MULTIVARIATE SURFACE GENERATION (MONTH 18)

Deliverable D4.4.1

Circulation:	PU: Public
Lead partner:	SINTEF
Contributing partners:	SINTEF, IMATI, IGN,
Authors:	Vibeke Skytt, Davide Sobrero, Simone Pittaluga, Guiseppe Patane, Clement Mallet
Quality Controllers:	Nicolas Paparoditis
Version:	1.0
Date:	April 30, 2014

Consisting of

SINTEF	STIFTELSEN SINTEF, Department of Applied Mathematics, Oslo, Norway
Fraunhofer	Fraunhofer Institute for Computer Graphics Research, Darmstadt, Germany
CNR-IMATI-GE	Institute for Applied Mathematics and Information Technologies of the National Research Council (CNR-IMATI), Genova, Italy
MOSS	M.O.S.S. Computer Grafik Systeme GmbH (MOSS), Munich, Germany
HRW	HR Wallingford Ltd (HRW), Wallingford, UK
FOMI	Hungarian National Mapping and Cadastral Agency (FOMI), 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 & 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
0.1	10.03.2014		First draft
0.2	11.04.2014		Most material included, issued for comments
0.3	26.04.2014		Submitted to quality control
0.4	30.04.2014		New version to quality control
1.0	30.04.2014		Version submitted to Project Officer

[1] Integers correspond to submitted versions

EXECUTIVE SUMMARY

This report describes the progress made in the first year of the development of the toolbox for Task 4.4 *Multivariate Surface Generation.*

A number of services have been implemented and the report starts with describing how these services fit into the value chain and the scenarios identified in the IQmulus project. The focus is on the first Land scenario and the Marine scenario, but several of the services are of a type that also can be utilized in a broader context. Some libraries used in the service development are presented: the GoTools library, which is a geometry toolkit developed by SINTEF, and the open libraries LibLAS, LidarFormat, GDAL, Triangle Software, Ann and RPly. Data sets from Liguria and Brittany were used for testing the software and are shortly described in this report.

Several of the services are directed towards generating a surface, which represents either an interpolation or approximation of the targeted continuous phenomena measured by data acquired by some instrument, while others act as utilities. The services developed so far offer solutions for surface generation with smoothness properties (e.g., LR B-splines or radial basis functions) as well as linear interpolation (triangulation). Depending on the physical surface to be approximated, the most appropriate approximation method can be chosen.

The services are described individually with some detail. The purpose and the method used in the implementation are explained and application examples are provided using the presented data sets. Some background material for the new technology of LR B-splines is given in an appendix, and uniform, tabular descriptions of all services can be found in a second appendix. All services have been submitted to the IQmulus service repository.

Geospatial data sets typically contain huge amounts of data. The main strategy for handling these data sets is parallelization and Cloud computing. This aspect of Task 4.4 will get larger emphasis in later deliverables. We also aim for creating geometry representations that offer good data compression while maintaining the characteristics of the input data. The table below relates this deliverable to the big data indicators.

Big data indicator	**Relevance**	**Explanation**
Volume	High	The specific examples run for the 10 services of the current toolkit are described in the sections 4.x.3 and results are evaluated in sections 4.x.4. The outlook for tackling very large data sets is discussed in section 5.
Variation	Medium	The input data is structured and geographically localized. The sample data sets used are described in section 3 and consist of LIDAR data from airborne laser altimetry, digital terrain models in an ASCII tabular format, and a LIDAR point cloud in LAS format.
Velocity	Medium to High	The velocity of the computations is part of the service evaluations in sections 4.x.4. Data streaming is not considered. Though rainfall data in 30 minute intervals are investigated in some services, they are not handled in real-time computations.
Analytics	High	Making big GIS data small to facilitate their efficient use in advanced workflows is the goal of some services. Identifying the information contained in the data through the use of triangulations and smooth spline surfaces is the main thrust of the foreseen surface generation services.

TABLE 1: The relation between this report and the big data indicators

TABLE OF CONTENTS

1 INTRODUCTION

This report describes the services developed for the Month 18 delivery of Task 4.4. In the remainder of the first section, the services are put into context with the IQmulus project as a whole and a first introduction to the services is given. In section 2, background libraries used in the service development are described, and in section 3 data sets used for testing are presented. In section 4, the services are presented in some detail. The continuation of the current work is discussed in section 5 and a conclusion can be found in section 6. Appendix A contains some background material on LR B-splines which is used in several services. Finally, a unified tabular description of the services can be found in Appendix B.

1.1 SCOPE OF TASK 4.4

Task 4.4 has a twofold focus: integration of data points from different types of sources, i.e. heterogeneous input, and surface generation. Different data sets are to be integrated and provided with the correct geo-spatial and temporal alignment. Furthermore, the data should be processed into a surface representation suitable for subsequent operations, for instance flooding simulations. Also additional information related to geospatial data is within the scope of Task 4.4; in this context, we focus on rainfall data.

For IQmulus as a whole a main issue is the fact that current data acquisition technology provides huge amounts of data. This aspect is important also for Task 4.4. Cloud computing and parallelization is an important strategy for handling large data sets and Hadoop and MapReduce are the selected tools for handling parallelization in IQmulus, see D2.3.1. In the context of Task 4.4, tiling of data sets is an important pre-process for parallelization. In addition to the problem of actually performing computations on large amounts of data, data compression is an important issue. In Task 4.4, we are working on generating surfaces that can lead to good data compression if the initial data are reasonably smooth.

1.2 RELATION TO USE CASES AND TO OTHER TASKS

In the WP1 deliverable D1.2.2 *Consolidated User Requirements*, a number of user stories are analyzed and three scenarios are presented, one marine scenario and two land scenarios. Later on, an urban scenario has been added as well. The use cases and scenarios are important input when defining a work plan but do not represent the only criteria for the planning. Three aspects are taken into consideration:

- The use cases and the need for tools in geo-spatial data processing;
- Huge amounts of data and a need for effective processing of large data sets. In this context cloud computing and parallelization are important.
- Research content.

The two scenarios that are primarily addressed in this deliverable are the first land scenario and the marine scenario. The developed land showcases are directed towards disaster management in the case of flooding, with the first one concentrating on the preparation phase. The marine showcase focuses on deriving an accurate elevation model from heterogeneous data to support further analysis and processing tasks. The context and showcase where a particular service is expected to contribute will be referred to in the presentation of the individual services and also in the appendix tables.

WP4 is the work package for the development of methods for data processing in IQmulus and Task 4.4 is one out of four development tasks. Task 4.4 is positioned internally in a value adding chain for geo-spatial data. Registration and pre-processing of the point clouds are required preconditions for obtaining good results from the Task 4.4 services. Service #7, static tiling of LAS file, of Task 4.3 will provide important pre-processing for the parallelization of surface generation. Feature points and lines, also results from Task 4.3, will help in guiding the generation of approximating surfaces, and outlier detection is crucial with regard to surface quality. Task 4.5 on change detection is parallel to or lies after Task 4.4 in the value chain. Finally the results of the services must be visualized using tools from WP5.

1.3 OVERVIEW OF SERVICES

Two areas within the field of surface generation are emphasized in the first project period. Triangulated surfaces have already performed well in the context of geographical information systems. They provide a flexible and efficient tool for the organization of scattered data points and handle large variation in the data points well. Spline surfaces, and in particular locally refined spline surfaces, are not much used in this field yet. However, due to their compact nature, they are able to represent smooth or mainly smooth terrains with small amounts of data.

In the context of heterogeneous input data, the work so far has been on rainfall data. Two services regarding this topic have been developed: one of them utilizes yet another surface type, namely radial basis function surfaces as another type of smooth surfaces with widespread applications.

The selection of the services being implemented in this period has been based on the user stories and scenarios defined in D1.2.2 and have been subject to discussion within WP4 (see D4.1.2 *Development guidelines for data integration and processing – Version 2*).

IQmulus has the ambition to handle very large data sets and in order to do this, cloud computing and parallelization using Hadoop and MapReduce are important tools. These concepts will play a more important rule later on in the project.

Below is a list of Task 4.4 services implemented for this deliverable. The services are available in the RedMine repository used by IQmulus as executables with documentation. The services will be presented in some detail in Section 4 of this report.

Id	Purpose	Lead Partner	Acronym	Initial test
49	Constrained triangulation	IMATI	Constrained triangulations	OK
51	Triangulation of gridded point cloud	IMATI	Triangulation of gridded point cloud	OK
66	Point cloud dimensionality	IGN	PCD	OK
9	Spline surface from parameterized point cloud	SINTEF	ParamPoints2Spline	OK
55	Spline surface from gridded point cloud	SINTEF	Grid2Spline	OK
56	Parameterize triangulated point set	SINTEF	Triang2ParamPoints	OK

84	Parameterize by projection	SINTEF	ParameterizeOnSurf	OK
57	Update spline surface with point cloud	SINTEF	UpdateSplinePoint	OK
40	Approximation of rainfall data with ordinary kriging	IMATI	krigingRain	OK
67	Approximation of rainfall data with radial basis functions	IMATI	RBF_RainFall	OK

TABLE 2: Overview over T4.4 services

The delivered services have been subject to an initial testing as indicated in column 5 in Table 2. It has been checked that the service actually exists in the repository and that it can be fetched and executed.

2 LIBRARIES

In the following, libraries used in the implementation and testing of the Task 4.4 services are presented.

2.1 GOTOOLS AND SISL

GoTools and SISL are in-house geometry libraries from SINTEF.

SISL is SINTEF's spline library. It is a mature library written in C. It handles NURBS curves and surfaces. It contains a number of methods to define geometry, but has its strength in operations related to curves and surfaces, in particular intersection functionality.

GoTools is a collection of geometry libraries which partly overlap with SISL. Both frameworks contain the creation of NURBS curves and surfaces and operations upon them but the functionality is not exactly the same: GoTools also offers volume entities, trimmed surfaces and topology structures as well as more functionality tailored for specific purposes. It supports LR B-spline surfaces and volumes. LR (Locally Refined) B-splines is a representation format that allows for local refinement of spline surfaces and volumes and will be explained in some detail in Appendix A. GoTools has interfaces to the standard transfer formats STEP and IGES. GoTools is written in C++.

SISL and parts of GoTools haven been made public as GNU GPL code. SISL is expected to work on most platforms. GoTools supports Linux and Windows with the Visual Studio compiler.

The IQmulus services related to spline surface generation, including locally refined splines, and parameterization of data points depend on GoTools. The related services are #9, #55, #56, #57 and #84. The following GoTools modules are used:

- Core functionality for curves and surfaces. This includes spline curves and tensor product spline surfaces, methods to construct these geometry entities and operations involving the entities.
- LR B-spline surfaces.
- Parameterization of scattered data points. The points are expected to be organized in a triangulation or an ordered graph.

Code developed in the IQmulus project will, when appropriate, be integrated into GoTools.

2.2 LIBLAS

LibLAS is a C/C++ library for reading and writing the very common LAS LiDAR format. The ASPRS LAS format is a sequential binary format used to store data from LiDAR sensors and by LiDAR processing software for data interchange and archival. LibLAS is open source and can be embedded in commercial/non-commercial applications as long as the restrictions of the BSD license are followed. LibLas is used to prepare data in LAS format for service #9 and service #57.

2.3 LIDARFORMAT

LidarFormat is an open source C++ library for handling 3D point clouds with a variable number of attributes. The library is provided by IGN. LidarFormat is used in service #66.

2.4 GDAL

GDAL (Geospatial Data Abstraction Library) is a library for reading and writing raster geospatial data formats. It provides support for a number of raster formats. GDAL can be used to convert GeoTiff data prior to the use of service #55 and is used also in other services.

2.5 TRIANGLE SOFTWARE

Triangle Software was developed by J.R. Shewchuk. It is a two-dimensional mesh generator and provides among other functionality the possibility of creating a number of Delaunay triangulations. Service #49 is based on Triangle Software.

2.6 ANN

ANN is a C++ library for exact and approximate nearest neighbour searching used in service #66.

2.7 RPLY

RPly is free software for reading and writing data in Ply format. This software is integrated into service #56 and used to read a triangulated surface.

3 DATA SETS

This section contains an overview of the data sets used for the testing of the Task 4.4 services.

3.1 TOPOGRAPHY OF A SMALL POCKET BEACH IN BRITTANY (PORSMILIN)

The LiDAR data from a small pocket beach in Brittany were obtained by airborne laser altimetry. The point cloud data set is provided by partner UBO. The data set contains two point clouds, which were acquired in 2009 and 2010. The point clouds are in xyz ASCII format. The point clouds from 2009 and 2010 contain 2 091 660 and 8 708 846 data points, respectively. In Figure 1, the two point clouds are shown in the same picture, the 2009 cloud in red and the 2010 cloud in blue.

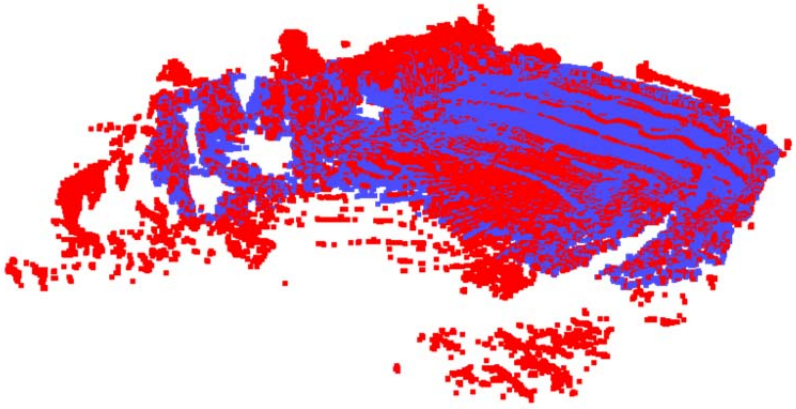


FIGURE 1: The two point clouds from Porsmilin

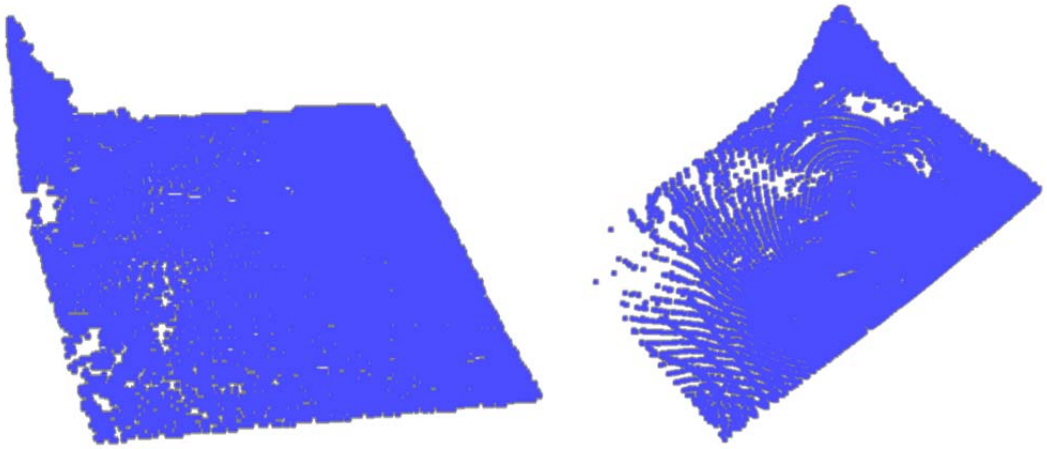


FIGURE 2: Sub clouds extracted from the Porsmilin 2010 and the Porsmilin 2009 cloud

A subset of the Porsmilin 2010 data set is used to test services #9, #56, #57 and #84. It is shown in the first picture of Figure 2. Another subset, this time from the 2009 cloud, which is shown in the second picture of Figure 2, is used in section 4.4.2.2. The first sub-cloud contains 322 619 points and the second one 842 532 points.

3.2 BATHYMETRY OF SUBMARINE SAND DUNES IN BRITTANY (FOUR)

FIGURE 3: The Four data set shown as xyz points

Two data sets from UBO are digital terrain models of submarine sand dunes. The data sets are provided in an ASCII tabular format. A data set from 2010 contains height information for 1 025 866 cells and the total height difference is about 34.5 metres. The cell size is 2x2 metres. The data set from 2011 has 3 332 806 cells. Figure 3 shows the data set from 2010. The data set is visualized as an xyz point cloud. The data set from 2010 is used in service #55.

3.3 LIDAR DATASET FROM LIGURIA

Figure 4 shows a LiDAR point cloud from Vernazza, Liguria. In the first picture there are 132 284 points given as xyz points. The version in the other picture is stored in LAS format and contains 4 134 193 points. In Figure 5, the latter cloud is coloured for better showing its features. This figure is identical to Figure 6 in the D4.2.1 report.

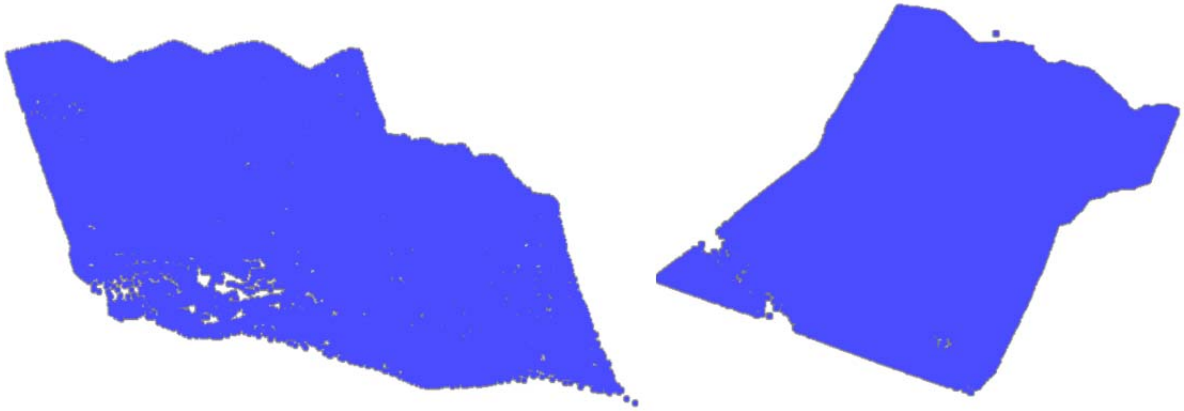


FIGURE 4: Two versions of a point cloud from Liguria

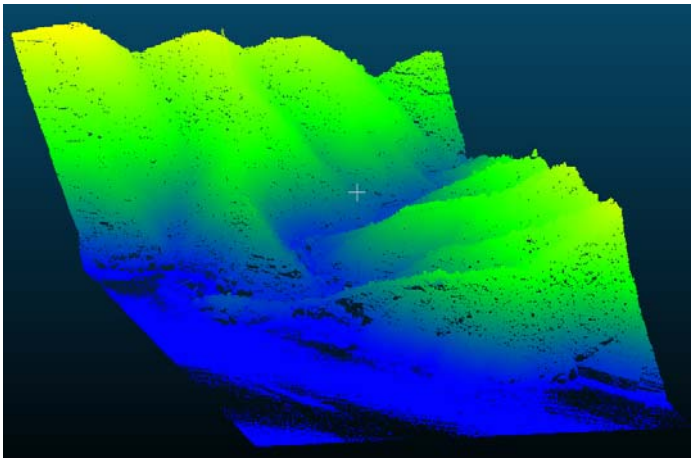


FIGURE 5: Coloured version of the largest point cloud

Both the full and the reduced version of the point set shown in Figure 4 are used in the presentation and testing of service #9. The data sets are also used in service #49.

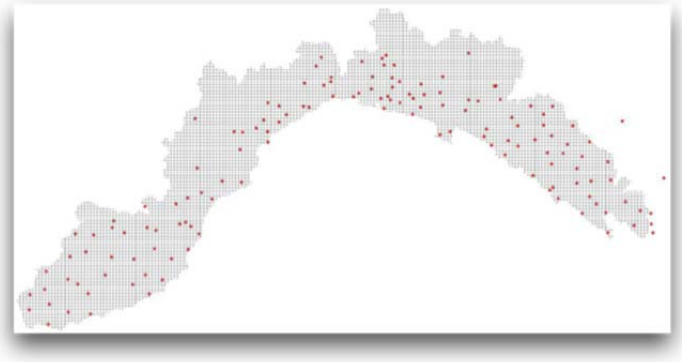


FIGURE 6: Rainfall measures $f(p_i)$ at 143 stations (regional level, red points) and 35 stations (municipality level)

Figure 6 shows rainfall data in the whole region of Liguria. This information is used in the services #40 and #67. Furthermore, service #51 creates a triangulation based on the corresponding geo-spatial data. This data set is an SRTM model with one point per 100 meters.

4 SERVICES

The individual services are described below. The level of detail in the presentation may vary from service to service depending on the complexity and type of a given service.

4.1 #49: CONSTRAINED TRIANGULATION

This is a utility service that, given as input a point cloud and a set of constraints in terms of segments and/or polygons, builds a constrained Delaunay triangulation, that is, a triangulation in which the constraints given as input are forced to be elements of the triangulation (edges of the triangulation).

4.1.1 Purpose and context

The aim of the service is to build a constrained triangulation starting from an input point cloud, usually resulting from a LiDAR acquisition pipeline of a terrain.

The Delaunay triangulation is based on a geometric optimization criterion which minimizes the occurrence of long thin triangles, a very good property for reducing the overall approximation error. However, since the criterion is purely geometric, the Delaunay triangulation might fail in respecting the morphology and semantics of the underlying surface. The most frequent problem is that the Delaunay triangulation produces an interpolation of the data whose boundary coincides with the convex hull of the data. Therefore, if we want to restrict the interpolation to a specific boundary (e.g., the actual administrative boundary of a given geographical area), we have to process the triangulation and trim the triangles that are not within the boundary of the geographical area.

Another important issue is that, if the acquisition density is not sufficient, important morphological characteristics, such as for instance ridges or ravines, might be "cut" by the Delaunay optimization criteria and in the end not be represented as a continuous line. Therefore, since we deal mainly with geographic data sets, it is likely that users need to take into account the presence of either morphological characteristics or cuts in the data set generated by the acquisition process. This led us to the idea to implement a service to build constrained triangulations.

This is a utility service that can be applicable in the context of all scenarios, for instance every time that a processing step requires a surface represented as a triangulation and users have specific constraints to be included explicitly in the surface model.

The service can be combined with other services, either to calculate the input of the service itself (i.e. service #11 from D4.2.1 *Spatio-temporal data fusion toolkit – Version 1*), or to build a triangular mesh needed by other processing services (for instance for service #64 from D4.3.1 *Feature extraction and classification toolkit – Version 1*).

Moreover, this functionality may also be useful for visualization, since the most common visualization techniques are triangle-based.

4.1.2 Method and implementation

The current implementation of the service takes as input a point cloud in the LAS file format and a set of constraints (lines or polygons) from a file in ShapeFile format. The service builds a constrained Delaunay triangulation of the set of input points projected into the xy plane, and then re-projects the result into space by adding the original z coordinate. The output is a PLY file with the triangulation and the original ShapeFiles of the constraints, provided for visualization or further processing purposes.

The current implementation is based on the Triangle Software developed by J. R. Shewchuk: for details, see [9] and [10]. We made this choice for the first implementation of the service because this software is currently the state of the art in Delaunay triangle mesh generations, it is fast and reliable, and it is stable in terms of numerical computation.

The service makes also use of the LibLas library presented in section 2.2 to read and convert the input point cloud.

4.1.3 Example(s)

We show the results of the execution of the service on an intermediate version of the data set of the Vernazza (Liguria) area introduced in section 3.3.

FIGURE 7: Triangulation of the Vernazza tile with the border constraint as shown in the planar projection on the right

In Figure 7, we can see the triangulation obtained by imposing the polygon shown on the right as border constraint to the service. The starting data set is a down-sampled version of the Vernazza LAS file, and the final triangulation is composed of 689,733 vertices and 1,378,303 triangles. The border polygon was extracted with the α-shape algorithm from the input point cloud.

4.1.4 Evaluation of the service

The “divide and conquer” algorithm that is implemented in the service is the fastest algorithm for Delaunay triangulation, and its theoretical complexity is $O(n \log n)$, where n is the number of the points to be triangulated. The enforcement of the polygon constraints and the removal of unwanted triangles do not affect the overall complexity.

	input		*output*	*time*		
name	**# points**	**size**	**Size**	**read**	**process**	**write**
vernazza	692,321	14 MB	51 MB	0.0882962	2.24969	1.47064
D441309678-0101	4,134,193	111MB	354MB	0.525587	11.6418	12.2385
strip40	10,851,327	290MB	950MB	1.28719	30.77	29.0053

TABLE 3: Size and calculation times for different input datasets

We have tested the service on some of the tiles of the dataset of Regione Liguria obtained from the LiDAR acquisition campaign organized by the Italian Ministry of Environment.

In Table 3 we can see some statistics for three of the data sets we have tested. We ran the tests on an Ubuntu 13.10 64 bit computer with an Intel® Core™ i7-3770 CPU @ 3.40GHz × 8 and 16 GB of RAM. We can see that also for the biggest input, the strip40 dataset, we had the desired result in about a minute of computation. Another important observation is the fact that the process time and the writing time are almost equivalent.

At the current implementation stage the service is not suitable to be executed in parallel: to adapt the service to a parallel framework, such as MapReduce, it will be necessary to redesign all the implementation. Nevertheless, we are considering the possibility of implementing a parallel version of this service, since the data sets we have successfully tested represent usually a small geographical area, and we are interested in increasing the size of the area to be processed.

The service can be found under WP4/49-CONSTRAINED_TRIANGULATIONS in the RedMine repository. It contains an executable and one data set. The use of the service is explained in a README file. The service works with Ubuntu 13.10. It is selective regarding the version of LibLAS used.

4.2 #51: TRIANGULATION OF GRIDDED POINT CLOUD

This is a utility service that, given as input a gridded point cloud in GeoTIFF format, builds a triangulation by using a simple algorithm that chooses one of the two diagonals of every square of the grid to create the two triangles covering the square itself.

4.2.1 Purpose and context

The aim of this service is to build a triangulation starting from a gridded point cloud in GeoTIFF format, usually a DTM (or DSM) of a selected geographical area.

This is a utility service that can be applicable in the context of all scenarios, for instance every time that a processing step requires a surface represented as a triangulation.

The service can be combined with other services that need a triangular mesh instead of data in a raster format for other processing steps (for instance service #64 from D4.3.1, service #40 and service #67 from D4.4.1).

Moreover, this functionality may also be useful for visualization, in particular to show the three-dimensional appearance of a DTM/DSM instead of a planar image.

4.2.2 Method and implementation

Since the input domain of the service is a regular grid of points in space, the algorithm to build a triangulation is straightforward.

The first step is to load the GeoTiff file and store all the points in a matrix; then, simply visiting the matrix, we decide for every rectangular block how to generate the two triangles needed. In this first version, we have chosen the shorter of the two diagonals of the square in the space, just to have a more regular structure of the triangulation.

Finally, we output the triangulation in PLY format.

4.2.3 Example(s)

We show an example of the result of the functionality on the SRTM model (downscaled 1 point every 100 meters) of the whole Regione Liguria: on the right, you can see the whole mesh, while on the left we show a detail of the regular structure of the mesh.

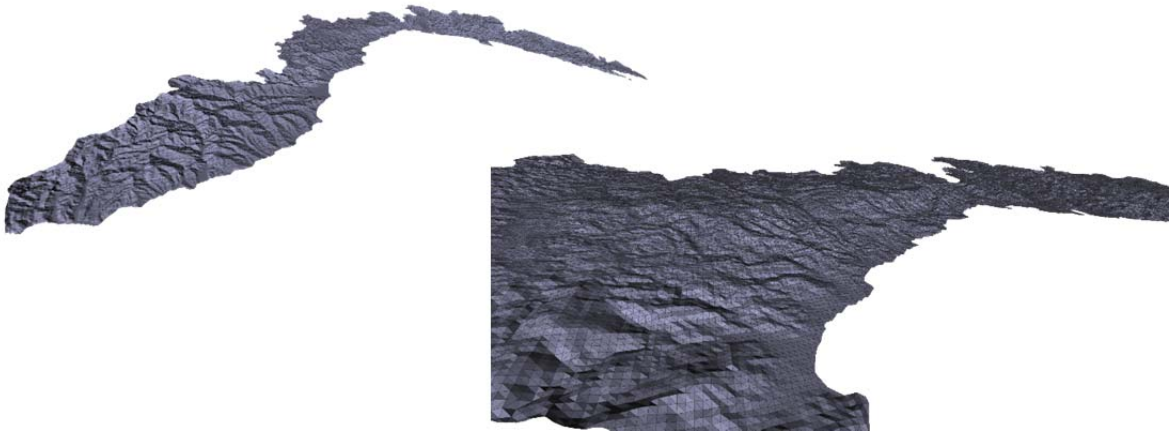


FIGURE 8: Triangle mesh of Regione Liguria and zoom-in

4.2.4 Evaluation of the service

At the current implementation state, the computation time of the service is linear in the dimension of the input; the same consideration applies also for memory consumption. Therefore we can manage data sets that are large as the main memory of the hardware infrastructure available.

	input [GEOTiff]			output [Ply]				
Name	**width**	**height**	**Size**	**points**	**triangles**	**size**	**time**	**dry run**
Aveto	2746	2270	12 MB	4,553,184	9,096,237	343 MB	28,6409	0.523631
la_spezia	6570	4018	51 MB	13,614,884	27,195,012	1.1GB	89,4468	1.55679
sestri_levante_alto_vara	9336	5857	105MB	21,880,400	43,729,461	1.7GB	143,743	2.70679
genova_portofino	12158	6316	147MB	25,449,659	50,843,592	2.0GB	178,032	3.21716

TABLE 4: Data size and calculation time for different GeoTiff input files

In Table 4 we show some experimental results: we tested the service on different portions of the whole DTM of Regione Liguria, which has a grid resolution of 5x5 meters (dataset #36 in the eRoom metadata table).

We performed our tests on an Ubuntu 13.10 64 bit computer with an Intel® Core™ i7-3770 CPU @ 3.40GHz × 8 and 16 GB of RAM. The calculation time (“time” column) includes also the writing of the output on the disk, which is the most time-consuming phase. Indeed, from the dry-run

execution time reported in the last column, we can clearly see that most significant part of the time is dedicated to the output phase.

Due to the regular input domain and to the simplicity of the algorithm, it seems to be feasible and relatively simple to adapt the service to the Map/Reduce paradigm: in this way, we could also work with a data set that does not fit into main memory.

The service can be found under WP4/51-TRIANGULATION_OF_GRIDDED_POINT_CLOUD in the RedMine repository. It contains an executable and two data sets. Two additional data sets can be found together with an earlier version of this service. The usage of the service is explained in a README file. The service is easily executed.

4.3 #66: POINT CLOUD DIMENSIONALITY

Point cloud dimensionality is a utility service which lies in a grey area between Task 4.4 and Task 4.3 on feature extraction, classification and correlation. It is applicable in the context of all scenarios. The service uses the LidarFormat library presented in section 2.3.

4.3.1 Purpose and context

This service is a multi-scale method that computes robust geometric features on LiDAR point clouds in order to retrieve the optimal neighbourhood size for each 3D point. Three dimensionality features are calculated on spherical neighbourhoods at various radius sizes.

4.3.2 Method and implementation

Based on combinations of the eigenvalues of the local structure tensor, the shape of the neighbourhood for each point is described, indicating whether the local geometry is more linear (1D), planar (2D) or volumetric (3D). A radius-selection criterion is used for finding automatically the optimal neighbourhood radius for each point. Such a procedure allows a dimensionality labelling, giving significant hints for classification and segmentation purposes. The service can be applied to 3D point clouds from airborne, terrestrial, and mobile mapping systems since no a priori knowledge on the distribution of the 3D points is required.

The input and output format is LidarFormat.

4.3.3 Example(s)

An example is provided in the RedMine service repository.

4.3.4 Evaluation of the service

Some parameters such as the minimum and maximum number of neighbours must be given.

The descriptors are low level features which cannot be directly evaluated as it depends on how they are used. However, they seem to outperform the state of the art when used for classification of urban scenes.

The service can be found under WP4/66-POINT_CLOUD_DIMENSIONALITY in the RedMine repository. It contains an executable and a data set. The usage of the service is explained in a README file. The service runs on Ubuntu 12.04, but not with Ubuntu 13.10. Some effort in addition to the information in the README-file is required to run the executable.

4.4 #9: SPLINE SURFACE FROM PARAMETERIZED POINT CLOUD

Approximation of scattered data points with spline surfaces is a well proven technology and works well if the data points are suited to be represented by a smooth surface. An LR B-spline surface is a new representation and using this technology to approximate terrain data is a research activity. Some background information on LR B-splines can be found in Appendix A. This service provides the foundation for all services related to LR B-splines and thus will be described in relative detail in the following.

The service is aimed at the Marine showcase and should appear in an appropriate chain of services.

4.4.1 Purpose and context

Locally refined spline surfaces are well adapted for the representation of smooth shapes, including terrains that are mainly smooth. In the right context, they provide good data compression compared to more point-based approaches. The effect is dependent on the smoothness in the data, the expected approximation accuracy and whether or not the data set is already reduced. It can vary from a factor of many hundreds to no real compression. The surfaces have got a global parameterization and a compact structure that are well suited for subsequent operations and comparisons between different representations of the same areas.

This service is directed towards the Marine showcase scenario. In general, the underwater structures are smoother than onshore landscapes even though some terrains in a Land scenario will also be well suited for approximation by locally refined splines. The service is not applicable to the Urban showcase. Spline surface approximations to geographical data are most appropriate for smooth terrains while in urban surroundings, the main characteristics are buildings with sharp edges. More than one surface would be required to represent that type of data and this is out of scope of the current service.

A parameterized point cloud is approximated by an LR B-spline surface. The point cloud should be pre-processed, meaning that it is expected that registration and removal of outliers have been performed. Feature information is not taken into consideration in this version of the service but the service will be extended in later versions to handle additional information.

The points may have associated parameter values. In that case a parameterized surface in 3D is produced. Otherwise, the xy-values of the points are used as a parameterization and a height function (1D surface) is generated. If a point cloud containing points without associated parameter pairs fails to be projectable onto the xy-plane but can be triangulated, service #56 can be used to parameterize the points. Then an approximating 3D surface can be computed by this service. The algorithm used in this service uses an iterative approach and the maximum number of iterations must be set as well as the expected accuracy of the approximation.

4.4.2 Method and implementation

The algorithm expects the data points to be provided either as

- xyz points where the x- and the y-coordinates are taken as parameter values and the z-coordinate gives the height value or
- each xyz point is associated with a parameter pair (u,v). The points are given as (u_i,v_i,x_i,y_i,z_i), i=1, ..., number of points.

If the service receives point clouds in LAS format, a pre-processing converting the data to the expected format is required. This can be done using libLAS.

- Translate the domain to be centred at (0,0) in the xy-plane. The purpose of this is to improve the numerical properties of the computations.
- The surface to be generated will be rectangular in the xy-plane. Thus, it will in many cases have a more regular shape than the given point cloud. As a pre-process to the approximation, extra points are generated in areas that will be covered by the surface, but where there are no or very few data points.
- Approximate the point cloud with a tensor product spline surface with few inner knots using a least squares approach with a smoothing term.
- Convert the surface to an LR B-spline surface.
- Until a prescribed number of iterations are reached, or the surface approximates the points within a given tolerance, or the algorithm fails to create an improved surface, perform the following:
    - Check if the accuracy is reached. In the functional case, the check is performed by evaluating the height function in the parameter given by the xy-value and comparing heights. In the 3D case, a closest point computation is required.
    - If not, insert new knot lines into the grid over which the surface is defined and update the surface accordingly.
    - If necessary, construct extra points in areas with few initial points, see section 4.4.2.5. These points will be treated separately.
    - Approximate the point cloud with the refined surface using a least squares approach with a smoothing term.

  Repeated iteration may not improve the accuracy of the surface. In that case the iteration is stopped to avoid increasing the data size without any gain. Small polynomial patches in areas with few or unevenly distributed data points may cause instabilities. If this occurs, the iteration should not be continued as the resulting surface will not possess the expected properties. In the next project period, an alternative approximation method based on the MBA algorithm, see [6], will be included in the algorithm to be used in such configurations.
- Translate the parameter domain of the function back to the initial domain in the 1D case. In 3D, the x- and y-components of the surface must be translated.
- Report on the accuracy of the approximation.

In many cases the surface should be trimmed to relate to the actual extension of the point set. This is not a part of the current version of the service but will be added later. Trimming information can be computed using service #66 or it can be provided from supplementary data sources as in service #49.

Several of the steps in the algorithm presented above need a more detailed explanation.

4.4.2.1 Least squares approximation with a smoothing term

Tensor product spline surfaces and LR B-spline surfaces are both expressed as combinations of coefficients and blending functions, i.e. the B-spline functions. The tensor product surface can be expressed as

$$F(u,v) = \sum_{i=1,j=1}^{n,m} P_{i,j} N_{i,d_1}(u) N_{j,d_2}(v) .$$

The surface has n coefficients in the first parameter direction and m in the second. The polynomial degrees are d_1 and d_2, respectively. Properties of an LR B-spline surface can be found in Appendix A and the surface is expressed as

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

In both cases the least squares approximation with smoothing is performed by minimizing the following expression

$$\alpha_1 J(\sigma) + \alpha_2 \sum_{r=1}^{R} (F(u_r, v_r) - a_r)^2$$

with respect to the surface coefficients. The data points a_r, $r=1, \ldots,$ *number of points,* are parameterized by their xy-coordinates if the surface represents a height function, otherwise they are parameterized with respect to the parameter domain of the surface. Several options for the smoothing term $J(F)$ exists, see [11]. We use the term

$$J(F) = \iint_{\Omega} \int_0^{\pi} \sum_{i=1}^{3} w_i \left(\frac{\partial^i F(u_0 + r\cos\varphi, v_0 \sin\varphi)}{\partial r^i} |_{r=0} \right)^2 d\varphi du_0 dv_0$$

The first, second and third partial derivatives of the surface are used in each point (u_0,v_0) in the parameter domain Ω to define directional derivatives which are integrated around the circle. The result is integrated over the parameter domain.

The minimization functional then leads to a linear equation system in the surface coefficients.

If appropriate knowledge exists, some coefficients can be assigned a value a priori and they will not be modified during the approximation. For instance, if an adjacent tile already has been approximated by an LR B-spline surface, the current surface can adapt to the boundary curve common to the surfaces and ensure continuity between them. Moreover, in areas where the required approximation accuracy has been reached or no data points exist, coefficients can be kept fixed to avoid unnecessary computations.

The approximation term and the smoothing term are weighed in the minimization functional by the factors α_1 and α_2. The importance of each term will depend on the characteristics of the current data set. If the corresponding terrain is smooth, the smoothing term can be prioritized higher than for data sets with larger variation. In general, the approximation term must be prioritized to achieve an acceptable accuracy. Consequently, there is a risk for oscillations in the surface in areas with no or few data points, especially if the data set has large variations in the neighbourhood. The need for another method to handle data sets of varying density arises. This is where the need for ghost points originates, see section 4.4.2.5.

4.4.2.2 Select new knot lines to insert

A part of the 2009 topographic point cloud from Brittany has been extracted and approximated with a tensor product spline surface. The major part of this area is very smooth, but some areas have a more complex configuration, in particular the upper left corner. The point set and the wire frame surrounding the polynomial pieces of the surface are shown in Figure 9. In addition the data points which are more distant to the surface than a threshold of 0.2 are shown. The surface clearly does not approximate the points well enough, not even with this relatively large tolerance. More degrees of freedom are required, but only in parts of the surface. This example illustrates the need for local refinement of the surface.

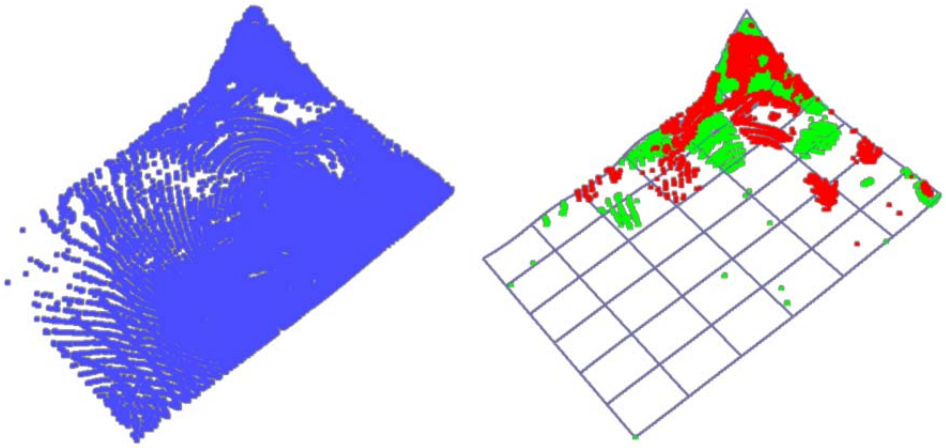


FIGURE 9: A subset of the topographic point cloud from 2009 of the Britanny pocket beach and a lean tensor product spline surface approximating this point cloud

Information about the approximation accuracy and the error distribution with regard to polynomial patches is collected. Some rules and guidelines govern the introduction of new degrees of freedom:

- A new knot line must be long enough to split the support of at least one B-spline.
- New knot lines must be introduced where the distance to the point cloud is large and above the given tolerance.
- Only a subset of the possible refinements is selected at each iteration step.
- B-splines with many elements in their support violate the property of local support applying to a surface.
- A rapid change in the size of neighbouring elements is not beneficial for the approximation procedure.

Figure 10 shows the wire frame of the surface and the most distant points in the next two steps of the iteration.

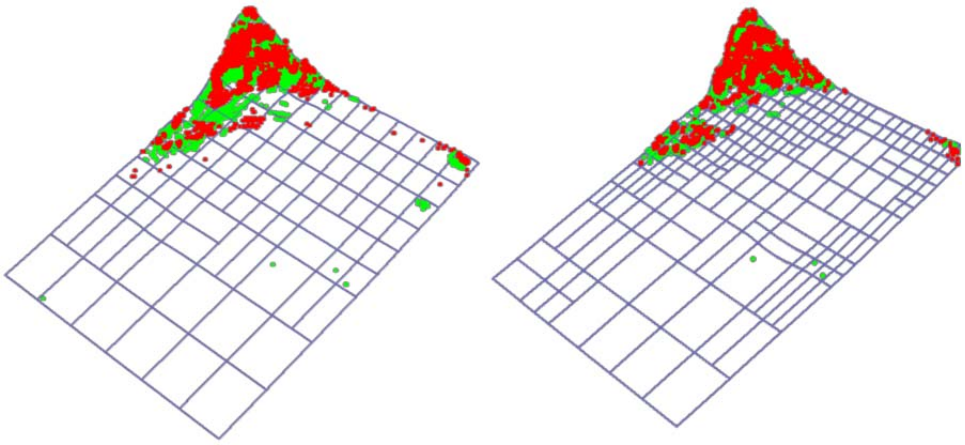


FIGURE10: Surface wire frame at the next two iteration steps

The refinement strategy will be subject to improvements in later versions of this service. In particular, the effect of outliers in the data must be minimized while the accuracy at isolated subsets of points needs more attention.

4.4.2.3 Knot insertion

A new knot line must span the width of the domain of at least one B-spline function. This B-spline function is a tensor product of two univariate B-spline functions. One of these is split into two by inserting the new knot. Let $B_i(u)$ be this univariate B-spline.

Let the knot vector corresponding to $B_i(u)$ be $u_i, ..., u_{i+d+2}$, where d is the polynomial degree of this B-spline. Let t be the new knot value to insert. Then $B_i(u)=\alpha_1 B_{i,,1}(u)+ \alpha_2 B_{i,2}(u)$, where $B_{i,1}(u)$ is the B-spline of degree d defined on the knot vector $u_i, ..., t, ..., u_{i+d+1}$ and $B_{i,2}(u)$ is defined on the knot vector $u_{i+1}, ..., t, ..., u_{i+d+2}$. The factors α_1 and α_2 are computed by the Oslo algorithm [3]. The combined B-splines are updated according to the new univariate B-splines. The coefficients and scaling factors of the LR B-spline surface are updated by accumulating the weights α_1 and α_2.

All B-splines having domains being split by the new knot line are split as well. More than one initial B-spline can give rise to the same new B-spline during this process. These occurrences are recognized and combined during the knot insertion process. Moreover, due to previous knot insertions and knot lines only partly spanning the domain of B-splines, the splitting of one B-spline may give rise to new splits where the knot line that used to span a part of the domain now spans the entire domain of the refined B-splines. Thus, the knot insertion process is a recursive process which continues until all B-splines are defined with minimal support.

4.4.2.4 Linear independence of B-spline functions, the peeling algorithm

The B-splines of an LR B-spline surface are not linearly independent by default and knot insertion into a surface where the B-splines form a basis for a spline space may lead to a situation with linearly dependent B-splines. A consequence of linear dependency is that the approximation algorithm will fail.

It is possible to define rules for knot insertions that will ensure that no linear dependency will arise but they tend to be more restrictive than necessary. Instead we choose to test the collection of B-splines at each step in the main iteration loop to decide if they are linearly independent or not.

An element, i.e. the parameter domain corresponding to one polynomial piece in the surface, is said to be overloaded if more than $(d_1+1)x(d_2+1)$ B-splines cover the element where d_1 and d_2 are the polynomial degrees of the B-splines in the first and the second parameter direction, respectively. A B-spline is overloaded if all the elements in its domain are overloaded.

The following observations gives rise to the peeling algorithm used to test for linear independency:

- If there are no overloaded B-splines, the B-splines are locally and globally independent.
- Only overloaded B-splines can occur in linear dependency relations.
- A linear dependency relation has to include at least two overloaded B-splines.

The algorithm systematically reduces the number of possibly overloaded B-splines until hopefully no candidates are left. A similar algorithm searches for overloaded B-splines at knot lines.

Experience indicates that a controlled insertion of new knot lines following the guidelines of subsection 4.4.2.2 will not give rise to any linear dependency but this observation is not formally proved. If a linear dependency situation occurs, additional knot lines designed to make the grid more regular will resolve the situation. So far in the experiments no additional knot insertion of this type has been required.

4.4.2.5 Construction of ghost points

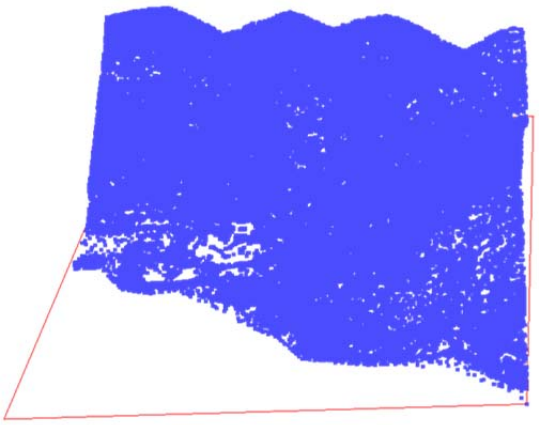


FIGURE 11: Point set and parameter domain

Consider Figure 11. A small point set from Liguria is shown together with a frame bounding the points in the xy-plane. This frame represents the parameter domain of the height function we want to represent as an LR B-spline surface. There is clearly a large area where the shape of the surface is undefined and it is required to stabilize the approximation in this area. We have chosen a general approach for stabilization and construct new points based on existing points in the neighbourhood.

Constructing new points outside the boundaries of an existing point set is an extrapolation operation, and this tends to be unstable. Given an area without data points, spline surfaces in nearby areas in both parameter directions are created and extrapolated. Several surfaces of varying degrees are combined and evaluated to create additional data points, called ghost points of level 1. The choice of using several different surfaces in this construction is based on the need to minimize the extrapolation effect.

The first picture in Figure 12 shows the initial point set and the ghost points belonging to level 1.

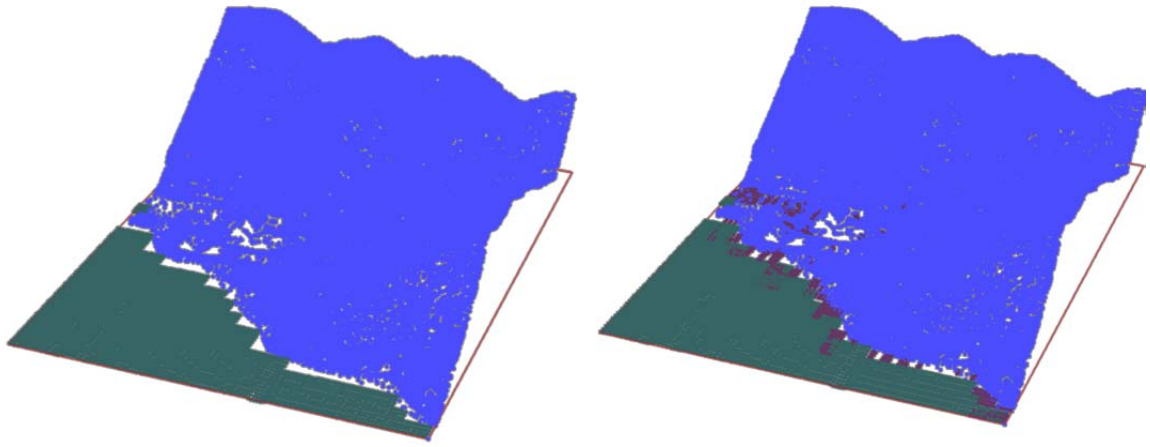


FIGURE 12: Point set and ghost points, level 1 and 2

As the refinement proceeds new polygonal pieces in the surface will be sparsely populated with data points. Ghost points of level 2 are added in these areas during the main iteration loop by evaluating points in the current approximating surface before the refined surface is updated. Ghost points belonging to level 2 are purple in the second picture of Figure 12.

The ghost points are included in the surface approximation at each step of the main iteration but they are not considered in accuracy computations or refinement.

The concept of ghost points is fruitful and is an important part of the algorithm. The current procedure does, however, have some drawbacks:

- Construction of ghost points, in particular at level 1, is time consuming.
- Extrapolation is an unstable operation. If additional information exists, for instance from adjacent data sets, this information should be preferred.
- The constructed ghost points are static. It turns out that they sometimes destroy the possibility of accurately approximating sparse initial points in an area where ghost points are constructed. If this situation occurs mechanisms for improving existing ghost points must be implemented.

The construction and handling of ghost points will be improved in later versions of this service.

4.4.3 Example(s)

The first example uses a subset of the Porsmilin topography data set from 2010 presented in section 3.1. Similar to the point cloud used in section 4.4.2.2, the subcloud is smooth in some areas and more complex in others. The subcloud does not have large areas without data points. Thus the need for ghost points, see section 4.4.2.5, is limited. In this example a height function approximating the elevation information will be generated, while an example where the points are approximated by a 3D surface will be presented in the sections 4.6.3, 4.7.3 and 4.8.3.

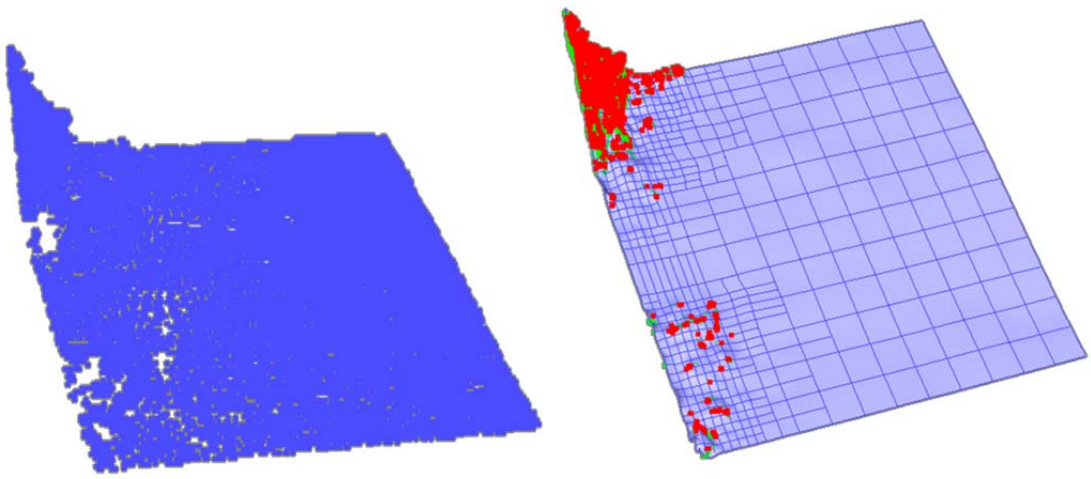


FIGURE 13: Point cloud and an intermediate approximating surface

Figure 13 shows the point cloud and an intermediate surface together with points being more distant to the surface than a threshold of 0.2. These points are concentrated in two areas and we can see that the surface already has more knots in these areas.

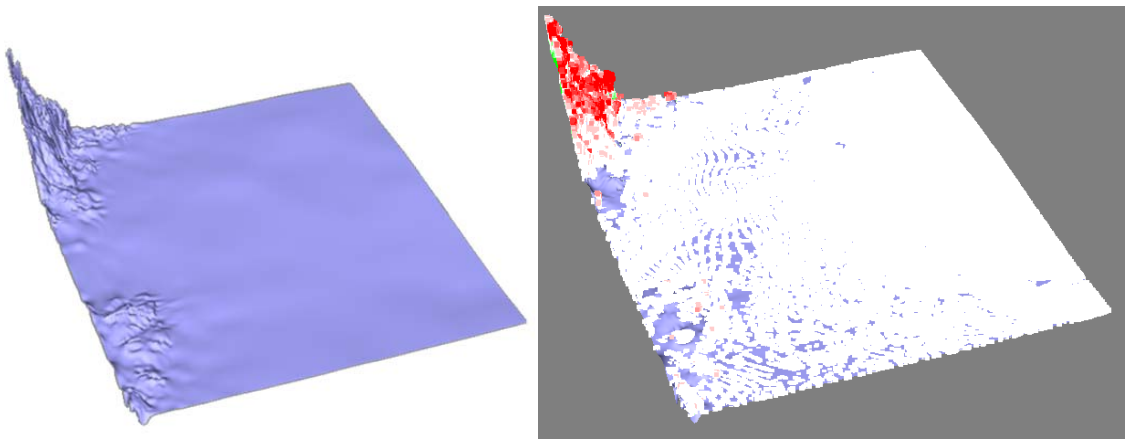


FIGURE 14: Final surface and the surface together with the point cloud

Figure 14 shows the final surface after 4 iterations. The white points lie within the threshold while red and green points lie above or below the surface, respectively. The maximum distance is 2.7 and the average distance in all points is 0.05. The surface has 2892 elements, i.e. polynomial pieces, and 2892 points out of 322 619 have a distance larger than the threshold. If we look at the details in the steep area where the largest errors can be found, see Figure 15, we see that there are very close points on both sides of the surface. In the second picture only points more distant than the threshold is shown to get a better view. The distances are measured in the

z-direction. If we convert the surface to 3D and looks at the closest distance between the points and the surface, the maximum distance becomes 2.3 while the average distance is 0.03.

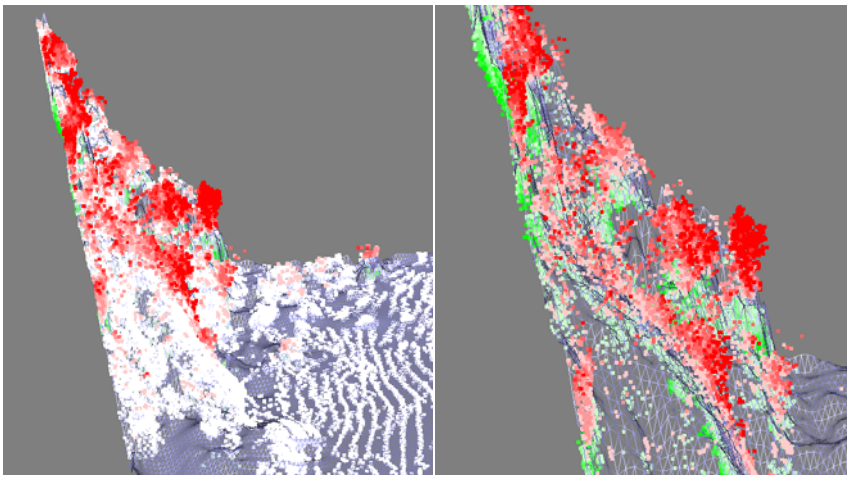
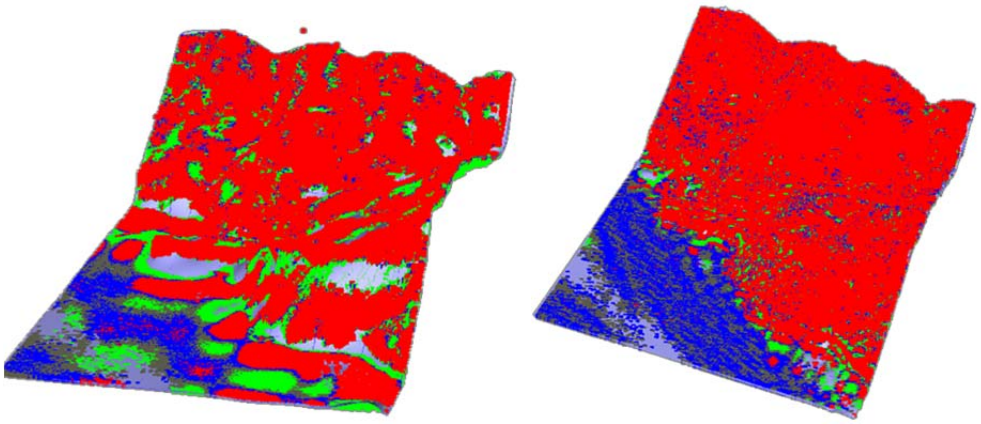


FIGURE 15: Details in the steep area, surface and point cloud



FIGUR 16: Surface and point cloud after 1 and after 6 iterations

The second example uses the LAS point cloud from Liguria, see section 3.3. A threshold of 0.5 is used and the height difference in this point set is 396.5. Figure 16 shows the first and the final approximating surface. The point cloud is included and coded as follows: red and blue points lie above the surface, the blue points lie within the threshold. Grey and green points lie below the surface, where grey points are within the threshold. The maximum distance remains at about 292 metres and is related to an outlier while the number of points lying outside the threshold reduces from more than 4 million to 2.8 million and the distance to the surface among these points reduces from 13.2 metres to 1.9. This distance is still large compared to the given tolerance, but looking at the distributions of points below and above the tolerance as shown in Figure 18, we see that there are no areas where the surface is particularly badly adapted. Points lying below and above the surface are scattered over the entire area. The trend of the data is well captured, but some details are probably lost. This data set is related to the Land scenarios and is not typical for the type of data where we in particular intend to apply this service. The surface itself is shown in Figure 17.

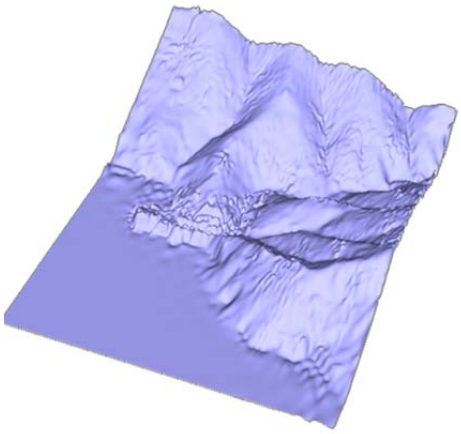


FIGURE 17: The approximating surface

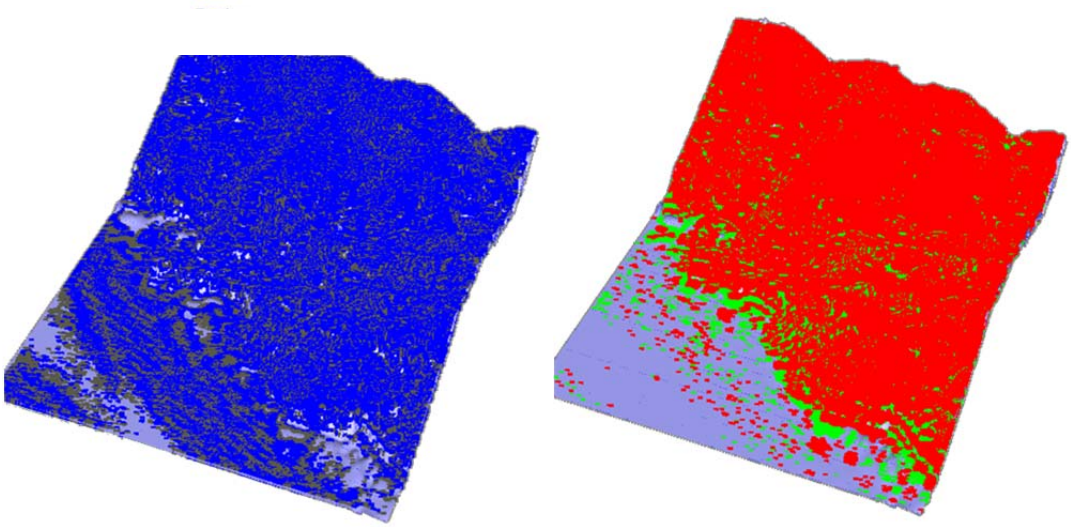


FIGURE 18: Distribution of points with respect to the tolerance

4.4.4 Evaluation of the service

The functionality performs well for smooth data sets and achieves good accuracy. For data sets with much variation, the service can be used to capture the trend of the terrain and details can be added using other surface types or additional LR B-spline surfaces. We have not pursued this approach yet.

The service does not work in real-time and is mostly suited for small to moderately sized data sets, up to a few million points. Most of the time is spent on checking the accuracy of the approximation in the given points and computing contributions from the point set to the global equation system. Thus, the computation time is mainly linear in the number of data points but the number of iterations also matters. 4 iterations of the small data set from Liguria (132 284 points) take about 5 seconds on a normal PC. A similar computation on the large data set (about 4 million points) takes about 2 minutes.

The algorithm creates a height function or a 3D surface depending on the format of the input points. The latter use is much more time consuming as accuracy in the points is computed with a closest point computation instead of a single evaluation. Large data sets need to be split before applying the service. The service is currently not prepared for MapReduce. On a detailed level MapReduce is not applicable for this service but MapReduce should be well suited for handling tiles where the algorithm is then applied once for each tile.

The level of data compression is related to the shape of the terrain represented by the point cloud and the expected accuracy. The point cloud from the first example in section 4.4.3 contains 322 619 points, i.e. 967 857 doubles. The approximating surface is quadratic and has 2892 polygonal pieces. Representing each polynomial as a Bezier function requires 46 272 doubles for the coefficients and 11 568 doubles for knot values. Other data sets will give a smaller or a larger data reduction effect.

Conversion of the data set from LAS to the xyz-format is required. This can be done using the LibLas library.

The iteration is stopped if the accuracy is reached, the maximum number of iterations is reached or the algorithm fails to create a better surface. Thus, the specified tolerance will not always be met, but detailed accuracy information can be achieved.

Outliers may fool the algorithm to do too much refinement in order to accurately approximate a subset of points that cannot be fitted with a smooth surface. In particular, if this configuration occurs close to areas with little data, the surface is likely to oscillate. We have implemented some mechanisms to handle areas without a sufficient number of points, and improvements are expected for further versions of this software. A fall-back strategy for outliers will be needed.

The service takes as input the maximum number of iterations and the expected accuracy. The accuracy should not exceed the accuracy of the input points. The number of iterations should be small, in the magnitude of 3 to 6. If this is not sufficient to reach a good result, problem areas should be treated separately.

The service can be found at WP4/09-SPLINE_SURFACE_FROM_PARAMETERIZED_POINT_CLOUD in the RedMine repository with an executable and two data sets. The usage of the service is explained in a README file. The service has been checked to run under Ubuntu 12.04.

4.5 #55: SPLINE SURFACE FROM GRIDDED POINT CLOUD

A spline surface approximating a gridded point cloud is generated. The service is very similar to service #9 and the same algorithm is used, but it differs with respect to the input data and the computation of the accuracy with respect to the point cloud. We will in this section describe how this service differs from service #9. For the algorithm in general, we refer to section 4.4.

4.5.1 Purpose and context

The service is primarily aimed at the Marine showcase scenario. The initial point cloud is pre-processed and represented as heights over a regular grid. It is expected that outliers are removed during pre-processing but areas without data may occur.

4.5.2 Method and implementation

The least squares approximation algorithm relates to parameterized data points. For gridded data only the functional approach described in section 4.4.2 applies. The input format describes the regular grid and height information. Prior to entering the least squares approximation, this information is transferred to xyz points where the x- and the y-coordinate represent the parameter pair and the z-coordinate represents the height.

The initial data relates the height information only to a particular cell and not to an exact xy-value. Thus, the computation of accuracy in the data points must relate to the cell as well. If any point within the cell that a given data point belongs to is within the prescribed tolerance, the accuracy in the point is accepted. The point is associated with the best accuracy computed in the cell.

4.5.3 Example(s)

The Four data set in Brittany described in section 3.2 is shown in the first picture of Figure 19

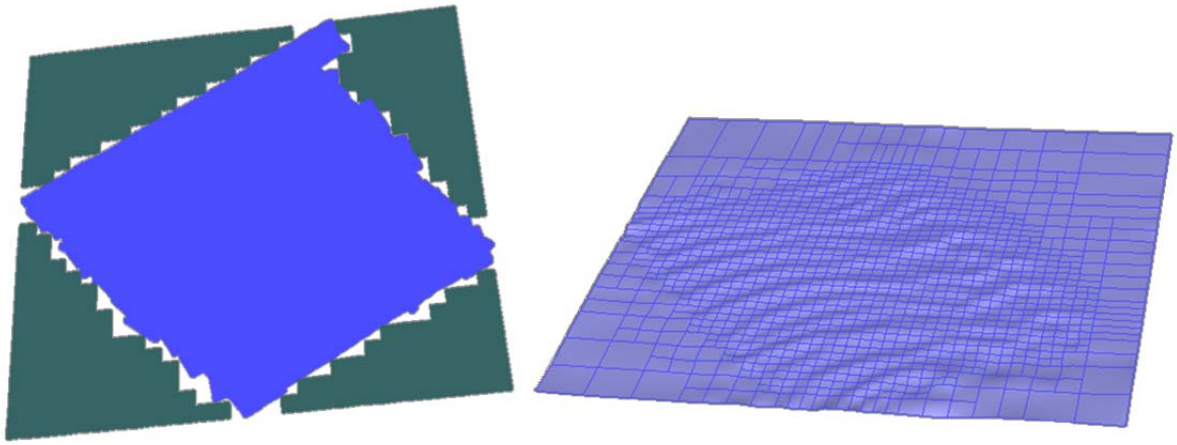


FIGURE 19: Submarine sand dunes and an LR B-spline approximation

along with ghost points completing the data set. An intermediate approximating spline surface is shown in the second picture. The ghost points are generated as described in section 4.4.2.5. The accuracy is not very good. About half the points are outside a tolerance of 0.2. The overall height of the data set is about 34.5. Still it is possible to see the structure of the sand dunes in this picture.

Figure 20 shows the surface obtained after 7 iterations and the corresponding parameter domain. This means more iterations than normally recommended and the surface consists of 55681 polynomial patches. The maximum distance from the surface to any cell is 0.95 and the average distance is 0.0089. 155 points lie outside of the threshold of 0.2, i.e. about 0.015%.

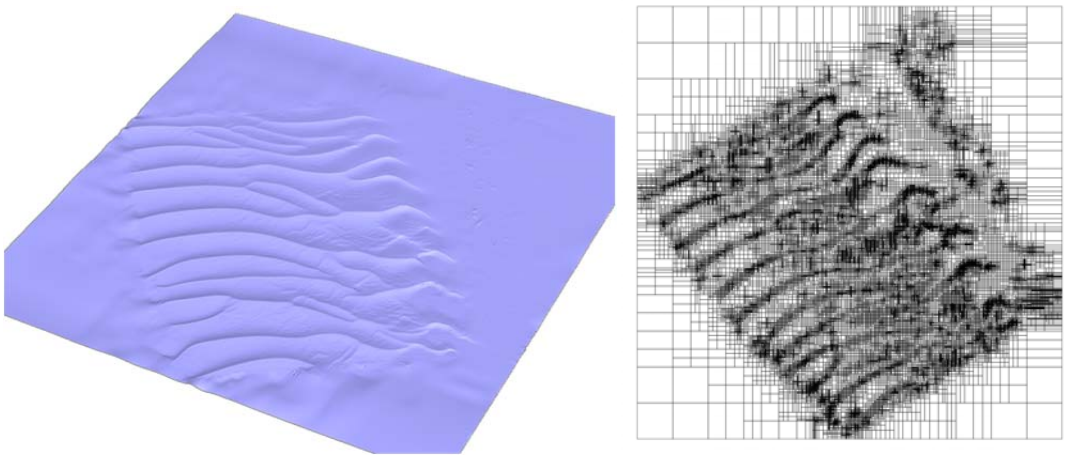


FIGURE 20: The final surface and parameter domain

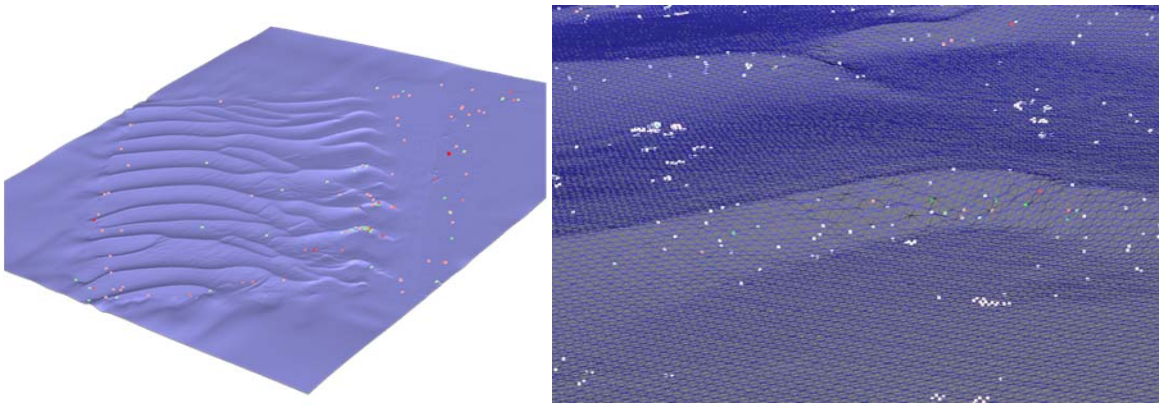


FIGURE 21: Accuracy of the surface with respect to the point cloud

In the first picture in Figure 21 the points being more distant to the surface than 0.2 are shown. The set consists, in general, of single points spread around the surface but has some concentration in two areas: to the right where the sand dunes are less pronounced and at the top of a couple of the dunes. The first of these areas has been subject to less refinement than the rest of the surface and indicates that the strategy for choosing new knot lines needs some attention. The second picture zooms in on the sand dune with the least accuracy. Now all points with a distance larger than 0.1 are shown. The green points lie below the surface while the red points lie above. We see that there are points both above and below the surface in this area which indicates local variations in the sand dunes.

The extent of the surface is larger than the point set. To complete the task of representing this point cloud with an LR B-spline surface, the surface must be trimmed. Trimming information can in this case be fetched from the gridded structure of the data set.

4.5.4 Evaluation of the service

This service uses the same algorithm as service #9 and the evaluation in section 4.4.4 applies also for this case. Extraction of the boundary information from the grid is not yet implemented. The service reads grid data from the ESRI ASCII format. GeoTiff data may be converted using the GDAL library presented in section 2.4.

At WP4/55-GENERATE_SPLINE_SURFACE_FROM_GRIDDED_POINT_CLOUD the service can be found in the RedMine repository with an executable and one data set. The usage of the service is explained in a README file. The service has been checked to run under Ubuntu 12.04.

4.6 #56: PARAMETERIZE TRIANGULATED POINT SET

Parameterization of scattered data points has been a subject for research in a number of years; an early contribution is Floater's shape preserving parameterization [4]. Later the concept of mean value coordinates [5] motivated by the Mean Value Theorem for harmonic functions was published. This service is based on an implementation of Floater's work in GoTools.

4.6.1 Purpose and context

This service is a general purpose algorithm for the parameterization of scattered data points organized in a triangulation. It serves as a support utility for the approximation of a point cloud with a 3D LR B-spline surface.

This functionality is most relevant in cases where the initial point set is not well parameterized by its xy-coordinates. This will be the case if the terrain has very steep slopes or overhangs. Then we may choose to represent it by a 3D surface instead of a height function. This will lead to higher computation times and a larger data size, but also to a more flexible surface representation.

The service is one piece in a chain. Prior to applying this service, the point set must be cleaned and thinned. The parameterization method is vulnerable to the size of the point set. A triangulation of the thinned point set must exist. Alternatively, the sparsest representation of a hierarchical triangulation can be used. After computing the parameterization, service #9 is used to create an approximating spline surface. Finally service #57 can be used to update this surface with the initial non-thinned point set.

4.6.2 Method and implementation

The point set is parameterized by expressing each point as a convex combination of its neighbours and solving a linear equation system to create a shape preserving parameterization. The size of the equation system reflects the number of data points, thus in IQmulus this method is not applicable on a complete data set.

The point set must be organized, in our case the points are triangulated, and the points at the surface boundary must be specified. Moreover we want to parameterize the point set on a rectangular domain. Thus the boundary vertices must be specified.

The algorithm works as follows:

- Read the input triangulation into the data structures of GoTools. The triangulation representation applies a half-edge data structure to represent adjacency.

- If necessary, re-orient the triangles to obtain a consistent direction of all triangle edges. The edges should be oriented counter clockwise around the triangles.
- Extract the boundary vertices from the oriented triangulation.
- Define the corner vertices. Currently an angular criterion is used, but more effort is required to get a good and stable selection of corners. Service #66 may be relevant in this context.
- The parameterization functionality requires that a boundary vertex has exactly two boundary vertices as neighbours. Ensure that this requirement is satisfied in the corner vertices.
- Use GoTools functionality to parameterize the boundary.
- Parameterize the inner vertices with respect to the boundary.

4.6.3 Example(s)

The current service is one component in a chain that is not yet completely put together. To mimic this chain in the current section and later example sections we create a triangulation from a rough surface approximation of a point cloud. Then the triangulated point set is parameterized and a 3D surface approximating the parameterized points is created. Finally, this surface is updated with respect to the initial point set. The second and third part of this example will be presented in sections 4.7.3 and 4.8.3.

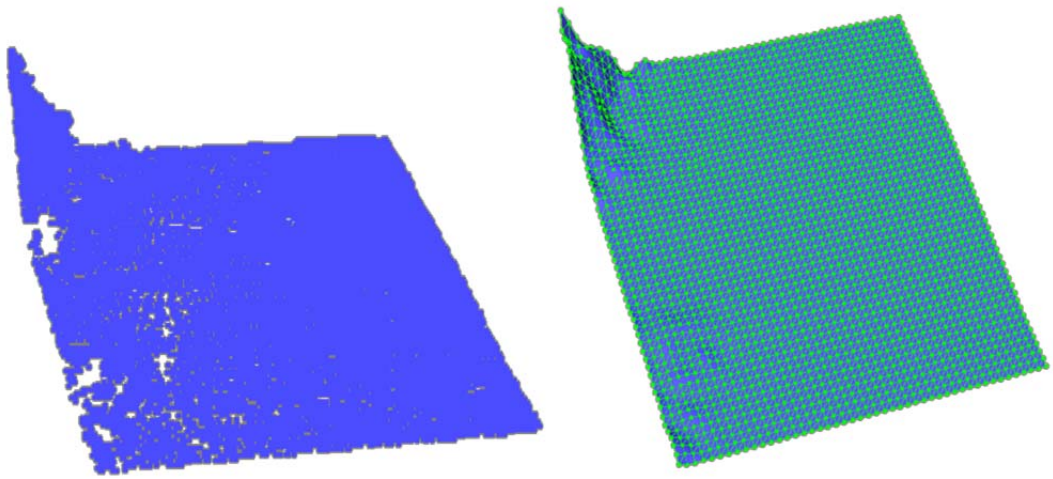


FIGURE 22: Point cloud, approximating surface and triangulation generated from the surface

The first picture in Figure 22 shows a point cloud extracted from the Porsmilin topography data set presented in section 3.1. This point set is loosely approximated by an LR B-spline surface shown in the second picture along with a regular point set evaluated from this surface. This point set is then triangulated.

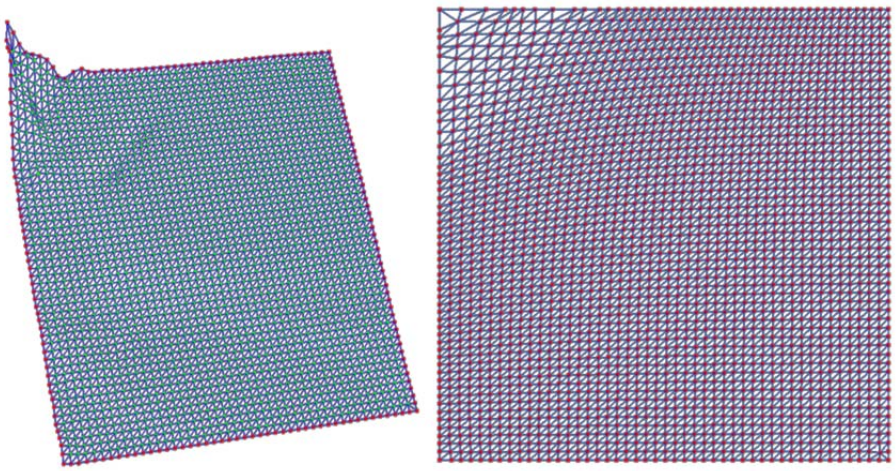


FIGURE 23: Oriented triangulation with marked boundary nodes and triangulated parameter values

Figure 23 shows the oriented triangulation where the boundary nodes have been recognized (red points). Some triangles in the corners may be swapped to satisfy the requirements of the parameterization algorithm. To the right is the corresponding parameterization in the parameter domain of the surface.

4.6.4 Evaluation of the service

The core algorithm is stable if the input requirements are met but it is slow so the number of data points must be low. The service has not yet been put into the real chain where it is expected to work, thus the test cases have so far been very regular. In this context, also the boundary recognition and corner recognition works well but we expect a need to update this part of the algorithm at a later stage. In particular, data sets without a rectangular shape will be challenging. Then the input triangulation may need to be extended with artificial corner points.

The core algorithm needs a number of parameters regarding parameterization methods. These parameters are not visible to the application.

The service can be found under WP4/56-PARAMETERIZE_TRIANGULATED_POINT_SET in the RedMine repository with an executable and one data set. The usage of the service is explained in a README file. The service has been tested and runs under Ubuntu 12.04.

4.7 #84: PARAMETERIZE POINTS BY PROJECTION

This is a utility service in the chain of creating a 3D surface from an xyz point cloud. A highly reduced point cloud, which is representative for the main shape of the initial one, is used to create a parameterized point set by applying service #56. These points are then approximated by a 3D LR B-spline surface using service #9. To update the surface with the initial points, the current service is applied before service #57 is used to create the final surface.

4.7.1 Method and implementation

A closest point algorithm is used to project the point onto the surface. Prior to the closest point computation a seed for the iteration is found by comparing each point with the control polygon of the given surface. Only a few iterations of the closest point algorithm are applied.

4.7.2 Example(s)

This service was used in a chain with service #56 and service #57 to approximate a part of the Porsmilin topography point set from 2010 by a 3D surface. See the example sections of those services for the presentation of the example.

4.7.3 Evaluation of the service

A point cloud and a corresponding surface must be given. No additional user input is required. The algorithm performs a closest point iteration for each point and is linear in time with respect to the number of points.

The service can be found under WP4/84-PARAMETERIZE_BY_PROJECTION in the RedMine repository with an executable and one data set. The usage of the service is explained in a README file. The service has been tested and runs under Ubuntu 12.04.

4.8 #57: UPDATE SPLINE SURFACE WITH POINT CLOUD

This service is very similar to service #9. In fact it starts at an intermediate stage of service #9 where an initial surface exists. This surface must correspond to the given point cloud.

4.8.1 Purpose and context

The context in which this service may be useful is:

- The point set is already approximated with a spline surface but a more accurate surface is wanted.
- Another point set, possibly a reduced one, is approximated by a surface but the surface should be updated due to new or extended information.
- A new point cloud from an area where a surface representation already exists is received. The surface is modified by the new data.
- A 3D surface is preferred in order to better represent steep areas. A parameterized point cloud is obtained from a reduced point set using service #56. Then an initial 3D surface is obtained by approximating the reduced and parameterized point set. The remaining points are parameterized by projecting them onto the initial surface. Finally, the initial surface needs to be updated to include the full point cloud in the approximation. The example illustrates this use of the service.
- A surface representing the trend of the terrain is obtained for a large area. A more detailed surface is required for a subcloud or tile. Then the initial surface can be updated in an area corresponding to the subcloud but remains the same outside of this area to maintain continuity to other tiles.

The service is aimed at the Marine showcase.

4.8.2 Method and implementation

The service uses the same algorithm and implementation as service #9 but the definition of an initial surface and ghost points of level one can be omitted. It is possible to keep the surface corners and even the surface boundaries fixed if the point cloud is sparse in these areas. This will be the case if a larger surface is modified locally.

4.8.3 Example(s)

Figure 24 shows a surface created by service #9 from a set of points parameterized by service #56. The initial point cloud, see Figure 13, is parameterized by projection onto this surface using service #84. The points with a distance larger than a threshold of 0.2 are shown in the second picture. The polygonal patches of the surface are also shown.

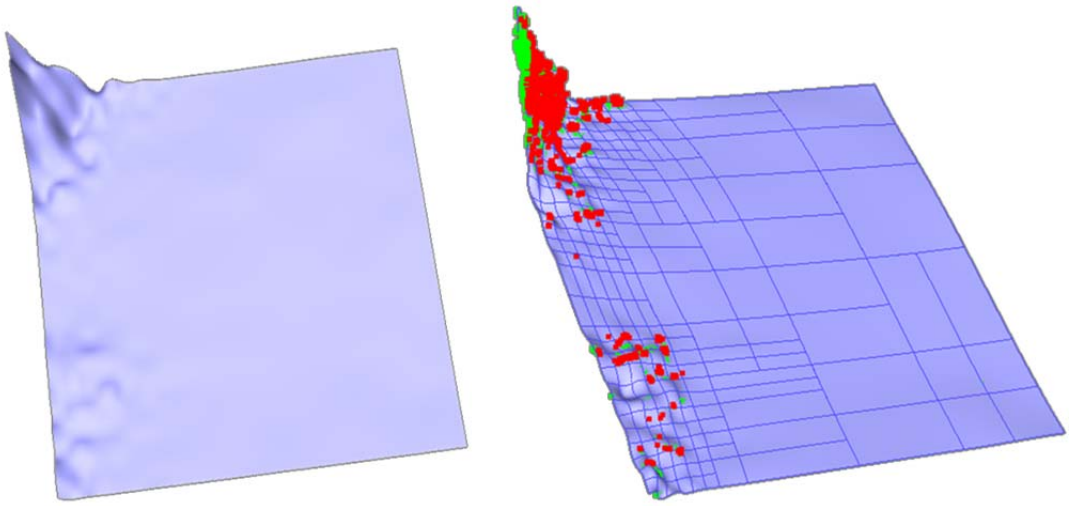


FIGURE 24: Input surface and generated surface shown together with the points where the distance between the points and the generated surface is larger than the threshold

In Figure 25, the final surface is shown together with the points lying outside of the threshold (picture 1) and with all points (picture 2). Two iterations are performed for this part of the process. The maximum distance is 1.5 and the average distance is 0.035. The surface has 790 polygonal patches. 5945 points, i.e. about 1.8% of the points, are outside the threshold. The accuracy is comparable to the function presented in section 4.4.3 but the size of the surface is less than one third.

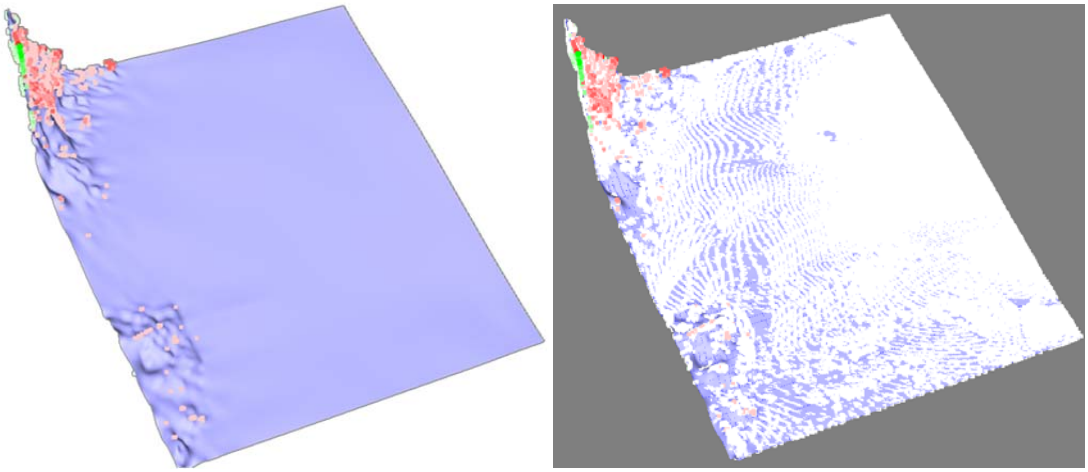
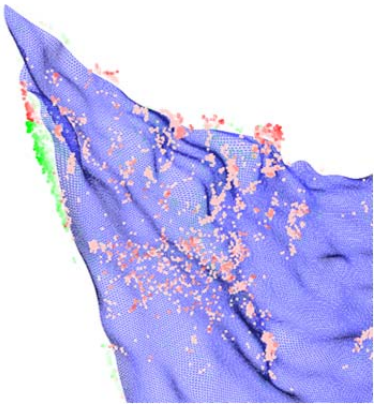


FIGURE 25: The final surface

Almost all the error points are situated in the steep area at the upper right corner of the surface. A closer look at this area in Figure 26 reveals that the surface is well placed between the points. Thus there is not much to gain by increasing the surface size and trying to approximate the points more closely.



FIGUR 26: Detail of final surface

4.8.4 Evaluation of the service

The same comments as for service #9 apply.

The service can be found at WP4/57-UPDATE_SPLINE_SURFACE_WITH_ADDITIONAL_POINTS in the RedMine repository with an executable and two data sets. The usage of the service is explained in a README file. The service has been checked to run under Ubuntu 12.04.

4.9 #40: APPROXIMATION OF RAINFALL DATA WITH ORDINARY KRIGING

This service provides an approximation of rainfall data, measured at a number of rain gauge sites. The problem is to use in a correct way data at different resolutions (very sparse rain data over the area of interest, different resolutions with respect to the DEM where the interpolation is needed). The DEM is used to get the points where the field will be approximated. The DEM should have enough details (resolution) to highlight main topographic features.

4.9.1 Purpose and context

The aim of this service is to provide a tool to interpolate sparse data in an unsampled location. Ordinary Kriging is a state-of-the-art technique to interpolate sparse values taking into account spatial continuity as a hypothesis on the phenomena to approximate. This estimator provides not only the best estimation for every point but also the error variance associated with the estimation.

The target of this service is the Land scenario. The input data are the rainfall data gathered from a network of rain gauges and/or sampled from radar measures maps, and the set of points where the field will be estimated. The output will be the estimated rainfall field at the desired resolution.

4.9.2 Method and implementation

Ordinary Kriging (OK) is a point estimator algorithm in the "best linear unbiased estimator" family. The estimate is a linear combination of the available data; it tries to be "unbiased" by having the residual mean equal to 0. Finally it is "best" because it tries to minimize the residual error.

The service steps can be summarized as follows:

- Read as input both the measured rainfall values and the list of points where the surface should be estimated.
- If the service runs on a multiprocessor computer, the points to be estimated are split into chunks determined by the number of avalaible processors.
- The OK algorithm is launched over all processors on each chunk.
- The estimated values are assembled together.

4.9.2.1 Ordinary Kriging

Ordinary Kriging is a point estimation method that aims to minimize error variance.

The point estimate is expressed as:

$$p_0 = \sum_{j=1}^{n} \omega_j \times p_j$$

where p_0 is the point at which the surface has to be estimated, p_j are the known samples and ω_j are the weights for all the known points.

The weights are computed as follows:

$$\omega = C^{-1} \times D$$

where C and D are the covariance matrices

$$C = \begin{pmatrix} C_{1,1} & C_{n,1} & 1 \\ C_{1,n} & C_{n,n} & 1 \end{pmatrix} \quad D = \begin{pmatrix} C_{1,0} \\ C_{n,0} \end{pmatrix}.$$

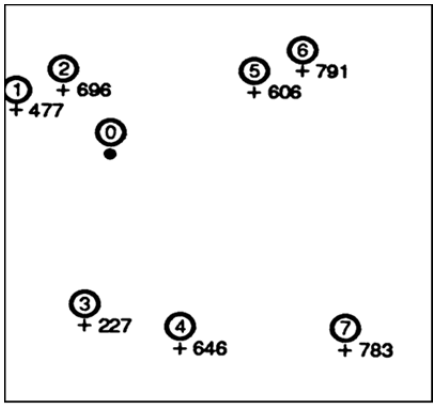


FIGURE 27: Example map to explain the OK algorithm. Point "0" is the point to be estimated and points from "1" to "7" are the known values

The covariance in C is calculated between all the known points, and matrix D is filled with the covariance between the point to be calculated and all the known data. The covariance is strictly correlated with the model variogram.

As show in Figure 27, the point tagged with "0" is the point to be estimated and points from 1 up to 7 are the known points. In this example, n will take values from 1 to 7 in both the C and D matrix and point 0 is to be estimated.

For this version, the model variogram is constant and it is defined at algorithm level; in a future release, the service will be able to select automatically the best model to use.

Our implementation can use all the measured rainfall values to compute the C and D matrix or it can be limited to the closest known values. This solution limits the computational resources for the calculation of the inverse of the C matrix but it requires building the list of nearest points.

The weights previously computed ensure the algorithm is "best linear unbiased".

4.9.2.2 Spatial continuity

The C and D matrices are calculated as functions of spatial continuity. The spatial continuity is measured with the spatial variance expressed as follows:

$$\gamma(h) = \frac{1}{\left(2 \times N(h)\right)} \sum_{\left((i,j)|h_{(i,j)} \approx h\right)} \left(p_i - p_j\right)^2$$

where *p* is the value taken by the variable in position *i* and *j* with i and j points that are separated by lag *h*. The γ values can be plotted against *h* to get the variogram plot that shows the spatial dissimilarity.

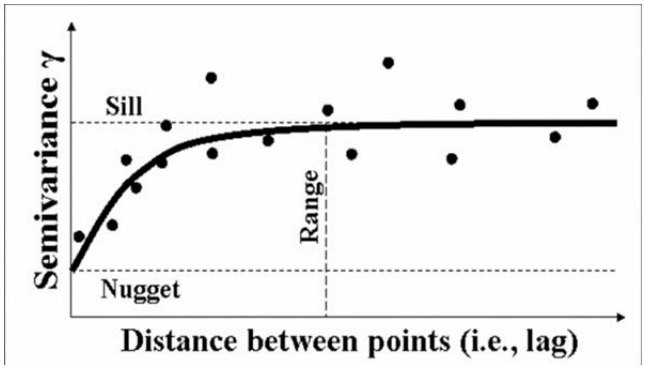


FIGURE 28: The dots represent the experimental variogram and the line the model variogram used for fitting. Sill, range and nugget are morphologic parameters used to set up the model.

The experimental variogram is fitted using a linear combination of basic variogram models. These basic models respect some constraints due to the requirements of the Ordinary Kriging process (positive definiteness of the covariance matrix).

The following expressions are the three most used model variograms. In addition to these models, we will consider vertical shifts, known as nugget effect, representing the local variance of the data.

$$Spherical: \gamma(h) = \begin{cases} c\left(1.5\frac{h}{a} - 0.5\frac{h^3}{a^3}\right), h \le a \\ c,\ h > a \end{cases}$$

$$Gaussian: \gamma(h) = \begin{cases} c(1 - e^{\frac{h^2}{a^2}}), if\ h \le a \\ c,\ h > a \end{cases}$$

$$Exponential: \gamma(h) = \begin{cases} c(1 - e^{\frac{h}{a}}), if\ h \le a \\ c,\ h > a \end{cases}$$

4.9.3 Example(s)

The test data set for this service is the same as for the service #67. We use rainfall data gathered from the Regione Liguria network (red stars) and from the Genova municipality network (purple circles) for a total of about 178 measure stations. The rainfall is cumulated every 30 minutes.

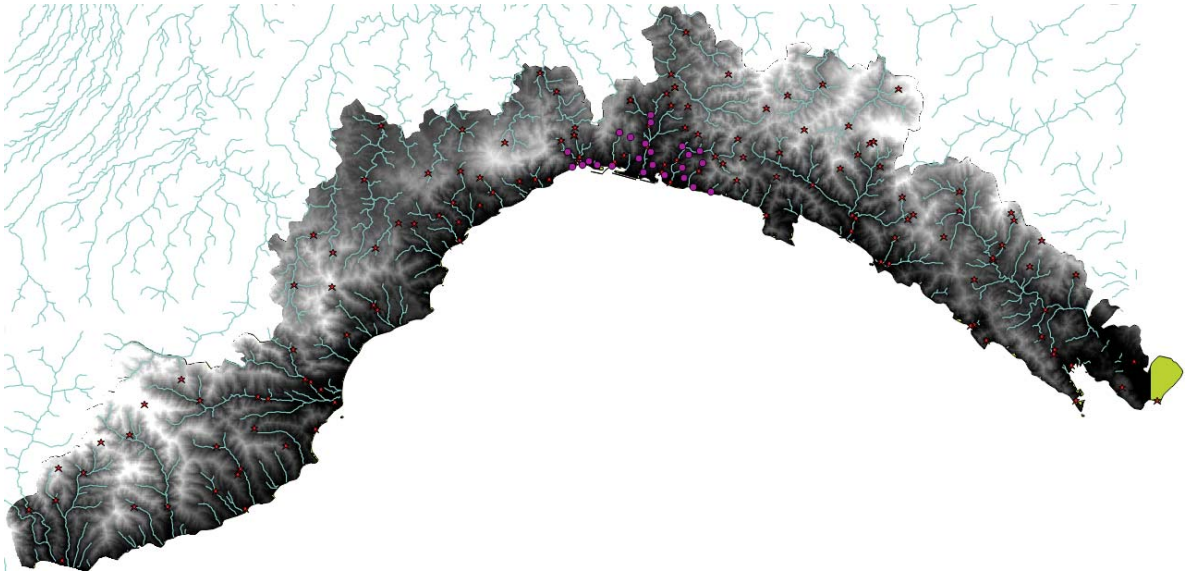


FIGURE 29: The rain gauges distribution: Regione Ligura networks (red stars); Genova municipality (purple circles)

The interpolation has been performed over the DEM of Regione Liguria from the SRTM dataset downscaled at 1 point every 100 meters.

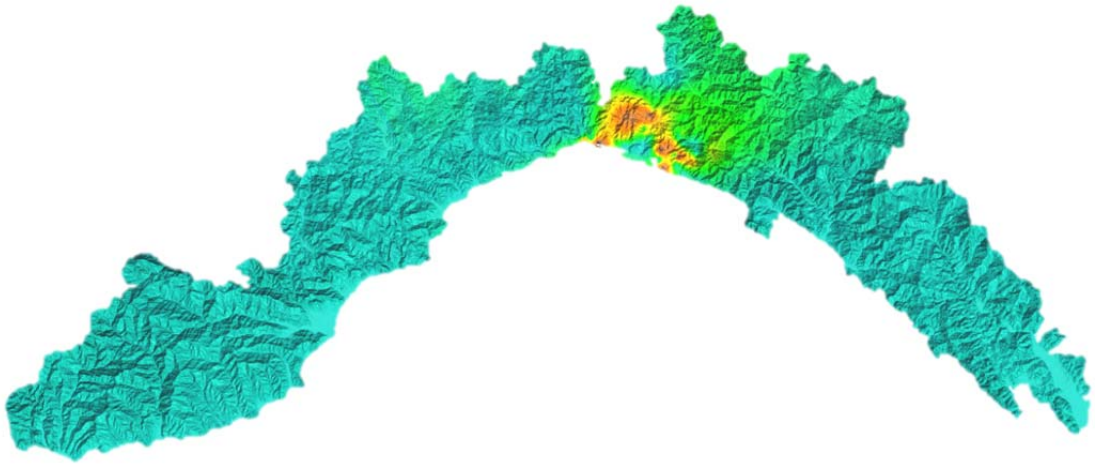


FIGURE 30: The OK approximation of rainfall over the whole region

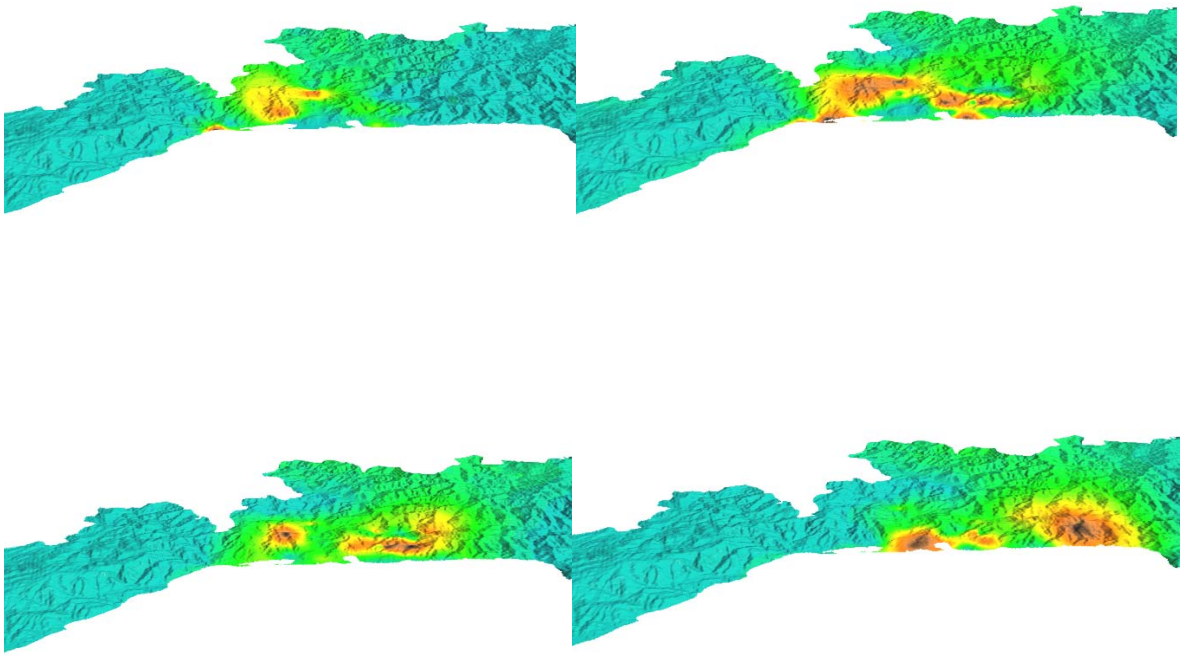


FIGURE 31: The rain field at steps of 30 minutes for pointing out the evolution of a thunderstorm over the city of Genova

4.9.4 Evaluation of the service

As for service #67, the proposed algorithms have been tested with different techniques in order to discuss the following items:

- *Evaluation of the accuracy of each approximation scheme* through the comparison between approximated and true values at every sample point.
- *Cross-validation of the method.* Every 30 minutes, each rain gauge has been turned off and the approximation function has been sampled at this position. Then, the obtained value has been compared with the true value.
- *Evaluation of the accuracy of the algorithms through the comparison with the ground truth*. For the validation of the approximation techniques, the rainfall data measured by the Genova municipality has been used as ground truth to validate the values interpolated from the regional data set. The two observation networks cover an overlapping region of the study area. The network of the Genova municipality is located within the boundary of the city and is denser than the one of Regione Liguria that covers the whole study area. Some stations of the regional network are located in the Genova area and allow us to interpolate the rainfall field over the city within a certain accuracy, which has been further estimated comparing the approximation results at these two scales.

Running this algorithm on an eight core computer takes 1.36 seconds to approximate 21593 points starting from 166 rainfall measures and it takes 6.58 seconds to estimate 113757 points from the same starting data sets.

The service can be found under WP4/40-APPROXIMATION_OF_RAINFALL_DATA in the RedMine repository with an executable and one data set. The executable is embedded in a Python script. The usage of the service is explained in a README file. The service is easily run after following the instructions in the README file.

4.10 #67: APPROXIMATION OF RAINFALL DATA WITH RADIAL BASIS FUNCTIONS

4.10.1 Purpose and context

The purpose of this service is to apply approximation techniques based on radial basis functions (RBFs), which are commonly used for the approximation of both sparse and dense data, to rainfall data. We also aim at comparing its accuracy and performance with general (e.g., Splines – Service #58: Spline representation of rainfall data, to be provided in the next project period) and specialized approximation methods (e.g., Kriging – Service #40: Approximation of rainfall data with ordinary kriging).

4.10.2 Method and implementation

For the approximation of rainfall data, which are represented as a scalar function $f: M \rightarrow R$ sampled at a discrete set of points $M:=\{p_i\}_{i=1}^n$, this service uses a global approximation scheme with radial basis functions and globally-supported kernels [2,8]. According to [1, 7] and choosing a kernel $\varphi: R^+ \rightarrow R$, the approximation $F: R^3 \rightarrow R$ of $f: M \rightarrow R$ is defined as $F(p) := \sum_{i=1}^n \alpha_i \varphi_i(p)$, $p:=(x,y,z)$; i.e., a linear combination of the radial basis functions $\varphi_i(p):=\varphi(||p-p_i||_2)$, centred at $\{p_i\}^n_{i=1}$. Then, the coefficients $\alpha=(\alpha_i)_{i=1}^n$ that uniquely satisfy the interpolation conditions $F(p_i)=f(p_i)$, $i=1,...,n$, are the solution of the linear system $A\alpha=f$, with $a_{ij}:=\varphi(||p_i-p_j||_2)$ and $p_i:=(p_{xi},p_{yi},p_{zi})$. We briefly recall that the support of an arbitrary map $g: R^3 \rightarrow R$ is defined as the set $supp(g):=\{p \text{ in } R^3 : F(p) \neq 0\}$. If $supp(g):=R^3$, then g has global support. Common choices of kernels with global support are the Gaussian $\varphi(t):=\exp(-t)$, the harmonic kernel $\varphi(t):=|t|^{-1}$, and the bi-harmonic $\varphi(t):=|t|^3$ kernel. For more details, we refer the reader to D4.1.1.

4.10.3 Examples

For the evaluation of this processing service, the data set is gathered from two different rain gauge networks with a different spatial distribution. The first one is the network developed by Regione Liguria over the whole region, with 143 measure stations (Figure 32). The second one is the measure system deployed by the Genova municipality within the city boundary, with 35 measure stations (Figure 32). Since the temporal interval is different for each network, station rainfalls are cumulated to a step of 30 minutes. The selected rainy day is September 29, 2013, which was characterized by light rain over Regione Liguria with two different thunderstorms that did not cause severe damages (only local flooding and landslides).

The interpolation process has been performed over two DEMs of Regione Liguria at different resolutions (Figure 33): (i) the SRTM model (downscaled at 1 point every 500m) and (ii) the DTM of Regione Liguria (with a resolution of a point every 5m). The interpolation accuracy has been tested over the 143&35 stations and the extrapolation test (approximation accuracy with respect to the ground truth and speed) has been performed over the two DEMs of Regione Liguria (Figure 34).

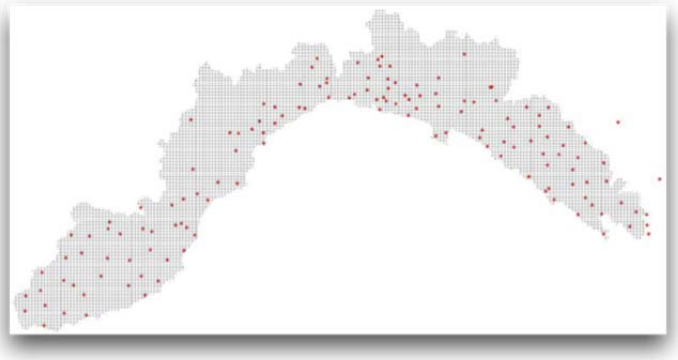


FIGURE 32: Input: rainfall measures f(p_i) at 143 stations (regional level, red points) and 35 stations (municipality level)

FIGURE 33: Triangle mesh of Regione Liguria and zoom-in

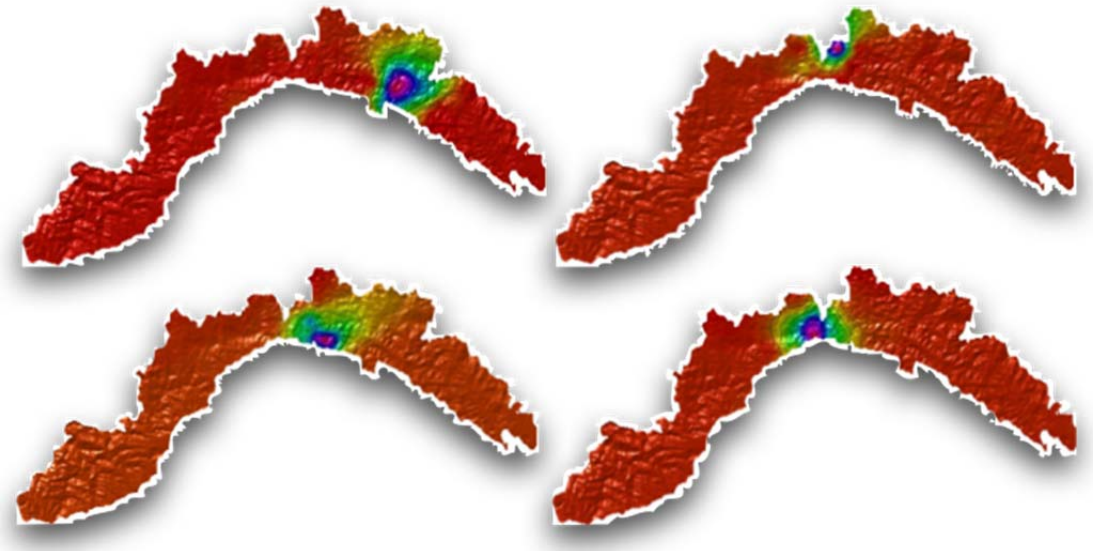


FIGURE 34: Colour map of the approximation of the rainfall data at 4 different times on the model of Regione Liguria

4.10.4 Evaluation of the service

The proposed algorithms have been tested with different techniques in order to discuss the following items:

Evaluation of the accuracy of each approximation scheme through the comparison between the approximated and true values at every sample point. Figure 35 (left) shows the maximum approximation error (y-axis) at the rainfall station at every hour (x-axis) during the selected rainy day. For most of the approximated rainfall data, the approximation error remains lower than 1% and the maximum variation is lower than 2%.

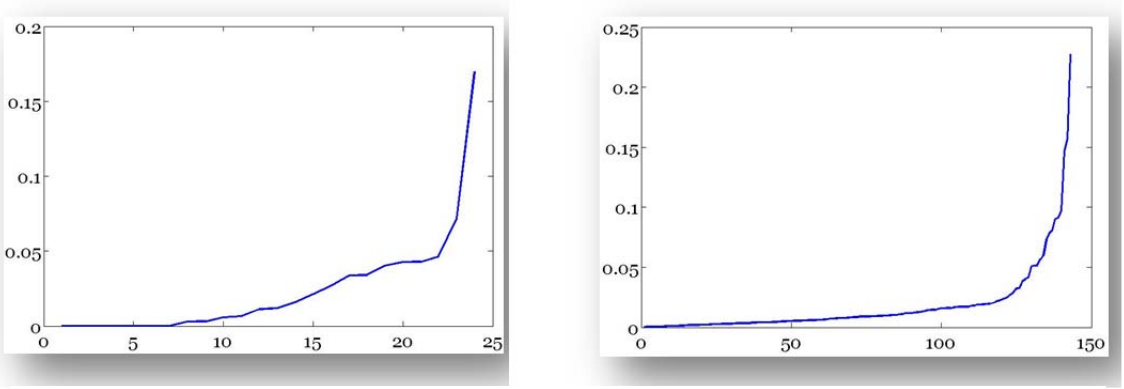


FIGURE 35: (left) Maximum approximation error (y-axis) at each rainfall station at every hour (x-axis) during the selected rainy day (right) Maximum approximation error (y-axis) at each rainfall station when it is turned off.

Evaluation of the accuracy of the algorithms through the comparison with the ground truth. Every 30 minutes, each rain gauge has been turned off and the approximation function has been sampled at this position. Then the obtained value has been compared with the true value. For the validation of the approximation techniques, the rainfall data measured by the Genova municipality has been used as ground truth to validate the values interpolated from the regional data set. The two observation networks cover an overlapping region of the study area. The network of the Genova municipality is located within the boundary of the city and is denser than the one of Regione Liguria that covers the whole study area. Some stations of the regional networks are located in the Genova area and allow us to interpolate the rainfall field over the city within a certain accuracy, which has been further estimated comparing the approximation results at these two scales.

Figure 35 (left) shows the approximation error (y-axis) at the removed rainfall station (x-axis) during the selected rainy day. For most of the approximated rainfall data, the approximation error remains lower than 1% and the maximum variation is lower than 2.5%. Indeed, the accuracy of the approximation scheme and the results of the cross-validation of the method are coherent and show the same order of accuracy. Figure 35 (right) reports the maximum approximation error (y-axis) at each rainfall station when it is turned off.

Computational cost. For the RBF approximation, we solve a linear system whose coefficient matrix is symmetric and positive-definite. Its unique solution is computed through a direct solver or pre-factorization of the coefficient matrix. In the current implementation, we apply a direct solver. The computational cost for the evaluation of F is $O(n^3)$, where n is the number of rainfall stations. The resampling of F on s samples is linear in the number s of samples. Since s>>n, the overall computational cost is linear in the number of samples (Figure 36). For the computation of the global approximation with a number of rainfall stations lower than 5K (as in real situations), the most time-consuming part is the evaluation of the implicit function at the sample points. To this end, a parallel implementation, which (i) subdivides the input samples into patches that are processed independently or (ii) assigns a set of RBFs to each processor, is

generally enough to efficiently handle large data sets in terms of memory and computational cost.

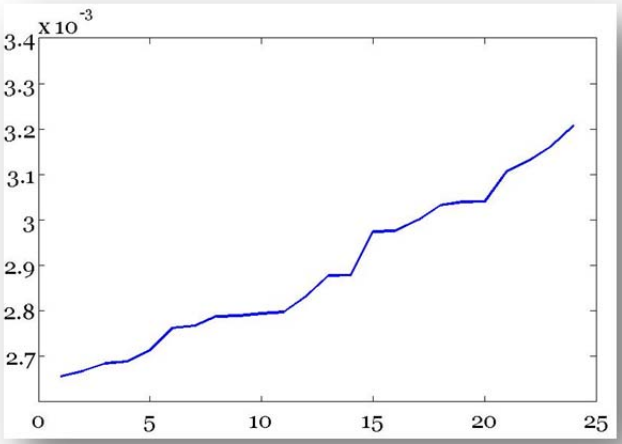


FIGURE 36: Computational cost (y-axis, in seconds) of the evaluation the RBF approximation at each hour (x-axis)

The service can be found in the RedMine repository under WP4/67-RBF_RAINFALL. It contains a MatLab executable and one data set given in two different formats. The executable can also be run using Octave. Instructions on how to run the service is given in pdf-format. The service runs nicely under MatLab but is more unstable using Octave. The data sets are given in .ply and .txt formats. The .txt file runs also under Octave but the input .ply file does not always work.

5 OUTLOOK/EVOLUTION OF THE TOOLKIT

The surface development will continue both for triangulations and for smooth approximations. For both classes of services parallelization with Hadoop and MapReduce will be applied wherever appropriate.

The service suite on triangulations will be extended with a multi-resolution triangulation. Continuity between different adjacent triangulated surfaces will be taken care of by functionality for tiling of huge point clouds and stitching of the corresponding triangulated surfaces. Furthermore, functionality is planned to improve the resolution of a digital elevation model with higher resolution data. The output will either be triangle based or grid based.

The basic functionality for approximations of point clouds with spline surfaces is provided by the services described in this deliverable. In the next project period, the focus will be to improve the current services and to make use of them in the work flow of the Marine scenario. This will involve some extensions to the current toolkit, particularly regarding the extraction of information from LR B-spline surfaces. The following extensions are foreseen:

- Create an initial large surface from a thinned point set combined from many tiles and update this surface tile by tile
- Create surfaces from overlapping tiles. Reduce and adapt the surface to get consistent behaviour at the boundaries for adjacent surfaces.
- Use Hadoop and MapReduce to handle several tiles in parallel.

- Compare point clouds to approximating surfaces and extract points lying outside a specified threshold from the surfaces. This will provide manageable point clouds for visualization.
- Provide input to feature extraction and/or services for outlier removal by identifying clusters of points lying more than a given threshold away from a surface representing the trend in the data.
- Integrate feature information by giving feature points identified by the services of Task 4.3 increased weight in the approximation procedure.
- Compute contour lines from the surfaces.
- Combine the current method for surface approximation with a local method lifting a current surface towards the point cloud in specified areas. The method is based on the MBA algorithm, see [6], and will typically be applied after a number of iteration steps in the least squares method.

In relation to the first phase of the Land scenario, representation of rainfall data is a prioritized task. To supplement the services already pursuing this task, a method for approximation of rainfall data with LR B-splines will be implemented.

Another approach for model generation will be pursued in the next year. This work is focused on automated fusion of multi-view volumetric point clouds (aerial and terrestrial, one or the other being generated from LiDAR or optical 3D surface reconstruction) and generation on the one hand of volumetric space occupancy and permeability grids and on the other hand of a dual watertight surface model. We will also discuss and possibly work on the interoperability of these new space occupancy models with simulation algorithms together with other partners of the IQmulus consortium.

The Urban showcase has so far not been addressed in the context of Task 4.4. We will study the scenario in detail and identify required T4.4 services.

6 CONCLUSION

The focus in the first year has been on services related to the Land scenario 1 and the Marine scenario. The Land show case services are related to triangulations and rainfall data while the services for the Marine show case are centred around the novel technology of LR B-splines. Even though the services have been selected in view of these two scenarios, they are applicable also in other contexts.

Until now Hadoop and MapReduce have not been prioritized in the development. These concepts will be included in the next period, but it is clear that the services differ in how easily they will adapt to this technology.

7 REFERENCES

[1] N. Aronszajn, *Theory of reproducing kernels*. Trans. of the American Mathematical Society 68 (1950), 337–404.

[2] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum and T. R. Evans, *Reconstruction and representation of 3D objects with radial basis functions*. In: ACM Siggraph (2001), 67–76.

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

[4] M.S. Floater, *Parameterization and smooth approximation of surface triangulations*. Computer Aided Geometric Design 14 (1997), 231-250.

[5] M. S. Floater, *Mean value coordinates*. Computer Aided Geometric Design 20 (2003), 19-27.

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

[7] T. Poggio and F. Girosi, *Networks for approximation and learning*. Proceedings of the IEEE, 78 (1990), 1481–1497.

[8] G. Turk and J. F. O'Brien, *Modelling with implicit surfaces that interpolate*. ACM Siggraph 21(2002), 855–873.

[9] J. R. 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, 203-222, Springer Verlag, Berlin, 1996

[10] J. R. Shewchuk, *Delaunay Refinement Algorithms for Triangular Mesh Generation*, Computational Geometry: Theory and Applications 22(2002), 21-74.

[11] G. Westgaard, H. Nowacki, *A process for surface fairing in irregular meshes*. Computer Aided Geometric Design 18 (2001), 619-638.

8 APPENDIX A – LR B-SPLINES

In this appendix some background material is provided for the services using LR B-spline surfaces.

A tensor product spline surface is a piecewise polynomial surface defined on a regular grid. The surface can also be extended with a rational term but this is mainly useful in the context where one wants to combine freeform and conic surfaces. Tensor product spline surfaces are well suited for a compact representation of smooth shapes but less adequate in a context with large variations in shape as we typically find it for geographic models.

Locally refined spline surfaces have, in contrast to tensor produce spline surfaces, the ability to represent local variations in shape without globally increasing the data size of the surface. A locally refined spline surface is typically created on the basis of a tensor product spline surface. New knot lines or coefficients are added adaptively until some conditions are satisfied. A number of approaches have been pursued:

- PHT splines [A1]
- General T-splines [A2]
- Analysis suitable T-splines [A3]
- Standard and semi-standard T-splines [A4 and references therein]
- Hierarchical B-splines [A5]
- LR B-splines [A6]

Spline surface generation in IQmulus uses LR B-splines. Starting from a tensor product surface defined on a regular grid in the parameter domain, local knot lines are inserted into this grid. The format also allows for local degree elevation. A new knot line is required to split the domain of at least one existing B-spline. New coefficients are computed based on the new grid.

A polynomial LR B-spline surface is given by:

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

where

K= number of control points,

P_i = control points, i=1, ..., K,

s_i= scaling factors,

$N_i^{d_1,d_2}(u,v)$ = B-spline defined in both parameter directions,

d_1= degree of the B-spline in the first parameter direction,

d_2= degree of the B-spline in the second parameter direction.

A B-spline function corresponding to an LR B-spline surface is composed of two univariate B-spline functions. The combined B-spline is minimal meaning that no B-spline function with a smaller domain can be defined by the knots in the domain of this B-spline.

The size of the domains over which the B-splines are defined varies and the number of B-splines defined over each element, i.e. the part of the parameter domain limited by consecutive knots in

both parameter directions, varies as well. One knot insertion may give rise to the split of several combined B-splines.

LR B-splines have the following properties:

- An LR B-spline is parameterized on a rectangular domain in R^2. The domain is a composition of rectangular boxes, but need not to be regular.
- If the domain is regular and all B-splines have the same degree in the first parameter direction and also the same degree in the second parameter direction, the corresponding surface will be a tensor product spline surface.
- New B-splines are created by refining existing B-splines.
- Nested spline spaces are guaranteed.
- Partition of unity is ensured by scaling the B-spline functions.
- An LR B-spline surface is contained in the convex hull of its coefficients.
- Linear independence of the B-splines is not guaranteed by default. It is ensured by:
    - restricted freedom in how to choose new knot lines;
    - counting the number of B-splines that possibly can be included in a linear dependency relationship in an element or over a knot line. If no such B-splines exist, the B-splines are linearly independent;
    - the dimension of the spline space being equal to the number of basis functions.
- If the B-splines corresponding to an LR B-spline surface are linearly independent the B-splines will define a spline space.

EXAMPLE:

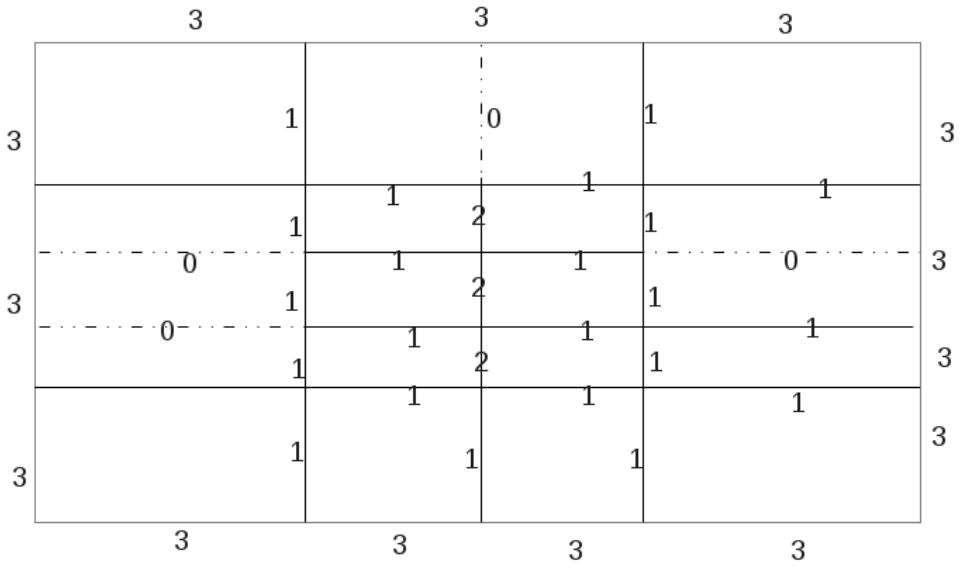


FIGURE 37: The parameter domain of a locally refined spline surface.

Figure 37 shows the parameter domain of a locally refined spline surface. The knot lines have varying multiplicity throughout the domain. The numbers show the maximum multiplicity of each knot. Zero multiplicity indicates that this knot does not exist in that part of the domain. This grid corresponds to a surface that is quadratic in both parameter directions for all B-splines.

The locally refined spline surface contains two sets of knot values, one for each parameter direction. Let the knot values be (u_1, u_2, u_3, u_4, u_5) in the first parameter direction and (v_1, v_2, v_3, v_4, v_5, v_6) in the second parameter direction.

The B-splines relate to the index of the knot values in the respective parameter direction, not the knots themselves and the knots may be repeated. The B-splines defined over this grid are starting from the lower left:

1. Degree1=2, degree2=2, knots1=(0,0,0,1), knots2=(0,0,0,1)
2. Degree1=2, degree2=2, knots1=(0,0,1,2), knots2=(0,0,0,1)
3. Degree1=2, degree2=2, knots1=(0,1,2,3), knots2=(0,0,0,1)
4. Degree1=2, degree2=2, knots1=(0,0,0,1), knots2=(0,0,1,4)
5. Degree1=2, degree2=2, knots1=(0,0,1,2), knots2=(0,0,1,4)
6. Degree1=2, degree2=2, knots1=(0,1,2,3), knots2=(0,0,1,4)
7. Degree1=2, degree2=2, knots1=(0,0,0,1), knots2=(0,1,4,5)
8. Degree1=2, degree2=2, knots1=(0,0,1,2), knots2=(0,1,4,5)
9. Degree1=2, degree2=2, knots1=(0,1,2,3), knots2=(0,1,4,5)
10. Degree1=2, degree2=2, knots1=(1,2,3,4), knots2=(0,0,0,1)
11. Degree1=2, degree2=2, knots1=(1,2,3,4), knots2=(0,0,1,2)
12. Degree1=2, degree2=2, knots1=(1,2,3,4), knots2=(0,1,2,4)
13. Degree1=2, degree2=2, knots1=(2,3,4,4), knots2=(0,0,0,1)
14. Degree1=2, degree2=2, knots1=(2,3,4,4), knots2=(0,0,1,2)
15. Degree1=2, degree2=2, knots1=(2,3,4,4), knots2=(0,1,2,4)
16. Degree1=2, degree2=2, knots1=(3,4,4,4), knots2=(0,0,0,1)
17. Degree1=2, degree2=2, knots1=(3,4,4,4), knots2=(0,0,1,2)
18. Degree1=2, degree2=2, knots1=(3,4,4,4), knots2=(0,1,2,4)
19. Degree1=2, degree2=2, knots1=(0,0,0,1), knots2=(1,4,5,5)
20. Degree1=2, degree2=2, knots1=(0,0,1,3), knots2=(1,4,5,5)
21. Degree1=2, degree2=2, knots1=(0,1,3,4), knots2=(1,4,5,5)
22. Degree1=2, degree2=2, knots1=(1,2,2,3), knots2=(1,2,3,4)
23. Degree1=2, degree2=2, knots1=(1,3,4,4), knots2=(1,2,4,5)
24. Degree1=2, degree2=2, knots1=(3,4,4,4), knots2=(1,2,4,5)
25. Degree1=2, degree2=2, knots1=(1 ,3,4,4), knots2=(2,4,5,5)
26. Degree1=2, degree2=2, knots1=(3,4,4,4), knots2=(2,4,5,5)
27. Degree1=2, degree2=2, knots1=(0,0,0,1), knots2=(4,5,5,5)
28. Degree1=2, degree2=2, knots1=(0,0,1,3), knots2=(4,5,5,5)
29. Degree1=2, degree2=2, knots1=(0,1,3,4), knots2=(4,5,5,5)
30. Degree1=2, degree2=2, knots1=(1,3,4,4), knots2=(4,5,5,5)
31. Degree1=2, degree2=2, knots1=(3,4,4,4), knots2=(4,5,5,5)

The scaling factors corresponding to the combined B-splines are 1 for all instances. The combined B-splines corresponding to this domain are not tested for linearly independence. The surface defined in this example is not a recommended one. There exist B-splines covering most of the domain. This results in a poor utilization of the local support property of spline surfaces and bad approximation properties. Besides, it is not beneficial regarding numerical properties.

8.1 REFERENCES

[A1] J. Deng, F. Chen, X. Li , C. Hu, W. Tong, Z. Yang and Y. Feng, *Polynomial splines over hierachical T-meshes*. Graphical Models 70 (2000), 76-86

[A2] T.W. Sederberg, J. Zheng, A. Bakenov and A. Nasri, *T-splines and T-NURCCs*. ACM Siggraph 22 (2003), 477-484.

[A3] X. Li and M.A. Scott, *Analysis-suitable T-spline: characterization, refineability, and approximation.* Mathematical Models and Methods in Applied Sciences 24 (2013), 1-24.

[A4] T.W. Sederberg, D. L. Cardon, J. Zheng and T. Lyche, *T-spline Simplification and Local Refinement.* ACM Transactions on Graphics 23 (2004), 276-283.

[A5] D.R. Forsey and R.H. Bartels, *Hierarchical B-Spline Refinement*. ACM Siggraph 7 (1988), 205-212.

[A6] T. Dokken, K.F. Pettersen and T. Lyche, *Polynomial splines over locally refined box-partitions*. Computer Aided Geometric Design 30 (2013), 331-356.
http://dx.doi.org/10.1016/j.cagd.2012.12.005

9 APPENDIX B – SERVICE INFORMATION TABLES

9.1 #49 CONSTRAINED TRIANGULATION

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Constrained triangulations		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Create a triangulation of a point cloud preserving user-defined linear constraints (e.g. boundaries, feature lines) We assume that the point cloud represents the sampling of a 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>**	• point cloud (LAS) • lines and/or polygons (ShapeFile)	*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>	• Triangle Mesh (PLY) • Lines and/or polygons (ShapeFile)	
	Functionality of the service: <text>	Triangulate the point cloud preserving specific features	
Algorithm	Use a Delaunay triangulation algorithm, accordingto the constraints imposedimposed		*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	Rendering of triangle mesh (and optional 1D curves)	
IQmulus Data	**Available IQmulus input data**: Dataset #22 and #23 (Lidar from Regione 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:** linear precision		*These fields are necessary to document all the characteristics of the services*
	Robustness: machine precision		
	Computational time in relation to		

	data size: == **Locality/globality of the algorithm:** local geometric criterion to generate triangles	*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.*
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	• Use case (IQmulus) #1042 (partially related) • User story (IQmulus) #1065	*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, Davide Sobrero [davide.sobrero@ge.imati.cnr.it]	*Partner ID and responsible person (include email)*
Involved Partners	====	

9.2 #51 TRIANGULATION OF GRIDDED POINT CLOUD

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Triangulation of gridded point cloud		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Apply a simple triangulation algorithm to triangulate a gridded 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>**	gridded point cloud, format: GeoTIFF	*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>	Triangle Mesh (PLY)	
	Functionality of the service: <text>	Triangulate a gridded point cloud	
Algorithm	Apply a standard algorithm to triangulate a gridded point cloud		*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*
	Dependencies with other	==	

	libraries		*library); constraints on the operating systems, and visualization modalities of the output.*
	Operating system	Linux	
	Visualization modalities of the output	Rendering of Triangle Mesh (PLY)	
IQmulus Data	**Available IQmulus input data**: <ID of the dataset as in the eRoom tables>		*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:** linear 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: machine precision		
	Computational time in relation to data size: ==		
	Locality/globality of the algorithm:		
	local geometric criterion to generate triangles. The regular structure of the input allows a straightforward generalization to parallel/distributed computing		
Alternatives	Service #3030		*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) #1050 • Use case (IQmulus) #10511051		*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, Davide Sobrero [davide.sobrero@ge.imati.cnr.it]		*Partner ID and responsible person (include email)*
Involved Partners	====		

9.3 #66 POINT CLOUD DIMENSIONALITY

IQmulus Service information		
Name of the metadata	**Content expected**	**Motivation/comments**
Service Acronym	PCD	*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Determination of the dimensionnality of each 3D point of the point cloud. The dimensionnality determines if the distribution of the set of points in the neighbourhood of a given point lies in a volume, on a surface or on an edge.	*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 LidarFormat 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]*	min and max number of neighbors, window size for streamed processing (all optional)	
	Output: <data representation and format>	Point cloud in LidarFormat format with attributes added	
	Functionality of the service: <text>	Computing dimentionality descriptors per point	
Algorithm			*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	LidarFormat, LibANN	
	Operating system	Ubuntu	
	Visualization modalities of the output	Any point cloud viewer with ability to visualize a scalar per point.	
IQmulus Data	**Available IQmulus input data**: not yet		*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 relevant		*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*
	Robustness: No issue		
	Computational time in relation to data size: a minute per million points		
	Locality/globality of the algorithm: local (result for a point depends on a limited neighborhood)		

		among more services that implement the same functionality but with different characteristics. See the following box.
Alternatives	None	*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	All, as this is a preprocessing service	*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	IGN, Bruno Vallet, bruno.vallet@ign.fr	*Partner ID and responsible person (include email)*
Involved Partners	IGN	

9.4 #9 SPLINE SURFACE FROM PARAMETERIZED POINT CLOUD

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	ParamPoints2Spline Generate spline surfaces from parameterized data		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Compute an LR B-spline surface approximating point cloud with parameter information. The parameterization can be given by the xy-data of an elevation data set or by additional parameter values		*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>**	ASCII, each entry is given as xyz or uvxyz where (u,v) is the parameter pair corresponding to the point. The number for each entry is separated by a space (as in xyz files) or by a comma (as in files converted from LAS by libLAS las2txt)	*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	

		• Maximum number of iterations • Number of values for each entry (xyz or uvxyz)	
	Output: <data representation and format>	• LR B-spline surface in g2-format • Information on surface size and accuracy	
	Functionality of the service: <text>	Surface approximation	
Algorithm	If the points are parameterized on the xy-plane, a function is created, otherwise a 3D surface. The point set is translated to origo. The surface or function is generated by least squares approximation with a smoothing term. Finally, the surface is translated back to its original 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	GoTools (SINTEF library)	
	Operating system	Linux Ubuntu	
	Visualization modalities of the output	Visualization by WP5	
IQmulus Data	**Available IQmulus input data**: 10, 11, 22, 23		*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:** Relates to the accuracy of the input points		*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: Depends on the data quality and smoothness		
	Computational time in relation to data size: Linear in data size, but there is multiplication factor depending on the complexity of the data. Some processing time must be expected.		
	Locality/globality of the algorithm: Global, but some operations like checking the accuracy of the points are local		
	Data partitioning will be necessary to address the scalability of the service.		
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 show case 1.2.2_SC1_1_UC2, 1.2.2_SC2_1_UC1 Can be a part of an alternative process: 1.1_22_UC6, 1.1_22_UC5		*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		

9.5 #55 SPLINE SURFACE FROM GRIDDED POINT CLOUD

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Grid2Spline Generate spline surfaces from gridded data		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Compute one B-spline or LR B-spline surface approximating a gridded data set		*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>**	Gridded point set (ESRI ASCII)	*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 • Maximum number of iterations	
	Output: <data representation and format>	• LR B-spline surface in g2-format • Information on surface size and accuracy	
	Functionality of the service: <text>	**Surface approximation**	
Algorithm	The points are parameterized using the grid information. The point set is translated to origo. The surface is generated by least squares approximation with a smoothing term. Finally, the surface translated back to its original 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	GoTools (SINTEF library)	
	Operating system	Linux, Windows	
	Visualization modalities of the output	Visualization by WP5	
IQmulus Data	**Available IQmulus input data**: 12, 13		*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:** Relates to the accuracy of the input points		*These fields are necessary to document all the*

	Robustness: Depends on the data quality and smoothness	*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.*
	Computational time in relation to data size: Linear in data size, but there is multiplication factor depending on the complexity of the data. Some processing time must be expected.	
	Locality/globality of the algorithm: Global, but some operations like checking the accuracy of the points are local	
	Data partitioning will be necessary to address the scalability of the service.	
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 show case 1.22_SC2_UC1 Can be a part of an alternative process: 1.1_22_UC4	*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		

9.6 #56 PARAMETERIZE TRIANGULATED POINT SET

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Triang2ParamPoints Parameterize triangulated point cloud		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Compute a parameterization on a sparse organized point set		*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>**	Triangulation (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]*	Specification of the triangulation boundary would be useful	
	Output: <data representation and format>	Enriched ASCII point cloud (u,v,x,y,z)	
	Functionality of the service: <text>	Mean value parameterization	
Algorithm	Identify (if necessary) the triangulation boundary and domain corners. The corners may extend the point set. Parameterize by M. Floaters mean value algorithm		*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 (SINTEF library) rply	
	Operating system	Linux, Windows	
	Visualization modalities of the output		
IQmulus Data	**Available IQmulus input data**: Generic		*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:**		*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: Requires a sparse triangulation		
	Computational time in relation to data size: The computational time grows fast with respect to data size		
	Locality/globality of the algorithm: Global		
	Data thinning will be necessary to address the scalability of the service.		
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			*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			

9.7 #84 PARAMETERIZE BY PROJECTION

IQmulus Service information		
Name of the metadata	**Content expected**	**Motivation/comments**
Service Acronym	ParameterizeOnSurf	*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Parameterize point cloud by projecting it onto a 3D 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 (g2), point cloud (currently xyz-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>	Enriched point set (u,v,x,y,z)	
	Functionality of the service: <text>	3D point cloud parameterization	
Algorithm			*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	
	Operating system	Ubunto	
	Visualization modalities of the output	None	
IQmulus Data	**Available IQmulus input data**: Any point cloud that is associated a 3D spline surface. This means that services #56 and #9 must be performed first. Data sets: 10, 11, 22, 23		*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:** Good		*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: Godd		
	Computational time in relation to data size: Linear, the constant includes a search for a starting point to a closest point computation and this computation		
	Locality/globality of the algorithm: Local given a global surface		
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	SINTEF Vibeke Skytt, Vibeke.Skytt@sintef.no	*Partner ID and responsible person (include email)*
Involved Partners		

9.8 #57 UPDATE SPLINE SURFACE WITH POINT CLOUD

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	UpdateSplinePoint Update spline surface with point cloud		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Compute an updated LR B-spline surface approximating a point cloud.		*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) Point cloud (ASCII: xyz or uvxyz depending on the dimension of the surface. The points can be separated by space or comma)	*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 • Maximum number of iterations	
	Output: <data representation and format>	• LR B-spline surface in g2-format • Information on surface size and accuracy	
	Functionality of the service: <text>	Closest point computations, surface approximation	
Algorithm	The parameter domain of the points and the surface must correspond. The updated surface is generated by least squares approximation with a smoothing term.		*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.,*

	Dependencies with other libraries	GoTools (SINTEF library)	*C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Operating system	Linux, Windows	
	Visualization modalities of the output	Visualization by WP5	
IQmulus Data	**Available IQmulus input data**: 10, 11, 22, 23		*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:** Relates to the accuracy of the input points		*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: Depends on the data quality and smoothness		
	Computational time in relation to data size: Linear in data size, but there is multiplication factor depending on the complexity of the data. Some processing time must be expected.		
	Locality/globality of the algorithm: Global, but some operations like checking the accuracy of the points are local		
	Data partitioning will be necessary to address the scalability of the service.		
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 show case		*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			

9.9 #40 APPROXIMATION OF RAIN FALL WITH ORDINARY KRIGING

IQmulus Service information		
Name of the metadata		**Motivation/comments**
Service Acronym	krigingRAIN	*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Approximation of rain fall data, measured at a number of raingauge sites, using as additional information the topography of the area. The problem is to use correctly data with different resolution (very sparse raindata with respect to the DEM). The DEM should have enough details (resolution) to highlight	*Brief textual description of the service: what it provides, what can be used for. This text could be used as a*

	main topographic features without loading the system too much.		*short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	Raindata: Enriched PC – ASCII - Shapefile DEM: gridded point cloud (GeoTIFF)	*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]*	Model variogram structure; the structure should be expressed as type (e.g. Gaussian, spherical, etc) and as its parameters (e.g. nugget, sill, isotropic or anisotropic rangw and orientation of anisotropy. User may select between different algorithms	
	Output: <data representation and format>	Rain field: Gridded point cloud (GeoTIFF)	
	Functionality of the service: <text>	Interpolation of point data with cokriging paradigm.	
Algorithm	• Ordinary kriging with external drift • Simple kriging with local varying mean • Ordinary cokriging		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++, Fortran	*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	Read and write libraries for GeoTIFF and shepefile, GSLIB, GsTl	
	Operating system	Linux-like	
	Visualization modalities of the output	• 3D visualization of the rainfall map with different colours, possibly in relation to rain value • possibility to overlap the rainfall map with 3D DEM and watershed map.	
IQmulus Data	**Available IQmulus input data**: Dataset 22, Dataset 27, Dataset 26		*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:** **Robustness:** **Computational time in relation to data size:** **Locality/globality of the algorithm:**	*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.*
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	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	• CNR-IMATI, Simone Pittaluga (simone.pittaluga@ge.imati.cnr.it) • TUDelft	*Partner ID and responsible person (include email)*
Involved Partners	• Regione Liguria	

9.10 #67 APPROXIMATION OF RAINFALL DATA WITH RADIAL BASIS FUNCTIONS

IQmulus Service information			
Name of the metadata	**Content expected**	**Motivation/comments**	**INSPIRE**
Service Acronym	RBF_RainFall	*Unique identifier of the service; necessary to call it within user-defined workflows*	*2.2.1.1 Resource Title (mandatory)*

Description	Approximation of heterogeneous rainfall data (punctual gauge measurements or radar measures) using radial basis functions		*Brief textual description of the service: what it provides, what it can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*	*2.2.1.2 Abstract (mandatory)*
Service functionality	**Input: <data representation and format>**	Sample points (xyz, as .LAS). The data loader also accepts samples as vertices of a triangle mesh (as PLY format) and xyz-coordinates(as a txt file). Rain fall measures (matrix as .mat)	*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]*	None		-
	Output: <data representation and format>	Rain fall approximation at sample points (matrix as .mat)		-
	Functionality of the service: <text>			*2.2.2.2 Classification of spatial data services (mandatory)*
Algorithm	Global approximation of discrete measure (e.g., rainfall data) with radial basis functions		*Explanation of the algorithm used to provide the expected functionality; alternative algorithms may be considered for the same functionality to provide adaptivity to the context in which the services in run (eg, dataset, size of the geographical area)*	-
Implementation details	**Implementation language**	Matlab (compatible with Octave software)	*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 (when relevant, indicate required version**)**	None		-

	Operating system	Anyone running Matlab or Octave		-
	Source code availability [yes | no]	yes		
Toolchain	**Operating System [Windows | GNU/Linux | Mac OS]**	Anyone running Matlab or Octave		
	Operating System Version	Anyone running Matlab or Octave		
	Operating System Architecture [32 bit | 64 bit]	Anyone running Matlab or Octave		
IQmulus Data	**Available IQmulus input data**: These new data will be uploaded during the next days		*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]*	-
	Max data set/tile size that can be handled by the service: (approximately) *5K* punctual gauge measurements (8GB main memory)			
Service characteristics	**Accuracy:** 10^{-10}		*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: The robustness of the method is related to the well-conditioning of the coefficient matrix, which can always be guaranteed through regularization techniques.			-
	Computational time in relation to data size: The computational cost for the evaluation of the interpolating field F() is $O(n^3)$, where n is the number of rainfall stations. The resampling of F() on s samples is linear in the number s of samples. Since s>>n, the overall computational cost is linear in the number of samples.			-
	Locality/globality of the algorithm: Global approach			-
Alternatives	• #40: Approximation of rainfall data with ordinary kriging • Service#58: Spline representation of rainfall data		*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 and scenario (Land, Marine, Urban)	• Use case (IQmulus) #1107 • User story (IQmulus) #1106 • User story (IQmulus) #1105 • User story (IQmulus) #1076 • User story (IQmulus) #1075	*Mention the user stories related to the usage of the service (see D4.1.1) and related use cases uploaded on Red Mine*	*-mandatory*
Responsible Partner	CNR-IMATI, Giuseppe Patanè (patane@ge.imati.cnr.it)	*Partner ID and responsible person (include email)*	*2.2.10 Responsible organisation (mandatory)*
Involved Partners	CNR-IMATI		-