SPATIO-TEMPORAL DATA FUSION TOOLKIT –VERSION 3

Deliverable D4.2.3

Circulation:	PU - Public
Lead partner:	TU Delft
Contributing partners:	FOMI , CNR-IMATI-GE, IGN, UCL
Authors:	Roderik Lindenbergh (TU Delft), Roberto Giachetta (Foemi), Simone Pittaluga, Giuseppe Patane, Michela Spagnuolo (Imati), Beril Sirmacek (TU Delft)
Quality Controller:	Vibeke Skytt (Sintef)
Version:	1.0
Date:	02.11.2015

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
0.1	Sep 8 2015		Initial Draft
0.2	Sep 25 2015		First partner contributions included
0.4	Okt 13 2015		Updated service tables; added illustrations
0.7	Okt 16 2015		Added Service #98, further updates throughout
0.8	Okt 23 2015		Added missing texts + minor improvements
0.9	Okt 29 2015		Added scalability results + MapReduce strategy
0.92	Okt 30 2015		Incorporated Giuseppe's and Michela's and Vibeke comments, corrected references, final summary + last scalability results
1.0	Nov 2 2015		Incorporated last quality check + last sc. results

[1] Integers correspond to submitted versions

1 EXECUTIVE SUMMARY

This report describes the third version of the IQmulus *Spatio-Temporal Data Fusion Toolbox* (Deliverable D4.2.3). This version is different from the previous two versions in the sense that no new services were released. The reason for this is twofold. First, the main showcases were already defined in the second year. Therefore new functionality at large scale was not required this year. Second, the main emphasis in this Toolbox is to make existing services work on large data and within the IQmulus infrastructure. Therefore in this toolbox parallel implementations were made available for a number of services. In addition one service is updated in a non-trivial way such that it is now able to take collections of files as input. Being able to operate on collections of files is one of the requirements that are a consequence of the choice of working with the IQmulus Hadoop Distributed File System (HDFS).

2 TABLE OF CONTENTS

3 TABLE OF FIGURES

4 INTRODUCTION

In this Introduction a short overview is given of the work within Task 4.2 that lead to this final toolbox report.

4.1 MOTIVATION OF SERVICE DEVELOPMENT

In the third year of the IQmulus project, service development has been further driven by the Marine, Land and Urban showcases. Emphasis is on making these showcases truly work on big data and on making the performance of both complete showcases and individual services measurable in an automated way using the scalability testing framework. Therefore considerable work of the service developers went in adding logging facilities and updating metadata procedures. This work on logging and metadata updating is not detailed in this deliverable; for more information we refer the reader to D1.2.4 and D6.5. In the descriptions below new service functionality in the algorithmic sense is reported on. Here three types of extensions of functionality can be distinguished

1. New parallel implementations of existing algorithms
2. Non-trivial extension of functionality to collections of files
3. Introduction of new options to existing methods

Notably the MapReduce version of Service #70 was implemented to make the entire showcase LS3 run in Hadoop. Services #71, #72 and #74 are planned to be used in a Year 4 version of showcase LS3 and are being prepared for that. The updated folder-enabled Service #98 would be very useful for an extension of showcase US2. The current version of US2 only takes Laser Mobile mapping data as input, but it would be an interesting challenge to add airborne LIDAR data as an additional input source in a large scale scenario. Note that Services #70 and #98 were also identified in D4.1.3 *Amendment* as targeted services w.r.t the aspect *size.* We meet this target in this toolbox release by releasing the MapReduce version of Service #70 and the version operating on tiles of Service #98. Also Service #8 was targeted in D4.1.3 *Amendment,* but it turns out that the tiling solution provided for Service #98 in this Toolbox will be largely portable to Service #8.

4.2 NEWLY RELEASED SERVICES IN YEAR 3

No new services have been developed in Year 3 within the scope of this Toolbox. Instead, the functionality of a number of existing services has been improved or extended to make them more suitable to handle large data.

4.3 UPDATED SERVICES IN YEAR 3

The table below gives an overview of services that have been updated with respect to their functionality or algorithm.

TABLE 1: UPDATED SERVICES

Number	Service Name	Lead	Showcase
#70	Radiometric Enhancement	Fömi	LS3
#71	Convolution Filtering	Fömi	LS3 (Y4)
#72	Image registration and resampling	Fömi	LS3 (Y4)
#73	Raster Preprocessing*	Fömi	LS3
#74	Spatial and Topological Operations	Fömi	LS3 (Y4)
#96	Store rainfall data	Imati	LS2
#98	Point cloud intersection	TU Delft	US2 (Y4)

*Combines methods in services #70, #71 and #72.

4.4 SERVICE CHARACTERISTICS & RELEASE STATUS

Work Package 4 has introduced and refined a set of rules for developing and releasing services documented in Deliverable D4.1.3 and its amended version. Requirements got further updated when the IQmulus infrastructure became available, which for example enabled automated testing. Details are described in the combined deliverable D6.1/6.2. One requirement is that each service must provide a description in a Metadata table. This table collects practical information on a service in a human readable format. These table are available in an appendix of the different toolbox reports. Computer readable information on services is available in a so-called *metadata.json* file. A so-called Jenkins test will be performed automatically on each new version of a service uploaded to artifactory, with the help of a *testing.json* file, to validate if a service is able to run on the IQmulus infrastructure. In addition, particular attention is given to the specification of service characteristics, which are expected to include:

- Quality
- Robustness
- Computation time in relation to data size
- Locality/Globality of the algorithm

In the table below an overview of the release status of Task 4.2 services is given w.r.t. the WP4 and WP6 requirements. For each service the latest version in Artifactory available at the date in the table caption is given. For that version it is indicated if the two required json files metatadat.json and testing.json are available. Jenkins test status is checked on 146.140.214.198:9090

TABLE 2: SERVICE RELEASE STATUS AT 21-10-2015

Service	Language	Target Architecture Ubuntu 14.04	Execution nodes	Metadata available	Artifactory uploaded	Jenkins passed*
#70	Mono/C#	yes	Single/multi (Mapreduce)	yes	yes	pass
#71	Mono/C#	yes	Single/multi (Mapreduce)	yes	yes	na
#72	Mono/C#	yes	single	yes	yes	na
#73	Mono/C#	yes	single	yes	yes	na
#74	Mono/C#	yes	single	no	yes	na
#96	Python	yes	Single/multi	yes	yes	na
#98	C++	yes	single	yes	yes	partly

*Some Jenkins testing got delayed because of the large supply of testing requests

4.5 SUMMARY OF THE FINAL TOOLBOX

In this Toolbox report several services are extended to better work on big data regarding several aspects. For services #70 and #71 a MapReduce implementation is released which make these services operate much faster on satellite images on a cloud infrastructure. Also for Service #98 a MapReduce implementation has been released. In this case the service is not necessarily operating faster but more robust, as the MapReduce version avoids memory exceedance. Service #96 has now been extended in a non-trivial way to take collections of files as input. Within HFDS, which is chosen to store data within the IQmulus project, this is key to make Service #96 suitable for large data sets. Finally, it should be recalled that the main job of all services within the different toolboxes is to perform processing of geospatial data. Therefore it is still important that Services #72, #73 and #74 have increased functionality.

5 LIBRARIES

In this toolbox, Mapreduce is used to parallelize some existing single node services.

5.1 HADOOP MAPREDUCE

FIGURE 1. HADOOP LOGO

Hadoop Mapreduce is used in Services #70, #71 and #98. Hadoop is the open source implementation of the Mapreduce model by the Apache Software foundation.In the map step a function is parsed to every element of a list. Next, the resulting list is given as input to one or more reduce steps that combine the resulting list elements. An explicit example for mapreducing rainfall data is given below in the description of Service #98. Hadoop Mapreduce is available from https://hadoop.apache.org/releases.html. Within Services #70 and #71 notably the Hadoop Streaming utility is used.

6 DATA SETS

The data used for testing in this Toolbox report was also used in the previous version, but the data distribution for one service was different, as explained below.

6.1 TILED TU DELFT AIRBORNE AND MOBILE LIDAR DATA.

This year a new version of Service #98, Point cloud intersection, is implemented that runs on folders containing files, instead of just on single files. To support the testing, a tiled version of the so-called TU Delft campus data has been created, compare Figure 2. As the figure shows, the airborne LIDAR data, from the open access, Dutch national LIDAR archive AHN, is divided into 6 tiles. The Laser mobile mapping data, provided for research purposes to TU Delft by Fugro BV, is divided into 10 tiles (not shown in the figure).

The folder version will be tested next to determine the intersection of the full LMMS campus data of 1.7 GB stored in a folder divided into tiles compared to a folder containing one original AHN3 LAZ tile of 2.5 GB divided into different tiles. The original AHN3 tiles released so far are indicated on the map of The Netherlands in Figure 2. AHN3 is the third version of the Dutch open source national airborne laser archive.

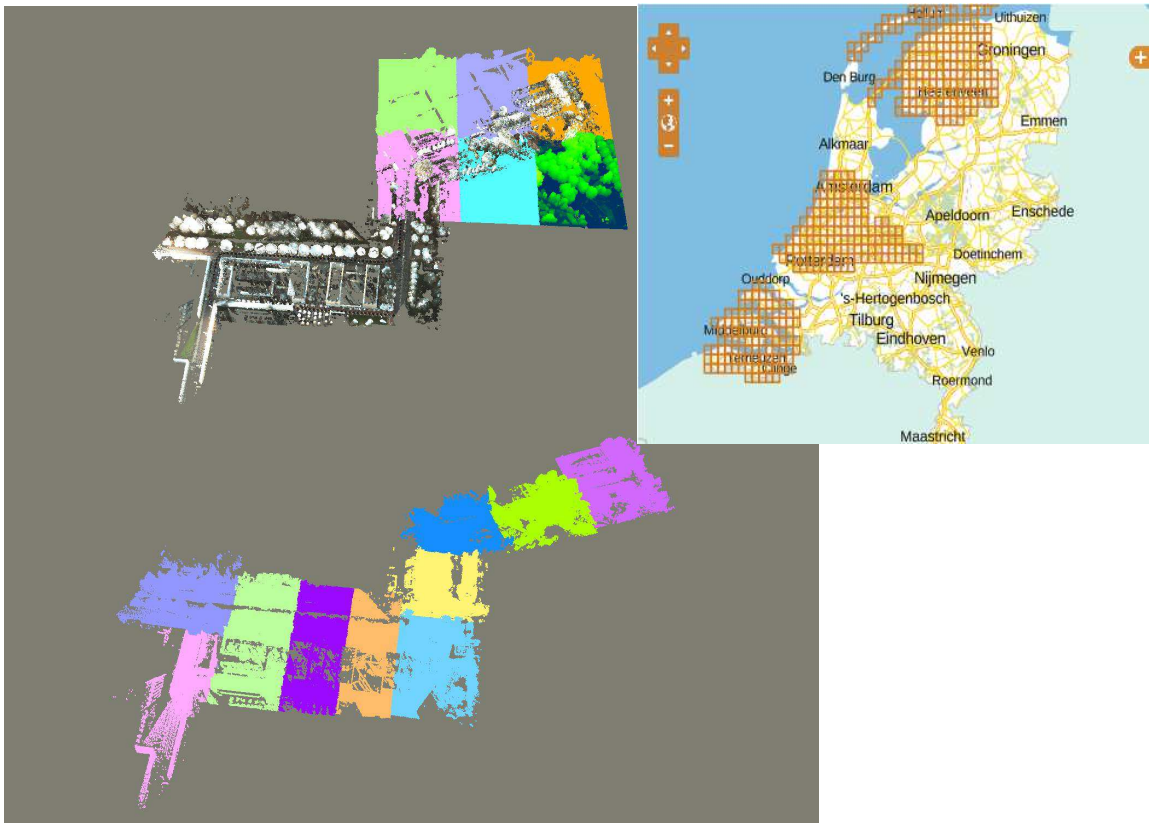


FIGURE 2. TILED VERSION OF THE TU DELFT CAMPUS DATA. THE TOP FIGURE SHOWS THE SIX AIRBORNE LIDAR TILES SUPERIMPOSED ON THE CONCATENATED MOBILE MAPPING DATA. THE BOTTOM FIGURE SHOWS THE 10 TILES FOR THE MOBILE MAPPING DATA. ON THE MAP OF THE NETHERLANDS, THE CURRENTLY AVAILABLE AHN3 TILES ARE INDICATED.

7 BIG DATA STRATEGY IN YEAR 3 AND BEYOND

7.1 BIG DATA STRATEGY IN YEAR 3

The main effort within this years Toolbox is to make existing services operate on big data. For Services #70, #71 and #96 this means that this year parallel MapReduce versions have been developed. Notably Services #70 and #71 could be easily exposed to big data, as these services perform basic processing of satellite images. Embedded within the MapReduce algorithm is also a tiling and stitching procedure, as explained in more detail below. Currently these services are tested on Spot and Landsat data, but after minor adaptations they should also work on ESA Sentinel-2 spectral satellite imagery which is coming freely available with revisit times of 2-3 days, compare Figure 4. This means that for an average European country, workflows should be anticipated in which new satellite images are incorporated at a daily automated basis.

MapReduce is not only a framework to parallelize algorithms, it is also a paradigm to process data in a streaming way. This has advantages when big chunks of data are processed. A traditional algorithm may attempt to load the full data chunk in memory at once, which may cause a crash. Processing the data in a streaming way avoid this. An example of advantage of using MapReduce is demonstrated in the new version of Service #96, Extraction of Rainfall.

Another approach is applied in Service #98, point cloud intersection. This service started as a simple brute force approach in project year 1 and was updated by implementing an algorithm that is scalable in the sense that it allows the user to obtain results at different levels of detail with corresponding processing times by determining intersection based on voxels of different user defined sizes. Still, these two instances of the point cloud intersection service could take only two single point cloud files as input. This year the service is extended to be able to operate on folders of files. For services that take single files as input this is in general trivial, but not in this case as explained in more detail in the service description below. Note that if Service #98 is operating on folders, it is also easy to parallelize it. Indeed, as a subresult it will report pairs of files from both input folders that have intersecting boundary boxes. Now for each such pair the file version of Service #98 can be launched and different instances for different pairs can be distributed over different nodes.

7.1.1 Data handling for services #70 and #71

In the Hadoop environment, processing is executed as a sequence of *MapReduce* operations, consisting of a *Map* and a *Reduce* phase. Both phases may have multiple instances executing on different nodes, identified by a *key*, which is propagated by the data. Data is shuffled between the phases by Hadoop and cashed using a built-in caching mechanism. Data is allocated in HDFS in blocks, with adjustable maximum size, and each block is processed in parallel.

When loading raster data into HDFS, one must consider the heterogeneous format of the files (e.g. GeoTIFF). The generic partitioning mechanism of Hadoop is not aware of the file content, and thus creates file blocks in a way that may be inconsistent with the file contents. To prevent this, partitioning is performed beforehand, creating individually processable blocks in a proper format (GeoTIFF). As most image processing operations function on the spatial domain of the images, partitioning is based on spatial extent. The resulting blocks are somewhat overlapping to enable operations working on multiple, neighbouring pixels (e.g. image filters, segmentation). These overlapping parts are denoted *buffer zones* (see Figure 3). The location of the buffer zones is also included in the image so that the algorithm is aware which parts of the block are actually

required to be processed. This information can be simply added to the GeoTIFF file as an additional transparency layer marked by a special flag. Additionally to the buffer zones, the resulting blocks contain several additional metadata derived from the original image, such as sensor properties and image histograms.

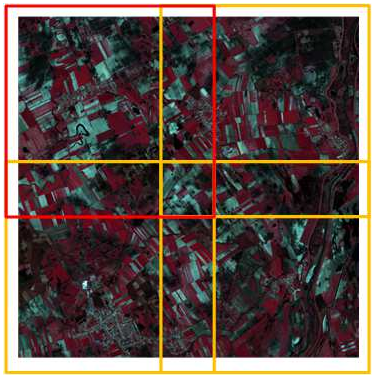


FIGURE 3. TILING AN IMAGE INTO 4 PARTS WITH BUFFER ZONES

The partitioned blocks serve as the input to the Map phase, which uses the buffer zones if specified. In some cases, the results generated by the Map phase require post-processing, which is then handled by the Reduce phase. The buffer zones may or may not be processed, depending on whether further processing of the image is planned. If no longer required, the buffer zones may simply be removed from the resulting images during processing or afterwards.

For example, assume that an image of 4000 × 4000 pixel size is partitioned into 16 parts with a buffer zone of 100 pixels. This results in images with dimensions ranging from 1100 × 1100 to 1200 × 1200 pixels, depending on their location within the original image. In case of histogram equalization, the transformation is applied pixelwise, thus (if the processing of the buffer zone is not required) the buffer zones are completely ignored, and therefore the transformation is applied to parts consisting of 1000 × 1000 pixels, and producing a result of the same image size. In contrast, a box filter with radius of 10 requires 10 additional pixels from the buffer zone, applying the filter on a region of up to 1010 × 1010 pixels for each part.

In addition to this form of partitioning, one must also consider the block based structure of HDFS. This structure requires information to be sequentially read from and written to the file system. Unfortunately, data processing is not sequential, requiring some form of in-memory buffering to enable seamless integration with Hadoop. Two possibilities come in mind:

1. The size of the input part is much smaller than the amount of available physical memory. In this case, the complete part can be loaded into memory beforehand, and all operations simply compute results in memory.
2. The size of the input part does not allow all data to be loaded into memory. In this case, the buffering mechanism should load only the required (currently accessed) amount of data into memory. The data is organized into fixed sized chunks (e.g. 1MB), and are stored in a map with flags indicating the bytes already processed in the chunk. If all bytes are processed, the complete chunk can be removed. The same method holds for writing. Although this approach is slower, as the chunks need to be individually tracked, it also enables to use only the required amount of memory.

The buffering mechanism can determine the available amount of memory, and automatically choose between these two methods. Tests indicate that the latter method may require up to 5 times as much time for data handling, but, with optimal file content layout, only a fraction of memory is used compared to the first method.

7.1.2 Advancing services #70 and #71 to MapReduce

Using the specified data handling methodology, advancing services #70 and #71 to the MapReduce paradigm is straightforward from a methodological point of view. All information required by the implemented processing methods is available in the individual blocks and no post-processing is required. Hence, each image part can be processed completely within the Map phase. From an implementation point of view however, many alterations were required compared to the original architecture and execution format.

As these services are implemented using the .NET/Mono frameworks, they cannot be directly executed by Hadoop MapReduce, but can only be incorporated using Hadoop Streaming[2], for which the executable is specified as Mapper (and optionally as Reducer). Hence, the service must enable both functionalities on demand. This was incorporated in the high level **`ServiceBase`** class, which serves as parent class for all services. **`ServiceBase`** determines the execution mode (based on the specified parameter), and instructs the engagement of the proper reading/writing mechanism. This makes it possible that a concrete service only implements its own proper processing functionality, without requiring knowledge of the exact environment it operates in.

Unfortunately, reading and writing of the binary image format (GeoTIFF) is not directly supported by Hadoop. Therefore it needs to be extended by specifying the correct input and output formats. This functionality was added in Java code, and is specified for each execution in terms of the **`resolver`**, **`inputformat`** and **`outputformat`** options. An example execution of service #70 (for histogram equalization) under Hadoop looks as follows:

```
hadoop jar hadoop-streaming-2.4.0.jar
               -libjars HadoopAegis.jar
               -D mapred.reduce.tasks=0
               -D stream.io.identifier.resolver.class=
                  hadoop.aegis.BytesIdentifierResolver
               -inputformat hadoop.aegis.CustomFileInputFormat
               -outputformat hadoop.aegis.CustomFileOutputFormat
               -input /data/input/
               -output /data/output/
               -mapper "mono AEGIS.Services.RadiometricEnhancement.exe
                              -execution-mode mapper
                              -method histogram-equalization"
```

As such execution would require too much technical knowledge of the user, the service executable also wraps the specified command line in a configuration file, by simply reducing the execution mode to a number of options with additional information (e.g. execution path).

[2]Hadoop Streaming: https://hadoop.apache.org/docs/r2.5.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/HadoopStreaming.html

7.2 BIG DATA STRATEGY IN YEAR 4

FIGURE 4. SENTINEL 2A SATELLITE IMAGE FROM THE PO VALLEY, ITALY (SOURCE: ESA)

At the start of project year 4 services as described in this Toolbox report are available and running on the IQmulus infrastructure. Services operate on test data and big data but not yet necessarily in the most efficient way. Reason for this is that for one given task consisting of a single service or full workflow operating on a collection of files there are many possible configurations that solve that task: Both input, intermediate results and output can be tiled in many different ways, while intermediate processing steps can be distributed over different nodes with different properties. At the same time, methods using LOD (Level Of Detail) processing as a strategy to handle big data can be varied in detail, with outputs of different quality as a result. The idea to find near optimal solutions in this wide range of possibilities is to systematically analyse logging results. Analyzing the logging results will help in at least two ways: first, separating efficient from less efficient configurations, and, second, identifying services that somehow perform suboptimal, are therefore slowing down a workflow and consecutively should be improved with priority by the service developers. In this way, the integrated IQmulus infrastructure is expected to provide feedback guiding development at Toolbox level.

Further possible guidance of service development is expected from requests to run newly composed workflows. Year 4 is typically devoted to extensive usage of the full IQmulus system by inside users and opening the system to external usage. Both user groups could define their own workflows given the available Toolbox services, but it is expected that in some cases additional service functionality or performance is needed to make a full input-output chain operationable. This will drive further requests for additional implementions in the different toolboxes.

8 UPDATED SERVICES

The following sections describe services that were updated w.r.t. their functionality or implementation since the last deliverable.

8.1 RADIOMETRIC ENHANCEMENT, FÖMI, #70

Radiometric enhancements transform image histograms as a preprocessing operation for further analysis. The techniques can differ, and there are also special forms of enhancement.

The service was updated both in terms of functionality (to server a broader user group) and performance on big data sets. The most important update is the introduction of distributed execution support within the Apache Hadoop framework. The service can be executed as MapReduce process with several input files, which execute in parallel. This service has received multiple additional updates:

- New algorithms were implemented.
- Support for native Landsat 7 & 8 and Spot 5 data formats was added.
- The file processing (reading/writing) performance and memory usage of the service was improved by the introduction of caching, compare Section 7.1.1.
- The service was adapted to fit the technical requirements of the IQmulus execution infrastructure, compare Section 7.1.2. This includes updating the logging mechanism and support for locally mounted network file systems, like notably HDFS.

8.1.1 Functionality & Algorithm

The following algorithms are supported:

- *Linear-contrast-enhancement*: Contrast enhancement using the manually specified offset and factor values for all pixels.
- *Histogram equalization*: Automated contrast enhancement using the commutative distribution function.
- *Histogram matching* (new): Automated matching of the image histogram to another (reference) image. [1].
- *Saturating-contrast-enhancement*: Automated contrast enhancement using affine transformation of the histogram.
- *Inversion*: Creates a negative image.
- *Top of atmosphere reflectance* (new): Automated enhancement using top of atmosphere reflectance based on imaging data, [2]. This method requires input metadata. Currently only SPOT and Landsat images are supported for processing.
- *Discrete Fourier transform* (new): The Fourier Transform is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain. The computation of the discrete Fourier transform is performed using the Cooley–Tukey algorithm, which enables the reduction of the computational complexity to $(n \cdot \log n)$, [3].
- *Temperature extraction* (new): Automated conversion of thermal infrared information to temperature information, [4]. This method requires input metadata, and currently only Landsat images are supported for processing.

8.1.2 Quality

The methods implemented in the service were tested on 6 Landsat 8 and 10 SPOT5 images (unmodified, full resolution). The quality of the resulting images was compared to multiple existing geospatial and image editing software (e.g. ENVI, QGIS) and also matched against result generated by existing implementations (e.g. Matlab). The comparison verified the 100% accuracy of all methods. This means the following: the methods in Services #70 until #74

implement analytical functionality, no approximation or randomization is involved. Therefore the result can be guaranteed to be 100 % as expected, based on the formula

8.1.3 Scalability

The service is available in two versions, standalone (single core) and MapReduce (distributed). The MapReduce version enables parallel execution of the service on multiple input files which were previously partitioned and allocated on HDFS. Results of the scalabity evaluation of this service within the IQmulus infrastructure are shown in Deliverable D6.5

The scalability of the service was evaluated on the IQmulus infrastructure in three variants:

1. Single core version. Loading and saving data on local disk.
2. Single core version. Loading and saving data on the shared HDFS file system.
3. MapReduce version, executed using the Hadoop runtime environment. Data operations on the shared HDFS file system.

Execution time was broken down into three steps:

a) Data reading,
b) Data processing
c) Data writing.

As input data, SPOT5 imagery was chosen with the caching mechanism loading the entire content into memory. The SPOT5 image of 250MB was tiled into 2, 4 and 8 parts with 10 pixels of buffer zone between the parts. Note that the size of the buffer zone is marginal compared to the original image size.

Two scenarios were evaluated:

A. Histogram equalization, which requires loading the initial histogram, then processing all pixels individually, resulting in an image with the same resolution and dimensions as the original.
B. ToA reflectance computation, which is applied to all pixels individually, thus can be computed slightly faster than histogram equalization. However, the resulting data contains 32 bit floating point values instead of 8 bit integer values, resulting in a data size of around four times the original input data size.

From the results shown in Table 3 the following conclusions are derived:

- The form of execution has no effect on the actual processing time. Both methods take about the same time in all cases. As anticipated, the ToA reflectance computation is slightly faster than histogram equalization, which requires some pre-processing.
- Reading and writing content from/to HDFS has about 90% overhead compared to local file access. As the caching mode was the same in both cases, this difference is only due to the network access time of HDFS.
- Reading and writing content on HDFS through MapReduce has only 10% to 15% overhead compared to local file access, which means it is much more efficient than simply accessing HDFS over the network. This is primarily due to the optimization of execution performed within Hadoop. It must be noted however, that Hadoop also requires some additional time (3-6 seconds) for execution initialization.

	Execution configuration	Data size	Execution time (sec)				
			Reading	Processing (A)	Processing (B)	Writing (A)	Writing (B)
1	Single core execution on local machine	250 MB	*8.63*	4.99	4.91	3.50	*14.00*
2		125 MB	*4.37*	2.47	2.43	1.81	*7.26*
3		63 MB	*2.25*	1.27	1.23	0.90	*3.60*
4		31 MB	*1.22*	0.65	0.63	0.45	*1.78*
5	single core execution on HDFS	250 MB	*18.64*	4.96	4.89	7.07	*28.24*
6		125 MB	*9.23*	2.48	2.42	3.79	*15.11*
7		63 MB	*4.72*	1.25	1.24	1.87	*7.49*
8		31 MB	*2.76*	0.65	0.63	0.93	*3.72*
9	MapReduce execution on HDFS	250 MB	*9.81*	4.93	4.91	3.72	*14.83*
10		125 MB	*4.85*	2.48	2.43	2.00	*7.95*
11		63 MB	*2.50*	1.25	1.24	0.99	*3.94*
12		31 MB	*1.46*	0.66	0.63	0.50	*1.96*

TABLE 3. EXECUTION TIMES FOR SERVICE#70 WITH DIFFERENT CONFIGURATIONS

The MapReduce version also offers the benefit of automated parallelization in case data is previously partitioned. Figure 5 presents results obtained by first partitioning the 250MB data in 2, 4 and 8 parts and next processing it on 2, 4 and 8 nodes respectively. Overall, the execution time of the MapReduce version decreases almost linearly with respect to the number of parts.

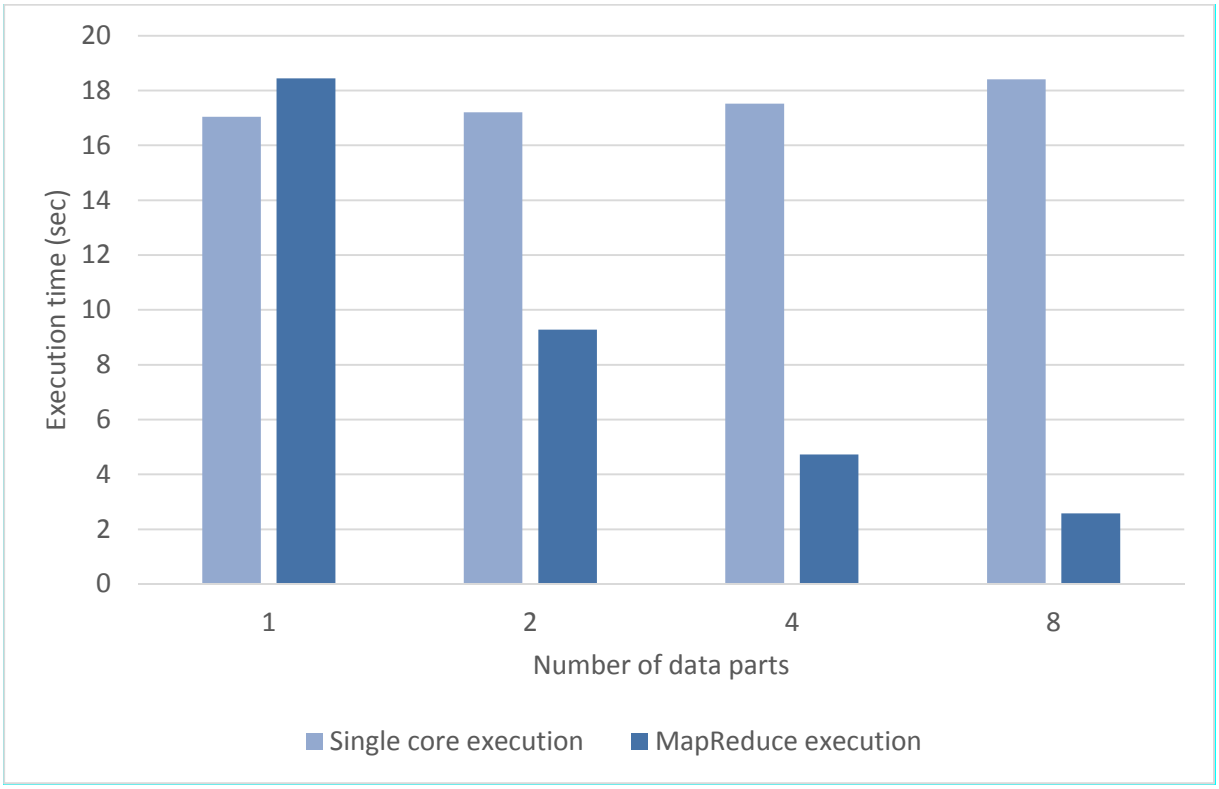


FIGURE 5. EXECUTION TIMES FOR SERVICE#70 WITH DIFFERENT NUMBER OF PARTS

8.1.4 Degree of Human Intervention

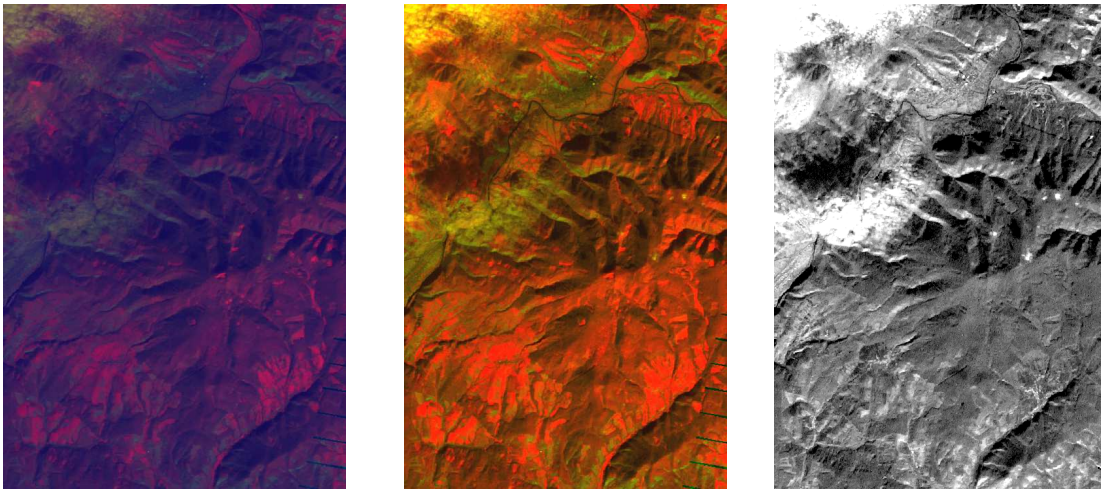
Execution is fully automated, only errors and help information are displayed during execution. If no messages are displayed, execution is successful. Options must be included beforehand. The following options are available:

- *method*: specifies the enhancement method.
- *band-index*: specifies the zero based band index which should be processed. This option can be used multiple times. Note that the zero based band index refers to band indices starting at band 0, not band 1, so indices range from 0 to n-1 for an image of n bands.
- *offset*: Specifies the offset used in linear contrast enhancement. Here, the offset is the value which is added to each pixel intensity in the image.
- *factor*: Specifies the factor used in linear contrast enhancement. Here, the factor is the value by which all pixel intensities are multiplied in the image.
- *reference*: Specifies the reference image used in histogram matching. The resulting image will have the same histogram properties as the reference image.

The syntax of the options is specified in the man page.

8.1.5 Examples

Figure 6 presents some examples of the new processing methods. The input is a Landsat 8 image, Figure 6a, containing surface reflectance. This false colour image is obtained by mapping the first three bands to RGB space. Next, the top of atmosphere reflectance is computed and the first three bands are colorized with the same colors, see Figure 6b. Finally, temperature data is extracted from bands 10 and 11 and equalized, see Figure 6c.



a) Input image b) ToA reflectance c) Temperature extraction

FIGURE 6. OUTPUT OF SERVICE #70.

Considering the quality of the results: note for example that the extracted temperature as shown in Figure 7c is derived from the Landsat 8 spectral image following the methodology described in [4]. The method in [4] is an estimation method, and is in itself not perfect. The implementation of this service is analytical, which means that the results in Figure 7c are guaranteed to follow the method in [4]. To fully evaluate the quality in for example Figure 7c a validation in geophyical sense would be required, which is beyond the scope of this project.

8.2 CONVOLUTION FILTERING, FÖMI, #71

In convolution filtering an odd sized template (known as kernel) is applied to image regions, and the central spectral value is computed based on the neighbouring values. By using different kernels, several results can be obtained, for instance, smoothing, sharpening, edge detection, line detection, etc.

The service was updated both in terms of functionality (to server a broader user group) and performance. The most important update is the introduction of distributed execution support within the Apache Hadoop framework. The service can be executed as MapReduce process with several input files, which execute in parallel. This service received multiple additional updates:

- New algorithms were implemented.
- Support for native Landsat 7 & 8 and Spot 5 data formats was added.
- Also for this service, the file reading/writing performance and memory usage was improved by the introduction of caching, compare Section 7.1.1.
- The service was adapted to fit the technical requirements of the IQmulus execution infrastructure, compare Section 7.1.2. This includes updating the logging mechanism and adding support for locally mounted network file systems like HDFS.

8.2.1 Functionality & Algorithm

The following algorithms are supported:

- *Box filter*: The box, box blur or mean filter the simplest matrix that computes the mean of filter values. It can be used as a fast approximation of the Gaussian-blur, as no actual matrix is required to be computed.
- *Gaussian-blur*: this filter performs blurring an image by a Gaussian function. It is used for enhancing image structures, removing noise, and it reduces the image's high-frequency components.
- *Median filter*: The median filter simply selects the median of the underlying values. It is primarily used for smoothening, as under some conditions, it preserves edges while removing noise.
- *Unsharp masking*: An image sharpening filter that uses a blurred (unsharp) positive image to create a mask of the original image. The unsharped mask is then combined with the negative image, creating an image that is less blurry than the original.
- *Mean removal*: This image filter sharpens the image by changing the values of neighbouring pixels with approximately the same value, so that small differences are evened out.
- *Sobel filter*: An edge detection algorithm that computes an approximation of the gradient of the image intensity function at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction
- *Prewitt filter*: Another edge detection algorithm using the same approach as the Sobel filter, with a slightly different kernel.
- *Discrete Laplace filter* (new): This operator is often used in image processing e.g. in edge detection and motion estimation applications. The discrete Laplacian is defined as the sum of the second derivatives Laplace operator and calculated as sum of differences over the nearest neighbours of the central pixel.
- *Canny filter* (new): This method is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.
- *Erosion* (new): Erosion is one of two fundamental operations (the other being dilation) in morphological image processing [5]. The value of the output pixel is the maximum

value of all the pixels in the input pixel's neighbourhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.

- *Dilation* (new): Dilation is one of two fundamental operations (the other being erosion) in morphological image processing. The value of the output pixel is the minimum value of all the pixels in the input pixel's neighbourhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.
- *Opening* (new): In mathematical morphology, opening is the dilation of the erosion of a set by a structuring element. Opening removes small objects within the image.
- *Closing* (new): In mathematical morphology, the closing of a set by a structuring element is the erosion of the dilation of that set. Closing removes small holes within the image.

8.2.2 Quality

The methods implemented in the service were tested on 6 Landsat 8 and 10 SPOT5 images, all unmodified and of full resolution. The quality of the resulting images was compared to results from different geospatial and image editing softwares like ENVI and QGIS and also matched against results generated by existing implementations in for example Matlab. The comparison verified the 100% accuracy of all methods, compare also Service #70.

8.2.3 Scalability

The scalability of the service was evaluated using the same methodology and testing infrastructure as in case of service #70. For testing, erosion was chosen, which operates each step on 9 pixels, thus it also uses some part of the buffer zone. The resulting image has the same resolution and dimensions as the input image.

Results are summarized in Table 4. From the results it is concluded that Service #71 scales in a similar way as Service #70.

	Execution configuration	Data size	Reading	Processing	Writing
1	Single core execution on local machine	250 MB	*8.63*	31.64	3.50
2		125 MB	*4.37*	15.80	1.81
3		63 MB	*2.25*	8.01	0.90
4		31 MB	*1.22*	4.07	0.45
5	single core execution on HDFS	250 MB	*18.64*	31.62	7.07
6		125 MB	*9.23*	15.78	3.79
7		63 MB	*4.72*	8.00	1.87
8		31 MB	*2.76*	4.08	0.93
9	MapReduce execution on HDFS	250 MB	*9.81*	31.64	3.72
10		125 MB	*4.85*	15.79	2.00
11		63 MB	*2.50*	7.99	0.99
12		31 MB	*1.46*	4.04	0.50

TABLE 4. EXECUTION TIMES FOR SERVICE#71 WITH DIFFERENT CONFIGURATIONS

8.2.4 Degree of Human Intervention

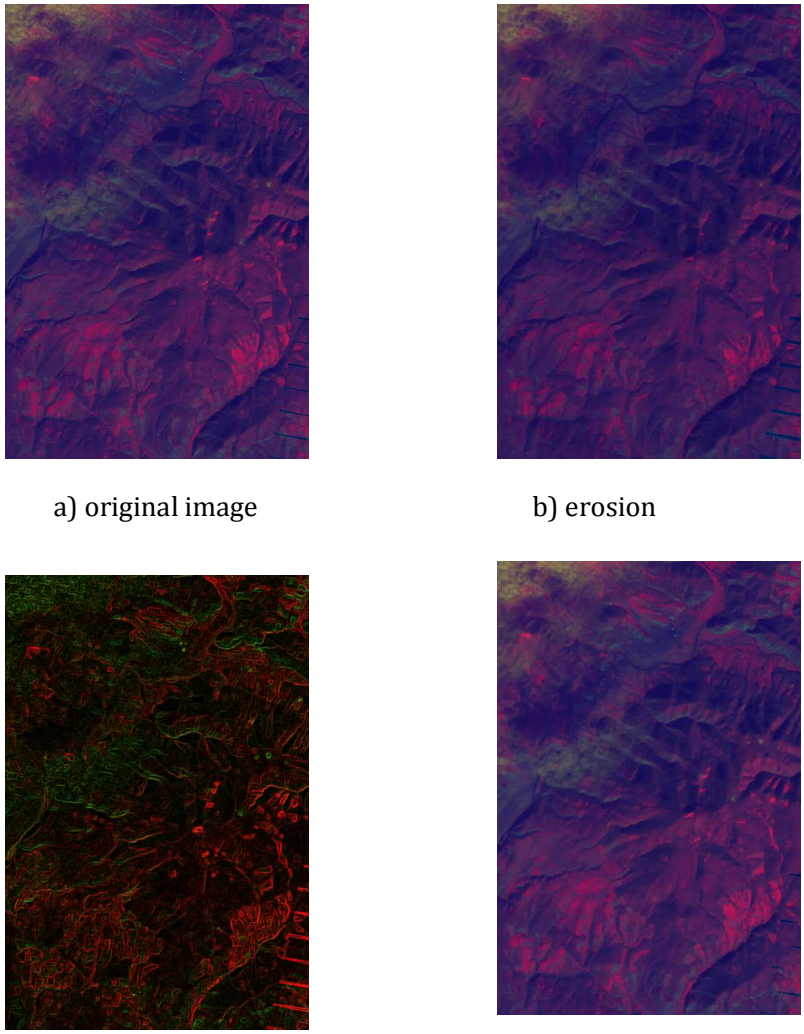
Execution is fully automated, only errors and help information are displayed during execution. If no messages are displayed, execution is successful.

Options must be included beforehand. The following options are available:

- *method*: Specifies the filtering method.
- *band-index*: Specifies the zero based band index which should be filtered. This option can be used multiple times.
- *radius*: Specifies the radius of the filter when applicable.
- *gaussian-standard-deviation*: Specifies the standard deviation value for the Gaussian blur filter. The default and also standard value is 1 which corresponds to a standard Gaussian kernel.
- *sharpening-amount*: Specifies the amount of sharpening in case of unsharp masking. The default and also standard value is 0.8, which results in a medium amount of sharpening.
- *sharpening-threshold*: Specifies the threshold at which sharpening takes effect in case of unsharp masking. The default and also standard value is 0, which will have the effect that all values are sharpened.
- *weight*: The weight of the central value in mean removal. The default and also standard value is 1, which causes all values to be weighted equally.

8.2.5 Examples

Figure 7 presents some examples of the new processing methods. The input, Figure 7a, is a Landsat 8 image colored by the surface reflectance in its first 3 bands. Figure 7b results from application of the erosion operator using a kernel size of 3×3 for all bands. Some minor differences can be noticed around edges compared to the original image. Next, closing is performed by applying dilation and erosion with kernels of size 3×3 for all bands. In the resulting Figure 7d it can be seen that some small artifacts were removed. Finally, Laplace filtering is applied to detect edges using the standard kernel (of size 3×3) for all bands. As the filter is applied individually for each band, different edges are detected. This can be observed by the different coloring in Figure 7c.



a) original image b) erosion

c) Laplace filter d) closing

FIGURE 7. OUTPUT OF SERVICE #71

8.3 IMAGE REGISTRATION AND RESAMPLING (#72)

In image registration, the task is to transform images into a single spatial reference system. Either the imagery is not initially referenced, or the reference system of the image does not match that of the dataset, and the image has to be transformed in order to be processable with other data available in the dataset.

The simplest method for performing registration is the allocation of ground control points (GCP), which have coordinates within the image space, and coordinates in the destination reference system. Another approach is to provide referenced imagery witch is matched to the source image using pattern recognition, resulting in GCP-s for the matched coordinates. This functionality is not yet supported in the initial release.

Resampling of the image is performed when the image dimensions, i.e. width and/or height, are changed. Resampling requires interpolation. The current version of the service supports four interpolation techniques: *nearest neighbour*, *bilinear*, *bicubic* (new) and *Lanczos*.

In addition to registering or resampling individual images, it is sometimes required for a set of images to have the same dimensions and spatial extent. This form of image matching registration can be peformed either by taking the intersection of all spatial extents, and cropping the images, or by taking an extent which is the union of all spatial extents, and extending all images. Additionaly to the spatial extent, the image dimensions can also be unified by downscaling larger images, or upscaling smaller images.

With respect to year 2, the service was slightly updated in functionality and performance. Improvements include:

- Support for handling multiple images, and image matching registration.
- Support for bicubic interpolation. Bibubic interpolation selects from the nearest sixteen mapped source pixels by building and evaluating two splines, one for each Cartesian direction.
- Support for native Landsat 7 & 8 and Spot 5 data formats was added.
- The file reading/writing performance and memory usage of the service was improved by the introduction of caching, Section 7.1.1.
- Adaptation to fit the technical requirements of the IQmulus execution infrastructure. Compare also Services #70 and #71.

8.3.1 Quality

The methods implemented in the service were tested on 6 Landsat 8 and 10 SPOT 5 images that were unmodified and of full resolution. The quality of the resulting images was compared to multiple existing geospatial and image editing software like ENVI and QGIS and also compared to results generated by existing implementations in e.g. Matlab. Further validation of the results is still required however. As a metric for the interpolation or resampling method, cross validation will be used, in which observed values are compared to to estimated values, where the latter are obtained by leaving out the observed value at the location considered. A metric for the registration method, distances in pixels between either control points or features extracted for the purpose of validation will be considered.

8.3.2 Scalability

The service is currently available as single core executable. Priority for multicore implementation and consecutive testing has been given to Services #70 and #71.

8.3.3 Degree of Human Intervention

Execution is fully automated given a number of predefined options, only errors and help information are displayed during execution. If no messages are displayed, execution is successful.

The following options are available:

- *source-reference*: Specifies the source reference system.
- *target-reference*: Specifies the target reference system.
- *gcp*: Specifies a ground control point. Multiple points may be specified.
- *matching-method:* Specifies the matching method. Examples are: spatial intersection/union, downscaling/upscaling.
- *interpolation-method*: Specifies the resampling method. Default is nearest neighbour resampling.

- *number-of-rows*: Specifies the numbers of rows (height) of the output image.
- *number-of-cols*: Specifies he number of columns (width) of the output image.

8.4 RASTER PREPROCESSNG FÖMI, #73

Preprocessing is a complex operation that is provided to enable multiple operations to be executed in a single process. The executable operations contain radiometric enhancements, geometric corrections, removal of atmospheric effects and instrumental errors, image registration and resampling.

This service combines the functionality of services #70, #71 and #72. The algorithms and other execution parameters match the ones specified previously.

8.5 SPATIAL AND TOPOLOGICAL OPERATIONS, FÖMI, #74

Spatial operations on 2D vector data are common features in most geospatial toolkits. The methods are described in the OGC Simple Feature Access standard and are composed of query (e.g. equals, intersects, touches) and analysis (e.g. union, difference, buffer) operations. The operations are computed by transforming the vector geometries to a topological representation, such as the half-edge model.

The service was slightly updated in functionality and performance. Improvements include:

- New algorithms were implemented as specified below.
- The (reading/writing) performance and memory usage of the service was improved by the introduction of caching , see section 7.1.1).
- Adaptation to fit the technical requirements of the IQmulus execution infrastructure. This includes updated logging mechanism and support for locally mounted network file systems (e.g. HDFS).

8.5.1 Functionality & Algorithm

The following spatial relation operations are supported: *equals*, *disjoint*, *intersects*, *touches*, *crosses*, *within*, *contains*, *overlaps*. In case of relation operations, the output of the service is a text file containing a single number: 1 (true) of 0 (false).

The following analysis operations are supported:

- *intersection*: Returns a geometric object that represents the Point set intersection of the source geometric objects.
- *union*: Returns a geometric object that represents the Point set union of the source geometric objects.
- *difference*: Returns a geometric object that represents the Point set difference of the source geometric objects.
- *symmetric-difference*: Returns a geometric object that represents the Point set symmetric intersection of the source geometric objects.
- *buffer* (new): Returns a geometric object that represents all Points whose distance from *the* source geometric object is less than or equal to distance. Note that because of the limitations of linear interpolation, there will often be some relatively small error in this distance, but it should be near the resolution of the coordinates used. Buffering uses the Minkowski-sum algorithm, [6]
- *convex hull* (new): Returns a geometric object that represents the convex hull of the source geometric object (s). Convex hulls, being dependent on straight lines, can be accurately represented in linear interpolations for any geometry restricted to linear interpolations. Convex hull computation uses the Graham-scan algorithm, [7]
- *approximate convex hull* (new): Returns a geometric object that represents the approximate convex hull of the source geometric object(s). The computation of the approximate convex hull is performed in $O(n)$ time, as opposed to the exact convex hull, which is computed in $O(n \cdot \log n)$ time. The computation is performed using the Bentley-Faust-Preparata algorithm, [8]
- *simplification* (new): Simplifies the source geometric object(s) by reducing the number of points in a curve that is approximated by a series of points. Simplification if performed using the Douglas-Peucker algorithm, [9]

In case of analysis operations, the result is written to the output file.

8.5.2 Quality

Quality of the methods is currently under investigation.

8.5.3 Scalability

Priority in scalability testing has been given to services with a new MapReduce implementation, which are Services #70 and #71 in this series of Fömi satellite data processing services. Note that for some of the implemented algorithms estimated running times are given above in terms of a computational complexity.

8.5.4 Degree of Human Intervention

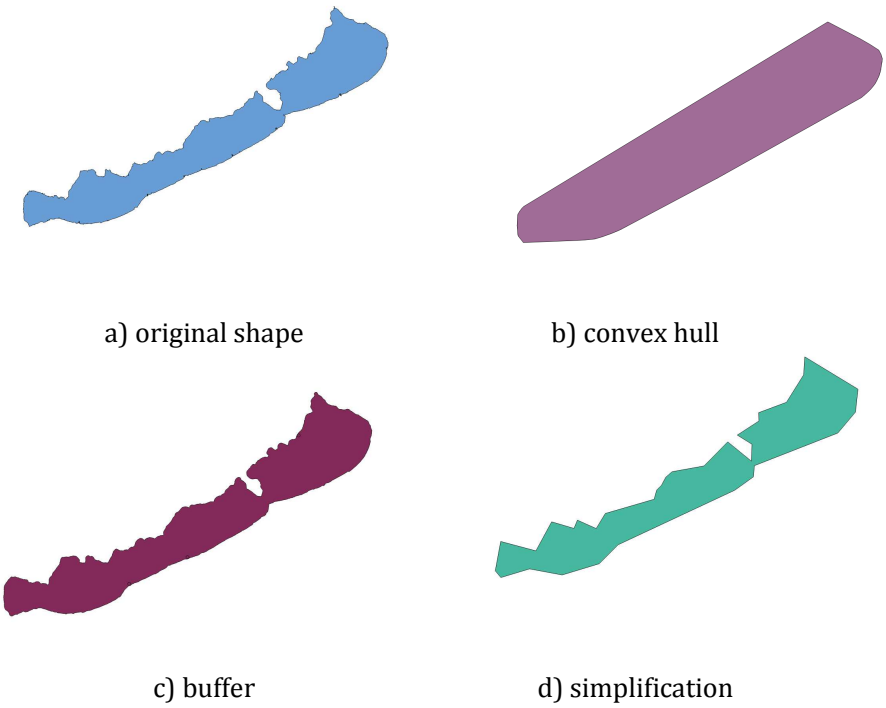
Execution is fully automated, only errors and help information are displayed during execution. If no messages are displayed, execution is successful.

Options must be included beforehand. The following options are available:

- *operation*: Specified the selected operation.
- *distance*: Specifies the buffer distance in case of buffering and the tolerance in case of simplification.
- *tolerance*: Specifies .
- *resampling-method*: Specifies the resampling method.
- *number-of-rows*: Specifies the numbers of rows (height) of the output image.
- *number-of-cols*: Specifies he number of columns (width) of the output image.

8.5.5 Examples

Figure 8 presents results of some newly added operations. The vector geometry of Lake Balaton in Hungary, Figure 8Figure 8 was taken as input. First its convex hull was obtained by applying the Graham-scan algorithm, Figure 8b. A determined buffer with a distance of 250 m is shown in Figure 8c, and a simplification with a distance of 500 m in Figure 8d.



a) original shape b) convex hull

c) buffer d) simplification

FIGURE 8. OUTPUT OF SERVICE #74

8.6 EXTRACTION OF RAINFALL, IMATI, #96

This service preprocesses rainfall data gathered from the meteorological service and prepares the data for further analysis. Service #96 is the start point of Showcase LS2. The service has been updated to make the management of the rainfall data as received from regional weather services more flexible. The main advantage of the new implementation is that now rainfall data is organized in files corresponding to individual weather stations. The resulting files may be updated one by one as needed by users or platforms that are not using a relational database. As a result, each weather station is represented at Geonetwork as an atomic dataset linked to some parent datasets.

8.6.1 Functionality & Algorithm

The service computes the cumulative rainfall over a time interval as selected by the user. The updated version of the service has been implemented in Java/Hadoop to improve performance respecting the possibility of handling large archives of rainfall records. The service loads input data from files where each of these refer to a recording from a single weather station. The mapper finds for each weather station the data belonging to the right time interval and forwards to the reducer all pairs <weatherstationname, rainfall>. The reducer computes the cumulative rainfall for each weather station and writes the result to a file. The whole task is schematized in Figure 9.

8.6.2 Quality

The update does not affect the quality of the service which only depends on the machine precision in those steps where numbers are manipulated, compare also the second year Toolbox report, D4.2.2

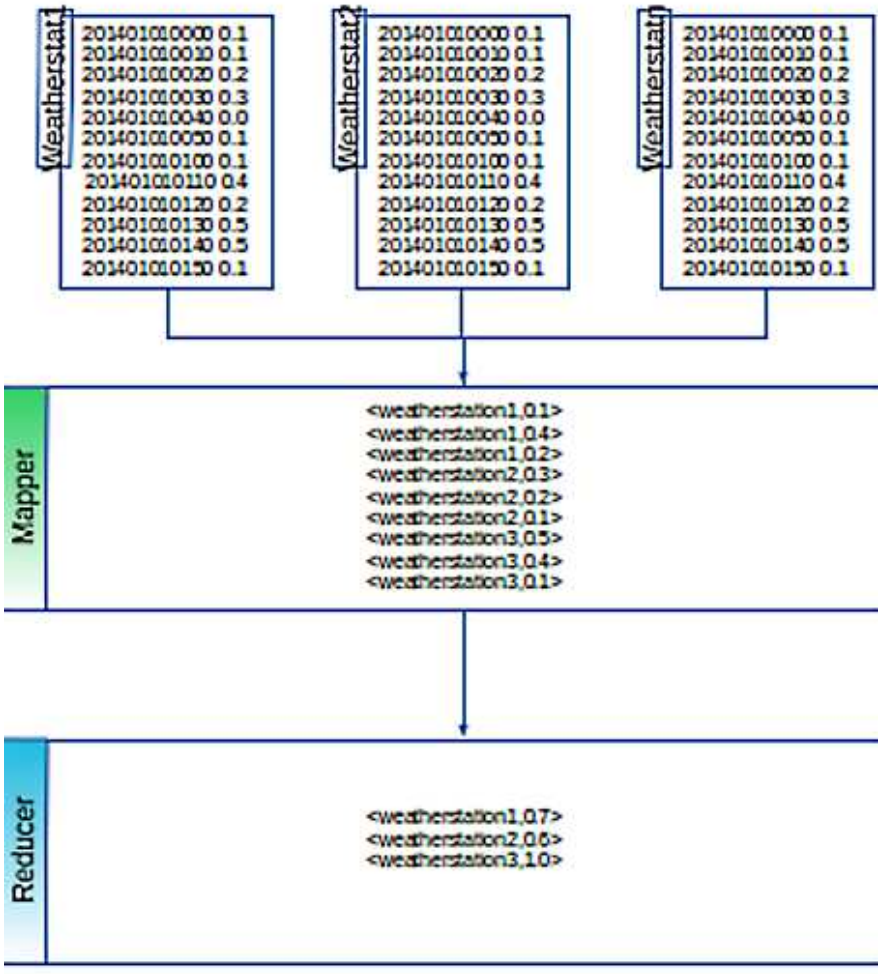


FIGURE 9. MAPREDUCE FOR RAINFALL EXTRACTION

8.6.3 Scalability

The updated service has been re-implemented in Java/Hadoop to take advantage from the parallel processing of the MapReduce paradigm. As at the time of this writing the updated service is not yet available on Artifactory, there are no `official` scalability results yet. But results of internal testing are presented in Table 1. The results show that the traditional approach is actually always faster then the Mapreduce approach. Note however, that the traditional approach crashes when it processes a dataset larger than available RAM. The traditional approach loads all input data into memory which causes the crash. Instead, Mapreduce processes the data in a streaming way, which avoids this memory overload.

The size limit on a FC node with 8 GB of RAM is between 1.3 GB and 1.7 GB corresponding, respectively, to 30*10^6 and 40*10^6 rainfalls records. The datasets potentially available are larger than this number of measurements, therefore using Mapreduce improves on robustness. Considering the processing speed: the traditionally approach is up to ten times faster than Mapreduce, that is, until it crashes. However, the processing speed of the Mapreduce version is such that it is feasible to use it, also for larger data sizes. We conclude that in this particular case,

a Mapreduce implementation does not result in faster processing but does result in a more robust approach.

Data Size	Traditional approach	Mapreduce approach
30 MB	2 s	9 s
600 MB	21 s	153 s
1300 MB	38 s	293 s
1700 MB	crash	580 s

TABLE 5. RUNNING TIME TRADITIONAL VS. MAPREDUCE FOR DIFFERENT INPUT SIZES

8.6.4 Degree of Human Intervention

Execution is fully automated, only errors and help information are displayed during execution. If no messages are displayed, execution is successful. The service will take as option the time interval and the path of both input and output folder

8.7 POINT CLOUD INTERSECTION, TU DELFT, #98

Earlier versions of this service were released in project years 1 and 2. The first version was Service #12, the second version was also called Service #98. This is a basic service that is for example used for change detection and for the evaluation of a registration at overlapping areas. It can be used in different showcases. The implementation in Year 1 used a brute force approach which was computationally not efficient. To improve on this a Level of Detail (LOD) algorithm was developed based on voxels in Year 2. To efficiently work on larger collections of data, the service is extended this year so that in can handle not only single files as input but also folders consisting of many files.

8.7.1 Functionality & Algorithm

In IQmulus year 3 the point cloud interserction service is being extended to make it work on collections of files. A description of how the service works on two individual files can be found in the second year Toolbox, D4.2.2. In addition, the service is also described in the following article, [10]:

> Wang, J., Lindenbergh, R., and Menenti, M.: Evaluating voxel enabled scalable intersection of large point clouds, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., II-3/W5, 25-31, (2015).

It is in general easy to make a service operate on a collection of files if it only takes one file as an argument. In most cases the services can be simply parsed from the folder containing the files to each individual file in the folder. This however doesn't hold for services that take more then one input file as an argument, such as point cloud intersection.

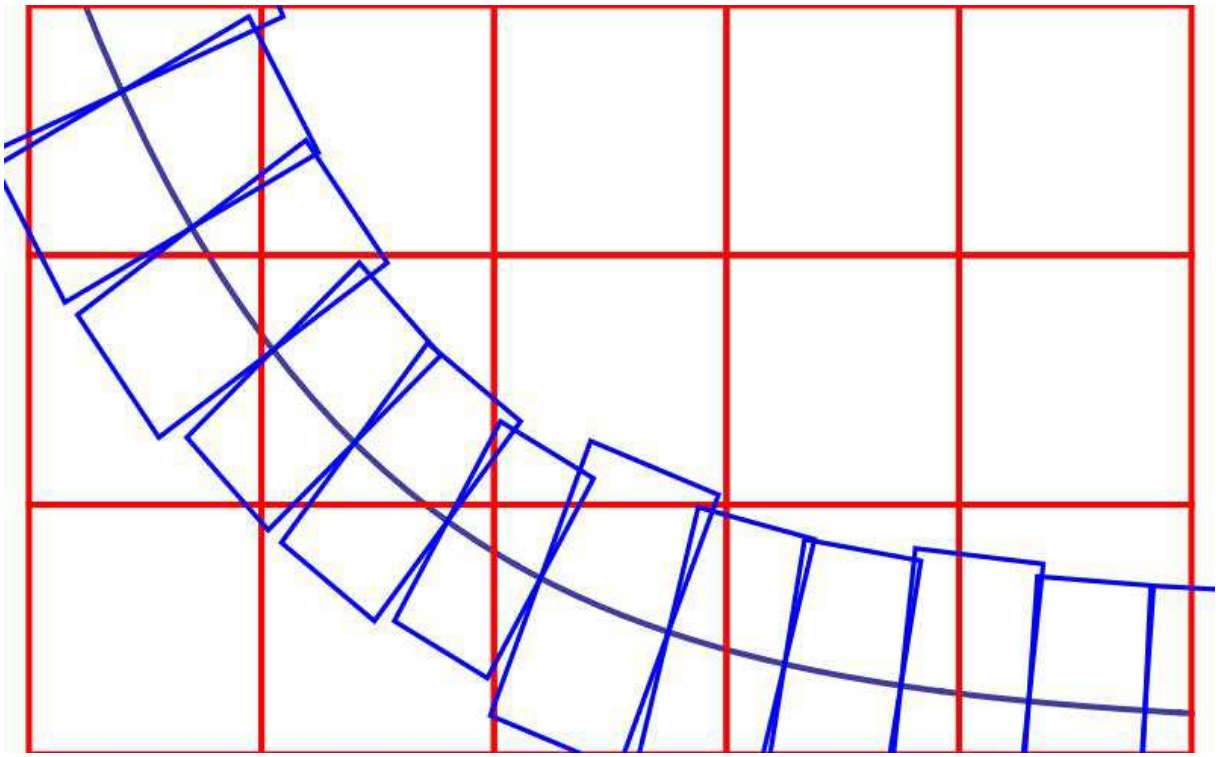


FIGURE 10. TWO COLLECTIONS OF TILED POINT CLOUD FILES GIVEN AS INPUT TO THE POINT CLOUD INTERSECTION SERVICE.

This is illustrated in a schematic way in Figure 10. Two collections of tiled point cloud files are given as input to the point cloud intersection service. In this figure, the red tiles could be part of an archive of airborne laser scan data, while the blue tiles could be collected by a laser mobile mapping system driving on the curved road. A common task in point cloud processing is to fuse the airborne and laser mobile data to obtain a complete 3D sampling of the built-up environment of in this case the neighbourhood of the scanned road. As a first step the intersection of the LMMS and airborne LIDAR data set needs to be acquired. To do so for collections of tiles as in the figure a two-step approach is used as follows:

1. Determine the 3D bounding boxes of all blue tiles
2. Determine the 3D bounding boxes of all red tiles
3. Sort both boxes in lexicographical order
4. Determine for each blue 3D box which red 3D boxes are intersecting it
5. For each pair of intersecting bounding boxes determine the file based intersection using the Year 2 version of Service #98
6. Store the results in a folder that is organised following the tile order of the blue tiles

Note that this is indeed a two step approach, as the folder version is implemented as a shell around the original file version. The rationale of the service is further illustrated in Figure 11.

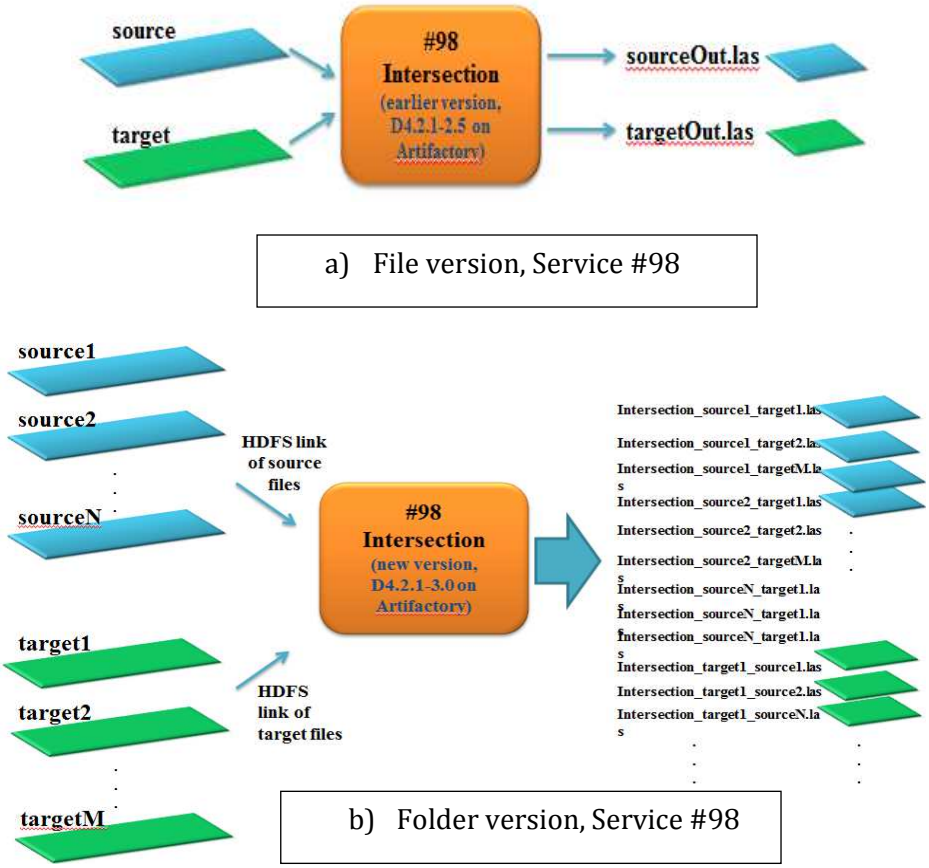


FIGURE 11. SCHEMATIC OF FILE VERSION AGAINS FOLDER VERSION, SERVICE #98.

8.7.2 Quality

The quality of the results on two sets of tiled files can be evaluated by comparing the result to what is obtained when the tiled files in each folder are concatenated to one file first and then passed on to the single file version of this services. At the time of writing the quality of the results is still under investigation.

8.7.3 Scalability

The performance of the extension of this service to collections of files notably depends on the efficiency of the bounding box operation. This operation is implemented in different versions in Service #11, Spatial Extent. In addition, simple rectangular 2D bounding boxes are expected to be available in the metadata in the data browser. Bounding box intersection is very fast as this step only evaluates a simple polygonal intersection for polygons consisting of a very small number of vertices.

Note that this folder version of the Point Cloud Intersection service can be parallellized in a trivial way, by distributing each resulting file-to-file instance of the service to a different node. It will be considered to compare the running time of different modalities of this option that result from using different tile sizes.

Scenario	Size	Reading, per tile	Processing, per couple	Writing, per tile
1 tile AHN **4 tiles LMMS**	2.4 GB 4 x 425 MB	28 sec ~11 sec	~8 min 10sec	**crash**
1 tile AHN **9 tiles LMMS**	4 x 0.5 GB 4 x 0.5 GB	27 sec ~7.8 sec	~6 min	**crash**
30 AHN tiles **20 LMMS tiles**	30 x 80 MB 20 x 85 MB	~6 sec ~6 sec	~2.95 min	~11 sec
56 AHN tiles **20 LMMS tiles**	56 x 40 MB 20 x 85 MB	~3 sec ~4.5 sec	~0.5 min	~6.45 sec

TABLE 6. RUNNING TIMES, DIFFERENT TILLING SCENARIO'S, INTERSECTION SERVICE

Some first testing results are summarized in Table 6. All tests were run on the IQmulus infrastructure at Fraunfhofer, on a single node, and using the default voxel size of 0.6 m as input parameter. The first two scenario's, working with original AHN tiles, crash because of memory limitations. This problem is obviously solved by using smaller tiles, like in the third and fourth scenario. It is not yet possible to indicate what an optimal tiling scenario would be as for this further analysis and maybe also further code optimization is required.

8.7.4 Degree of Human Intervention

The point cloud intersection service has one parameter, the voxel size, for which a default choice is available. As dicussed in the previous toolbox report, D4.2.2, a smaller voxel size will lead to more accurate results, but will be computationally less efficient. In addition, the user can use different tilings, compare Table 6.

9 NEW SERVICES

No new services are implemented this year

10 REFERENCES

[1] *Gonzalez, Rafael C.; Woods, Richard E. , Digital Image Processing (3rd ed.). Prentice Hall. p. 128, (2008).*

[2] *Liang, Shunlin, Hongliang Fang, and Mingzhen Chen. "Atmospheric correction of Landsat ETM+ land surface imagery. I. Methods." Geoscience and Remote Sensing, IEEE Transactions on 39.11, p2490-2498, (2001).*

[3] *Cooley, James W., and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series." Mathematics of computation 19.90 , p297-301, (1965).*

[4] *Jimenez-Munoz, Juan C., et al. "Land surface temperature retrieval methods from Landsat-8 thermal infrared sensor data." Geoscience and Remote Sensing Letters, IEEE 11.10 , p1840-1843, (2014).*

[5] *Mathematical Morphology and Its Applications to Image Processing, J. Serra and P. Soille (Eds.), proceedings of the 2nd international symposium on mathematical morphology (ISMM'94), ISBN 0-7923-3093-5, (1994)*

[6] *Hazewinkel, Michiel, ed. , "Minkowski addition", Encyclopedia of Mathematics, Springer, ISBN 978-1-55608-010-4, (2001)*

[7] *Graham, R.L. , An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. Information Processing Letters 1, p132-133, (1972)*

[8] *Jon Bentley, G.M. Faust & Franco Preparata, "Approximation Algorithms for Convex Hulls", Comm. ACM 25, p64-68, (1982)*

[9] *David Douglas & Thomas Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer 10(2), p112–122, (1973)*

[10] *Wang, J., Lindenbergh, R., and Menenti, M., "Evaluating voxel enabled scalable intersection of large point clouds", ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., II-3/W5, p25-31, (2015)*

11 SERVICE TABLES

11.1 RADIOMETRIC ENHANCEMENT, #70

IQmulus Service information: #70, Y3			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Radiometric enhancement		*Radiometric enhancement is used in many processing algorithms to enhance image processing and visualization.*
Description	Radiometric enhancements transform image histogram as a preprocessing operation for further analysis. The technique can differ, and there are also special forms of enhancement.		*User is able to choose from the specified enhancement methods.*
Service functionality	**Input:**	Raster image (GeoTIFF).	*Can work on any general TIFF file.*
	Input parameters: *[optional]*	Enhancement method, and additional parameters (for the specified method). Optionally the band number.	
	Output:	Enhanced raster image (GeoTIFF).	
	Functionality of the service:	Spectral transformation.	
Algorithm	The radiometric enhancement is applied to all spectral values of the image. For the execution, the generation and analysis of image histogram may also be required. The algorithm can be performed in-place and out-place. Each value modification can be performed independently (in parallel). The following algorithms are supported: Linear-contrast-enhancement, Histogram equalization, Histogram matching (new), Saturating-contrast-enhancement, Top of atmosphere reflectance (new), Discrete Fourier transform (new), Temperature extraction (new).		*Algorithms can be executed on a single machine (standalone) or on a Hadoop cluster with multiple machines (distributed).*
Implementation details	**Implementation language**	C#, Java	*The implementation is performed using the AEGIS [2] framework.*
	Dependencies with other libraries	None for standalone, single machine execution. Hadoop with HDFS and MapReduce for distributed execution.	
	Operating system	Any operation system running .NET	

		Framework or Mono.	
	Visualization modalities of the output	-	
Service characteristics	**Accuracy:** High, as algorithms compute precise relations.		
	Robustness: The distributed version of the service enables the processing of any input size if the required tiling was performed on the input.		
	Computational time in relation to data size: Radiometric enhancement is usually a linear operation with respect to image size.		
	Locality/globality of the algorithm: The operation is local for every spectral value (on each band of the image), but the histogram must be computed for each band of the image, which is a global operation. The execution time is usually within a minute.		
Alternatives	Use functionality from other toolkit (e.g. Orfeo Toolbox).		
Responsible Partner	FÖMI, Roberto Giachetta		
Involved Partners	FÖMI		

11.2 CONVOLUTION FILTERING, #71

IQmulus Service information: #71			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Convolution filtering		*Convolution filtering usually servers as a step in complex image operations (e.g. feature detection).*
Description	In convolution filtering an odd sized template (known as kernel) is applied to image regions, and the central spectral value is computed based on the neighbouring values. By using different kernels, several results can be obtained, for instance, smoothening, sharpening, edge detection, line detection, etc.		*User must be able to choose the method, the size of the filter (in some cases), and additional parameters of the kernel (e.g. the amount of sharpening).*
Service functionality	**Input:**	Raster image (GeoTIFF).	*Can work with other raster data source even without spatial reference.*
	Input parameters: *[optional]*	Convolution method, filter size (with some filter), and additional parameters of the kernel (with some filters). Optionally the band number.	
	Output:	Filtered image (GeoTIFF).	
	Functionality of the service:	Spectral transformation.	
Algorithm	The kernel is applied to all spectral values, requiring the operation to be out-place (neighbourhood values are also needed for computation). Each value modification can be performed independently (in parallel). The following algorithms are supported: Box filter, Gaussian-blur, Median filter, Unsharp masking, Mean removal, Sobel filter, Prewitt filter, Discrete Laplace filter (new), Canny filter (new), Erosion (new), Dilation (new), Opening (new), Closing (new).		*Algorithms can be executed on a single machine (standalone) or on a Hadoop cluster with multiple machines (distributed).*
Implementation details	**Implementation language**	C#, Java	*The implementation is performed using the AEGIS [2] framework.*
	Dependencies with other libraries	None for standalone, single machine execution. Hadoop with HDFS and MapReduce for distributed execution.	

	Operating system	Any operation system running .NET Framework or Mono.	
	Visualization modalities of the output	-	
Service characteristics	**Accuracy:** High, as algorithms compute precise relations.		
	Robustness: The distributed version of the service enables the processing of any input size if the required tiling was performed on the input.		
	Computational time in relation to data size: Convolution filtering is a linear operation in terms of the image size and filter size.		
	Locality/globality of the algorithm: The operation is focal for every spectral value, thus the pixel neighbourhood (matching the size of the kernel) is required. The filter can be applied individually for each band.		
Alternatives	Use functionality from other toolkit (e.g. Orfeo Toolbox).		
Responsible Partner	FÖMI, Roberto Giachetta		
Involved Partners	FÖMI		

11.3 IMAGE REGISTRATION AND RESAMPLING, #72

IQmulus Service information: #72			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Image registration and resampling		*Image registration is required for non-referenced rasters.*
Description	Image registration is the process of transforming different sets of data into one reference system (by using control points or reference imagery). Registration is followed by resampling (scaling) with multiple supported interpolations.		*User may input the control points, or use a set of referenced images to perform the registration.*
Service functionality	**Input:**	Raster image (GeoTIFF) (can be general TIFF image without reference system).	*Can work with other raster data source.*
	Input parameters:	Target reference system, ground control points (with reference system data), or referenced images (in GeoTIFF format).	
	Output:	Raster image (GeoTIFF) with spatial reference.	
	Functionality of the service:	Geometric transformation.	
Algorithm	If the control points are not specified, the referenced images are looked up for possible control points. There are then located on the source image resulting the control points. A polynomial transformation, and resampling is applied to the image (based on the number of control points).		
Implementation details	**Implementation language**	C#	*The implementation is performed using the AEGIS [2] framework.*
	Dependencies with other libraries	None.	
	Operating system	Any operation system running .NET Framework or Mono.	
	Visualization modalities of the output	-	
Service characteristics	**Accuracy:** Depends on the set of registered images, and the method used for control-point		

	allocation.	
	Robustness: Reliable to work with images up to multiple GB-s (depending on system memory size).	
	Computational time in relation to data size: The lookup and allocation of control points depends on the number and size of the referenced images. The computation of the approximation polynomial depends on the number of control points. The registration procedure depends on the interpolation method. Neares-neighbour is computed the quickest, whilst Lanczos resampling is the slowest (but is the most accurate). The execution time can take multiple minutes.	
	Locality/globality of the algorithm: The lookup and allocation of control points is performed globally on the referenced images and the source image (but can performed in parallel). Resampling is performed locally on each pixel.	
Alternatives	Use functionality from other toolkit (e.g. Orfeo Toolbox).	
Responsible Partner	FÖMI, Roberto Giachetta	
Involved Partners	FÖMI	

11.4 PREPROCESSING OF RASTER DATA, #73

IQmulus Service information: #73			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Preprocessing of raster data		*Preprocessing is required to remove the effect of radiometric and geometric distortion.*
Description	Corrections of radiometric and geometric properties must be applied to all remotely sensed images before further processing and analysis can be performed. These corrections include the removal of atmospheric effects, instrumental errors and geometric effects (such as earth rotation, curvature and panoramic distortion).		*It is possible to have pre-defined correction parameters for sensors. Therefore the user only needs to input the sensor type and image capture parameters (e.g. time, look angle).*
Service functionality	**Input:**	Raster image (GeoTIFF).	*Can work with other raster data source.*
	Input parameters: *[optional]*	Sensor type and image capture parameters.	
	Output:	Preprocessed raster image (GeoTIFF).	
	Functionality of the service:	Geometric and spectral transformation.	
Algorithm	For atmospheric effects radiometric enhancement techniques are applied. For geometric distortion affine transformations, or control-point based polynomial transformations may be used. The image is resampled in the process.		*The used techniques heavily depend on the type of sensor used. Hence, a predefined sensor parameter collection must be specified.*
Implementation details	**Implementation language**	C#	*The implementation is performed using the AEGIS [2] framework.*
	Dependencies with other libraries	None.	
	Operating system	Any operation system running .NET Framework or Mono.	
	Visualization modalities of the output	-	
Service characteristics	**Accuracy:** Depends on the sensor, the input parameters and other properties of the image.		
	Robustness: Reliable to work with images up to multiple GB-s (depending on system		

	memory size).	
	Computational time in relation to data size: The execution time can take multiple minutes depending on the number and type of corrections.	
	Locality/globality of the algorithm: Most corrections can be applied locally, but it depends on the methods used.	
Alternatives	Use functionality from other toolkit (e.g. Orfeo Toolbox).	
Related use cases	**[Generic data manipulation service]**	
Responsible Partner	FÖMI, Roberto Giachetta	
Involved Partners	FÖMI	

11.5 SPATIAL/TOPOLOGICAL OPERATIONS ON 2D DATA SETS, #74

IQmulus Service information: #74			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Spatial/topological operations on 2D data sets		*Topological analysis is a common feature in GIS toolkits.*
Description	The topological representation and analysis of 2D vector geometries is a common feature in most geospatial toolkits.		*The analysis can be extended to 3D vector data.*
Service functionality	**Input:**	2D vector data (Shepfile), or feature coordinates.	*The operation (id) can be passed as a parameter with operation-specific values (e.g. the distance in case of buffering).*
	Input parameters: *[optional]*	The operation, and optional parameters.	
	Output:	Result of analysis (Shapefile), or topological relations.	
	Functionality of the service:	Topological analysis.	
Algorithm	Multiple algorithms: • Bentley-Faust-Preparata (convex hull) • Bentley-Ottmann (line intersection) • Douglas-Peucker (polyline simplification) • Graham scan (convex hull) • Greiner-Hormann (polygon clipping) • Shamos-Hoey (line intersection)		
Implementation details	**Implementation language**	C#	*The implementation is performed using the AEGIS [2] framework.*
	Dependencies with other libraries	None.	
	Operating system	Any operation system running .NET Framework or Mono.	
	Visualization modalities of the output	-	
Service characteristics	**Accuracy:** High, as algorithms compute precise relations. Can be adjusted with geometry precision.		
	Robustness: As the construction of the topological representation is required, additional data structures are stored beside the original vector geometries. Thus, the		

	processable number of geometries is limited (depending on the number of points, number of intersection, etc.). Further versions will include caching to enable more data to be processed.	
	Computational time in relation to data size: As multiple algorithms, including transformations are executed, the process is time consuming, within the range of $O(n^2)$.	
	Locality/globality of the algorithm: Most algorithms function on the entire dataset.	
Alternatives	Use functionality from other toolkit (e.g. GeoTools).	
Responsible Partner	FÖMI, Roberto Giachetta	
Involved Partners	FÖMI	

11.6 EXTRACTION OF RAIN DATA, #96

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Extract_rainfall		*Unique identifier of the service; necessary to call it within user-defined workflows*
Description	This service reads the input data provided by the official Meteorological service and prepares the dataset for later elaborations. The service transposes the input table, computes the cumulative rainfall over the selected time interval, and groups the dataset by weather station.		*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>**	Enriched point set (MAT); Each row is represented as follow: time stamp (DTRF); rainfall (RAINC); each file have to be named with the ID of weather station	*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*

		(ANAG.CODE);	*this field.*
	Input parameters: *[optional]*	• Time interval over which the cumulative rainfall has to be calculated	
	Output: <data representation and format>	Enriched point set (MAT); Each row is represented as follow: ANAG.CODE RAINC	
	Functionality of the service: <text>	Pre-process input data;	
Algorithm	The service loads input data from files which each file refer to a weather station record. The mapper find for each weather station the data belonging to the right time interval and put to the reducer the pairs <weatherstationname,rainfall>. The reducer computes the cumulative rainfall for each weather station and right it to a file.		*The same functionality may in principle be implemented by different algorithms.*
Implementation details	**Implementation language**	Python	*Include information related to the implementation of the service, such as language (e.g., C, C++, etc); dependencies with other libraries (e.g., ANN library); constraints on the operating systems, and visualization modalities of the output.*
	Dependencies with other libraries		
	Operating system	Linux, Windows	
	Visualization modalities of the output		
IQmulus Data	**Available IQmulus input data**: #27;#42		*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. It depends on the accuracy of the input dataset. The service perform only sorting and arithmetic operations (sum) over the dataset.		*These fields are necessary to document all the characteristics of the services that are important to assess the quality of the results. Also, these fields could be used to select a specific service among more services that implement the same functionality but with different characteristics.*
	Robustness: Depends on the input data quality and smoothness		
	Computational time in relation to data size: $O(N^2)$		

	Locality/globality of the algorithm: Each weather station can be processed regardless of the other	*See the following box.*
Alternatives	No 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	Land scenario	*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	IMATI simone.pittaluga@vs30.it	*Partner ID and responsible person (include email)*
Involved Partners		

11.7 POINT CLOUD INTERSECTION, #98

IQmulus Service information			
Name of the metadata	**Content expected**		**Motivation/comments**
Service Acronym	Point Cloud Intersection		*Determine intersected area from the two imported point cloud datasets.*
Description	Two folders or single files containg overlapping point cloud datasets are imported. First intersecting files are identified by a bounding box analysis. Next the intersection for each pair of intersecting files is determined		*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 and *.las point cloud data Or, two folders containing such data	*The voxel parameter makes the method scalable. The smaller the voxel, the more accurate the results, but the higher the processing time*
	Input parameters: *[optional]*	Voxel size	
	Output: <data representation and format>	Two *.las files storing the points of the intersected area in each	

		input dataset Or, in case two folders were provided as input: an output folder is generated containing a series of files corresponding to the files of the 1st input folder	
	Functionality of the service:	Acquire the intersected area of the two imported point cloud datasets.	
Algorithm	This service is implemented based on voxels. Firstly, two voxels are generated with the imported point cloud datasets. Then the relative position of the vertices of each voxel cell is determined based on the geometry of a cell. Thirdly, all labelled vertices are categorized and a decision is made whether the cell is a overlapping voxel cell. Finally, the points in the voxel cell are then classified and exported. In case the input conists of two folder collections, first the bounding boxes of all files in the input folders are compaired pairwise. Files from different folders with non-empty bounding boxes are forwarded to the single file procedure.		*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	
	Operating system	Linux	
	Visualization modalities of the output	Points from the intersected area could also be outputted and displayed in a new window.	
IQmulus Data	**Available IQmulus input data**: For tests we use the following data; (1) AHN-2 data. Also in *.xyz format. (2) TU Delft campus LMMS point cloud data		
Service characteristics	**Accuracy:** a) The median or mean of the point to		

	point distances (Service #8) between the two output point clouds is small. b) The spatial extent (Service #11) of the two output point clouds is almost the same.	
	Robustness: Reliability is verified by testing several different point cloud data.	
	Computational time in relation to data size: A first analysis is given in the paper in the Appendix.	
	Locality/globality of the algorithm: The program can be used to quickly analyse if the point cloud data is intersecting. If so, this intersection is determined.	
Alternatives	No alternative within IQmulus	
Related use cases	This is a basic processing service	
Responsible Partner	TUDelft, Jinhu Wang (Jinhu.wang@tudelft.nl) and Beril Sirmacek	
Involved Partners	TUDelft, Roderik Lindenbergh (algorithm development support)	