SPATIO-TEMPORAL DATA FUSION TOOLKIT (MONTH 12)

Deliverable D4.2.1

Circulation:	PU - Public
Lead partner:	TU Delft
Contributing partners:	CNR-IMATI-GE, IGN, UCL, FOMI
Authors:	Roderik Lindenbergh, Beril Sirmacek (TUDelft). Jan Boehm, Simon Julier, Kun Liu, Ziyi Jiang (UCL); Clement Mallet et al., (IGN), Simone Pittaluga, Giuseppe Patanè, Andrea Cerri, Michela Spagnuolo (CNR-IMATI-GE); Daniel Kristof, Istvan Fekete (FOMI);
Quality Controller:	Ewald Quak (SINTEF)
Version:	1.0
Date:	01.11.2013

Consisting of

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

DOCUMENT HISTORY

Version[1]	Issue Date	Stage	Content and Changes
0.9	Nov 1 2013		Version for quality control
1.0	Nov 1 2013		Version to be submitted to the Project Officer

[1] Integers correspond to submitted versions

EXECUTIVE SUMMARY

This report describes the first version of the IQmulus *Spatio-Temporal Data Fusion Toolbox* (Deliverable D4.2.1). This toolbox actually consists of a series of implemented algorithms, spread over 13 different services.

In the introduction of this report the process that led to the identification of these services is described together with an outlook to the second year of the IQmulus project, in relation to next year's version of this toolbox.

To develop these services, extensive use has been made of a number of open source libraries that are therefore shortly discussed as well.

The services have been developed in relation to certain test data that again relate to the IQmulus showcases. Also this test data is shortly described.

The main part of this toolbox report is, however, dedicated to the different services themselves. For each service a short description of its purpose is given, followed by a more detailed description of the algorithm used and a short discussion of current shortcomings and expected improvements. Almost every service is also demonstrated on test data.

In addition, service tables are provided for each service in an appendix that lists in a uniform way all information relevant for a service.

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

1.1 SCOPE OF TASK 4.2

This report describes the services provided for IQmulus in Work Package 4, Task 4.2. Within Work Package 4, which considers functional and domain processing services of IQmulus, Task 4.2 is focussed on spatio-temporal data fusion. As the Description of Work states, this task focusses on processing methods aiming at creating a single data set containing information of the same type (univariate data set) out of different data sources (multi-source fusion). The processing should deal with different resolutions, spectral dimensions and temporal dimensions. Quality assessment of the resulting data (localized, at best at point level) should be provided both qualitatively and quantitatively. Various knowledge sources will be considered to perform fusion: sensor characteristics or known characteristics of the entities measured; knowledge about features, which might be available for some of the data sets to be integrated. This task will also propose new ways to incorporate methods of interpolation and extrapolation into the workflow, in order to match different attributes sampled at different spatial and/or temporal resolution.

1.2 FROM USER STORIES TO SERVICES

The first year of the IQmulus project started with an analysis of the needs of users. These needs were described in user stories, which were condensed into three main showcases, two corresponding to a land scenario and one corresponding to a marine scenario. A breakdown of user stories into single components led to the identification of so-called services. A service is an IQmulus software program that provides some atomic functionality. Two types of services are distinguished, functional and domain services. A functional service provides a general functionality that is expected to be needed in different WP 4 tasks and may even be needed for other work packages; think notably of WP 5 Interactive Visual Decision Support. A domain service is a service that is directly connected to the theme of Work Package 4. An example of a functional service is Service 11 (see the Appendix for details), Spatial Extent, which determines the boundaries of a dataset. An example of a domain service is Service 6 (see again the Appendix), ICP registration, which aligns two point clouds in a common coordinate system, which typically belongs to the domain of data fusion.

1.3 CHOICE AND SCOPE OF INDIVIDUAL SERVICES

A choice had to be made on which services to address in the first-year toolbox. Priority has been given to services that are directly related to the identified showcases. One service may consist of different instances. There are for example different ways to define the Spatial Extent of a data set, see Service 11, depending on the dimension of the data set, but also within the same dimension, there are alternative definitions. Each service corresponds to an executable. To create these executables either existing Open Source software repositories have been used or new code has been developed. If possible the code, and therefore the service, has been tested on data from the IQmulus data repository. Datasets in this repository are also directly linked to the showcases.

1.4 SERVICE DESCRIPTIONS

The different services are described below in a uniform way. For each service, a Service Table is provided in the Appendix that gives the service name and ID, a description of its functionality, a description of the algorithm used and details on the implementation, like the use of Open Source code. It is also described on which IQmulus data the service was or could be tested and if alternative services exist with related functionality. Each service is described in terms of it computational efficiency, quality of the output and potential future improvements. Finally it is essential for what purpose a service is designed, therefore also links to user stories are provided. Several services have already been tested on data from the IQmulus data repository. If available, examples of the results of a service as executed on IQmulus data are shown in figures.

1.5 OUTLOOK TO THE SECOND-YEAR TOOLBOX

We conclude this summary with an outlook towards the second-year Toolbox. At this stage, most service executables have only been tested internally, by the developing partner. As a next step the service executables should be tested by the different partners in WP 4. This testing phase is expected to lead to adaptations in both the Service Tables and in the functionality of some services. Then it is needed to test complete workflows, corresponding to relevant user stories. To implement a user story, a certain sequence of services will be needed. Workflow testing is expected to result in the identification of some missing functionalities, which will result in the design of new services, to be developed for the second-year Toolbox.

Besides the basic functionality of a service also its performance should be evaluated, notably in terms of quality and computational aspects. This testing will take place within Task 4.2, as part of WP 4, but also in cooperation with WP 8 Dissemination and Outreach. WP 8 has already started to organize contests in which different algorithms, both from within and from outside the IQmulus consortium, are tested on fixed data sets on a fixed computer. These types of tests will often reveal the need for adaptations in the algorithms, implementations and scope of options. That is, in the next years new versions of some services will be released. Also new testing possibilities will emerge together with the maturing of the IQmulus project. We will move to larger datasets, and a first version of the IQmulus infrastructure will be released. Meanwhile, analysis of additional user stories will lead to the identification and implementation of new services. These are in short the steps that will contribute to the second-year Toolbox.

2 LIBRARIES, SOFTWARE & COMPILERS

In the following, we briefly review libraries, software and compilers used to build the services for *spatio-temporal data fusion*. These libraries have been collected and extensively discussed in collaboration with other tasks in WP4 and will be further reviewed during the development of the processing services.

2.1 AEGIS

AEGIS is not yet used in the services but is shortly described here, as it is expected to be employed in Year 2 of IQmulus, being currently in use at partner FOMI.

The following features are currently available in the AEGIS system, or under development to be included in the near future: the spatial data model is based on the OGC Simple Feature Access standard including general spatial operations on 2D/3D features and metadata support. Spatial indexing solutions are implemented for enhanced query support in geometry collections. Support is also available for raster geometry (including histogram and spectral space calculations). Graph representations for both vector and raster data can be used with multiple general graph algorithms for traversal, computing shortest paths, spanning tree, etc.

The spatio-temporal data model is an extension to the standard by introducing time as an additional dimension and temporal validity for all geometries. Temporal alteration of metadata is also supported. Spatio-temporal indexing solutions are introduced on multiple levels (collection and geometry). Reference system and coordinate transformation management are based on the OGC Abstract Specification and EPSG guidelines. An extendable operations library for both vector and raster operations, including geometry transformation, segmentation, clusterization, spectral space manipulation, etc., is offered. Some special operations are available including vehicle routing, agent based simulation and traffic forecast. There is I/O support for multiple spatial data formats (WKT/WKB, GML, Shapefile, GeoTIFF, etc.). Processing services are offering a high level scripting language for creating batch operations and custom algorithms working with AEGIS operations. Operations are interpreted to an internal DSL, which can be transformed to .NET IL or OpenCL code. The latter may be executed in parallel on GPGPU architecture. Application services including revision control of spatial and spatio-temporal data with branching support are in place. An efficient data model is implemented for managing raster data.

See for more information http://www.aegistg.com/Geospatial_Programs.htm

2.2 ANN

Used in Service #6 aiming at fine registration

ANN is a library written in C++, which supports data structures and algorithms for both exact and approximate nearest neighbour searching in arbitrarily high dimensions. In the nearest neighbour problem a set P of data points in d-dimensional space is given. These points are preprocessed into a data structure, so that given any query point q, the nearest or generally k nearest points of P to q can be reported efficiently. The distance between two points can be defined in many ways. ANN assumes that distances are measured using any class of distance functions called Minkowski metrics. These include the well-known Euclidean distance, Manhattan distance, and max distance. ANN performs quite efficiently for point sets ranging in size from thousands to hundreds of thousands, and in dimensions as high as 20. The library implements a number of different data structures, based on kd-trees and box-decomposition

trees, and it employs a couple of different search strategies. The library also comes with test programs for measuring the quality of performance of ANN on any particular data sets, as well as programs for visualizing the structure of the geometric data structures.

More information and the download folder can be found at
http://www.cs.umd.edu/~mount/ANN/

2.3 BOOST

Referred to in Services #6, #11, #18, #23, #29 and #31

Boost is a set of libraries for the C++ programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions and unit testing. Boost is required by libLAS and PCL, and it is also used by commercial packages including Matlab and Adobe Acrobat. The libraries are mostly released under the Boost Software License, which allows for the use of the libraries in commercial products. Given its widespread use by multiple IQmulus libraries, this library will be used throughout the duration of the project.

More information and the download folder can be found at http://www.boost.org/

2.4 BUNDLER

Bundler is used for service #31, point cloud generation from multiple images

Bundler is a structure from motion (SfM) system for unordered image collections (for instance, images from the Internet, or images from different time and sensor sources). It is written in C and C++. Bundler takes a set of images, image features, and image matches as input, and produces a 3D reconstruction of camera and sparse scene geometry as output. The system reconstructs the scene incrementally, a few images at a time, using a modified version of the Sparse Bundle Adjustment library package as the underlying optimization engine. Bundler is distributed under the GNU General Public License.

The Bundler library folder can be downloaded from
http://www.cs.cornell.edu/~snavely/bundler/#S2

2.5 FAST

Referred to in service #29, image to point cloud matching

FAST (Features from Accelerated Segment Test) is used for corner detection in 2D images. It operates by considering a circle of sixteen pixels around the corner candidate p. FAST requires the libCVD (computer vision) library. The libCVD is a very portable and high performance C++ library for computer vision, image, and video processing. The emphasis is on providing simple and efficient image and video handling and high quality implementations of common low-level image processing functions. The library is designed in a loosely-coupled manner, so that parts can be used easily in isolation if the whole library is not required.

The FAST library can be downloaded from http://www.edwardrosten.com/work/fast.html

2.6 GDAL

GDAL is used for service #23, format conversion.

GDAL (Geospatial Data Abstraction Library) is a library for reading and writing raster geospatial data formats, and is released under the permissive X/MIT style free software license by the Open Source Geospatial Foundation. GDAL is considered as major free software for its extensive capabilities of data exchange and also in the commercial GIS community due to its widespread use and comprehensive set of functionalities. GDAL as of version 1.9.0 provides at least partial support for more than 120 raster geospatial data formats. In the IQmulus project, we benefit from the GDAL library for reading and writing raster geospatial data formats such as GeoTIFF, Erdas Imagine, Ascii Grid, Ascii Gridded XYZ, ERMapper, ESRI, SDTS, EXW, MrSID, JPEG2000, DTED, NITF, etc.

More information and the download folder can be found at http://www.gdal.org/

2.7 HADOOP

Used in services #35 and #36, point cloud thinning and resampling

The Apache Hadoop is a project which develops open source software for reliable and scalable distributed computing. The Apache Hadoop software library allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. A wide variety of companies and organizations use Hadoop for both research and production. Hadoop consists of a number of modules. In the IQmulus project, we focus on the usage of two components: the Hadoop Distributed File System and the MapReduce engine.

The Hadoop library can be downloaded from http://hadoop.apache.org/releases.html

2.8 LIBANN

Used in service #6 aiming at fine registration

LIBANN (Artificial Neural Network Library) is a C++ library which provides implementations of Artificial Neural Networks. Artificial Neural Networks are used in many systems. They have applications in Speech Recognition, OCR and robotics. LIBANN allows creating and using Neural Nets without going into complicated details of the implementation. The library is written using the C++ Standard Template Library. LIBANN is a free library.

The LIBANN library can be downloaded from
http://download.savannah.gnu.org/releases/libann/

2.9 LIBLAS

Used in services #23, #35 and #36

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 provides a number of attractive features for a software developer looking to incorporate support for the ASPRS LAS format in their software. LibLAS is open source and can be embedded in commercial/non-commercial applications as long as the restrictions of the BSD license are followed.

More information and the download package can be found at http://www.liblas.org

2.10 NANOFLANN

Used in services #8 and #12, cloud to cloud distances and point cloud intersection

NanoFlann is a C++ header-only library for building KD-Trees, mostly optimized for 2D or 3D point clouds. Queries for neighbors around any arbitrary location in space can then be solved quickly and efficiently using Approximate Nearest Neighbor (ANN) algorithms. This library is a subset of the Flann library. Following the original license terms, NanoFlann is distributed under the BSD license. The power of the original Flann library comes from the possibility of choosing between different ANN algorithms. The cost to pay is the declaration of pure virtual methods, which (in some circumstances) impose run-time penalties. In NanoFlann all those virtual methods have been replaced by a combination of the Curiously Recurring Template Pattern (CRTP) and inlined methods, which are much faster.

More information and the download package can be found at
http://code.google.com/p/nanoflann/

2.11 OPENCV

Is used in services #23, #29 and #31

OpenCV is a library of programming functions which are mainly developed for real-time computer vision applications. The OpenCV library's wide solution spectrum includes functions for 2D and 3D feature extraction segmentation, stereo vision, classification and learning methods. In the IQmulus project, we will benefit from the OpenCV library almost in every task for processing data (i.e., gridded 2D and 3D data, iPhone images, range images).

More information and the library package can be found at http://opencv.org/

2.12 OPENGL

Referred to in service #29 image to point cloud matching

OpenGL is a cross language, multiplatform API for rendering 2D and 3D computer graphics. The library helps with texture mapping on 3D surfaces and the generation of realistic 3D scenes. For example, shadow and illumination properties can be controlled easily.

More information and the library package can be found at http://www.opengl.org/

2.13 PCL

Referred to in services #11, #18, #23 and #29

PCL is a standalone open source library for handling point clouds and 3D geometry processing. The library includes algorithms for filtering, feature estimation, surface reconstruction, surface

normal extraction, registration, model fitting and segmentation. For instance, the ICP (Iterative Closest Point) algorithm is a popular method for registering point clouds by iteratively minimizing the distance between paired or corresponding points in the cloud and the Sample Consensus Initial Alignment (SAC-IA) is also another popular registration method which can be found in the PCL framework. We believe that the provided functions will be helpful to develop our experimental code implementations. PCL also provides 3D interest point and shape descriptor extraction methods such as NARF (Normal Aligned Radial Feature), Point Feature Histograms (PFH) and Fast Point Feature Histograms (FPFH), which can be of use in our experiments.

More information and the library package can be found at http://pointclouds.org/

2.14 SFM

Referred to in service #32 multiple images to point cloud

Libmv is an open source structure from motion library, which plans to take raw video or photographs, and produce full camera calibration information and dense 3D models. Libmv consists of different modules (image/detector/descriptor/multiview) that allow resolving part of the SfM process. The user can choose to use the available implementation or to customize the module by writing her own framework using a general interface.

Libmv library can be downloaded from https://code.google.com/p/libmv/

2.15 VTK

Referred to in services #8, #11, #12, #18, #23, #29, #31

VTK is an open source framework for 3D computer graphics, image processing and visualization. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture and volumetric methods and advanced modelling techniques such as implicit modelling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation.

More information and the library download package can be found at http://www.vtk.org/

2.16 COMPILERS

For writing, editing and compiling C++ codes and libraries we use Microsoft Visual Studio 2010. Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms or WPF (Windows Presentation Foundation) applications, web sites, web applications and web services in both native code together with managed code for all platforms which are supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight. Visual Studio includes a code editor supporting IntelliSense and code refactoring. The integrated debugger works both as a source-level debugger and a machine level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level including adding support for source control systems and adding new toolsets like editors and visual designers for domain specific languages or toolsets for other aspects of the software development life cycle. Visual Studio supports different programming languages by means of language services, which allow the code editor and

debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists.

Visual Studio can be downloaded from
http://www.microsoft.com/visualstudio/eng/products/visual-studio-2010-express,
however a license must be bought and registered online.

3 TEST DATA SETS

Here an overview is given of the data sets used for testing the services.

3.1 AHN2, DELFT

This data set has been used in Services #8, cloud to cloud distance and #12, dataset intersection

This point cloud is part of the Dutch airborne laser altimetry archive called the Actual Height model of the Netherlands (AHN). The first AHN data was acquired in the years 1996-2003 under leaf-off conditions with a point density of at least 1 point per 4x4m2 area. Starting from 2007, AHN2 has been acquired over the Netherlands. In Figure 1, we represent a screenshot of point cloud data which samples part of the city of Delft, The Netherlands. The original data is in the Dutch RD (Amersfoort) coordinate system.

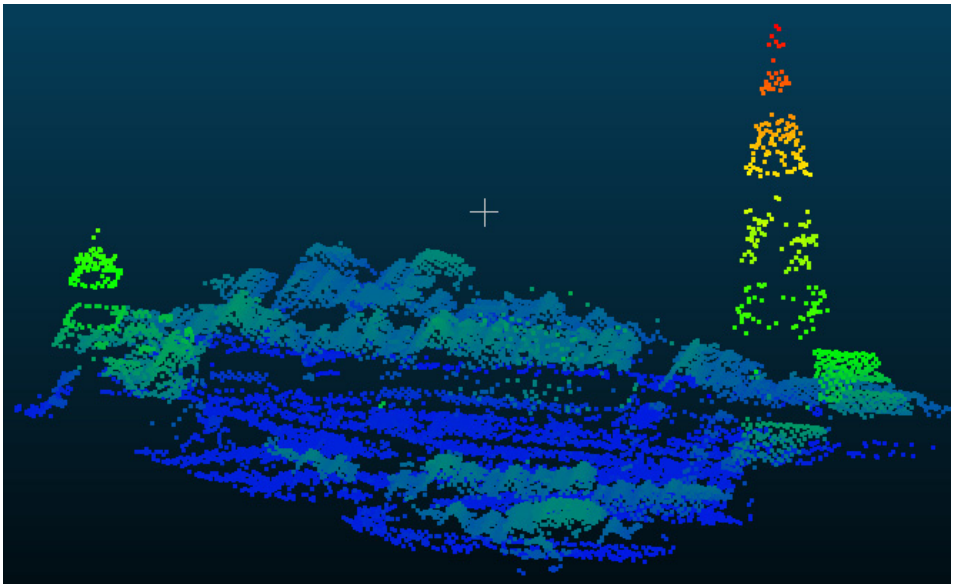


FIGURE 1: AHN2 AIRBORNE LASER SCANNING POINT CLOUD SAMPLING DELFT, THE NETHERLANDS

3.2 IPHONE IMAGE DATA SET, DELFT

This data set is used in service #31 (multiple images to point cloud conversion).

We benefit from smartphone camera sensors to obtain photographs for point cloud generation and texture registration on 3D urban models. In this way, we would like to update archived point cloud data. So far for our experiments we have generated data sets which contain multi-view iPhone 3G camera photographs of some urban structures. Those iPhone photographs can be read with their metafiles which are written in exchangeable image file format (Exif). Exif is a standard that specifies the formats for images, sound and other digital records like videos or scanner data. The metadata contain a wide spectrum of tags like data and time information, camera properties, GPS position, looking angle, image resolution and properties, device properties, etc.

In Figure 2, we provide an example for multi-view images. The photographs are showing a historical windmill in Delft city center.

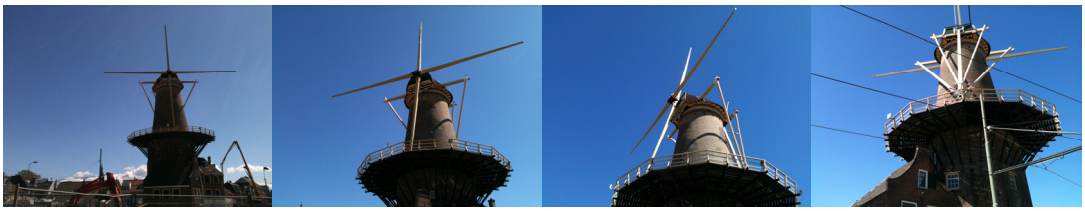


FIGURE 2: IPHONE PHOTOS FROM A HISTORICAL WINDMILL IN DELFT, THE NETHERLANDS.

3.3 BRITTANY POCKET BEACH

A pocket beach in Brittany, France, is sampled both on- and offshore by bathymetric and topographic data.

3.3.1 Bathymetric Data

This data set has been used in Services #8, #11, #18, #23 and #29.

One of our point cloud data sets is a bathymetric digital terrain model (DTM) of a small pocket beach in Brittany, which is obtained by a mobile laser sensor on a ship. The point cloud data set is provided by the project partner UBO. The data set contains two point clouds which were acquired in the years 2010 and 2011. The original point clouds are in ASCII format. The point clouds of 2010 and 2011 contain 61741 and 451653 data point,s respectively. In Figure 3, we provide a screenshot view of the point cloud which is acquired in 2011. The data set uses the EPSG:2154 (RGF93/Lambert93) coordinate system.

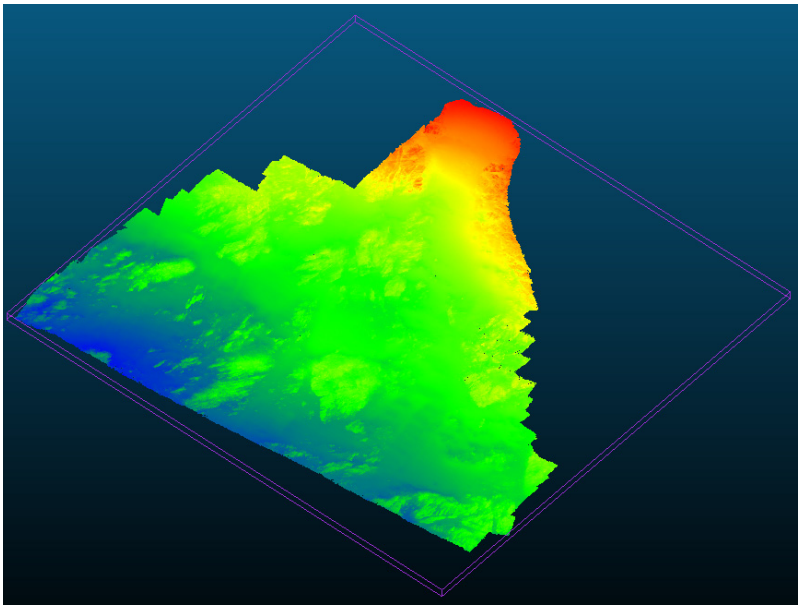


FIGURE 3: BATHYMETRIC POINT CLOUD FROM A BRITTANY POCKET BEACH, ACQUIRED IN 2011.

3.3.2 Topographic Data

This data set has been used in Services #11, #12, #18, #23 and #29.

Digital terrain models (DTM) of the land part of the same small pocket beach in Brittany were obtained by airborne laser altimetry. The point cloud data set is again provided by partner UBO. The data set contains two point clouds, which were acquired in 2009 and 2010. The original 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 4, we provide a screenshot of the point cloud from 2009. The data set uses the EPSG:2154 (RGF93/Lambert93) coordinate system.

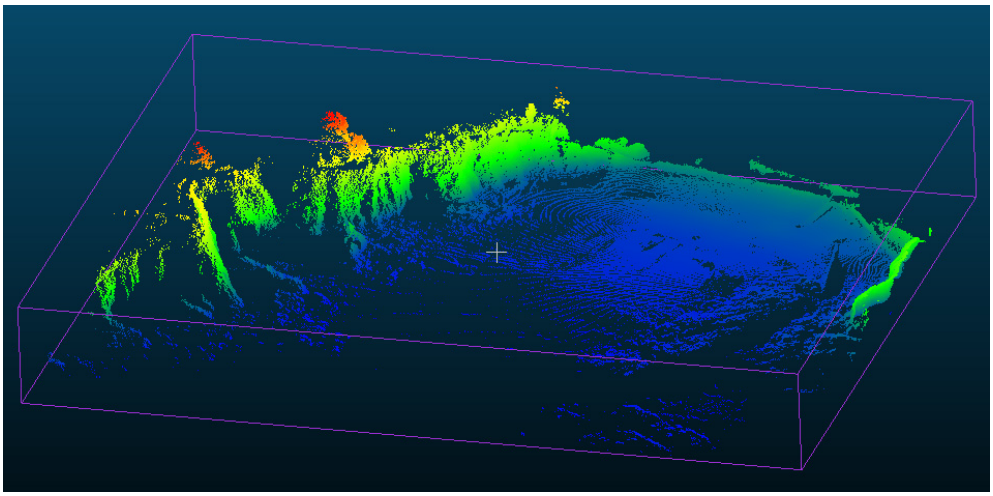


FIGURE 4: TOPOGRAPHIC POINT CLOUD FROM A BRITTANY POCKET BEACH, ACQUIRED IN 2009.

3.4 IGN CITY POINT CLOUDS

Two data sets are provided by partner IGN that cover parts of French cities.

3.4.1 IGN airborne laser scanning data

This data set has been used in Service #6, Fine Registration

The dataset has been acquired over a dense urban area in Amiens, France, at two different dates, and with distinct airborne Lidar scanners, resulting in two different ground patterns. The first acquisition was completed in 2003 (point density of 7.5 pts/m^2, 400 000 pts), with a Toposys fiber scanner. Spatial sampling is therefore very inhomogeneous (1.5 m between two fibers, and 0.15 m between two measurements of the same fiber). The second acquisition occurred in 2008 (point density of 2 pts/m^2, 90 000 pts, oscillating mirror) with an Optech 3100 EA device.

3.4.2 IGN LMMS data

This data set has been used in Service #6, Fine Registration

These point clouds are acquired with the Stereopolis vehicle. This dataset covers one building facade in an urban area in Paris, France. The mobile mapping system acquired the same area twice on the same day but with a time shift of one hour. A RIEGL LMS-Q120i lidar has been used for that purpose, and oriented towards the roof top (pavement and road surfaces omitted). The challenges here are: (1) not exactly the same parts of the buildings are sampled and (2) a 3D shift between both point clouds naturally exists, due to the georeferencing process. The point cloud density is very variable, even within the same point cloud, depending on the angle of incidence of the variable distance between objects and the MMS. However, the typical density on the facades of the buildings is near 100 pts/m^2 (around 200,000 pts per point cloud).

3.5 LIGURIA DATA

This data set is used in service #11, #18, #23, #29, #35, #36

Another complementary point cloud data set consists of both bathymetric and topographic point clouds of a study area in Vernazza, Liguria, Italy. The data set is provided by the project partner Liguria.

3.5.1 Liguria bathymetric data

The data set contains a point cloud which is acquired by Multibeam Echosounding. The original point cloud is in ASCII format and contains 28.349.605 data points. In Figure 5, we provide a screenshot of the Liguria bathymetric point cloud. The data is provided in the Gauss Boaga (EPSG: 3004) coordinate system.

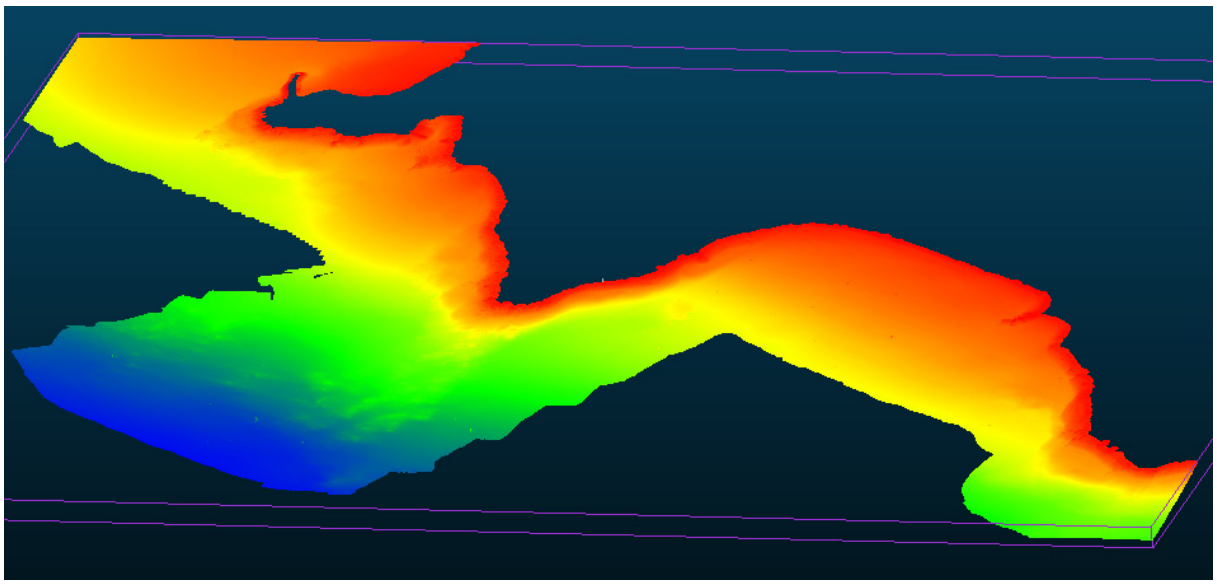


FIGURE 5: LIGURIA, BATHYMETRIC POINT CLOUD

3.5.2 Liguria topographic data

The data set contains a point cloud which is acquired with an airborne laser sensor. The original point cloud is in .las format and contains 4 134 193 data points. In Figure 4, we provide a screenshot view of the Liguria topography point cloud. The data is probably also given in the Gauss Boaga (EPSG: 3004) system.

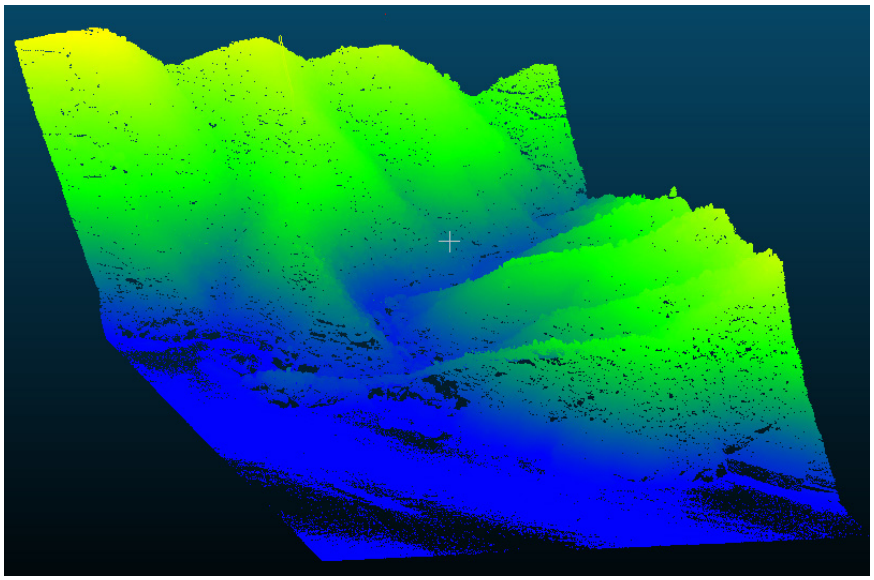


FIGURE 6: LAS POINT CLOUD SAMPLING TOPOGRAPHY AT LIGURIAN COAST

4 INDIVIDUAL SERVICES

In this chapter the IQmulus Task 4.2 services that were developed for Toolbox D4.2.1 are described in some detail. For each service a short description is given, an overview of the method and some examples using the available test data sets. In addition, for each service a service table is provided in the Appendix that contains a full description of all aspects of a service. The ID of each service is used for internal communication purposes, and it refers to the ID used in the eRoom table of services.

4.1 ICP REGISTRATION (SERVICE #6)

Registration is the process of aligning notably 3D point cloud data into a common coordinate system. Therefore registration aims at identifying an optimal rigid body transformation that transforms a second scan into the coordinate system of a reference scan. Typically first a coarse registration is used to align two point clouds in an approximate way. In a second step fine registration further optimizes the coarse registration. A coarse registration of sufficient quality is needed to ensure convergence of the fine registration. ICP, which stands for Iterative Closest Points, is the best-known method of fine registration. It minimizes in an iterative way corresponding point-to-point or point-to-surface distances. The service table can be found in the appendix section 5.1.

4.1.1 Current algorithm and possible improvements

This is a more detailed description of the algorithm:

Until a given accuracy threshold is reached/ Until a given number of iterations is reached:

- Sub-sample the 2 point clouds keeping only high entropy values (service #66);
- Match the closest points in both point clouds, using the Euclidian distance;
- Weigh each pair of points using a constant value;
- Keep only the 50% best pairs wrt their difference in terms of ellipsoid volume (service #66).
- Find the transformation that minimizes the sum of squared differences between each couple of points and by taking into account the local normal estimates.
- Repeat.

To deal with very large point clouds it will be considered to implement a hierarchical approach in the future. On the other hand, a multi-scale/decimation approach will be considered so as to deal with dense point clouds of small spatial extent.

4.1.2 Examples, Service #6

Figure 7 and Figure 8 demonstrate the service; both on airborne and terrestrial laser scan data for two different scenes in Paris.

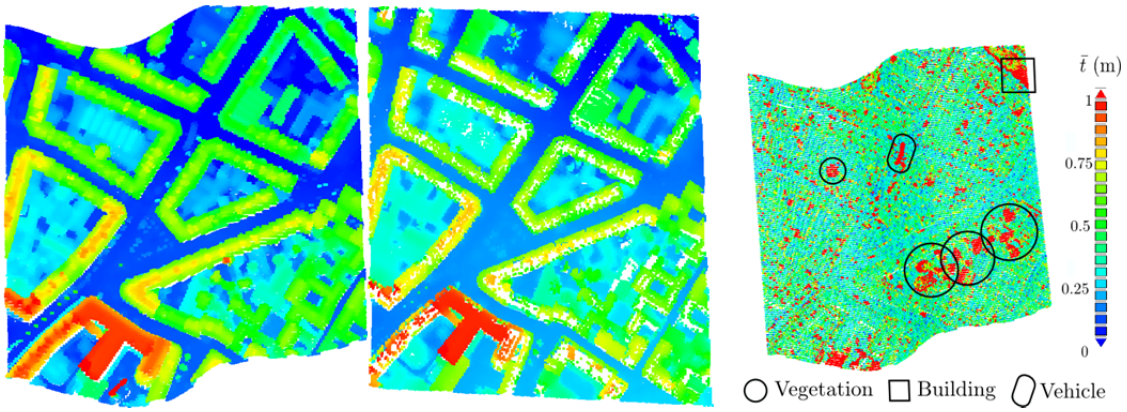


FIGURE 7: RESULTS ICP FINE REGISTRATION ON AIRBORNE LASER DATA (LEFT) 2003 DATA (MIDDLE) 2008 DATA, (RIGHT) QUALITY MAP, SHOWING ABSOLUTE DIFFERENCES AFTER FINE REGISTRATION IN METER. THE INDICATED FEATURES CORRESPOND TO ACTUAL CHANGES, E.G., FEATURE IN ELLIPSE IS A BUS.

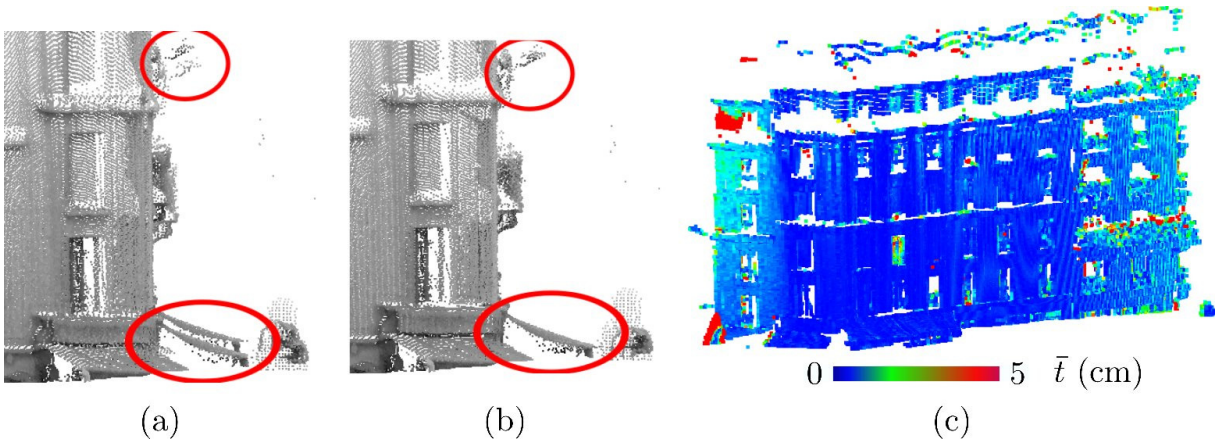


FIGURE 8: RESULTS ICP FINE REGISTRATION ON LMMS DATA (LEFT) BEFORE FINE REGISTRATION LOCAL MISMATCHES ARE CLEARLY VISIBLE IN THE ELLIPSES (MIDDLE) AFTER FINE REGISTRATION MISMATCHES HAVE DISAPPEARED. (RIGHT) QUALITY OF FINE REGISTRATION AS CLOUD TO CLOUD DISTANCES IN CM.

4.2 POINT CLOUD TO POINT CLOUD DISTANCE (SERVICE #8)

Once two point clouds A and B are sampling the same region of interest and are given in approximately the same coordinate system, the cloud-to-cloud distances have many applications. By cloud-to-cloud distance is meant either the distance between a point in cloud A to the nearest point in cloud B, or the distance between a point in cloud A and some surface representation of cloud B. The resulting distances can be used as an evaluation step in a fine registration, for validating a data set of unknown quality against a reference data set of superior quality or to detect changes between two data sets acquired at different moments. The service table can be found in the appendix section 5.2.

4.2.1 Current algorithm and possible improvements

- Use the LibLas library to read in two point cloud data sets in ".las" file format
- Find the closest points between the two data sets
- Calculate 3D Euclidean distances between the closest points of the two data sets
- Use the VTK library to display the point clouds and color coded distances

- Use the VTK library to provide 3D data rendering, zooming and clicking to the user

This functional service should be very efficient as it will be used over and over again. There are many implementations conceivable, e.g., using MapReduce, using spatial data structures and using approximate methods. We should consider organizing a contest for this service.

4.2.2 Examples, service #8

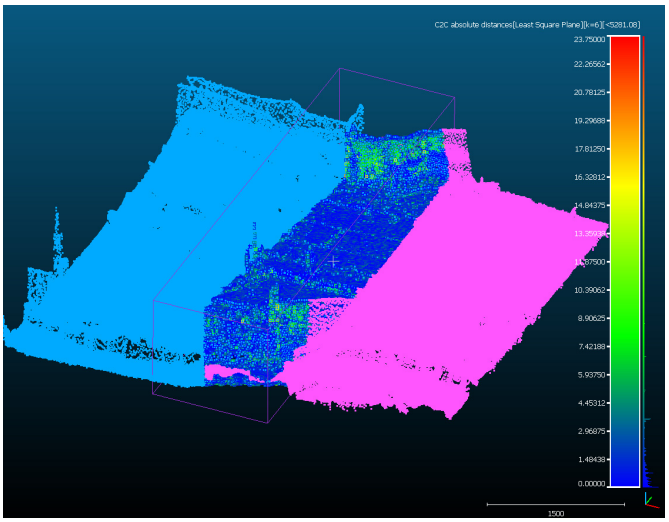


FIGURE 9: CLOUD TO CLOUD DISTANCES AT THE INTERSECTION OF THE TWO POINT CLOUDS FROM FIGURE 12

4.3 SPATIAL EXTENT (SERVICE #11)

The spatial extent of a spatial data set is a description of the area covered by the data set. Different descriptions are possible, like the minimal and maximal value of each coordinate, leading to a bounding box, or the convex hull, the smallest convex set that fully contains a set of spatial locations. A more advanced concept is that of alpha shapes. Holes in a spatial data set are for example not detected by bounding boxes or convex hulls. Alpha shapes can fit more tightly around a spatial data set, where the fit is controlled by the parameter alpha. The service table can be found in the appendix section 5.3.

4.3.1 Current algorithm and possible extensions

For this service different algorithms have been implemented. All algorithms start and end similarly:

- Read the point cloud which is in ".las" file format using the libLas library
- Use the VTK library for 3D visualization of the points

Spatial Extent – 2D Alpha Shape Extraction

- Use the PCL library for a coordinate transformation based on PCA analysis
- Use the PCL library to extract 2D alpha shape

Spatial Extent – 3D Alpha Shape Extraction

- Use the PCL library to extract 3D alpha shape

Spatial Extent – 2D Bounding Box Extraction

- Use the PCL library for the coordinate transformation based on PCA analysis
- Extract minimum and maximum point coordinate values on the main plane surface to detect the 2D bounding box shape

Spatial Extent – 3D Bounding Box Extraction

- Extract minimum and maximum point coordinate values on the main plane surface to detect the 3D bounding box shape

Spatial Extent – 3D Convex Hull Extraction

- Use the PCL library to extract 3D convex hull

In all cases the method ends with:

- Use the VTK library to provide 3D rendering, zooming and clicking to the user
- Use the VTK library to plot the detected spatial extent on the 3D display

This is a typical functional surface, which is expected to be frequently used. The performance of the current implementations will be extensively tested in year 2 of IQmulus. If necessary, alternative implementations will be considered. For the alpha shape algorithms, choosing a correct alpha value is quite important. We will consider developing automatic algorithms to suggest default alpha values to the user.

4.3.2 Examples, service #11

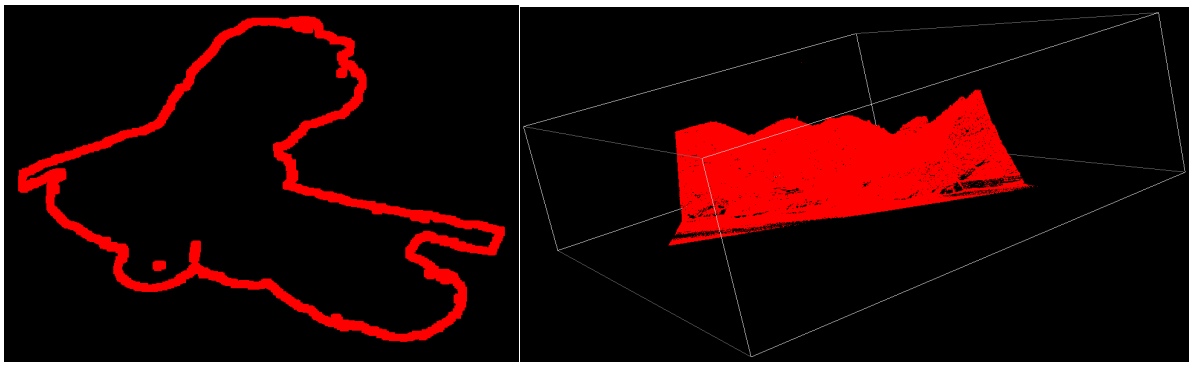


FIGURE 10: (LEFT) ALPHA SHAPE OF THE BRITTANY BATHYMETRY POINT CLOUD (RIGHT) 3D BOUNDING BOX OF THE LIGURIA POINT CLOUD DATA

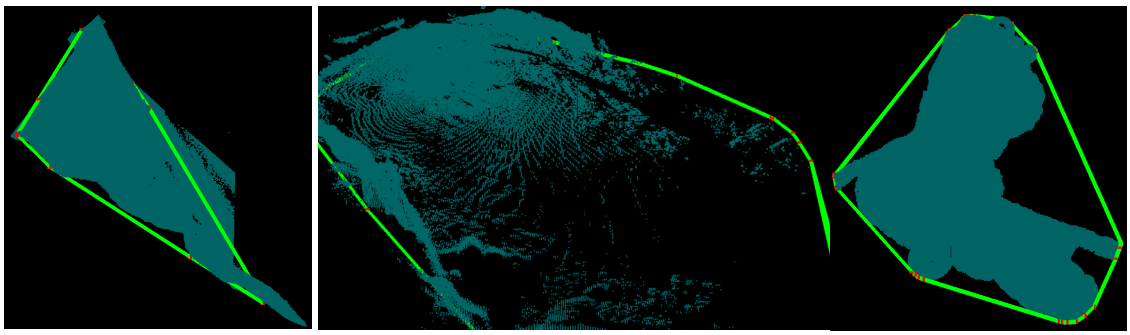


FIGURE 11:(LEFT) 2D CONVEX HULL OF THE LIGURIA POINT CLOUD;(MIDDLE) 2D CONVEX HULL OF THE BRITTANY TOPOGRAPHY POINT CLOUD;(RIGHT)2D CONVEX HULL OF THE BRITTANY BATHYMETRIC POINT CLOUD.

4.4 DATASET INTERSECTION (SERVICE #12)

Given two spatial datasets A and B, the dataset intersection consists of the region covered by both datasets A and B. The service table can be found in the appendix section 5.4.

4.4.1 Current algorithm and possible improvements

Current algorithm:

- Using the LibLas library to read in two point clouds in ".las" file format
- Find the closest points between the two data sets
- Check if the closest point distances are smaller than a previously set threshold value
- Label the boundary of the detected points as intersection area
- Use the VTK library to display the point clouds and the intersection area
- Use the VTK library to provide 3D data rendering, zooming and clicking to the user

Initial testing indicated that the current implementation may not be sufficiently efficient. Notably finding the closest points is time consuming. In year 2 of IQmulus it should be considered to integrate efficient data structures with these kind of services. For the current method, the choice of a threshold value has a large impact on the result. In the future, we will consider suggesting a default threshold value to the user.

4.4.2 Examples, service #12

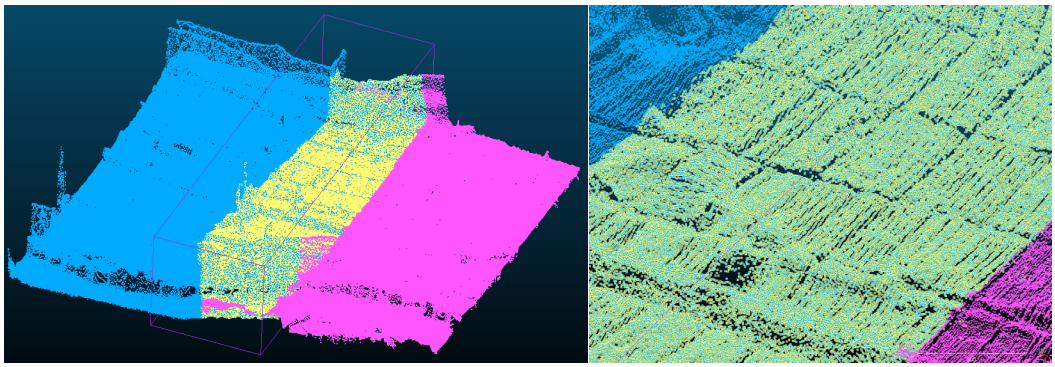


FIGURE 12: (LEFT) TWO INPUT POINT CLOUDS IN BLUE AND PINK AND THEIR INTERSECTION IN YELLOW, (RIGHT) ZOOM IN FROM THE INTERSECTION REGION

4.5 MATCHING 3D KEYPOINT DESCRIPTOR VECTORS (SERVICE #18)

3D Keypoints are low-level descriptors of features in 3D point clouds. A descriptor is typically a list of values that somehow characterize a point in its neighbourhood in terms of texture or local differential geometry, often at different scales. For the purpose of coarse registration of point clouds A and B, 3D keypoints can be matched, which means that correspondences between similar descriptors are identified. The service table can be found in the appendix section 5.5.

4.5.1 Current algorithms & possible improvements

- Use the LibLas library to read in two point cloud data in ".las" file format
- Read the matrices which hold the 3D keypoint locations and their descriptor vectors for each point cloud

- Calculate the similarities between two descriptor vector matrices
- Detect matching 3D keypoint locations
- Use the VTK library to display the input point clouds
- Use thr VTK library to label the locations of extracted local features on the 3D display

In the future different similarity measurements will be tested.

4.5.2 Examples, service #18

No figure available at the moment

4.6 GEOTIFF TO LAS (SERVICE #23)

Spatial data sets come in different ASCII and binary formats. ASCII formats have the advantage that they are directly readable in an editor, but binary formats typically occupy much less space. For both ASCII and binary formats different ways exist, e.g., to represent coordinates and metadata. The format conversion service is able to convert between the different formats that are supported by the IQmulus project. The service table can be found in the appendix section 5.6.

4.6.1 Current algorithm & possible improvements

- Read input file name
- Detect the file name extension
- Read the desired format entry
- Read the input file using related libraries (PCL, Liblas, GDAL…)
- Apply transformation using related libraries (PCL, Liblas, GDAL…)
- Save the converted point cloud format into the work folder

4.6.2 Example, service #23

No figure available at the moment

4.7 REGISTERING 2D IMAGES ON POINT CLOUDS (SERVICE #29)

2D images can be used together with 3D point clouds, for example, and an RGB image can be used to color a point cloud, or the structure in the point cloud can be compared to the structure in the images. For this purpose the 2D image should be aligned with the 3D point cloud. This means that the location of the camera during acquisition should be determined in the coordinate system of the point cloud, for example by matching 2D features detected in the photo to 3D features derived from the point cloud. Then the field of view of the 2D image can be overlaid on the point cloud and their area of intersection can be determined. The service table can be found in the appendix section 5.7.

4.7.1 Current algorithm & possible improvements

- Use the LibLas library to read in the point cloud data in ".las" file format
- Read input iPhone image using the OpenCV library
- Read iPhone image metafile

- Use iPhone camera orientation and GPS position to determine the region of interest in the point cloud
- Prepare a smaller point cloud which contains the points from the region of interest only
- Apply surface reconstruction to the small point cloud using the PCL library
- Extract low level local structural features from the reconstructed surface
- Extract low level local structural features from the iPhone image
- Apply feature matching between 3D surface and iPhone image
- Use the RANSAC algorithm to eliminate false matches
- Find the transformation matrix to register the iPhone image on the 3D surface
- Use the registered iPhone image as texture on the 3D surface
- Use the VTK library to display the 3D result

There are alternatives available for matching and outlier removal. Also calibration of the smartphone camera should be considered. The current method is only developed for iPhone images, in future also other smart phone images should be considered.

4.7.2 Example, service #29

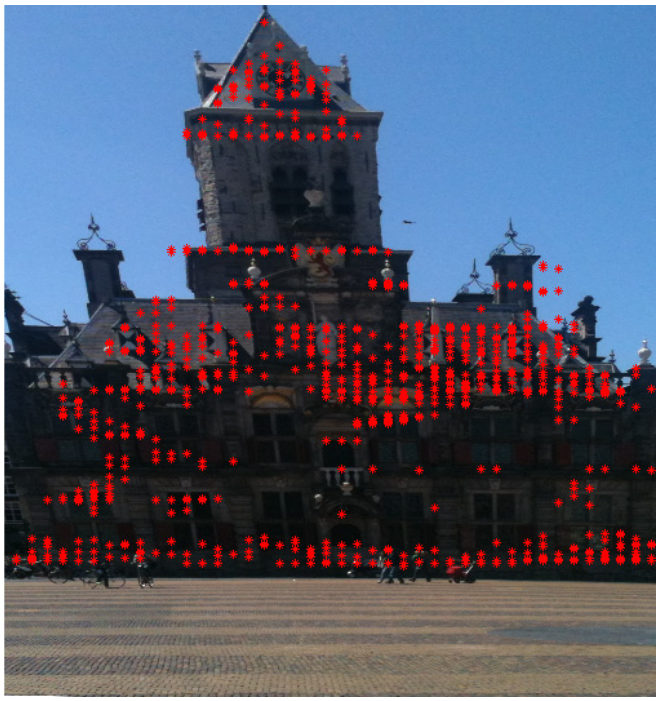


FIGURE 13: IPHONE PHOTO REGISTERED ON AIRBORNE LIDAR POINT CLOUDS (RED POINTS)

4.8 POINT CLOUD GENERATION FROM MULTI-ANGLE IMAGES (SERVICE #31)

Given 2D images with sufficient overlap corresponding points between the images can be determined. These corresponding points are used to estimate the interior and exterior orientation parameters, which identify the different locations of the camera during acquisition relative to the positions of the corresponding points in 3D. Once the 3D geometry of the scene is determined for cameras and corresponding points, 3D coordinates of remaining pixels are determined from a process called forward intersection. This in short describes the workflow

from multiple images to a 3D point cloud. The service table can be found in the appendix section 5.8.

4.8.1 Current algorithm & possible improvements

- Read input iPhone images using the OpenCV library
- Apply SIFT keypoint extraction and keypoint matching
- False match removal using RANSAC
- Find the extrinsic camera parameters
- Calculate the 3D locations of the points
- Use the VTK library to display 3D results

This workflow could be improved by the characterization of the camera properties, calibrating and evaluating it. SIFT is just one matching method, alternative methods could be used as well. The same holds for RANSAC, alternative methods are available.

The quality of the final point cloud depends on the camera properties, the quality of the matches and on the ability of RANSAC to remove weak matches. It should be possible, but not easy, to give a priori estimates of the quality of the final point cloud products. This is still a topic of further research.

The current computation time on a standard desktop computer is 5 minutes for a data set consisting of 49 IPhone images. So also from a computational viewpoint improvements are interesting.

4.8.2 Example, service #31

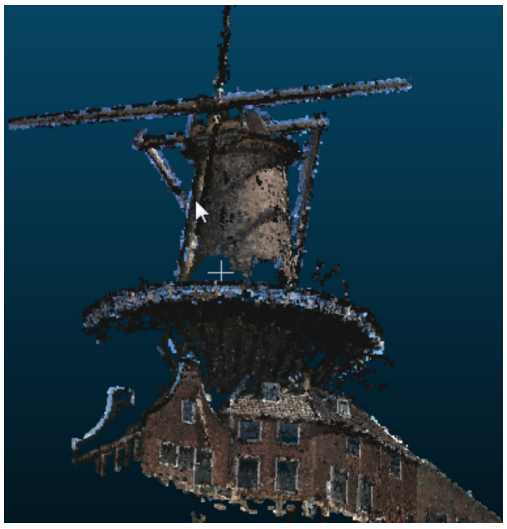


FIGURE 14: POINT CLOUD OF A DUTCH WINDMILL GENERATED FROM 49 IPHONE IMAGES WITH DIFFERENT VIEWING ANGLES

4.9 RE-SAMPLING OF POINT CLOUDS (SERVICE #35)

Point clouds obtained by scan instruments or cameras often result in coordinates that seem unorganized given some arbitrary Cartesian coordinate system. If, e.g., a surface height estimate is required on some given 2D grid, the available height values should be resampled to the given coordinates, a process also referred to as interpolation. There is a vast literature on the topic of

interpolation; this service gives some first functionality of the resampling of point clouds. The service table can be found in the appendix section 5.9.

4.9.1 Current algorithm and possible improvements

Re-sampling is a method to decrease the size of a point cloud or to change the support of a data set. In this implementation, first local regions are computed, such as grid cells, then a point is chosen to represent each local region by a certain criterion. Here the centroid of all points in the local region is computed as an approximation of the local points it represent.

The method is implemented using Hadoop, and a snippet of Java code to generate local regions is shown in Algorithm 1. In the implementation, a point cloud is partitioned into grid cells and thus only x- and y- coordinates are considered.

```
double lower_grid_x = Math.floor((val.x / grid_res_x)) * grid_res_x;
double lower_grid_y = Math.floor((val.y / grid_res_y)) * grid_res_y;

String output_key = String.format("% 15.6f", lower_grid_x) + " "
                  + String.format("%15.6f", lower_grid_y);
String output_val = String.format("% 15.6f", val.x) + " "
                  + String.format("% 15.6f", val.y) + " "
                  + String.format("% 15.6f", val.z);
```

ALGORITHM 1. SNIPPET OF JAVA CODE TO GENERATE LOCAL REGIONS FOR SUB-SAMPLING AND RE-SAMPLING.

Actually, re-sampling can be considered as a special case of interpolation. As such, there are many possible alternatives for this service, compare also the IQmulus state of the art analysis in D4.1.1.

4.9.2 Examples, Service #35

Figure 15 (left) illustrates the results of re-sampling. The used grid cell width is 5 meters.

4.10 SUB-SAMPLING (THINNING) OF POINT CLOUDS (SERVICE #36)

Often point clouds are too large. Especially when testing a workflow it is more efficient to work with a thinned version first, which gives fast results. The sub-sampling service determines point cloud sub-samples in a systematic way. The service table can be found in the appendix section 5.10.

4.10.1 Current algorithm and possible improvements

Sub-sampling is a method to decrease the size of a point cloud. First local regions are computed, such as grid cells, then a point is chosen to represent each local region by a certain criterion. In sub-sampling, in each local region an original point is selected to represent all points present. Depending on the application, the representative point can be chosen randomly or as the one with, e.g., the maximum height value.

The method is implemented using Hadoop, and a snippet of Java code to generate local regions is contined in Algorithm 1. In the implementation, a point cloud is partitioned into grid cells and thus only x- and y- coordinates are considered.

The current algorithm does not consider the actual data values. An alternative method is to make the sub-sample adaptive to the data characteristics, for example, it could be decided to keep many original points in highly varying terrain, but much fewer points on a very flat terrain.

4.10.2 Examples, Service #36

Figure 15 (right) illustrates the results of sub-sampling. The used grid-cell width is 10 meters.

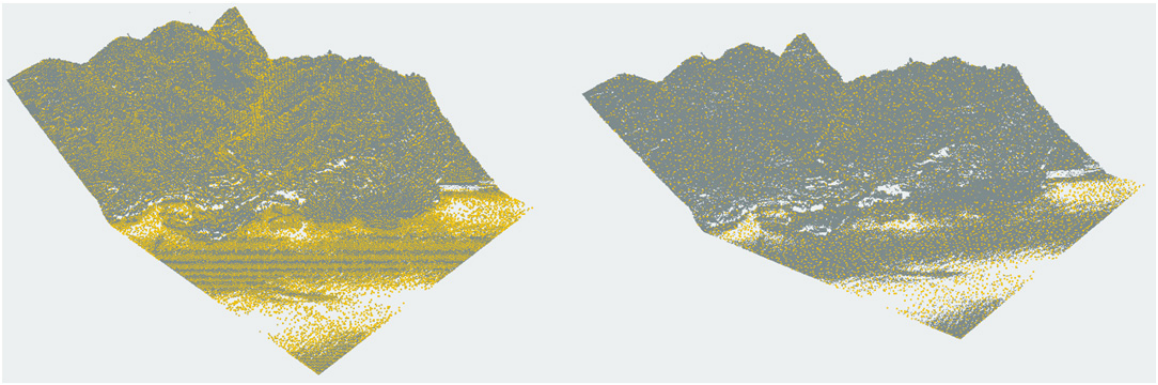


FIGURE 15: OUTCOME OF RE-SAMPLING (LEFT) AND SUB-SAMPLING (RIGHT). THE RESULTING POINT CLOUDS AFTER PROCESSING ARE COLORED IN YELLOW AND THE DARK GRAY POINTS ARE ORIGINAL.

4.11 ISOSURFACE EXTRACTION (SERVICE #46)

Given a data set that is truly 3D in the sense that it occupies a non-zero volume, it is often required to extract from such a volume a surface that has some given value for a function that maps from the volume to, say, the real numbers. An example is the surface of zero degree Celsius in the atmosphere above a country, where the height of such a surface typically varies from day to day but also varies with the local weather conditions. The service table can be found in the appendix section 5.11.

4.11.1 Current algorithm and possible improvements

The main references for this algorithm are

- W.Lorensen, H.Cline, Marching Cubes: *A High Resolution 3D Surface Construction Algorithm,* Computer Graphics, 21 (4): 163-169, July 1987
- Chien-Chang Ho, Fu-Che Wu, Bing-Yu Chen, Yung-Yu Chuang and Ming Ouhyoung. *Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data*, In Computer Graphics Forum *(Proc. EUROGRAPHICS 2005)*, 24(3), pp 537-545, Dublin, Ireland, August 2005.

4.11.2 Examples, Service #46

FIGURE 16: INTEGRATED VISUALIZATION OF ISO-CONTOURS AND ISO-SURFACES ON A TERRAIN MODEL (TOP VIEW).

4.12 ISOCONTOUR EXTRACTION (SERVICE #47)

Isocontours are curves where a given value of a surface mapping to, say, the real numbers, has a constant value. The best-known examples are contour lines on a map, connecting locations with the same height. The service table can be found in the appendix section 5.12.

4.12.1 Current algorithm and possible improvements

The algorithm extracts a list of iso-contours from a triangle mesh that represents a digital elevation model. Linear interpolation of the height values at the mesh vertices will be used to trace the iso-contours. Possible adaptation to uniform grids will be taken into account. Input parameters of this service are the number of iso-values or the list of iso-values. If input parameters are not provided, then pre-defined values will be used (uniform sampling of the iso-values). The method has machine precision accuracy for the height function; degeneracies in the connectivity of the triangle mesh (e.g., non-manifold cases) can create iso-contours with a wrong topology and/or wrong results in the shape of the iso-contours. The computational cost of the service for the extraction of k iso-contours is O(nlogn+kn), where n is the number of input vertices and k is the number of extracted iso-contours. With this low computational cost, the service is capable of processing large data sets. In short the following steps are followed:

- Reorder the function value in increasing order
- Find an edge intersected by the iso-contour
- By ajacency relations, find the next intersected edge
- Repeat the previous step until no edge is intersected by the iso-contour or the surface boundary is reached

As future improvement, we foresee the encoding of the information related to the surface sampling in a neighbor of the iso-contour, which can be visualized using a color-coding or a different width of the contour itself. Preliminary examples are shown below. A possible C++ implementation will be discussed for the next revision of D4.2.1.

4.12.2 Examples, Service #47

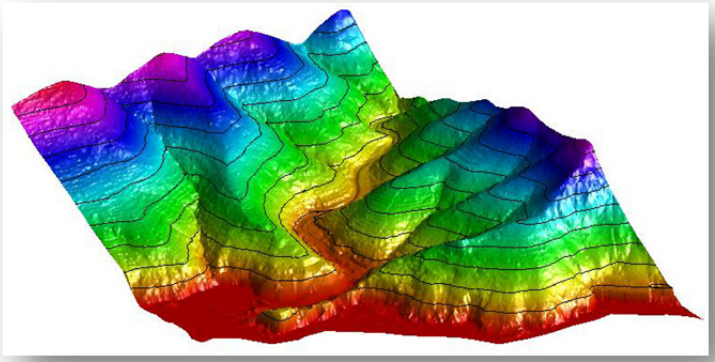
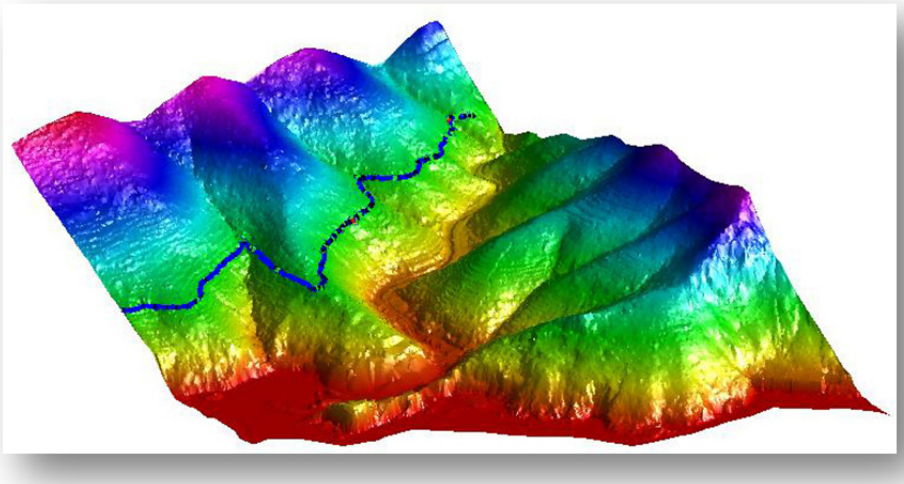


FIGURE 17: ISO-CONTOURS ON THE VERNAZZA MODEL (INPUT: PLY), PROVIDED BY REGIONE LIGURIA.



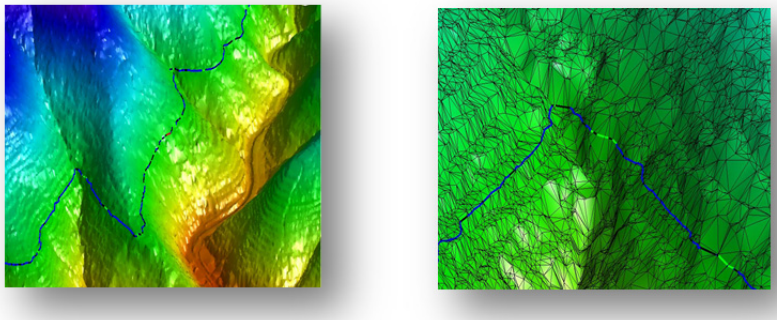


FIGURE 18: COLOR-CODING OF AN ISO-CONTOUR ACCORDING TO THE SAMPLING DENSITY OF THE SURFACE IN A NEIGHBOUR OF THE CONTOUR ITSELF, WHICH VARIES FROM BLUE AND BLACK (HIGHER LOCAL SURFACE SAMPLING) TO GREEN (AVERAGE) AND RED (LOWER LOCAL SURFACE SAMPLING)

4.13 POINT CLOUD COLORING (SERVICE #65)

This method maps the color information in 2D RGB images on 3D point clouds. This service is related to Service #29, but as this service is more specific and fine-tuned for this particular application, it is listed separately here. This method is for example systematically used when acquiring laser mobile mapping data by a car-based system. In this case photographs and 3D point clouds are acquired simultaneously and the photographs are systematically integrated to color the 3D LMMS point cloud. The service table can be found in the appendix section 5.13.

4.13.1 Current algorithm and possible improvements

This point cloud algorithm deliberately focuses on scalability by proposing a process that handles each point independently, without requiring any preliminary surface reconstruction or any more global analysis. Occlusion handling is tackled by sampling preferentially from the image which projection center is closest to the sensor position when the measured point was captured.

1- Read all image metadata
2- Read the volumetric point cloud
3- For each point of the volumetric point cloud :
- For each image metadata
  - if the point projects within the image, compute the lidar sensor position to image projection center distance
  - update the best image id for this point
- Add this point to the set of points colored by its best image

4- For each image with a non-empty point set :
- Read the image data
- For each point colored by this image
  - sample the image at the point projection

5– Write out the color-enriched volumetric point cloud

The proposed approach is very scalable as a single image is open simultaneously (in step 4). To handle large point clouds in a distributed way, the service is able to process only a range of point indices, and output the corresponding color-enriched volumetric point cloud subset. A possibility would be to launch this service on a partition of the point set and merge the resulting point clouds in the end.

Possible future developments:

Streaming implementation: instead of loading the whole point set (or subset if an index range is selected), steps 2,3,4,5 could be implemented in a streaming paradigm enabling the processing of point clouds of arbitrary size in constant memory.

Image metadata scalability: if the number of images is so large that the image metadata itself do not fit into memory (step 1), the same algorithm could be implemented in multiple passes by partitioning also the set of image metadata. This will involve adding an extra output attribute which is the distance between the lidar and the image sensors.

Example, Service #65

The following figure demonstrates the service in a mobile mapping context. The 6 million points have been colored using 355 2Mpix images in 7:10 minutes in a single thread of a standard desktop PC.

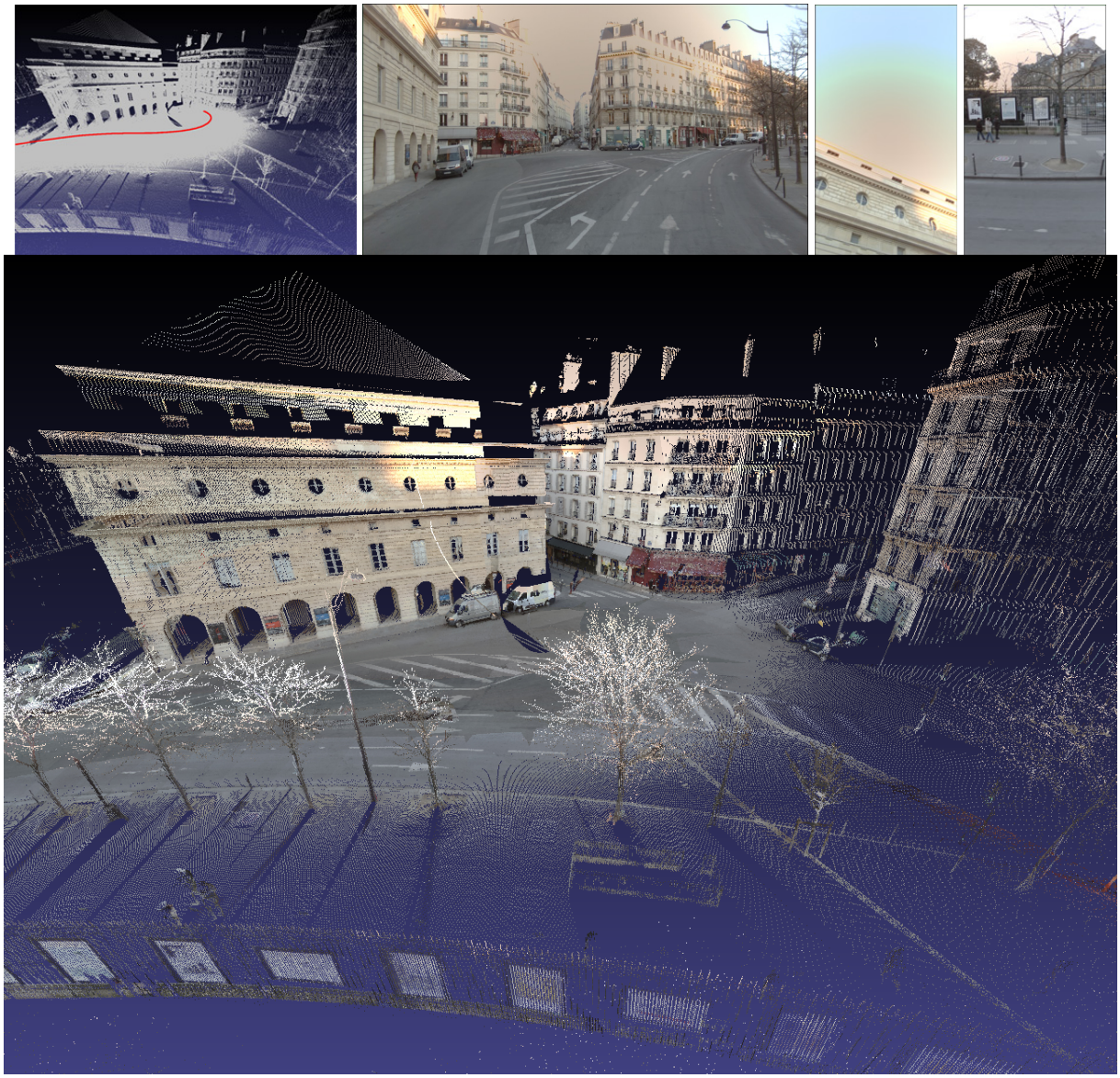


FIGURE 19: (TOP LEFT) INPUT VOLUMETRIC POINT CLOUD OF 6 MILLION POINTS, WITH THE SENSOR TRAJECTORY IN RED. (TOP RIGHT) 3 OUT OF THE 355 IMAGES OF THE CALIBRATED IMAGE SET. (BOTTOM) RESULTING COLOR-ENRICHED VOLUMETRIC POINT CLOUD.

5 APPENDIX: SERVICE TABLES

5.1 SERVICE #6: ICP FINE REGISTRATION

<table>
<tr><th colspan="5">IQmulus Service information: #6</th></tr>
<tr><td>Name of the metadata</td><td colspan="2">Content expected</td><td colspan="2">Motivation/comments</td></tr>
<tr><td>Service Acronym</td><td colspan="2">ICP Fine Registration</td><td colspan="2">Unique identifier of the service; necessary to call it within user-defined workflows</td></tr>
<tr><td>Description</td><td colspan="2">Given two 3D point clouds with appropriate precomputed 3D features, it performs a fine registration between these two point clouds using the Iterative Closest Point algorithm.</td><td colspan="2">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</td></tr>
<tr><td rowspan="4">Service functionality</td><td>Input: <data representation and format></td><td>2 unorganized point clouds in LAS format</td><td rowspan="4">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.</td></tr>
<tr><td>Input parameters:</td><td>(1) Number of neighbors for k-NN search;

(2) Number of neighbors for final quality assessment.</td></tr>
<tr><td>Output: <data representation and format></td><td>Transformation matrix (.mat) and 2 point clouds registered (.LAS)

+ Quality measure based on the mean of distances of nearest points between both point clouds.</td></tr>
<tr><td>Functionality of the service: <text></td><td>Fine pair-wise 3D point cloud registration</td></tr>
<tr><td>Algorithm</td><td colspan="2">An improved version of the ICP algorithm for fine pair-wise registration based 3D multi-scale dimensionality criteria.</td><td colspan="2">The same functionality may in principle be implemented by different algorithms.</td></tr>
</table>

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	(1) The data IO libraries lidarformat: reading and writing the point clouds (2) ANN library for nearest neighbour computation (3) Matrix handling & computation: boost, libann, libeigen	
	Operating system	Coded on Linux, probably multi-platform.	
	Visualization modalities of the output	3D Visualization of the registered point clouds	
IQmulus Data	**Available IQmulus input data**: Unregistered 3D point clouds both for marine and land cases (Liguria #22-#23 and UBO #10-12)	*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:** provided at the end of the process	*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics. See the following box.*	
	Robustness: not robust to highly misaligned point clouds. First requires a coarse registration step.		
	Computational time in relation to data size: Depends on the dataset type (airborne VS terrestrial + point density): ~ 3 minutes for 1M points.		
	Locality/globality of the algorithm: the algorithm applies a global registration between 2 point clouds but for a limited amount of points (<5M). If it is necessary to address the scalability of		

	the service, then the point clouds can be downsampled. If the point clouds have to be splitted to achieve good results, then an additional fusion of transformations should be performed.	
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	#1095 - 1_1__50_UC1 #1093 - 1_1_48_UC1 #1092 - 1_1_47_UC1 #1041 - 1.2.2_SC1_1	*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, Clément Mallet clement.mallet@ign.fr	*Partner ID and responsible person (include email)*
Involved Partners	TU Delft	

5.2 SERVICE #8: POINT CLOUD TO POINT CLOUD DISTANCE

IQmulus Service information: #8			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Point Cloud Distance Computation		*Compute the distance between two points.*
Description	A pair of point cloud data, in ASCII format, is imported automatically by browsing the files path. Then the distance between the two point cloud data is computed. The results will be directly displayed in the popped window. The distance is colorized in scalar colour from red to blue, which denote the range from maximum distance to minimum distance of the two point clouds.		*User can directly know the relative spatial position of the imported point cloud data and, also the mean and standard deviation of the point clouds.*
Service functionality	**Input:**	ASCII format *.xyz point cloud data	*Quickly obtain the spatial distance of two imported distance and the mean and standard deviation of the two point clouds. For registration and combination of different spatial datasets (The algorithm can be used to test the performance of a registration method.*
	Input parameters: *[optional]*	File path of the two point cloud data.	
	Output: <data representation and format>	Dialog based input GUI and automatic popped window display	
	Functionality of the service: <text>	Directly obtain the relative spatial distance of two point clouds and compute the mean and std. of the distance.	
Algorithm	Two point cloud data are imported into the tool, implemented and the internal computations are implemented using C/C++ functions. KNN searching is performed using nanoflann library, which is open source. A popped window which for the results display is implemented by using Visualization Tool Kit (VTK).		*OpenGL and OSG, which are also open source and can also be used to display the computed results.*
Implementation details	**Implementation language**	C/C++	*ANN and FLANN can also be used for the KNN searching.*
	Dependencies with other libraries	Nanoflaan and VTK are needed.	
	Operating system	Windows 64 bits OS, single platform	
	Visualization modalities of the	Visualization of the point cloud distance	

	output	computation results cloud also be exported and displayed in another open source software or tool.	
IQmulus Data	**Available IQmulus input data**: For tests we have used the following data; (1) UBO – Bathymetry in Brittany (ID-12-13) but converted the data in *.xyz format. (2) AHN-2 point cloud data, also in *.xyz file format.		
Service characteristics	**Accuracy:** -		
	Robustness: Reliability is verified by testing several other point cloud data.		
	Computational time in relation to data size: About twenty seconds for approximately 1 million point cloud data on standard PC		
	Locality/globality of the algorithm: The tool could be used to quick obtain the distance of two imported point cloud data.		
Alternatives	Compute the point cloud distance and then exported the results with an extra channel for distance, and then display the results in another tool, such as CloudCompare and QuickTerrain.		
Related use cases	User Story #1037 Land Showcase User Story 2 (1.2.2_SC-2		
Responsible Partner	TUDelft, Jinhu Wang (Jinhu.wang@tudelft.nl)		
Involved Partners	• TUDelft, Roderik Lindenbergh (algorithm development support) • TUDelft, Beril Sirmacek [b.sirmacek@tudelft.nl] testing		

5.3 SERVICE #11: SPATIAL EXTENT

<table>
<tr><th colspan="4">IQmulus Service information: #11</th></tr>
<tr><th>Name of the metadata</th><th colspan="2">Content expected</th><th>Motivation/comments</th></tr>
<tr><td>Service Acronym</td><td colspan="2">Spatial Extent</td><td>Extracting spatial content information and displaying detection result on screen.</td></tr>
<tr><td>Description</td><td colspan="2">Following spatial information is extracted automatically for the input point cloud.

- 3D Bounding box
- Convex hull
- Alpha shape

Currently 3 different .exe files, one for each option</td><td>User can have a quick overview about the spatial content of the input data.</td></tr>
<tr><td rowspan="4">Service functionality</td><td>Input:</td><td>.pcd point cloud file format</td><td rowspan="4">Quick display of the spatial content of the 3D input data.</td></tr>
<tr><td>Input parameters: [optional]</td><td>Path of the folder where the input 3D data is located.</td></tr>
<tr><td>Output: <data representation and format></td><td>.ply polygon files of the borders and screen display.</td></tr>
<tr><td>Functionality of the service: <text></td><td>Quick calculation and display of spatial content of the input point cloud file.</td></tr>
<tr><td>Algorithm</td><td colspan="2">Reading point cloud data using Point Cloud Library (PCL) functions.

Creating 3D display screen by using Visualization Kit (VTK) library and adding data rendering features.

Extracting spatial content of the point cloud using PCL functions.

Visualization of the spatial content on the original point cloud.

Saving the spatial content into a polygon file.</td><td>PCL and OpenGL also provides functions for 3D display.</td></tr>
<tr><td rowspan="4">Implementation details</td><td>Implementation language</td><td>C++</td><td rowspan="4">It is also possible to display data and its spatial content by using commercial software like QuickTerrain, MeshLab, CloudCompare, etc.</td></tr>
<tr><td>Dependencies with other libraries</td><td>PCL, VTK, Boost Libraries are needed.</td></tr>
<tr><td>Operating system</td><td>Windows, single platform</td></tr>
<tr><td>Visualization</td><td>Visualization and rendering</td></tr>
</table>

	modalities of the output	of the point cloud. Calculation and visualization of the spatial content of the input point cloud.	
IQmulus Data	**Available IQmulus input data**: For tests we have used the following data; (3) UBO – TLS Topography of a small beach in Brittany (ID-10-11) (4) UBO – Bathymetry in Brittany (ID-12-13) (5) Linguria – Lidar data for the study area of Vernazza, Liguria, Italy (ID-22-23)		*eRoom id's are provided.*
Service characteristics	**Accuracy:** -		
	Robustness: reliable to work with large point clouds		
	Computational time in relation to data size: * 2D Alpha Shape: < 5 seconds for point cloud with ~600000 points. 3D Bounding Box: : ~8 seconds for same point cloud 2D Convex Hull: ~5 seconds for same point cloud All on standard PC		
	Locality/globality of the algorithm: the program can be used to see spatial content of the input data or the processed data quickly.		
Alternatives	Using 3D display functions from other libraries like PCL, ANN, CGAL and OpenGL at the code generation. Another option is using commercial software packages for point cloud display; i.e. QuickTerrain, MeshLab, CloudCompare.		
Related use cases	Each user story in the redmine list can benefit from quick 3D visualization application for displaying the input or the output data with their spatial content.		
Responsible Partner	TUDelft, Beril Sirmacek [b.sirmacek@tudelft.nl]		
Involved Partners	• TUDelft, Roderik Lindenbergh (algorithm development support) • TUDelft, Jinhu Wang (testing the developed algorithms)		

5.4 SERVICE #12 DATASET INTERSECTION

IQmulus Service information: #12			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Point Cloud Intersection region computation		*Compute intersected area from the two imported point cloud datasets.*
Description	A pair of point cloud data, in ASCII format, is imported automatically by browsing the files path. Then the distance between the two point clouds data is computed. If the two point cloud have intersected region, the intersected region will displayed in blue colour since it has the minimum distance of the two point cloud datasets.		*Users can directly know if the two imported point cloud data have intersection region and where the intersection is.*
Service functionality	**Input:**	ASCII format *.xyz point cloud data	*Quickly acquire the intersected area of two imported point cloud data.*
	Input parameters: *[optional]*	File path of the two point cloud data.	
	Output: <data representation and format>	Dialog based input GUI and automatic popped window display	
	Functionality of the service: <text>	Acquire the intersected area of the two imported point cloud data.	
Algorithm	A pair of point cloud data, in *.xyz format are imported into the tool, all the implementation and the internal computations are in C/C++ function. KNN searching is performed using nanoflann library, which is open source. A popped window which for the results display is implemented by using Visualization Tool Kit (VTK) .		*OpenGL and OSG, which are also open source and can also be used to display the computed results.*
Implementation details	**Implementation language**	C/C++	*ANN and FLANN can also be used for the KNN searching.*
	Dependencies with other libraries	Nanoflaan and VTK are needed.	
	Operating system	Windows 64 bits OS, single platform	
	Visualization modalities of the	Points from the intersected area could	

	output	also be outputted and displayed in a new window. This could be useful when editing the point cloud data.	
IQmulus Data	**Available IQmulus input data**: For tests we have used the following data; (6) UBO – Bathymetry in Brittany (ID-12-13) but converted the data in *.xyz format. (7) AHN-2 data. Also in *.xyz format.		
Service characteristics	**Accuracy:** -		
	Robustness: Reliability is verified by testing several other point cloud data.		
	Computational time in relation to data size: ~20 seconds for 2 x 1 million points		
	Locality/globality of the algorithm: The program can be used to quickly determine if the point cloud data have an intersection area, if they have, obtain the intersected region.		
Alternatives	Perform the internal computation and then output the intersected area and display in other point cloud visualization software, such as openmesh, open flipper, and QuickTerrain.		
Related use cases	User Story: #1037 Land Showcase User Story 2 (1.2.2_SC2_2)		
Responsible Partner	TUDelft, Jinhu Wang (Jinhu.wang@tudelft.nl)		
Involved Partners	• TUDelft, Roderik Lindenbergh (algorithm development support) • TUDelft, Beril Sirmacek [b.sirmacek@tudelft.nl] testing.		

5.5 SERVICE #18: MATCHING 3D KEYPOINT DESCRIPTOR VECTORS

IQmulus Service information: #18			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Matching 3D keypoint descriptor vectors		*Finding matching couples where there are two local feature sets with their descriptor vectors.*
Description	The 3D point clouds, their key features and the descriptor vectors are read from the folder path which is provided by the user. Best matching descriptor vectors are found and a list of matching pair indices is kept. Point clouds and the matching key features are displayed on the screen in 3D space which enables rendering, zooming like mouse actions.		
Service functionality	**Input:**	.pcd point cloud file format .txt file (x,y,z) local feature coordinates and a descriptor vector for each local feature.	*Very large point clouds can be displayed after downsampling in order to increase the speed of zooming and rendering.*
	Input parameters: *[optional]*	Path of the folder where the input point clouds and their key feature positions with descriptor vectors are located in the computer.	
	Output: <data representation and format>	.txt file keeping a list which holds the indices of the matching key features. Screen display of the point clouds and the matching local key features.	
	Functionality of the service: <text>	Point clouds and the matching key features are displayed.	
Algorithm	Reading point cloud data using Point Cloud Library (PCL) functions. We plan to write our own mathematical vector distance calculation functions to select the descriptor vectors which have the minimum distance (the highest similarity) to each other. Creating 3D display screen by using Visualization Kit (VTK) library and adding data rendering features.		*RANSAC algorithm can be implemented to get rid of the false matches.* *PCL and OpenGL also provides functions for 3D display.*

	Displaying the matching key feature couples on point clouds.		
Implementation details	**Implementation language**	C++	
	Dependencies with other libraries	PCL, VTK, Boost Libraries are needed.	
	Operating system	Windows, single platform	
	Visualization modalities of the output	Visualization and rendering of the point cloud with matching local key features.	
IQmulus Data	**Available IQmulus input data**: For tests we have used the following data; (8) UBO – TLS Topography of a small beach in Brittany (ID-10-11) (9) UBO – Bathymetry in Brittany (ID-12-13) (10) Linguria – Lidar data for the study area of Vernazza, Liguria, Italy (ID-22-23)		*eRoom id's are provided.*
Service characteristics	**Accuracy:** Tests will be done.		
	Robustness: reliable to work with large point clouds		
	Computational time in relation to data size: == expected to be less than a minute for point cloud data with ~600000 points.		
	Locality/globality of the algorithm: the program can be used with object detection, registration, fusion applications.		
Alternatives	-		
Related use cases	User stoy (IQmulus) #1037 Land Showcase User Story 2 (1.2.2_SC2_2) For registration and combination of different spatial datasets.		
Responsible Partner	TUDelft, Beril Sirmacek [b.sirmacek@tudelft.nl]		
Involved Partners	• TUDelft, Roderik Lindenbergh (algorithm development support) • TUDelft, Jinhu Wang (testing the developed algorithms)		

5.6 SERVICE #23: GEOTIFF TO LAS

IQmulus Service information: #23			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	GeoTIFF to LAS		*Reading the image or the point cloud and converting the data into another format which is asked by the user.*
Description	The data is read from the folder path which is provided by the user. Data is automatically displayed on the screen and if it is 3D then rendering, zooming like mouse actions are considered for visualization of the different views and the details. The target file format is chosen by the user. The data is saved on the computer harddisk in the chosen file format.		*User can apply quick file format conversion.*
Service functionality	**Input:**	.pcd point cloud file format	*Very large point cloud data can be displayed after downsampling in order to increase the speed of zooming and rendering.*
	Input parameters: *[optional]*	Path of the folder where the input 3D data is located. Name of the target file format	
	Output: <data representation and format>	Screen display Point cloud or 2D image in the format which is chosen by the user	
	Functionality of the service: <text>	Quick visualization of the 3D data.	
Algorithm	Reading point cloud data using Point Cloud Library (PCL) functions. Creating 2D or 3D display screen using Visualization Kit (VTK) library and if data is 3D adding data rendering features. Reading the user preference for the target file format. Saving the data into a new file in the chosen format.		*OpenCV, PCL and OpenGL also provides functions for 2D and 3D display.*
Implementation details	**Implementation language**	C++	
	Dependencies with	Liblas, GDAL, OpenCV, PCL,	

	other libraries	VTK, Boost Libraries are needed.	
	Operating system	Windows, single platform	
	Visualization modalities of the output	Visualization and rendering of the point cloud.	
IQmulus Data	**Available IQmulus input data**: For tests we have used the following data; (11) UBO – TLS Topography of a small beach in Brittany (ID-10-11) (12) UBO – Bathymetry in Brittany (ID-12-13) (13) Linguria – Lidar data for the study area of Vernazza, Liguria, Italy (ID-22-23)		*eRoom id's are provided.*
Service characteristics	**Accuracy:** -		
	Robustness: reliable to work with large point clouds		
	Computational time in relation to data size: ~3 seconds for 600000 points		
	Locality/globality of the algorithm: Currently the program can convert xyz, ascii and las data formats to pcd format. We plan to implement more file format conversion options.		
Alternatives	It is also possible to display data and apply file format conversion using commercial software like QuickTerrain, MeshLab, CloudCompare, etc.		
Related use cases	Each user story in the redmine list can benefit from file format conversion.		
Responsible Partner	TUDelft, Beril Sirmacek [b.sirmacek@tudelft.nl]		
Involved Partners	• TUDelft, Roderik Lindenbergh (algorithm development support) • TUDelft, Jinhu Wang (testing the developed algorithms)		

5.7 SERVICE #29: REGISTERING 2D IMAGES ON POINT CLOUDS

IQmulus Service information: #29			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Registering 2D images on point clouds		*Mapping texture on point cloud*
Description	Point cloud and the 2D image are read from the folder paths which are provided by the user. The key features and descriptor vectors are extracted from the point cloud and the 2D image. After finding the list of matching key feature point locations, the transformation matrix is found to register 2D image on the mesh face of the point cloud. Finally, point cloud mesh surface is covered with texture which is the content of 2D image.		
Service functionality	**Input:**	.pcd point cloud file format .jpg 2D input image with EXIF file content which includes the GPS position and the orientation information.	Adding textures on point cloud by using smartphone images for detailed representation of the urban structures.
	Input parameters: *[optional]*	Path of the folder where the input point cloud and the 2D image are located.	
	Output: <data representation and format>	Screen display of the textured point cloud mesh surface. List of matching keypoints and their locations. .pcd format textured mesh file.	
	Functionality of the service: <text>	Adding textures on point cloud by using smartphone images.	
Algorithm	Algorithm first finds the interested point space in the point cloud by reading the GPS location and the orientation of the smartphone image. Looking angle and the point cloud of the building façade is corrected by using coordinate transformation. Local key features are extracted from the point cloud and the 2D smartphone image. Matching key features are listed by using descriptor		*We will work on selection of different local feature extraction methods depending on the input data (marine, urban, dunes, mountains, etc.)*

	vector matching. Transformation and rotation matrix is generated by using the list of the matching features. Mesh data is generated from the interest points and the registered 2D smartphone image is used as a texture on it.		
Implementation details	**Implementation language**	C++	
	Dependencies with other libraries	PCL, VTK, FAST, OpenGL, OpenCV, Boost Libraries are needed.	
	Operating system	Windows, single platform	
	Visualization modalities of the output	Visualization of the 3D textured mesh surface is provided automatically.	
IQmulus Data	**Available IQmulus input data**: For tests we have used the following data; (14) UBO – TLS Topography of a small beach in Brittany (ID-10-11) (15) UBO – Bathymetry in Brittany (ID-12-13) (16) Linguria – Lidar data for the study area of Vernazza, Liguria, Italy (ID-22-23) *Tested on internal AHN + IPhone data*		*eRoom id's are provided.*
Service characteristics	**Accuracy:** Inaccuracy of smartphone GPS location and orientation information causes a small shift on the actual looking angle to the urban structures of the interest. That causes slight error on the registration results.		
	Robustness: reliable to work with large point clouds		
	Computational time in relation to data size: <30 seconds for one iPhone image and one point cloud with ~45000 points.		
	Locality/globality of the algorithm: the program can be implemented as a smartphone application which enables any end user to upload photos and register the most recent view of the area on 3D models.		
Alternatives	Commercial software like (QuickTerrain) can help to register the texture on 3D models if matching tie points are provided.		

Related use cases	User Story (IQmulus) #1037, Land Showcase User Story 2 (1.2.2_SC2_2); User can register up-to-date scene on to the 3D models.	
Responsible Partner	TUDelft, Beril Sirmacek [b.sirmacek@tudelft.nl]	
Involved Partners	• TUDelft, Roderik Lindenbergh (algorithm development support) • TUDelft, Jinhu Wang (testing the developed algorithms)	

5.8 SERVICE #31: POINT CLOUD GENERATION FROM MULTI-ANGLE IMAGES

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Point cloud generation from multi-angle images		*Generating terrestrial view point clouds by using multiple multi-angle images acquired by smartphones.*
Description	Point cloud of the urban structures can be generated easily by smartphones of the users in order to update the 3D archive data.		*User can generate the point cloud of an urban structure easily by using any smartphone sensor.*
Service functionality	**Input:**	Smartphone image data set (including ~25-35 images) in .JPEG format and with EXIF file tags.	
	Input parameters: *[optional]*	Path of the folder where the image data set is located.	
	Output: <data representation and format>	Point cloud in .pcd file format (can be converted to .las format) Screen display of the generated point cloud	
	Functionality of the service: <text>	Generation and quick visualization of the urban structure point clouds with RGB color information.	
Algorithm	- Read input iPhone images and their EXIF file tags using OpenCV library - Apply SIFT keypoint extraction and keypoint matching - Apply RANSAC algorithm to remove the false matches - Find the extrinsic camera parameters - Calculate 3D locations of the points using bundler and structure from motion (SFM) libraries - Use VTK library to display the generated		*OpenGL also provides functions for this 3D display.*

	point cloud		
Implementation details	**Implementation language**	C++	
	Dependencies with other libraries	PCL, VTK, Boost, OpenCV, Bundler, SFM Libraries are needed.	
	Operating system	Windows 7	
	Visualization modalities of the output	Visualization and rendering of the urban structure point cloud with RGB color information.	
IQmulus Data	**Available IQmulus input data**: For tests we have used multi-angle iPhone image data set of a windmill, the old municipality building and the new church of the Delft city in the Netherlands.		*More tests will be added for the regions which have already air-/spaceborne point clouds in the eRoom.*
Service characteristics	**Accuracy:** Tests will be done by comparing the generated point clouds with terrestrial laser scan point clouds.		
	Robustness: reliable to generate urban structure point clouds rapidly.		
	Computational time in relation to data size: ~5 minutes for the iPhone image data set which includes 49 images.		
	Locality/globality of the algorithm: the program can be used to generate terrestrial 3D information of the urban structures quickly without having need of expensive and bulky sensor devices.		
Alternatives	Another option is using commercial software packages for point cloud generation; i.e. Pix4D, CloudCompare.		
Related use cases	Each user story in the redmine list can benefit from quick 3D visualization application for generating new terrestrial point clouds of the region of interest. Generated new point clouds can be used for updating the archive point cloud data and for displaying the specific locations and amounts of the changes to the user.		
Responsible Partner	TUDelft, Beril Sirmacek [b.sirmacek@tudelft.nl]		
Involved Partners	• TUDelft, Roderik Lindenbergh (algorithm development support) • TUDelft, Jinhu Wang (testing the developed algorithms)		

5.9 SERVICE #35: RE-SAMPLING OF POINT CLOUDS

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	35		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Resample the input point cloud using a predefined resolution		*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>**	Unordered point cloud, format: LAS	*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]*	Grid size (resolution)	
	Output: <data representation and format>	Gridded point cloud, format: LAS; 2.5D raster, format: GeoTIFF	
	Functionality of the service: <text>	Resample the input point cloud	
Algorithm	1. Projecting 3D points to plane Z=0. 2. Create 2D grids with the predefined resolution to cover the projected points on x-y plane. 3. In each grid, the centroid of points presented is computed. By this way, a resampled point cloud is generated.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	Java, C++, Bash	*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	Hadoop, libLAS	
	Operating system	Unix/Linux	
	Visualization modalities of the	pcl_viewer, meshlab	

	output		
IQmulus Data	**Available IQmulus input data**: <ID of the dataset as in the eRoom tables> 23, 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:** N/A **Robustness:** Robust **Computational time in relation to data size:** O(n) (the size of new point cloud is supposed to be smaller) **Locality/globality of the algorithm:** Native Hadoop implementation under MapReduce framework		*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 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	<Use cases ID as in RedMine>		*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	Ziyi Jiang (ziyi.jiang@ucl.ac.uk) Kun Liu (kun.liu@ucl.ac.uk)		*Partner ID and responsible person (include email)*
Involved Partners	Other Partners and responsible persons (if any, emails) involved in the development and/or testing of the service		

5.10 SERVICE #36: SUB-SAMPLING OF POINT CLOUDS

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Sub-sampling of point clouds		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Reduce the size of the input point cloud by extracting a subset from it		*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>**	Unordered point cloud, format: LAS	*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]*	Grid size (resolution)	
	Output: <data representation and format>	Gridded point cloud, format: LAS; 2.5D raster, format: GeoTIFF	
	Functionality of the service: <text>	Reduce the size of the input point cloud	
Algorithm	1. Projecting 3D points to plane Z=0. 2. Create 2D grids with the predefined resolution to cover the projected points on x-y plane. 3. In each grid, a point is randomly selected.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	Java, C++, Bash	*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	Hadoop, libLAS	
	Operating system	Unix/Linux	
	Visualization modalities of the output	pcl_viewer, meshlab	

IQmulus Data	**Available IQmulus input data**: <ID of the dataset as in the eRoom tables> 23, 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:** N/A **Robustness:** Robust **Computational time in relation to data size:** O(n) (the size of new point cloud is supposed to be smaller) **Locality/globality of the algorithm:** Native Hadoop implementation under MapReduce framework	*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 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	<Use cases ID as in RedMine>	*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	Ziyi Jiang (ziyi.jiang@ucl.ac.uk) Kun Liu (kun.liu@ucl.ac.uk)	*Partner ID and responsible person (include email)*
Involved Partners	Other Partners and responsible persons (if any, emails) involved in the development and/or testing of the service	

5.11 SERVICE #46: ISOSURFACE EXTRACTION

IQmulus Service information, Service #46			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	IsoSurfaceFromVolumeGrid		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Isosurface extraction from volumetric scalar fields through a variation of the Marching Cubes method, which preserves sharp features and maintains consistent topology of the extracted iso-surfaces.		*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>**	Matrix: scalar values at the node of indices (I,j,k)	*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>	Extraction of a triangle mesh from a volume grid	
Algorithm	Adaptive variation of the Marching Cubes algorithm		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	Matlab, 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	Windows, Linux, OS	
	Visualization modalities of the output	Rendering of triangle meshes	
IQmulus Data	**Available IQmulus input data**:		*If there are examples*

	Volume data extracted/extrapolated from 24-28 (Regione Liguria), through interpolation/approximation methods.	*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 **Robustness:** machine precision **Computational time in relation to data size:** == **Locality/globality of the algorithm:** the algorithm applies a local refinement of the volumetric grid. Analogously to the Marching Cubes, this service is capable of processing large data set if the local refinement is not too heavy. If it is necessary to address the scalability of the service, then data partitioning can be addressed in a quite simple way as a matter of the regularity of the input grid.	*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	• User story (IQmulus) #1105 • User story (IQmulus) #1106 • Use case (IQmulus) #1107 • User story (IQmulus) #1075 • User story (IQmulus) #1076	*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, Giuseppe Patanè [patane@ge.imati.cnr.it]	*Partner ID and responsible person (include email)*
Involved Partners	• Regione Liguria (analysis of rainfall data) • SINTEF (for approximation schemes)	

5.12 SERVICE #47: ISOCONTOUR EXTRACTION

IQmulus Service information: #47			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	IsocontourFromTIN		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	The algorithm will extract a list of iso-contours from a triangle mesh that represents a digital elevation model. Linear interpolation of the height values at the mesh vertices will be used to trace the iso-contours. Possible adaptation to uniform grid will be taken into account.		*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>**	Triangle mesh (OFF)	*Include here the input/output of the service, the name of the functionality (e.g., registration, fusion), and input parameters (if any). In the future, we may consider defining a taxonomy of the functionalities to harmonize the terminology used to fill this field.*
	Input parameters: *[optional]*	• Number of iso-values or • List of iso-values If input parameters are not provided, then pre-defined values will be used (uniform sampling of the iso-values).	
	Output: <data representation and format>	List of 1D curves (ShapeFile)	
	Functionality of the service: <text>	Iso-contour extraction	
Algorithm	Iso-contour extraction with linear interpolation of the height values.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	Matlab, 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*
	Dependencies with other libraries	==	
	Operating system	Windows, Linux, single platform	
	Visualization modalities of the	• Visualization of the curves and input TIN • Visualization of the	

	output	digital elevation model as a triangle mesh (optional) • Visualization of the colour map associated to the height values (optional)	*of the output.*
IQmulus Data	**Available IQmulus input data**: • 7 (FOMI) • 22, 23 (RegioneLiguria) after triangulation • 8, 9, 12, 13 (UBO)		*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:** machine precision for the height function **Robustness:** Degeneracies in the connectivity of the triangle mesh (e.g., non-manifold cases) can create iso-contours with a wrong topology and/or wrong results in the shape of the iso-contours. **Computational time in relation to data size:** The computational cost of the service for the extraction of the iso-contours is O(nlogn+kn), where n is the number of input vertices and k is the number of extracted iso-contours. **Locality/globality of the algorithm:** • Since the computational cosy of the service is O(nlogn+kn), the service is already capable of processing large data sets. • if data partitioning will be necessary to address the scalability of the service, iso-contours on adjacent tiles will be glued together.		*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	Service 33 (TUDelft)		*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	• User story (IQmulus) #1076 • User story (IQmulus) #1069		*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, Giuseppe Patanè, patane@ge.imati.cnr.it		*Partner ID and responsible person (include email)*

Involved Partners	Possible tests on data sets provided by Regione Liguria, UBO, and FOMI.	

5.13 SERVICE #65: POINT CLOUD COLORING

IQmulus Service information: #65		
Name of the metadata	**Content expected**	**Motivation/comments**
Service Acronym	Point Cloud Coloring	*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	Given a volumetric point cloud and a collection of georeferenced and calibrated images, enrich the point cloud by adding a color attribute to each 3D point.	*Brief textual description of the service: what it provides, what can be used for. This text could be used as a short "help text" in the User Interfaces of IQmulus*
Service functionality	**Input: <data representation and format>**	- a volumetric point cloud (ply) - a collection of 2D multi-channel images (unconstrained georeferencing) (tiff+xml)
	Input parameters:	none
	Output: <data representation and format>	- the initial volumetric point cloud enriched with a new per-point color attribute (ply)
	Functionality of the service: <text>	Volumetric Point Cloud Coloring
Algorithm	An improved version of the ICP algorithm for fine pair-wise registration based 3D multi-scale dimensionality criteria.	*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	C++
	Dependencies with other libraries	(1) The data IO libraries (to be defined, as they depend on the actual supported data types, lidarformat for the moment): reading and writing the point clouds (2) libTIFF for reading the images (3) TinyXML for reading the XML-encoded image georeferencing and calibration
	Operating system	Coded on Linux, probably multi-platform.
	Visualization modalities of the output	3D Visualization of the colored point clouds
IQmulus Data	**Available IQmulus input data**: Corresponding dataset is not uploaded 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:** depends on the input data registration 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: not robust to geometric and radiometric misregistration, not robust to diachronism or dynamic scenes.	
	Computational time in relation to data size: O(#images x #points)	

	Locality/globality of the algorithm: Data partitioning on the point cloud is trivial (each point is processed independently). Further data partitioning on the image set is feasible if really needed.	
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	This service does not address a particular user story but may be used in 2 cases : – point cloud visualization is more intuitive if points are represented with their visible color. – Fusing image colors within a point cloud adds an extra attribute that may be used for further processing (classification, segmentation, registration...)	*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, Mathieu Brédif mathieu.bredif@ign.fr	*Partner ID and responsible person (include email)*
Involved Partners	IGN	