Testbed-12 LiDAR Streaming Engineering Report

Table of Contents

Publication Date: 2017-03-09

Approval Date: 2016-12-07

Posted Date: 2016-10-05

Reference number of this document: OGC 16-034

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A071

Category: Public Engineering Report

Editors: Simon Jirka, Arne de Wall, Christoph Stasch

Title: **Testbed-12 LiDAR Streaming Engineering Report**

OGC Engineering Report

COPYRIGHT

WARNING

LICENSE AGREEMENT

that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license

without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Abstract

This Engineering Report describes how developments of the Community Sensor Model Working Group (CSMW) can be harmonized with the latest SWE specifications and developments in order to support streaming of LiDAR data with SWE technologies. The report will therefore provide an overview on both initiatives and then describe different options how to integrate LiDAR data streams and SWE technologies. In particular, the ER will consider the results of the activities SOS Compression (LiDAR) Server (A012) and LiDAR Streaming Client (A010) and infer recommendations for future developments.

Business Value

This ER offers providers of LiDAR data guidance how their data can be shared within Sensor Web infrastructures (e.g. for delivering LiDAR data efficiently to their customers). Furthermore, it describes an approach how application developers can make use of streaming-based provision of LiDAR data to develop a broad range of applications that benefit from incremental data loading mechanisms.

What does this ER mean for the Working Group and OGC in general

The Testbed-12 LiDAR Streaming ER will describe the provision and handling of LiDAR data using the OGC Sensor Web Enablement (SWE) standards. As the Point Cloud DWG is intended as a forum for the discussion and presentation of pilots and implementations of OGC standards in the context of point cloud data, this ER would fit well into the scope of this DWG. On the one hand the editors hope to receive feedback and suggestions from the DWG members and on the other hand the ER is intended to contribute to the discussion by summarizing the Point Cloud related activities of Testbed-12.

How does this ER relate to the work of the Working Group

The Point Cloud DWG is dealing with the handling of point cloud data within the framework of OGC standards. Thus, the SOS-based streaming approch described in this ER is a data delivery mechanim which is relevant to the scope of this DWG.

Keywords

ogcdocs, testbed-12, SWE, SOS, streaming, point clouds, LiDAR

Proposed OGC Working Group for Review and Approval

This engineering report shall be submitted to the Point Cloud DWG for review

and comment.

Chapter 1. Introduction

1.1. Scope

This OGC® document gives guidelines how the OGC Sensor Observation Service 2.0 interface can be used for enabling a JPIP-like streaming of point cloud data. It summarizes the findings gathered during the implementation of a LiDAR Streaming SOS server during the OGC Testbed-12.

1.2. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Table 1. Contacts

Name	Organization
Simon Jirka	52°North GmbH
Arne de Wall	52°North GmbH
Christoph Stasch	52°North GmbH
Charles Heazel	WiSC Enterprises

1.3. Future Work

Based on the findings during the development of the LiDAR streaming SOS, there are several work items we recommend for the future. These comprise:

- Allow further response types for the GetResult operation of the SOS interface: At the moment the returned point cloud data chunks are transmitted as binary payload within an XML GetResult respose. This required additional encoding/decoding and reduces performance. Thus, a recommendation is to investigate how binary responses of the SOS can be enabled.
- Provide further guidance on metadata provision for LiDAR data: During Testbed-12 the focus was on enabling a JPIP-like streaming of point cloud data through the SOS interface. As a next step we would recommend to develop a more comprehensive guidance how relevant metadata could be transmitted based on SensorML.
- Security/Access control: The question how to enable security during the SOS-based streaming of LiDAR data was not within the focus of Testbed-12. However, in the future it would be valuable to investigate how the proposed access workflow (based on the GetResult operation) could be secured.
- Due to the time and budget constraints of the project, there was only limited time for optimizing the output of the SOS. Thus, in future activities we suggest to further analyse different compression mechanims for more efficient data delivery.

1.4. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC 06-121r9, OGC® Web Service Common Implementation Specification 2.0.0
- OGC 08-094r1, OGC® SWE Common Data Model Encoding Standard 2.0
- OGC 09-001, OGC® SWE Service Model Implementation Standard 2.0
- OGC 10-004r3, OGC® Abstract Specification, Topic 20: Observations and Measurements 2.0
- OGC 10-025r1, Observations and Measurements - XML Implementation 2.0
- OGC 12-000, OGC® SensorML: Model and XML Encoding Standard 2.0
- OGC 12-006, OGC® Sensor Observation Service Interface Standard 2.0

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9] and in OGC® Abstract Specification Topic 20: Observations and Measurements 2.0 shall apply. In addition, the following terms and definitions apply.

3.1. LAS

```
LAS is a data format for LIDAR (or other) point cloud data.  It is a data format which
is independent from specific LiDAR hardware and software.
```

3.2. LAZ

```
LAZ files are a compressed version of LAS files.
```

3.3. Point cloud

```
A point cloud is a set of data points in a coordinate system. In this ER, three-
dimensional coordinate systems are considered. Point clouds are often generated by 3D
scanners.
```

3.4. Sensor Web

```
Sensor Web refers to the integration of observation data (e.g. point clouds generated
by 3D scanners) into Web-based infrastructures. The OGC Sensor Web Enablement
standards are intended to build interoperable Sensor Web infrastructures.
```

Chapter 4. Conventions

4.1. Abbreviated terms

- ASCII American Standard Code for Information Interchange
- ASPRS American Society of Photogrammetry and Remote Sensing
- DWG Domain Working Group
- ER Engineering Report
- HDF Hierarchical Data Format
- JPIP JPEG 2000 Interactive Protocol
- LiDAR Light detection and ranging
- O&M Observations and Measurements
- OGC Open Geospatial Consortium
- SensorML Sensor Model Language
- SIPC Sensor Independent Point Cloud
- SWE Sensor Web Enablement
- SOS Sensor Observation Service
- WKB Well-known-Binary

Chapter 5. Overview

This ER summarizes the findings of the LiDAR SOS client and server developments to enable the streaming of point cloud data. It comprises the following clauses which introduce different aspects of the LiDAR streaming SOS work:

- Relevant Standards: This section introduces the relevant standards used in the development of the LiDAR streaming approach based on the SOS.
- Requirements: In this clause, the requirements that influenced the LiDAR streaming SOS developments are introduced.
- Encoding: Here, an overview of the used encodings is shown.
- SOS Operations: This clause describes the workflow and necessary SOS operations to enable a streaming access to LiDAR data via the SOS interface.
- Server: This section describes the realisation of the LiDAR streaming SOS.
- Client: Complementary to the server section, this clause explains the corresponding client developments.
- Recommendations: In this section, the reader finds recommendations derived from the experiences gained during the LiDAR SOS developments.

Chapter 6. Relevant Standards

This section describes the standards and specifications that were used for building the LiDAR Streaming SOS client and server

6.1. OGC Sensor Web Enablement (SWE) Standards

The OGC SWE standards comprise several specifications to allow the integration of sensors and their measurements into spatial data infrastructures. This comprises both, encodings for metadata and observation data as well as Web service interfaces for functionality such as downloading observation data, event subscriptions as well as controlling/tasking sensors. Within the context of streaming access to LiDAR data, three standards are relevant, which are introduced in the following sub-sections:

6.1.1. OGC Sensor Observation Service

The OGC Sensor Observation Service is an interface specification of the OGC for enabling the interoperable access to observation data as well as associated metadata about the processes which have generated the offered observation data (e.g. sensor metadata). Especially the following operations are considered for this ER:

- GetCapabilities: Access a self-description of a SOS server (e.g. supported operations, available data).
- GetObservation: Query and download observation data based on a broad range of filters (e.g. observed properties, temporal filters, spatial filters).
- DescribeSensor: Retrieve metadata about a process (e.g. sensor) which has generated data offered by a SOS server.
- GetFeatureOfInterest: Retrieve the geospatial objects, to which certain observations are associated.
- GetResultTemplate: Often, a client intends to retrieve data for the same sensor, observed property, etc. with only a few changing parameters. To avoid the repetitive download of metadata that is common to multiple requests of the same style, it is possible to download such common metadata as a result template.
- GetResult: Retrieve observation data in an efficient and compact way, encoded according to a previously retrieved result template.

6.1.2. OGC Observations and Measurements (O&M)

While the SOS interface deals with the functionality for accessing observation data, it is also necessary to define how these observation data sets shall be modelled and encoded. The provision of such a common data model and encoding for observation data is the purpose of O&M. For example O&M defines important properties such as:

- Different time stamps
- Result (the observed value including unit of measurement (if applicable))

- Information about the property that is observed
- References to the associated feature and procedure (sensor)

6.1.3. OGC Sensor Model Language (SensorML)

To ensure the correct interpretation of observation data, it is important to provide metadata about the processes which have generated the data. A data model and (XML) encoding for such metadata is provided by the OGC SensorML standard. This standard offers a wide range of different metadata properties to describe and characterize sensing processes:

- Identifiers (e.g. serial numbers)
- Characteristics (e.g. size, weight) and capabilities (e.g. resolution, precision) of a sensor
- Calibration information
- Documentation (e.g. links to users guides)
- Contact information (e.g. operator of a sensor)
- Inputs and outputs
- Etc.

6.2. LAS/LAZ

LAS is a public, binary file format for the interchange of 3D point cloud data. It is one of the most widely used approaches for encoding and transferring point cloud data. LAS is available as an open specification which is published, maintained and copyrighted by the American Society of Photogrammetry and Remote Sensing (ASPRS) [3]. In addition, a compressed version of LAS is available as the LAZ format.

Chapter 7. Requirements

This section describes the requirements that need to be covered by the LiDAR Streaming SOS implementations.

Testbed-12 shall determine the optimal method to deliver LiDAR data through an OGC Sensor Observation Service. The LiDAR SOS shall provide a streaming mechanism, similar to JPIP. This work is divided into three deliverables:

- LiDAR Streaming ER: Describe how streaming of LiDAR data can be realized using the OGC Sensor Web Enablement suite of standards
- SOS server supported for streaming LiDAR data
- LiDAR streaming client that interacts with the streaming SOS server

7.1. Objectives

The development described in this ER addresses two main objectives:

7.1.1. Objective 1: Development of a SOS-based protocol for streaming LiDAR data

This objective comprises the development of an approach describing how the operations of the OGC SOS interface can be used for enabling JPIP-like streaming of LiDAR data. This includes, for example, mechanisms for loading data of a LiDAR scene in different resolutions (first coarse resolution for a quick overview and subsequently more detailed data for specific areas, e.g. when zooming in). On the other hand, there is a need for an efficient encoding to transfer Point Cloud data and the associated metadata from the SOS Streaming server to a client.

7.1.2. Objective 2: Development of client applications for consuming the LiDAR data streams

For this objective there are two application scenarios to consider:

- Visualisation of LiDAR scenes in a viewer: In this case a viewer may benefit from streaming based data loading so that a first view of the scene can be shown quickly while subsequent (more detailed) elements of the scene can be loaded on demand to increase the resolution.
- Applications performing further processing/analysis with the LiDAR data: In this case, the interoperable access to subsets of a scene would be the main focus because an algorithm (e.g. offroad routing) may require the highest possible level of detail for certain parts of a data set as quickly as possible.

7.2. General Considerations

7.2.1. Level-Of-Detail Access

For visualizing a LiDAR scene, it is not useful to completely download a potentially large data set,

even if the zoom level or the spatial extent shown in a viewer only requires a small subset of the whole LiDAR scene. Thus, a mechanism is needed to allow the delivery of subsets of data that fit to a requested area of interest and detail level. An octree-based data structure could be suited to enable such a type of data access. This is further investigated in the remaining sections of this ER.

7.2.2. Compression

LiDAR data may comprise huge data volumes. Thus, compression mechanisms shall be investigated. Within the work presented in this ER, LAZ was considered for this purpose.

7.2.3. Lossless data transmission

Even though the data of a LiDAR scene must be subsetted to enable a level-of-detail access, it must still be ensured, that a full set of all points in a certain region must be downloadable. The octree-based approach shown in this ER, ensures that this requirement is fulfilled (no points are lost during the structuring of the LiDAR data scene into sub-sets).

7.2.4. Why to use a SOS server?

To enable a JPIP-like streaming access to LiDAR data, it is necessary to sub-divide an existing LiDAR scene into hierarchically organised subsets (in this case into an octree structure). For reflecting this structure to clients, the SWE feature concept is well suited. This way, a client is capable of determining which sub-sets of a LiDAR scene are located within the currently displayed area. Furthermore, the client is able to determine and specifically request additional sub-sets of LiDAR data based on the hierarchy level. Consequently, the SOS allows one to intelligently combine a point cloud data set with a feature model that supports its hierarchical sub-setting.

Other advantages of using the SOS to access LiDAR data comprise the provision of comprehensive metadata through the dedicated DescribeSensor operation.

7.3. Demonstration scenario

The scenario for the demonstration of the LiDAR SOS deals with an offroad routing application that is used in an emergency situation:

Due to an earthquake, it has been necessary to evacuate a part of San Francisco. However, due to the earthquake the road surface has been damaged. There are no alternative roads available and there are already evacuees on the route. How do we get the stranded evacuees to safety? Since there are no roads available, we need a cross-country evacuation route.

7.4. Data sets to be used

For testing and demonstrating the LiDAR SOS implementation, an area in California north of Richmond and Oakland, and South of Vallejo California was selected. This area lies on the most direct route from Oakland to Travis Air Force Base.

The LiDAR data sets were obtained from the USGS as LAS files:

Table 2. LiDAR data sets

Title	Metadata URL
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_001007 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301ee4e4b092f17df87090
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_000650 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301ddae4b092f17df86816
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000748 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a1ae4b092f17df77e4d
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000650 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009dee4b092f17df77cc1
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000648 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009dde4b092f17df77cbd
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000750 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a1ae4b092f17df77e51
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000739 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a17e4b092f17df77e3b
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_000604 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301dcde4b092f17df86751
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000737 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a16e4b092f17df77e37
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000644 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009dce4b092f17df77cb5
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_000600 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301dbee4b092f17df86674
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000642 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009dce4b092f17df77cb1
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_000930 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301e9fe4b092f17df86f69
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000756 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a2ee4b092f17df77ec1
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000649 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009dde4b092f17df77cbf
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000651 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009ede4b092f17df77d27
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000749 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a1ae4b092f17df77e4f

Title	Metadata URL
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000751 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a2de4b092f17df77eb7
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000745 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a19e4b092f17df77e47
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000757 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/54300a2fe4b092f17df77ec3
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_001017 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301ee7e4b092f17df870b8
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000638 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009dbe4b092f17df77ca9
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000645 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009dde4b092f17df77cb7
USGS Lidar Point Cloud (LPC) CA_ContraCostaCo_2007_000646 2014-08-27 LAS	https://www.sciencebase.gov/catalog/item/543009dde4b092f17df77cb9
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_000934 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301ea0e4b092f17df86f7b
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_000645 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301dd8e4b092f17df867f0
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_000150 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301c75e4b092f17df8544a
USGS Lidar Point Cloud (LPC) CA_SanFranBayFEMA_2004_001020 2014-08-26 LAS	https://www.sciencebase.gov/catalog/item/54301ee8e4b092f17df870c2

Chapter 8. Encoding

This chapter deals with the encodings of LiDAR data sets that had to be published through the SOS (as input data). Furthermore, the output formats of the SOS are explained.

8.1. Encodings of the Input Data Sets

The input data used for the LiDAR SOS developments were provided in the LAS/LAZ format (see previous section).

8.2. Encoding of the SOS Outputs

8.2.1. LiDAR Data

Within the GetResult operation, the SOS relies on the LAZ format (see above) as an output encoding.

However, as the GetResult operation has to deliver an XML response, the LiDAR SOS needs to perform a Base64 encoding of the LAZ payload.

8.2.2. Metadata

In order to allow the interpretation of a LiDAR data set returned by an SOS server, it is necessary to deliver certain metadata. For this purpose, relevant metadata about a LiDAR data sets (modelled as an SOS offering), are encoded as part of the SOS capabilities. More specifically, this comprises metadata how the hierarchical octree representation of the LiDAR data set has been calculated.

1. Example of the encoding of metadata of a LiDAR scene in the SOS Capabilities

```
<swes:extension xmlns:ns="http://www.opengis.net/swe/2.0"
xsi:type="ns:DataRecordPropertyType">
    <ns:DataRecord definition="spacingScaleHierarchy">
        <ns:field name="spacing">
            <ns:Count definition="spacing">
                <ns:value>25</ns:value>
            </ns:Count>
        </ns:field>
        <ns:field name="scale">
            <ns:Quantity definition="scale">
                <ns:uom xlink:href="http://www.opengis.net/def/nil/OGC/0/unknown"/>
                <ns:value>0.01</ns:value>
            </ns:Quantity>
        </ns:field>
        <ns:field name="hierarchyStepSize">
            <ns:Count definition="hierarchyStepSize">
                <ns:value>5</ns:value>
            </ns:Count>
        </ns:field>
    </ns:DataRecord>
</swes:extension>
```

Chapter 9. Required SOS Operations and Parameters

This chapter described the SOS operations that are necessary for enabling the streaming of LiDAR data.

The point cloud data delivered by the SOS service is grouped in different offerings, where each offering exposes data from a specific point cloud scene. For example, a single point cloud scene for San Francisco is considered as a single offering. The point clouds of each offering are stored in the octree data structure presented earlier, where each node of the octree is bound to a Feature of Interest (please refer to section 7 for more details on the sub-division of LiDAR scenes).

A typical workflow of using the SOS service to request point cloud data looks like the following:

1. Initial information such as service metadata and a detailed description of the point cloud contents (i.e. offerings) that is available on the SOS service can be received by calling the GetCapabilities operation.
2. For each offering in the GetCapabilities response there exists a point cloud stored in the octree data format.
3. A listing of all octree nodes that exist for a specific offering can be requested by using the GetFeaturesOfInterest operation. This operation returns the unique identifiers with the specific indexation endings of all nodes within the octree.
4. Finally, to request the data of a specific node of an offering, the GetResult operation is used. This operation finally returns the content of the requested node as XML where the resultValues attribute contains the point cloud data as Base64 encoded LAZ.

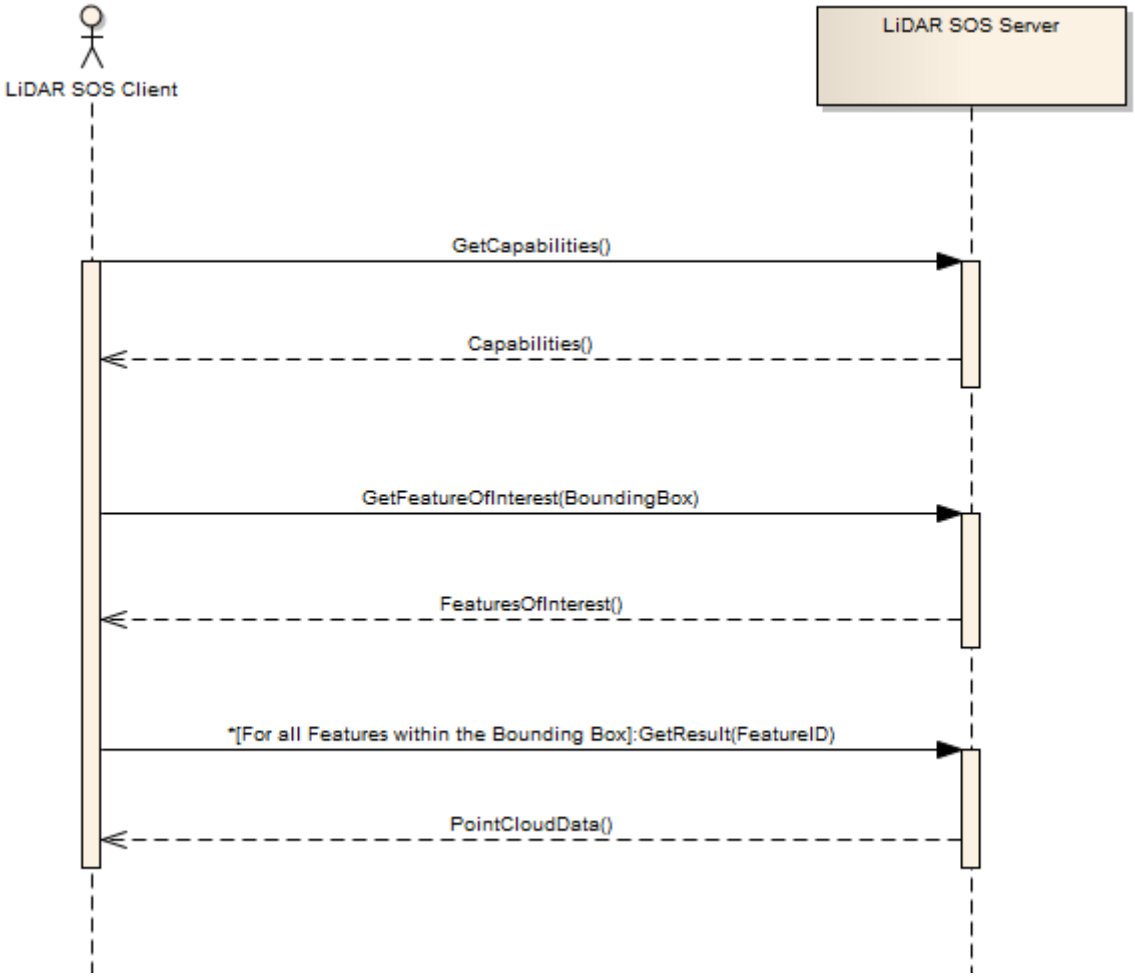

Figure 1. Illustration of the interactions between LiDAR SOS client and server.

The following subsections describe the operations to access the point cloud data in more detail.

9.1. GetCapabilites

Connecting to a SOS server for getting service metadata and a description of its data contents is done by calling the GetCapabilities operation. This operation allows clients to retrieve the metadata of the SOS and detailed information of the point clouds that are accessible over the SOS interface.

The contents section describes the service as a collection of offerings. Each offering contains the following essential information:

- The Offering identifier
- The related Procedure identifier
- The collection of Observable Properties (their identifiers)
- An extension attribute that holds the general information of the point cloud scene.

The content of the extension attribute holds the necessary information of the converting process of a specific point cloud scene to the octree data format. This includes information such as the cubic bounding box, point attributes and the number of hierarchy levels. All this information (especially the cubic bbox) is essential to set up the octree structure at the client.

Example of a GetCapabilities request

```
http://pilot.52north.org/52n-sos-webapp/service?service=SOS&request=GetCapabilities
```

Shortened example of a GetCapabilities response. For the LiDAR SOS the sos:contents section is especially relevant. For each LiDAR scene there is a separate offering.

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:Capabilities ...>
   <ows:ServiceIdentification>
      ...
   </ows:ServiceIdentification>
   <ows:ServiceProvider>
      ...
   <ows:OperationsMetadata>
      ...
   </ows:OperationsMetadata>
   <sos:filterCapabilities>
      ...
   </sos:filterCapabilities>
   <sos:contents>
      <sos:Contents>
         <swes:offering>
            <sos:ObservationOffering xmlns:ns="http://www.opengis.net/sos/2.0">

<swes:identifier>http://www.52north.org/test/offering/pointcloud</swes:identifier>
               <swes:name codeSpace="eng">ContraCosta</swes:name>
               <swes:extension xmlns:ns="http://www.opengis.net/swe/2.0"
xsi:type="ns:DataRecordPropertyType">
                  <ns:DataRecord definition="spacingScaleHierarchy">
                     <ns:field name="spacing">
                        <ns:Count definition="spacing">
                           <ns:value>25</ns:value>
                        </ns:Count>
                     </ns:field>
                     <ns:field name="scale">
                        <ns:Quantity definition="scale">
                           <ns:uom
xlink:href="http://www.opengis.net/def/nil/OGC/0/unknown"/>
                           <ns:value>0.01</ns:value>
                        </ns:Quantity>
                     </ns:field>
                     <ns:field name="hierarchyStepSize">
                        <ns:Count definition="hierarchyStepSize">
                           <ns:value>5</ns:value>
                        </ns:Count>
                     </ns:field>
                  </ns:DataRecord>
               </swes:extension>
               <swes:extension>
```

```
                  <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
                     <gml:lowerCorner>6056998.350000001 2201998.35
-2.88</gml:lowerCorner>
                     <gml:upperCorner>6069500.0 2214500.0
12498.769999999908</gml:upperCorner>
                  </gml:Envelope>
               </swes:extension>

<swes:procedure>http://www.52north.org/test/procedure/pointcloud</swes:procedure>

<swes:procedureDescriptionFormat>http://www.opengis.net/sensorml/2.0</swes:procedureDe
scriptionFormat>

<swes:observableProperty>http://www.52north.org/test/observableProperty/pointcloud_1</
swes:observableProperty>
               <sos:observedArea>
                  <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
                     <gml:lowerCorner>2201998.35 6056998.35 -2.88</gml:lowerCorner>
                     <gml:upperCorner>2214500.005 6066999.9975
706.92</gml:upperCorner>
                  </gml:Envelope>
               </sos:observedArea>
               <sos:phenomenonTime>
                  <gml:TimePeriod gml:id="phenomenonTime_1">
                     <gml:beginPosition>2012-07-31T17:45:15.000Z</gml:beginPosition>
                     <gml:endPosition>2012-07-31T17:45:15.000Z</gml:endPosition>
                  </gml:TimePeriod>
               </sos:phenomenonTime>
               <sos:resultTime>
                  <gml:TimePeriod gml:id="resultTime_1">
                     <gml:beginPosition>2012-07-31T17:45:15.000Z</gml:beginPosition>
                     <gml:endPosition>2012-07-31T17:45:15.000Z</gml:endPosition>
                  </gml:TimePeriod>
               </sos:resultTime>
               <sos:responseFormat>http://www.opengis.net/om/2.0</sos:responseFormat>
               <sos:observationType>http://www.opengis.net/def/observationType/OGC-
OM/2.0/OM_Measurement</sos:observationType>

<sos:featureOfInterestType>http://www.opengis.net/def/samplingFeatureType/OGC-
OM/2.0/SF_SamplingPoint</sos:featureOfInterestType>
            </sos:ObservationOffering>
         </swes:offering>
      </sos:Contents>
   </sos:contents>
</sos:Capabilities>
```

9.2. DescribeSensor

If a user wants to retrieve the description of the sensor or process that has generated the LiDAR scene, the DescribeSensor operation of the SOS can be used. Within the response of this operation,

additional (usually SensorML encoded) metadata from the SIPC/HDF5 file may be returned. Please note: in this testbed a stronger focus was on the development of the streaming so that a comprehensive metadata mapping to SensorML would be part of future work.

Example of a DescribeSensor request for retrieving metadata about the sensor with the id "http://www.52north.org/test/procedure/pointcloud"

```
<?xml version="1.0" encoding="UTF-8"?>
<swes:DescribeSensor xmlns:swes="http://www.opengis.net/swes/2.0" version="2.0.0"
service="SOS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/swes/2.0
http://schemas.opengis.net/swes/2.0/swes.xsd http://www.opengis.net/swe/2.0
http://schemas.opengis.net/sweCommon/2.0/swe.xsd">
   <swes:procedure>http://www.52north.org/test/procedure/pointcloud</swes:procedure>

<swes:procedureDescriptionFormat>http://www.opengis.net/sensorml/2.0</swes:procedureDe
scriptionFormat>
</swes:DescribeSensor>
```

Example of a DescribeSensor response containing SensorML encoded metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<swes:DescribeSensorResponse xmlns:swes="http://www.opengis.net/swes/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml/3.2"
xsi:schemaLocation="http://www.opengis.net/swes/2.0
http://schemas.opengis.net/swes/2.0/swesDescribeSensor.xsd
http://www.opengis.net/sensorml/2.0
http://schemas.opengis.net/sensorML/2.0/sensorML.xsd http://www.opengis.net/gml/3.2
http://schemas.opengis.net/gml/3.2.1/gml.xsd">

<swes:procedureDescriptionFormat>http://www.opengis.net/sensorml/2.0</swes:procedureDe
scriptionFormat>
   <swes:description>
      <swes:SensorDescription>
         <swes:validTime>
            <gml:TimePeriod gml:id="tp_C46F290752705F4FD4A80191AC502ECF666BE99C">
               <gml:beginPosition>2016-08-10T08:13:34.757Z</gml:beginPosition>
               <gml:endPosition indeterminatePosition="unknown"/>
            </gml:TimePeriod>
         </swes:validTime>
         <swes:data>
            <sml:PhysicalSystem xmlns:sml="http://www.opengis.net/sensorml/2.0"
xmlns:swe="http://www.opengis.net/swe/2.0" xmlns:xlink="http://www.w3.org/1999/xlink"
gml:id="ps_C100C9EF114F0F998014F67A53ECFD443DA8176B">
               <!--Unique identifier-->
               <gml:identifier
codeSpace="uniqueID">http://www.52north.org/test/procedure/pointcloud</gml:identifier>
               <sml:keywords>
                  <sml:KeywordList>
```

```
                     <sml:keyword>contracosta-r4022</sml:keyword>
                     <sml:keyword>contracosta-r4264</sml:keyword>
                     <sml:keyword>contracosta-r6200</sml:keyword>
                  </sml:KeywordList>
               </sml:keywords>
               <sml:identification>
                  <sml:IdentifierList>
                     <sml:identifier>
                        <sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
                           <sml:label>uniqueID</sml:label>

<sml:value>http://www.52north.org/test/procedure/pointcloud</sml:value>
                        </sml:Term>
                     </sml:identifier>
                  </sml:IdentifierList>
               </sml:identification>
               <sml:validTime>
                  <gml:TimePeriod
gml:id="tp_E73B44F3289C053ECA4ED7130D652DF6DF09348D">
                     <gml:beginPosition>2016-08-10T08:13:34.757Z</gml:beginPosition>
                     <gml:endPosition indeterminatePosition="unknown"/>
                  </gml:TimePeriod>
               </sml:validTime>
               <sml:capabilities name="offerings">
                  <sml:CapabilityList>
                     <sml:capability name="ContraCosta">
                        <swe:Text
definition="http://www.opengis.net/def/offering/identifier">
                           <swe:label>ContraCosta</swe:label>

<swe:value>http://www.52north.org/test/offering/pointcloud</swe:value>
                        </swe:Text>
                     </sml:capability>
                  </sml:CapabilityList>
               </sml:capabilities>
               <sml:featuresOfInterest>
                  <sml:FeatureList
definition="http://www.opengis.net/def/featureOfInterest/identifier">
                     <swe:label>featuresOfInterest</swe:label>
                     <sml:feature xlink:href="contracosta-r4022"/>
                     <sml:feature xlink:href="contracosta-r4264"/>
                     <sml:feature xlink:href="contracosta-r6200"/>
                  </sml:FeatureList>
               </sml:featuresOfInterest>
               <sml:inputs>
                  <sml:InputList>
                     <sml:input name="test_observable_property_pointcloud">
                        <sml:ObservableProperty
definition="http://www.52north.org/test/observableProperty/pointcloud"/>
                     </sml:input>
                  </sml:InputList>
```

```
               </sml:inputs>
               <sml:outputs>
                  <sml:OutputList>
                     <sml:output name="test_observable_property_pointcloud_1">
                        <swe:Quantity
definition="http://www.52north.org/test/observableProperty/pointcloud_1">
                           <swe:uom code="NOT_DEFINED"/>
                        </swe:Quantity>
                     </sml:output>
                  </sml:OutputList>
               </sml:outputs>
            </sml:PhysicalSystem>
         </swes:data>
      </swes:SensorDescription>
   </swes:description>
</swes:DescribeSensorResponse>
```

9.3. GetFeaturesOfInterest

The GetFeatureOfInterest operation can be called to retrieve all the features of interest for which the SOS offers data. It accepts a procedure as parameter and returns all the Features of Interests, i.e. nodes within the octree, related to that procedure. This operation provides a sort of list consisting of the featureIDs of all nodes within the octree. All featureIDs always have the same structure like follows: <name>-<octree_identifier>, where the name is any arbitrary name shared for all features and the octree_identifier corresponds to the location within the octree.

In addition, to determine the nodes of the octree that cover a certain area, the GetFeatureOfInterest operation also supports spatial filters within the query.

Example of a GetFeatureOfInterest call. In this case all features for which the sensor with the id "http://www.52north.org/test/procedure/pointcloud" is delivering data are requested.

```
http://pilot.52north.org/52n-sos-
webapp/service?service=SOS&version=2.0.0&request=GetFeatureOfInterest&procedure=http:/
/www.52north.org/test/procedure/pointcloud
```

Example of a GetFeatureOfInterest response (shortened so that it only contains one feature of interest).

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:GetFeatureOfInterestResponse xmlns:sos="http://www.opengis.net/sos/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sams="http://www.opengis.net/samplingSpatial/2.0"
xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:sf="http://www.opengis.net/sampling/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/sos/2.0
http://schemas.opengis.net/sos/2.0/sosGetFeatureOfInterest.xsd
http://www.opengis.net/sampling/2.0
http://schemas.opengis.net/sampling/2.0/samplingFeature.xsd
http://www.opengis.net/samplingSpatial/2.0
http://schemas.opengis.net/samplingSpatial/2.0/spatialSamplingFeature.xsd
http://www.opengis.net/gml/3.2 http://schemas.opengis.net/gml/3.2.1/gml.xsd">
   <sos:featureMember>
      <sams:SF_SpatialSamplingFeature
gml:id="ssf_1DD31564DE9BC0E539C07E6C16CDDCFDCB8453AD">
         <gml:identifier
codeSpace="http://www.opengis.net/def/nil/OGC/0/unknown">contracosta-
r</gml:identifier>
         <sf:type xlink:href="http://www.opengis.net/def/samplingFeatureType/OGC-
OM/2.0/SF_SamplingSurface"/>
         <sf:sampledFeature
xlink:href="http://www.opengis.net/def/nil/OGC/0/unknown"/>
         <sams:shape>
            <ns:Polygon xmlns:ns="http://www.opengis.net/gml/3.2"
ns:id="polygon_ssf_1DD31564DE9BC0E539C07E6C16CDDCFDCB8453AD">
               <ns:exterior>
                  <gml:LinearRing xsi:type="ns:LinearRingType">
                     <ns:posList
srsName="http://www.opengis.net/def/crs/EPSG/0/4326">2201998.35 6056998.350000001
2214499.98 6056998.350000001 2214499.98 6066999.970000001 2201998.35 6066999.970000001
2201998.35 6056998.350000001</ns:posList>
                  </gml:LinearRing>
               </ns:exterior>
            </ns:Polygon>
         </sams:shape>
      </sams:SF_SpatialSamplingFeature>
   </sos:featureMember>
   ...
</sos:GetFeatureOfInterestResponse>
```

9.4. GetResultTemplate

The GetResultTemplate operation may be used for determining the encoding and the structure of the data in GetResultResponses. However, for this implementation in the context of Testbed-12, the

GetResultTemplate operation is not needed.

9.5. GetResult

The GetResult operation is the final operation to request the data of a specific node in the octree. As input parameters it requires the unique identifier (i.e. featureID) of the specific node in octree, which can be requested by the GetFeatureOfInterest operation. In addition, it requires the unique identifier for the observedProperty that can be found in the corresponding offering of the GetCapabilities response.

The GetResult operation returns the requested node of the octree as Base64 encoded LAZ string. Therefore, in order to make use of the encoded LAZ files, it is first required to decode the Base64 string in the resultValues xml attribute.

We use the feature_id to select the subset of point cloud data that shall be returned (e.g., r.01, r.02). This allows clients to dynamically navigate through the structure of an octree to request more detailed data for certain areas if a users zooms in.

Currently the payload of the GetResult response is in the LAZ format. However, the support of further output formats (e.g. WKB, EXI) could be easily enabled by adding a dedicated result format request parameter to the GetResult implementation.

Example of a GetResult request

```
http://pilot.52north.org/52n-sos-
webapp/service?service=SOS&version=2.0.0&request=GetResult&offering=http://www.52north
.org/test/offering/pointcloud&observedProperty=http://www.52north.org/test/observableP
roperty/pointcloud_1&featureOfInterest=contracosta-r
```

Example of a GetResult response (the LAZ payload has been shortened for better readability of this ER).

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:GetResultResponse xmlns:sos="http://www.opengis.net/sos/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sos/2.0
http://schemas.opengis.net/sos/2.0/sosGetResult.xsd">
   <sos:resultValues xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">TEFTRgAAAAAAAAAAAAAAAAAAAAAAAAAAQJQREFMAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAFBEQUwgMS4yLjAgKDY4YzkzNSkAAAAAAAAAAAAAAAAAEAHgB+MARwIAAAQAAACDIgB+lAIAoK0
AALfmAQAnAAAAAAAAAAAAAB7FK5H4XqEP3sUrkfheoQ/exSuR+F6hD8AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ADiehT+zSRXQWdmZpYJG1dB16Nw/THlQEHNzMwsx8xAQZDC9ShcEYZAzMzMzMzMAMAAAExBU0ZfUHJvamVjdGl
vbgCvh0AAR2VvVGlmZiBHZW9LZXlEaXJlY3RvcnlUYWcAAAAAAAABAAEAAAAHAAAEAAABAAIAAQQAAAEAAQAAC
AAAAQDmEAEIsYcHAAAABggAAAEAjiMJCLCHAQABAAsIsIcBAAAAAABMQVNGX1Byb2plY3Rpb24AsIcQAEdlb1R
pZmYgR2VvRG91YmxlUGFyYW1zVGFnAAAAAAAAiG10lh2kckAAAABAplRYQQAATEFTRl9Qcm9qZWN0aW9uALGHC
ABHZW9UaWZmIEdlb0FzY2lpUGFyYW1zVGFnAAAAAAAAAFdHUyA4NHwAAABsYXN6aXAgZW5jb2RlZAAAvFY0AGh
0dHA6Ly9sYXN6aXAub3JnAAAAAAAAAAAAAAAAAAAAAgAAAAICAAAAAAAAUMMAAP//////////////////8DA
AYAFAACAAcACAACAAgABgACADxJEAAAAAAAu0sdJIVCMA2dAQAACgASAgAABgAAAAAAAAAAAAAAAAAAAEDc0rK
W9OmNoEwTYzlcXtYWBrgNSwzWFUYJF7UKUjy+MdNagF8b0MKlqfLYI2qvJtqkPKWSv4SP+ok5Klf1tgjNK5PKW
5Z4hyODc4/nn1jwokTKsPwNqcyyDqKPvIT27UqCk88EIkkbpERzsuHz8LafnpDjxkPFJq2iwRXetsgLyaW1Qj2
5Pavhbz3j8dkL3TM32eY3t/SP4AlqrQkZrcGwa3ANyUvDN/ma2fccfIbj5KQaO5ZPfTbyTvQjPXZvjIw0GyYB5
xbyE7mJMLW/R/aEcLscdJUYPYHGkhsvLHUoQmurrKF1Ys7YgrazR7Tj+yRIshcoagUSTyqqsNJ+1+mVmOdi6Hw
fjtobZ2oK5vK43QCWHGdF93Km61V4wuTEQihHWZHrR83RIU1DJPFEjyE+H2GasHWntM/Ey6Czfy6o+08Ff550S
rcJAWpEmoxOX/L/BPNRtx/COJ+Y5YJ5KY/ReO42f80IuFF7cN0bKgHKf2A+UXi2j1KERKf+LrhaxYPrwNthW3e
oJU/X9tFxt71fYDWaWjlehnoT1RRqQ1T2CtCx/70GitHXoL+gXEOa8T2Dq9D+o8bW7FXitcw0VF644h9VOKlZG
p+Opoa69QA46ZEzM85n94dhEFJheT9BCHFXKWk2uYhekV/3ozoYHYieDvFW+e5OMP3QmwEdcVJyDIy92DPUWa+
OGYIrvqe/j/huo/Su+D+IrL9ulec1m1O8vc4m6gmW/N50mWEhfppdIhF+qtRyAwMfbXJvIA4p5B4lVBWFSMcXs
/yfbBQNAZwZgyUurSAi7fskIrr2DgY13ezneGIFMc6ezwQQXDLGF/SiUkipEZ0zFFpgEy9x8RV63IBTpw3RSFY
ErWOY/QAaYT9pbdr3IQ8VLbzhcEcyLep/CcpNfK7s891+NyRxbIMTg0yEuAW1EHBxMVBJWfvSav3QaqsbCYlJF
dRRFaHlV1ZeQmVVgd5OvdqROUj3ADaKrdLoMd2R1aQSMxbBT/iQktAtJrniDQexBg59nXTTXAbOIiclStXlEUG
yA8GXoj1pbrPxNkb8JH4xOLEQK/KNzj50EOn2u2l60+zAvB5JQ/AEc91Uzg1yIDZGZHTaJ4P1JjFxbUNAbH6hU
ZJp96JUy0V34ezc9/khNVk/H7eIFJBVQYB3FGN0ep/71bzi3bztUwSguivrbFKTT+zFHQ8m7yVPQfcP6KzwBQF
vtPjz+NxJA/ySaWNc0zt4Vn0nqirv8sBG60VEnengycnsuQ0abdQGFAjYFem8KzEMQ3BA9HSw12YsDoBGazfnX
+UvKOV9B6Th6it1nDZyEoTwu1j7ypXoeDivaGXMMbmzwT14QtQTDXAzi5jZXzw1db/FrOvXxzwvuD+RIShp6IG
0vq26c1YYBgBN3tLb6GWF34ERlKwS895N2m2KuQHcgw//iIMlevH6lMot2EihWoH+R4wokmSABz+/YIv5EIq3g
m+ [...]
  </sos:resultValues>
</sos:GetResultResponse>
```

Chapter 10. SOS Compression (LiDAR) Server Implementation

This section describes the implementation of the LiDAR Streaming SOS Server. First the general approach for the development of the server is described. After this, more detailed information about the implementation is provided.

10.1. Organization of the Point Cloud Data

An essential requirement to the streaming-based provision of point cloud data is to enable multi-resolution access to the data. However, since a point cloud scene may contain several million data points, an on-demand level of detail selection and data fetching process may become very CPU intensive and time-consuming.

A predominant approach to tackle these limitations in dealing with large point clouds is to pre-process the input model in order to provide structural access to a hierarchy of pre-computed point clouds with different coverages and resolution-levels. The SOS server makes use of this idea and therefore relies on the octree data format of Potree.

The generalization of the Potree data structure works as follows:

- For each point, if the distance to any other point inside the root node is larger than a specific spacing, then the point is added to the root node.
- Otherwise, the node is passed to one of the eight child nodes in the next level and the same check is applied with a smaller spacing.

In this way, each point of the initial point cloud is allocated to a different level of details where low-level nodes contain low resolution models while the resolution increases and the covered area decreases at higher levels. This allows the combination of several point cloud levels with a stateless top-down octree traversal to locally improve sample densities.

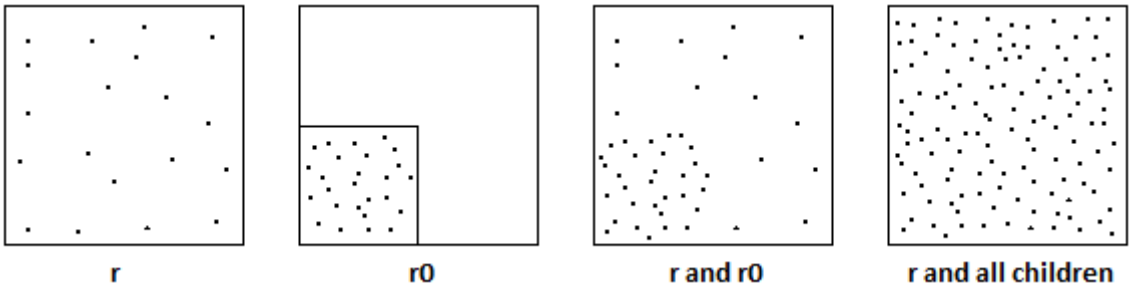

Figure 2. Illustration of the approach of subdividing the LiDAR scene for a simplified 2-dimensional case. The same approach is applied for the 3-dimensional case: r contains a rather coarse resolution covering the whole area while its children contain higher resolution subsets for parts of the area. The highest level of detail is achieved if the content of root and all its children is combined.

The Potree file format is actually a collection of nodes and sub-nodes of distinct point clouds, where each node can be addressed by an unique identifier that indicates its exact position in the hierarchy. This identifier looks like the following: Depending on the spatial position of the child

nodes relative to its parent node, each child node can be addressed by a number ranging from 0 to 7.

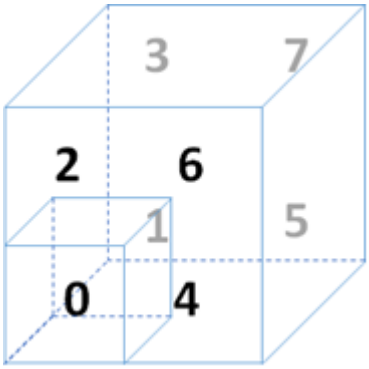

Figure 3. Illustration of the indices of the children in the octree

To demonstrate this principle, consider a root node of the octree indexed with the character r. Each of r's child nodes can now be assigned a fix number ranging from 0 to 7. In this way, the third child of r is addressed by r2 and the first child of r2 is addressed by r20.

10.2. JPIP-Like Streaming

With the aforementioned octree-based data format, we can provide a JPIP-like streaming approach as required by the project. One of the major characteristics of JPIP-based streaming is that the server does not send data that is already in the client cache. Instead, only the missing information is sent to the client. Therefore, the server is responsible to identify parts of requests that have already been send to the client in previous responses. This is called cache modelling and requires the server to be stateful. However, the SOS server does not provide any session-based capabilities.

For this reason, we introduced along with the Potree data format a static hierarchy of point clouds that are completely distinct from each other, meaning that each point only exists once in the whole data structure. This way of organizing the data has the benefit that it enables the easy caching of contents transferred by the server in responses without any retransmissions of overlapping requests.

In addition, our approach enables the multi-resolution access to the data similar to JPIP. Clients initially get a low-resolution view of the point cloud and the resolution can be increased by requesting more sub-nodes of the specific part of the scene. By following this approach, the client provides access to a high-resolution image, but only transfers the subset of the point cloud that has a specific resolution and that is actually required by the client. The same intention is followed by JPIP.

10.3. Database Structure for Storing Point Clouds in PostgreSQL

LiDAR scenes may consist of hundreds of millions points, which leads to the difficult problem of managing massive number of points in PostgreSQL, because considering each point as a single row in relational databases leads to significant scalability issues. For this reason, we used the PostgreSQL extension pgPointCloud to overcome the limitations of a straight-forward point cloud

integration based on single points. It introduces a new datatype called pc_patch that creates an abstraction layer by storing group of several hundred to thousands of points rather than single point entities. By making use of this new datatype, the overhead due to the great number of rows is omitted.

As presented before, we used an octree-based data format for organizing the point clouds and providing means of different resolution levels. Along with this structure, the pgPointCloud extension is used to store each node in the octree as a single entity in the database. A new table has been added to the database schema that stores the content of each node in the octree. This way, each node is referenced by a feature id that is linked to a so called pc_patch which contains the point cloud data for the node. The import to and the export from the database is handled by the so-called "Point Cloud Data Abstract Library" Each feature_id is assigned an ending that corresponds to the octree address invented by the Potree data structure.

For storing and loading the LAS/LAZ file, we rely on the PDAL library. This library offers a comprehensive set of functionality for translating and manipulating point cloud data.

Chapter 11. LiDAR Streaming Client Implementation

This section contains a description of the client application to consume data from the LiDAR Streaming SOS Server.

11.1. General Things

The client implementation for lidar point clouds is a Web viewer for large point clouds that uses standard web technologies and requires no additional plugins. The client is a modified version of the Potree client framework, which is a free and open source WebGL-based viewer for large point clouds. It is based on the octree data format described in the previous section. The use of this multi-resolution enabled data structure allows the client to cull away data and to only load what is visible within the rendered scene. This approach significantly reduces required bandwidth as well as the amount of data to be cached and handled by the client.

11.2. Loading data from the SOS

As mentioned in the previous section, the developed client is based on the existing Potree implementation. In order to make use of the developed streaming capabilities of the SOS, we have replaced the general file-based loading mechanism with an additional adapter for the SOS interface. Initial information such as service metadata and detailed descriptions of the point cloud contents (i.e. offerings) that are available on the SOS service, are received by calling the GetCapabilities operation. For each offering in the GetCapabilities response there exists a point cloud stored in the octree data format on side of the SOS. The client takes this information and provides the selection capabilities to the user.

Once a specific point cloud (offering) has been selected by the user, the client starts retrieving additional information about the requested point cloud. Therefore, it uses the GetFeatureOfInterest operation to request a listing of all octree nodes related to the selected offering. When the GetFeaturesOfInterest response is retrieved by the client, the client starts loading the scene by first requesting the root element r of the octree and then requesting its child elements r0 to r7. Based on the users view frustum in the client, the client decides on its own, which parts of the octree gets requested from the SOS and rendered. For the requests to download the different chunks of the LiDAR data set it makes use of the GetResult operation provided by the SOS.

Each GetResult response that is retrieved by the client holds the information of the requested node element as Base64 decoded LAS or LAZ. To make use of the encoded content, the client first decodes the resultValue element of the response back to bytes and then interprets the binary content as LAS or LAZ.

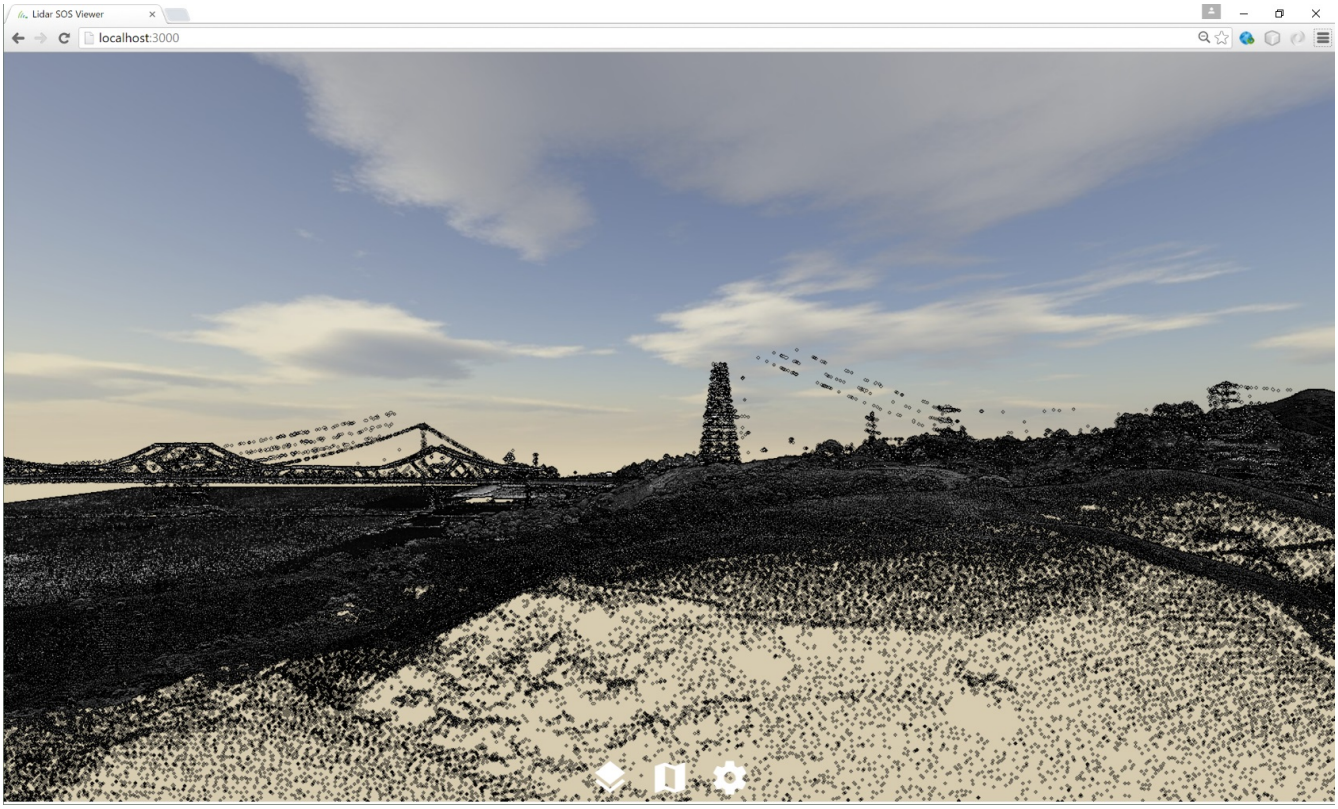

Figure 4. Screenshot of the modified Potree client

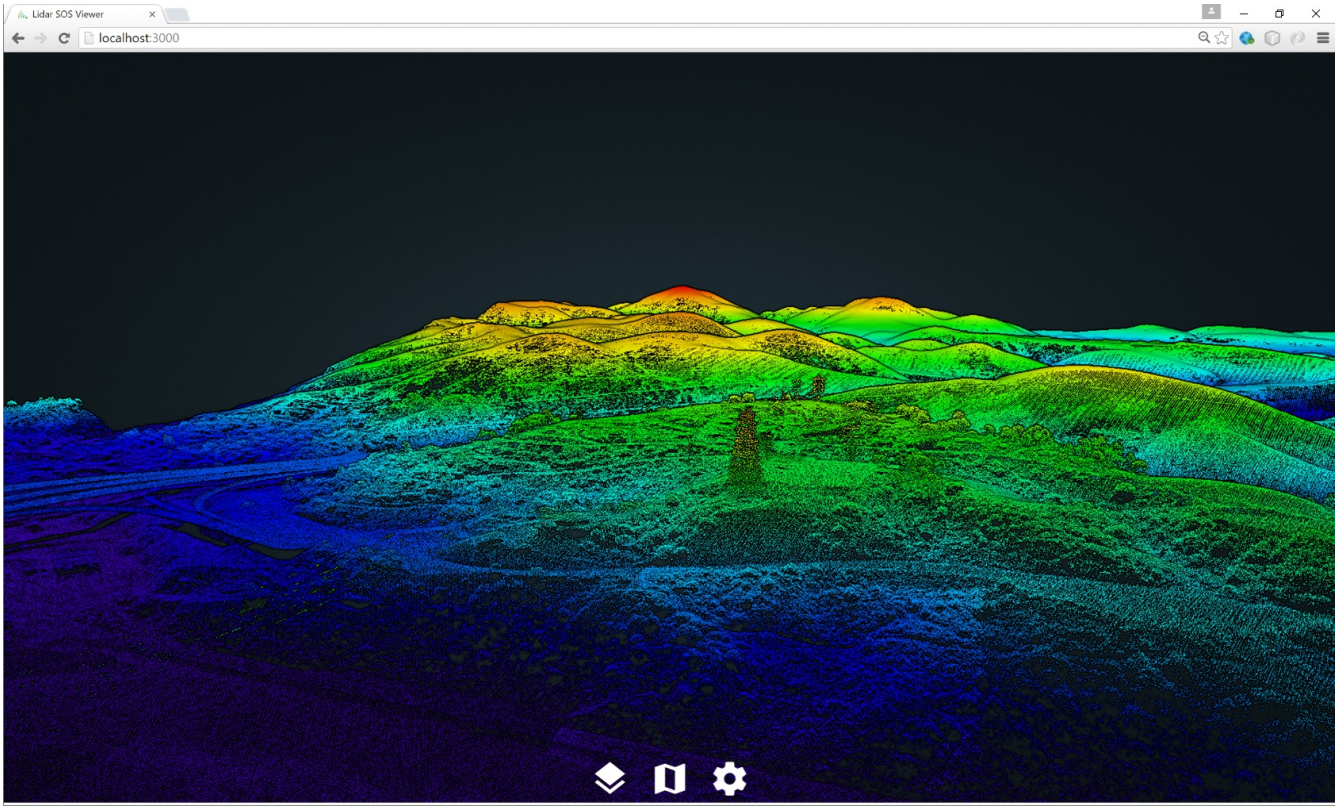

Figure 5. Screenshot of the modified Potree client showing elevation information

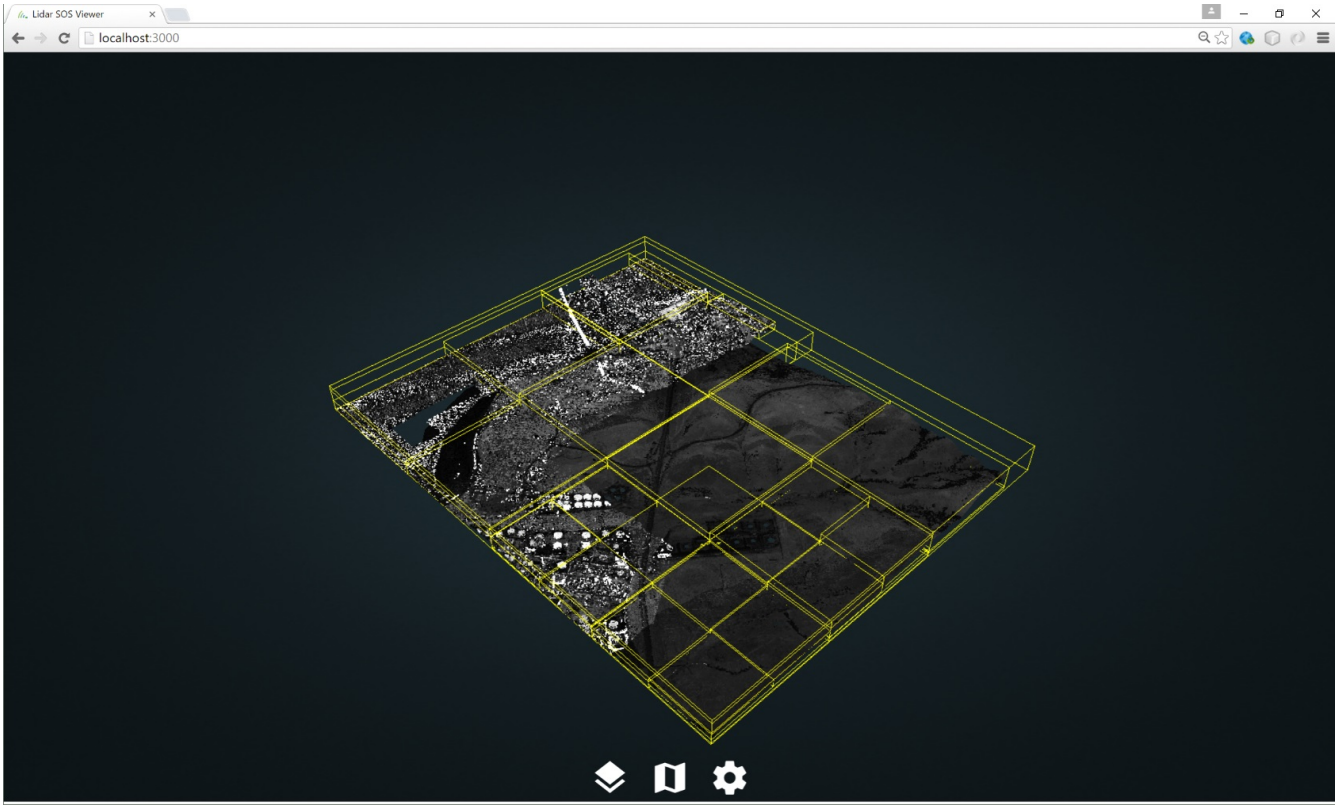

Figure 6. Screenshot of the modified Potree client showing the underlying octree structure

Chapter 12. Recommendations

This section summarizes recommendations resulting from the developments performed on the LiDAR SOS server and client.

12.1. SOS Interface

The SOS interface is suited for serving LiDAR data in a JPIP-like streaming approach.

The requesting of individual chunks of a LiDAR scene can be achieved through the GetResult operation. A requirement for applying this approach is the pre-processing of the LiDAR scene to be published so that it is subdivided in patches with different spatial extent and resolution. Subsequently it is possible to treat every patch as a distinct feature which can be combined by the client with other data chunks.

Further SOS operations needed for this approach comprise:

- GetCapabilities for retrieving an overview of the offered LiDAR scenes and their metadata
- GetFeatureOfInterest for retrieving the geometries of the LiDAR data chunks that cover a certain area of interest

As the ResultHandling operations are not a mandatory element required by the SOS standard, they should be made mandatory by a dedicated SOS 2.0 LiDAR Streaming Profile.

12.2. Data Models and Format

The Sensor Web data models and formats can cover the requirements for streaming LiDAR data. However, it is recommended for the future to support native (binary) data formats for LiDAR data as output of the GetResult operation.

12.3. Performance

The performance observed during the implementation and testing was suitable for the purposes of the testbed (loading chunks of the LiDAR scene within a few seconds). However, a certain delay was induced by the need to en-/de-code the binary LiDAR data into a text-based XML document. By avoiding this overhead, it would be possible to retrieve the requested data chunks from the database in order to write it directly into an output stream. We expect that removing this bottleneck by allowing binary GetResult response contents, a significant performance improvement would be achieved.

12.4. Comparison with Streaming of other Data Types

Compared with other data streaming approaches such as JPIP, the data streaming through the SOS interface as it was developed in Testbed-12 has a few differences:

- While clients in JPIP specify the area of the image that shall be downloaded, the LiDAR streaming SOS uses pre-defined patches. A client has first to identify the relevant patches via the

GetFeatureOfInterest operation and subsequently can download the data for the selected patches via the GetResult operations.

- JPIP may use stateful server side business logic to determine which data was already sent to the client. In case of the LiDAR streaming SOS, it is the responsibility of the client to keep track which data it already has downloaded.

In summary, the LiDAR SOS requires slightly more business logic on the client side compared to other approaches. However, during the implementation of the LiDAR SOS client, it became clear, that this additional logic can be implemented in a straightforward manner with reasonable efforts.

12.5. Summary

As a final statement, it is possible to conclude that the SOS 2.0 Standard is suitable for enabling the streaming of LiDAR data. Based on pre-processing of the LiDAR data sets that shall be published and the SOS result handling operations it is possible to subset a LiDAR scene into smaller chunks that can be step-wise loaded by a client. It has to be noted, that this mechanism requires a certain amount of business logic on the client side but as demonstrated by the LiDAR SOS client, the implementation is feasible with reasonable efforts.

Recommendations on the improvement of the underlying standards comprise especially the idea to allow the GetResult operation responses to contain binary data without the requirement to embed it into an XML container. This would avoid an additional en-/de-coding step during data transmission which would increase performance.

Another topic would be to develop a dedicated SensorML profile for the specific metadata of LiDAR data. While the encoding of the necessary metadata is generally feasible, formal guidance how to achieve this would further increase interoperability.

As a final step of future work, these recommendation could result in a dedicated SOS 2.0 Profile for serving LiDAR data (e.g. including the ResultHandling operations which should be made mandatory for those SOS servers that shall server LiDAR data).

Appendix A: Revision History

Table 3. Revision History

Date	Release	Editor(s)	Primary clauses modified	Descriptions
April, 2016	S. Jirka, C. Stasch	0.1	all	initial version
September 29, 2016	Simon Jirka	0.7	all	extended version
October 04, 2016	Simon Jirka	0.8	various	added further content
October 05, 2016	Simon Jirka	0.9	various	added further content
October 31, 2016	Simon Jirka	1.0	various	adressed comments from review

Appendix B: Bibliography

[1] Bröring, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T., Stasch, C., Liang, S., Lemmens, R.: New Generation Sensor Web Enablement. Sensors 2011, 11, 2652-2699.

[2] 52°North: SOS Server 4.x implementation, http://52north.org/communities/sensorweb/sos/

[3] American Society for Photogrammetry & Remote Sensing: LAS SPECIFICATION, VERSION 1.4 – R13 15 July 2013, http://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf

[4] TU Wien, Institute of Computer Graphics, Potree 1.3, http://potree.org/wp/