INTERACTIVE VISUAL DECISION SUPPORT TECHNIQUES

Deliverable D5.1.3

Circulation:	PU: Public
Lead partner:	Fraunhofer
Contributing partners:	SINTEF
Authors:	Frank Michel, Thomas Gierlinger, Sascha Räsch, Miguel de Sousa (Fraunhofer) Vibeke Skytt, Sverre Briseid (SINTEF)
Quality Controllers:	Mathieu Bredif (IGN)
Version:	1.0
Date:	16.11.2015

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 (FOMI), Institute of Geodesy, Cartography and Remote Sensing, Budapest, Hungary
UCL	University College London (UCL), Research centre for Photogrammetry, 3D Imaging and Metrology, London, UK
TUDelft	Delft University of Technology (TUDelft), Department of Earth and Climate Sciences & 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	08.10.2015	Draft	Initial draft with deliverable structure
0.2	13.10.2015	Draft	Updated structure
0.3	17.10.2015	Draft	Initial content
0.4	06.11.2015	Draft	Updated content
0.5	12.11.2015	QC	Version ready for QC
1.0	16.11.2015	Final	Ready for submission

[1] Integers correspond to submitted versions

EXECUTIVE SUMMARY

This report presents the progress made in WP5 Interactive Visual Decision Support Techniques during the third year of IQmulus.

Due to the end-user requirements, to also look at original, non-processed big data, the WP5 work in year three of IQmulus was focused around processing, handling and accessing big data related to the show cases. In addition, we report on our continued work to extend and adapt the visualization modules, integrate with the IQmulus Infrastructure, measure the performance of the visualization and pre-processing, and extend the data import capabilities.

TABLE OF CONTENTS

1 INTRODUCTION

In year two of the *IQmulus* project the developed visualization techniques had been enhanced and new techniques had been added to the *IQmulus Fat Client,* to support more requirements from the show cases. However, due to the requirements to not only look at the final results, e.g. already processed and scaled-down data, but also the need for inspection of the original big data, the focus of WP5 was shifted to look into the pre-processing and handling of big data for visualization. This shift was also recommended by the reviewers at the period two review of the project.

Therefore, the WP5 work in year three of *IQmulus* was focused around processing, handling and accessing big data related to the show cases. In addition, work was conducted to extend and adapt the visualization modules, integrate with the *IQmulus Infrastructure* and measuring the performance of the visualization.

This is also expressed in the structure of this report which is split into the five parts

- Visualization and Big Data
- Large Data Handling
- Integration
- Benchmarking
- Visualization and Interaction

In the upcoming sections, we will describe which work has been done related to these topics.

Section 2 gives a short motivation about what needs to be taken into account when visualizing big data. The work done regarding pre-processing the data for visualization and how to access it is reported in section 3. How the integration with the *IQmulus Infrastructure* was achieved and how the necessary information for visualization is stored is detailed in section 4. Section 5 covers the work conducted to measure the performance of the *IQmulus Fat Client* and the pre-processing, and the new developments for visualization of the showcases and handling specific data sets is reported in section 6.

2 VISUALIZATION AND BIG DATA

When visualizing big data, especially big volumes of data, the size of the complete data sets exceed the capabilities of the used hardware by orders of multiple magnitudes. Typically a modern desktop computer is equipped with 16 GB of RAM and 2 GB of GPU RAM, while a larger data model can approach TB sizes.

However, in the case of visualization, the needed data to generate “a picture” is defined by the current viewpoint and view direction of the user. For a chosen view position, only the visible parts of the data are needed to generate the visualization. To be able to efficiently access the data in such a manner, that only the currently needed data is loaded into CPU/GPU RAM, the data needs to be processed into a certain structure. The most common structures for this task are octrees, kd-trees, and optimizations thereof. These structures divide space into smaller volumes, e.g. cubes, which contain only small parts of the data. When the structure is generated, the maximum number of data, e.g. points for point clouds, contained in each small volume can be set, resulting in an upper bound of data to be transferred for each volume. In addition, the bounding boxes of theses volumes are known and can be used to decide which parts of the data set are actually seen from the current viewing position. Using these bounding boxes a visualization system starts with accessing the data in the centre of the viewing frustum, and then gradually populates the "world". Larger data models are handled by dynamically moving data sets in and out of memory when the viewing position changes. This approach allows a smooth experience when navigating the model, with data fetching/discarding being handled in the background.

If the data is located on a remote infrastructure and should be visualized with non-PC hardware, e.g. in a web browser on a tablet, additional constraints come into focus. On the one hand, data access and transfer is a limiting factor when the visualization should be interactive. For example, accessing and dynamically moving gigabyte sized data chunks into memory is fast within a device, but transferring this amount of data through an internet connection is prohibitive. On the other hand, moving from specified PC hardware to mobile devices or utilizing the web browser for visualization, the capabilities for visualization can greatly differ. To cope with these constraints, the data needs to be segmented into even smaller chunks and converted to a format which can directly be used to generate the visualization. In addition the provisioning of the data to the client needs to be more flexible and adaptable to the available hardware and bandwidth resources.

In IQmulus we followed different approaches to address these topics, depending on the given scenarios, which are detailed in the following chapters.

3 LARGE DATA HANDLING

To cope with the size of the data, different approaches have been implemented, to take the data type as well as the intended use into account.

The data types with the highest priority due to its size and necessity to visualize them for the end users are:

- Point Clouds (LS1)
- LR B-splines + original points (MS1/MS2)

The data access can be separated into:

- local data sets
- data located at the remote infrastructure

In the following sections we describe the different approaches used for handling and providing the data to the end user.

3.1 VISUALIZATION PRE-PROCESSING

Local data

In the Marine Showcase, the data to be visualized consists of a number of partly overlapping data surveys (point clouds) and generated surfaces (LR-B splines). The pattern and properties of these data sets vary. In addition the accumulated data sizes may be too high, both for the purpose of surface generation and visualization.

Prior to surface generation, each data survey is divided into tiles. Individual surfaces are created for each tile and post-processed such that adjacent surfaces meet with C^1 continuity. Distance fields for the survey point clouds are also computed and stored for each tile. The data tiles, surfaces and distance fields represent a large amount of data. To create an interactive user experience, we have developed an acceleration data structure based upon the Hybrid Spatial Indexing (HSI). The HSI data structure consists of JSON metadata files, which describe survey sets, individual surveys and individual tiles. The same metadata files are used both during geometry processing and visualization. The metadata includes information about the geometries, and in particular a bounding box for each tile and distance field. This enables us to efficiently decide which tile is within the current viewing volume and calculate other metrics, such as the distance from our focus point.

The following metadata files are supplied:

- survey_set.json: This file contains a list of all the relevant data surveys including information about the number of points in each surface and a corresponding bounding box.
- survey.json: For each survey, we have metadata which references the actual data file in addition to information about the number of points and the bounding box of the data. If the data survey is tiled, a list of tile metadata files are given. Each tile entry also includes the number of points in the tile and the corresponding bounding box.
- tile.json: For each tile, we have metadata that contains a reference to the tile data, the number of points and a bounding box. If the tile also has a surface representation, the surface geometry is referenced and the bounding box of the surface is given. The surface can be trimmed or not, which means that the LR-B spline surface either has a rectangular or irregular shape. If a distance field computation is performed, the point cloud with distance field information is also referenced and some summary information regarding the distance between the surface and the points are given.

All data and metadata files corresponding to a model are stored in a folder structure in which file references within the metadata files are relative. This means that the metadata can be stored either separately from, or in the same folder structure, as the actual data. Both the metadata and the data itself can be stored on a local hard drive or the Hadoop Distributed File System (HDFS).

Remote data

In the Land Showcase (LS1) the data to be visualized consists of a large number of big point clouds from LIDAR surveys and generated high resolution triangulations resulting from a partitioning of the survey data into basins.

The point cloud data resulting from the processing services is not structured in a way, which can be directly used for visualization. Therefore, we implemented a visualization pre-processing service for point clouds. For the processing of the point clouds, we make use of the "PotreeConverter" [Potree] to generate an additive octree containing small chunks of the original data. The converter allows to set thresholds for point spacing, number of points per node and the number of octree levels to generate, which allows us to control the size of the resulting data chunks. The structure generated by the converter is afterwards transformed into our access structure (HSI) and the data chunks are converted into a binary format (SRC) which can directly be filled into the GPU buffer on the client side.

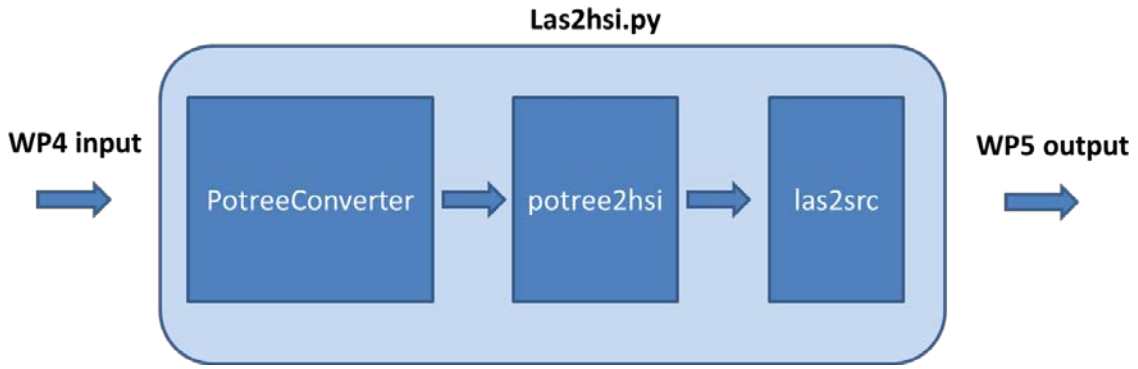

FIGURE 1: VISUALIZATION PRE-PROCESSING FOR LARGE POINT CLOUDS

The HSI is a tree structure, which contains for each octree node the bounding box, number of points, and a link to the actual data. This information is employed to efficiently identify the needed data for a current visualization view point. The generated structure is either accessed directly by the *IQmulus Fat Client* using the *IQmulus Data Access Service* or it is used as the basis for further preparation of the data to be visualized in the *IQmulus Thin Client*.

For the further preparation of the data to be used in the *IQmulus Thin Client* we employ the *webVis/instant3Dhub* [BMP*2015] platform. The platform combines a novel Web-Components-based framework with a Visual Computing as a Service (VCaaS) infrastructure, to deliver a powerful and comprehensive solution for interactive 3D data visualization. The system adapts to a given environment by providing client, server and hybrid visualization techniques and optimizing the delivery of complex data sets, resulting in an improved user experience. The *instant3Dhub* service platform provides network interfaces used by the web client framework *webVis*.

The development of the visualization pre-processing service only started in the current project period and is still in progress.

3.2 ACCESSING DATA

Local data

In the Marine scenario all data files are assumed to be accessible as files on a local or network mounted disk. The outcome of the visualization should be seamless

- surfaces
- point clouds including distance information, both originating from individual data surveys and combination of surveys
- surfaces and point clouds in the same view

The tiling is, thus, not visible in the view, but is used to select the files that, given a particular view, are required for visualization. This is especially important for point clouds which in most cases have much larger data sizes than the surfaces.

Smaller data models will fit entirely into both the RAM and GPU RAM. If only surfaces are to be visualized, the limits for keeping the model in memory is not likely to be reached, but the data size of point clouds with distance fields can be very high.

Larger data models are dynamically handled by moving data sets in and out of memory, and we process the geometry entities of the model until we reach a predefined CPU RAM threshold. Using the bounding box of each entity we start with the objects in the centre of our viewing frustum, and then gradually populate our "world". Geometry entities are then transferred to the GPU and added to the scene until we reach a predefined GPU RAM threshold. This threshold is typically much smaller than the CPU RAM threshold as the GPU has far less memory. This approach allows a smooth experience when navigating the model, with data fetching/discarding being handled in the background.

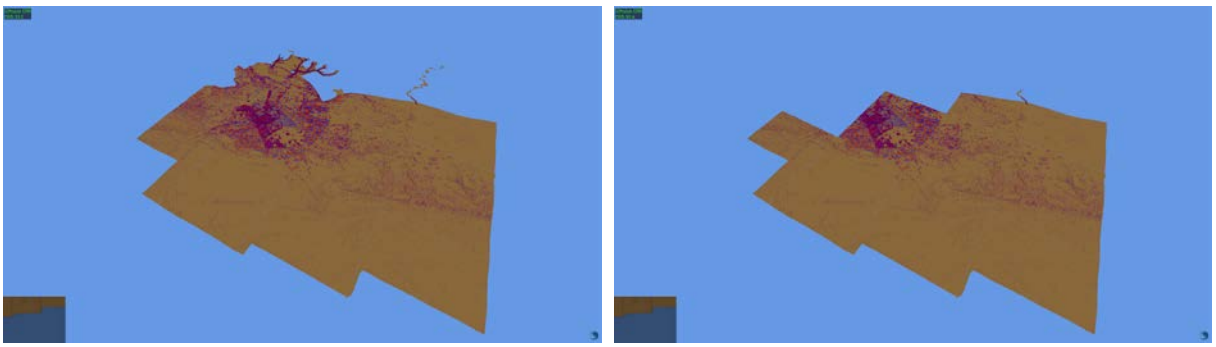

FIGURE 2: THIS FIGURE SHOWS SEVERAL LR B-SPLINE SURFACES WHICH REPRESENT 60 PARTLY OVERLAPPING POINT CLOUD SURVEYS. (LEFT) ALL SURFACES AND DATA POINT SETS ARE SHOWN. THE POINTS ARE COLOURED ACCORDING TO THEIR POSITION WITH REGARD TO THE SURFACE. RED POINTS LIE BELOW THE SURFACE AND BLUE POINTS ABOVE. NOTE THAT LR-B SPLINE SURFACES CAN HAVE A NON-TRIVIAL BOUNDARY, REFERRED TO AS A TRIMMED SURFACE. (RIGHT) ILLUSTRATES THE DYNAMIC DATA LOADING AND OFFLOADING. COMPARED WITH THE FIGURE ABOVE, THE ENTITIES FAR AWAY FROM THE CENTRE OF THE VIEW FRUSTUM ARE REMOVED TO CONSERVE GPU MEMORY.

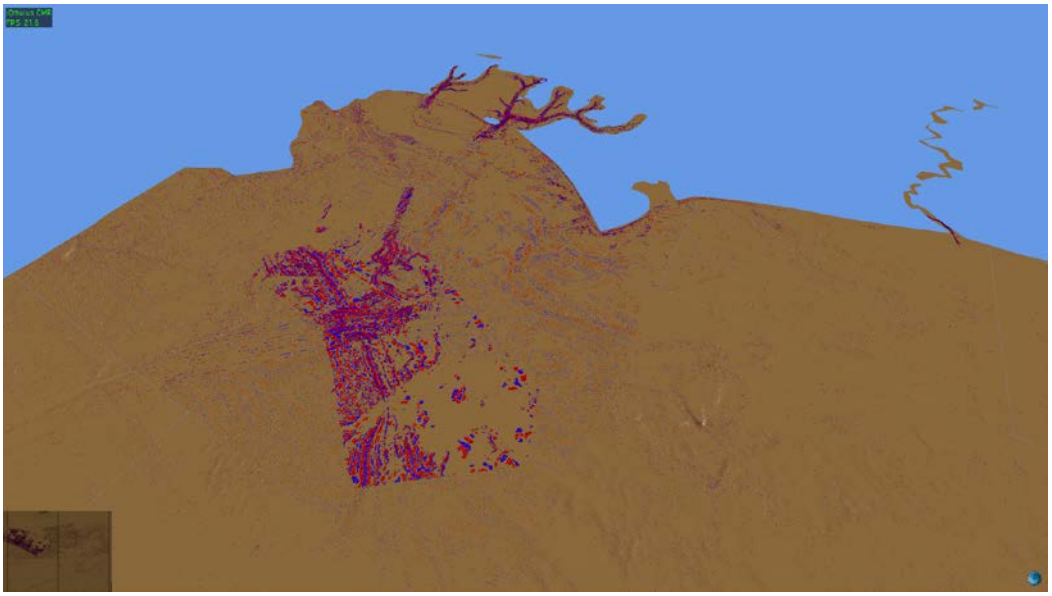

FIGURE 3: THIS FIGURE SHOWS HOW DYNAMIC DATA LOADING ALTERS THE SCENE. WHEN ZOOMING IN, ADDITIONAL SURFACES AND POINT CLOUDS ARE SENT TO THE GPU FOR RENDERING. NOTICE THAT THE POINTCLOUD DENSITY VARIES DRAMATICALLY BETWEEN DIFFERENT SURVEYS. IN THIS VIEW, ONLY POINTS THAT LIE OFF THE SURFACE ARE SHOWN. POINTS ABOVE THE SURFACE ARE COLOURED BLUE, WHILST POINTS BELOW THE SURFACE ARE RED.

Remote data

In order to access the data sets with the *IQmulus Thin Client* we employ the *webVis/instant3Dhub* platform, which uses out-of-core rendering, making it possible to visualize very large data-sets. The web client sets appropriate memory limits both on CPU and GPU and streams data from the server as necessary to generate an image from a given viewpoint. If the amount of data surpasses the available memory, it will iteratively refine the rendered image. In such cases, the web application is still responsive at all times. The user may move around the model, in which case the available data is rendered and the refinement will restart when the navigation stops.

The *webVis* framework provides a solution for embedding 3D visualization of structured data in Web applications with minimal effort. Based on JavaScript, it runs on modern web browsers without the need for external plug-ins. The *webVis* API offers all the functionality needed for accessing, visualizing and manipulating data originating from the *instant3Dhub* infrastructure.

4 INTEGRATION

To integrate the modules of the visualization into the IQmulus architecture, different interfaces to the infrastructure have been implemented in year 3. These can be divided into:

- *IQmulus Fat Client* Data Access to the *IQmulus Infrastructure*
- Meta data for visualization

4.1 DATA ACCESS TO THE IQMULUS INFRASTRUCTURE

In order to access the data provided by the processing services of the *IQmulus Infrastructure*, we developed a data access module for the *IQmulus Fat Client*. The module utilizes Qt 5 for user authentication and data access via the web-based interface to the HDFS of the *IQmulus Infrastructure*. More specifically, we developed a connection manager to establish secure communication with the *IQmulus Data Access Service* (DAS) and adapted the Webkit-based HTML engine provided by Qt (QWebView) to download and visualize data sets from the HDFS. The main components involved in HDFS data access in the *IQmulus Fat Client* are shown in Figure 4:

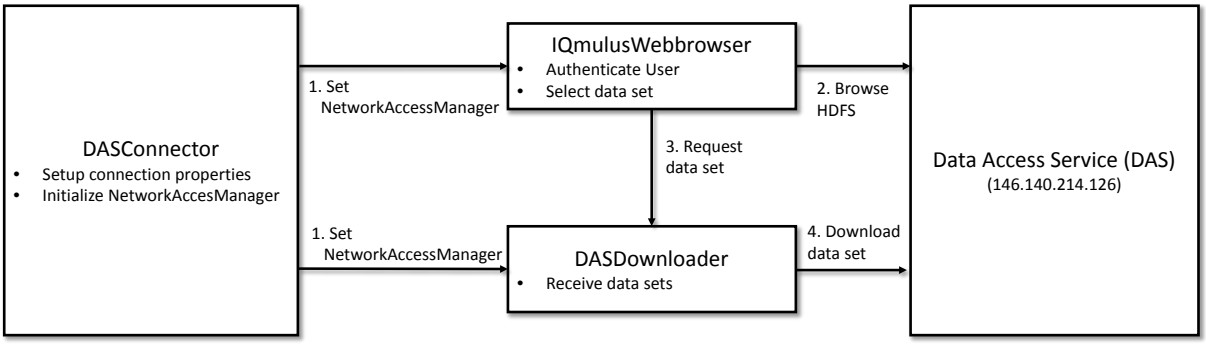

FIGURE 4: COMPONENTS OF THE *IQMULUS FAT CLIENT* INVOLVED IN ACCESSING DATA IN THE *IQMULUS INFRASTRUCTURE*.

In the following we provide an overview of the functionality of the components of the data access module:

DASConnector

The DASConnector is responsible for configuring a secure connection to the *IQmulus Data Access Service*. This requires the initialization of SSL parameters (we currently use TLS1/SSL3), loading the certificate of the *IQmulus Data Access Service*, and setting up security / functionality related properties of Qt's HTML engine (e.g. configuration of JavaScript capabilities, WebGL, etc.). The connection relevant settings are subsumed in a Qt class termed QNetworkAccessManager. A relevant property of this class is the internal handling of cookies, which are required during the authentication process of the *IQmulus Data Access Service* and must remain accessible for each

connection request after authentication. The configured NetworkAccessManager is provided the other components for secure access to the *IQmulus Data Access Service*.

IQmulusWebbrowser

The IQmulusWebbrowser is an extension to the HTML engine provided by Qt (QWebView). It is the front end the user utilizes to submit authentication information to the *IQmulus Data Access Service*, browse the HDFS and initiate the download and visualization of a data set. Inheritance from and extension of the default QWebView implementation was necessary in order to gain full control of the sizing policies and for the implementation of custom context menus (as shown in Figure 5).

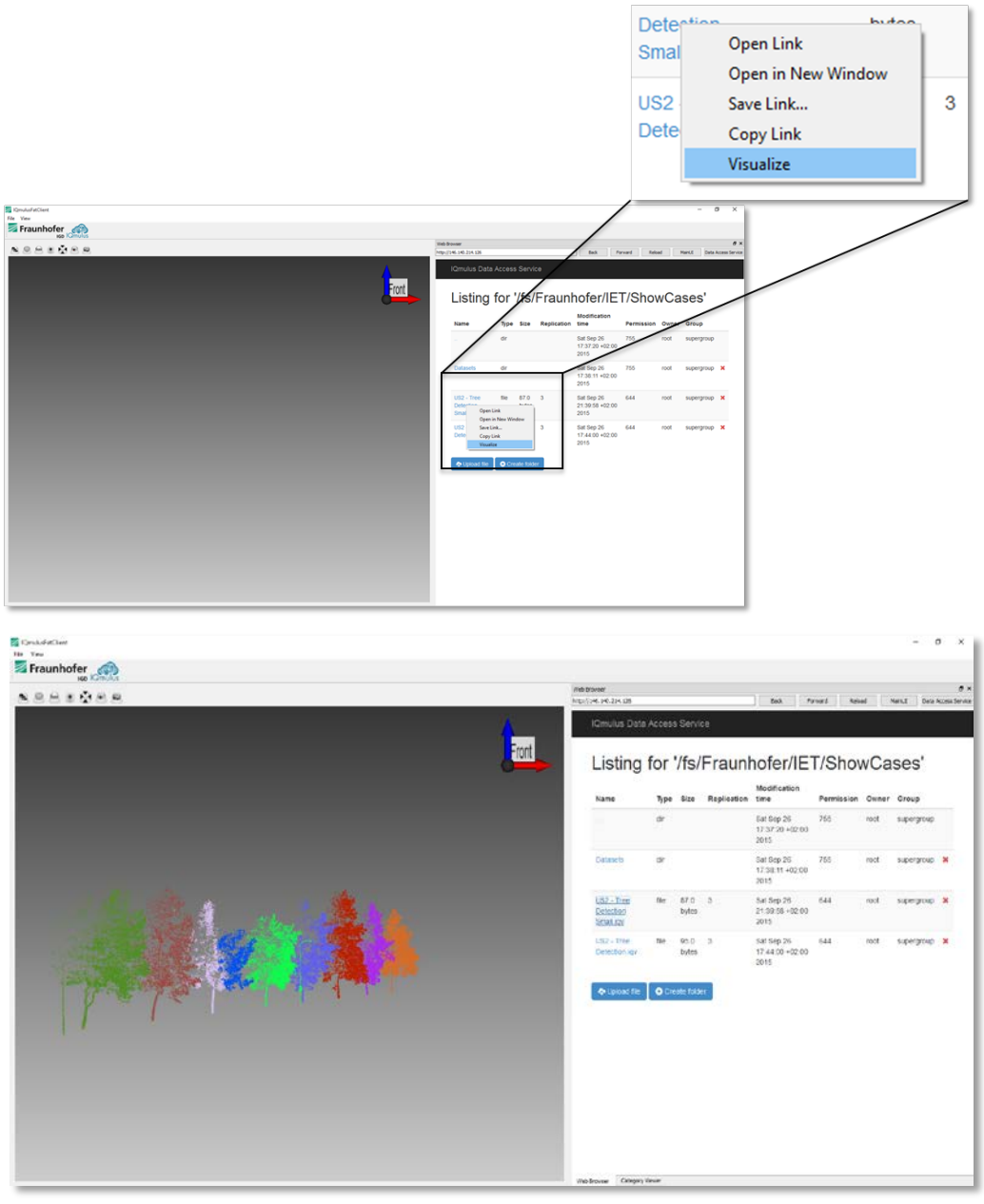

FIGURE 5: IQMULUS WEBBROWSER WITH CUSTOM CONTEXT MENU FOR VISUALIZATION OF DATA SETS.

DASDownloader

When the user has selected a data set in the HDFS, the corresponding link is passed to the DASDownloader for asynchronous download. Once the data download is finished, it is loaded into the *IQmulus Fat Client* for visualization.

An interesting opportunity provided by the availability of an integrated web browser in the *IQmulus Fat Client* is the possibility to display and use the *IQmulus Main UI* from "inside" the *IQmulus Fat Client*. Furthermore, it may be possible to directly preview *IQmulus Thin Client* visualizations (depending on the current status of QWebView). Currently there are minor issues with the integrated web browser including restricted image format support.

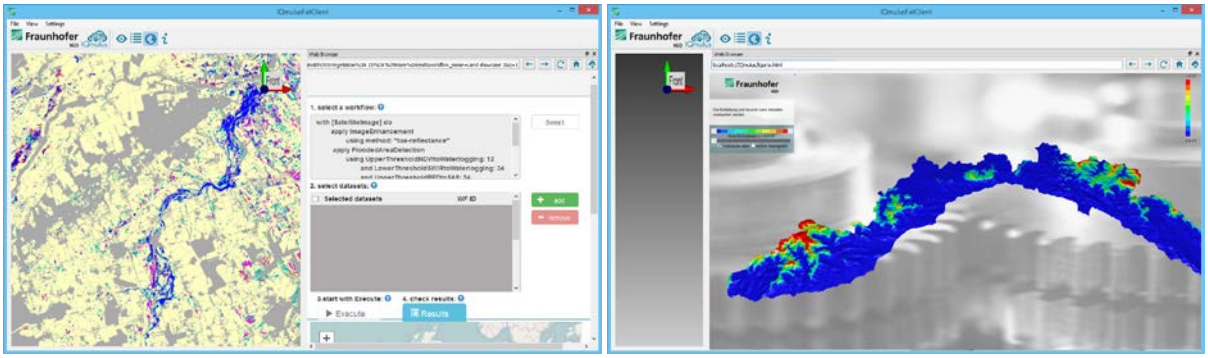

FIGURE 6: *IQMULUS FAT CLIENT* INTEGRATED WEB BROWSER. (LEFT) IQMULUS MAIN UI, (RIGHT) X3DOM / WEBGL.

4.2 IQMULUS VISUALIZATION FILE

To be able to generate, save and load customized visualizations we defined the *IQmulus Visualization File* (IQV/.iqv), which describes all the necessary information for a visualization scenario. The metadata includes basic visualization settings (such as the showcase context), file locations/names (on the local hard drive or the remote hdfs) as well as customizable settings (e.g. thresholds for colour mapping, view positions, etc.).

```
{
    "Context": "MS4",
    "URLForHeightImage": "Datasets/MS4/MBES_2010_Four_UTM30_formated_sf.tif",
    "URLForXOffset": "Datasets/MS4/Px1_Num2_DeZoom1_LeChantier.tif",
    "URLForYOffset": "Datasets/MS4/Px2_Num2_DeZoom1_LeChantier.tif",
    "URLForMaskImage": "Datasets/MS4/Masq_LeChantier_DeZoom1.tif"
}
```

FIGURE 7: EXAMPLE IQV FILE FOR THE MS4 SHOW CASE.

An example of an IQV file is given in Figure 7. It currently provides links to the sub data sets required for the visualization of the dune migration vector field. Future iterations will include correlation data for colour mapping as well as colour map settings (colour map, mapping method, min / max mapping values).

The IQV file can either be generated by a workflow to provide initial settings for the visualization of the workflow results or the user can save a visualization he prepared using the *IQmulus Fat Client* to store and share the settings.

5 BENCHMARKING

The visualization components developed within *IQmulus* utilize the possibilities of modern graphics cards and thus inherently need a GPU. In addition, the visualization components run on the user's local computer/hardware which results in different preconditions for testing. Therefore, the methodology used to measure the performance of the visualization components differs from the methodology for processing services running on the *IQmulus Infrastructure*.

Due to the differences in measurable attributes for the visualization modules, we implemented a benchmarking framework for the *IQmulus Fat Client*, which is currently being adapted to the different visualization modules.

The development of the visualization pre-processing only started in the third project year as a task not originally planned in WP5. Because of this, the visualization pre-processing service and the benchmarking framework is still in development/optimization so the final benchmarking results are not complete at this time.

5.1 FAT CLIENT BENCHMARKING FRAMEWORK

In order to support benchmarking the *IQmulus Fat Client* with multiple data sets and multiple hardware configurations without manual interaction, we developed a framework for automatic benchmarking and logging. The framework instruments key components of the *IQmulus Fat Client*, namely the data import and rendering module, and gathers performance data. Performance indicators are sampled over time (the time between samples may be provided by the user, the default is 1 sec.). The resulting values are stored in an Excel sheet (.csv file). Report generation, i.e. generating graphs from the measured data, is not yet automated but foreseen for the final version of the framework. We also implemented a simple animation (zoom in/out of the data set, and rotate around the data set bounding box) which may provide insight into the run-time behaviour of the system w.r.t. the portion of the dataset being visible. Finally we are taking screenshots at key events during the benchmark, namely after loading, at the start of the animation, half way through the zoom animation (zoomed-in state), and at the end of the animation. The screenshots are required for verification of a benchmark run, i.e. making sure the correct model was loaded and visible. The performance indicators tracked by the framework are

- Scene characteristics (measured once)
    - Loading time: The time required to load the data set
    - Number of primitives in the data set
        - Number of Lines

        - Number of Points
        - Number of Triangles
- Run-time characteristics (measured continuously)
    - CPU usage
    - CPU RAM usage
    - GPU RAM usage
    - Rendering speed (frames per second)

The performance indicators to track and the use of animations can be programmatically changed. We may expose this configuration possibility via command line switches in the future. The GPU memory consumption is measured using vendor-specific OpenGL extensions. The used extensions are GL_NVX_gpu_memory_info for NVIDIA cards and GL_ATI_meminfo for AMD cards. We tested the benchmarking framework with data from the Basin-127 data set (scanned point clouds related to a specific area / basin in Liguria). We use five different point clouds, the characteristics of the sub data sets are depicted in Figure 8.

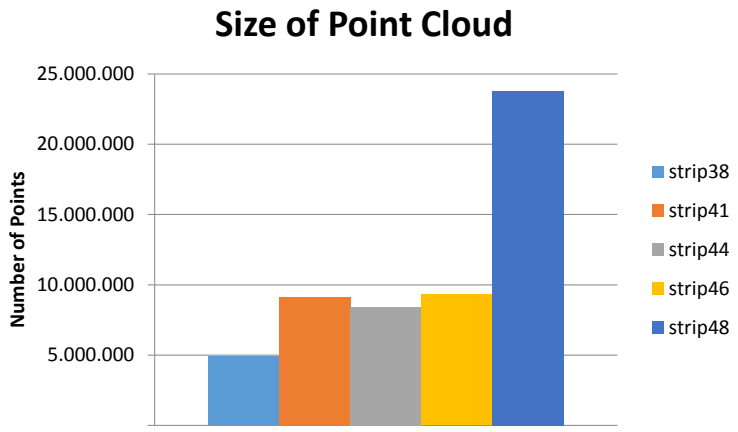

FIGURE 8: BASIN-127 TEST DATA SET SIZES

Figure 9 shows an example measurement for the files from the Basin-127 data sets on a Core-i7-5820K, 16GB RAM, NVIDIA GTX 750Ti (2GB) machine on the top and the same configuration with a different graphics card (AMD Radeon 6450 (1GB)) at the bottom.

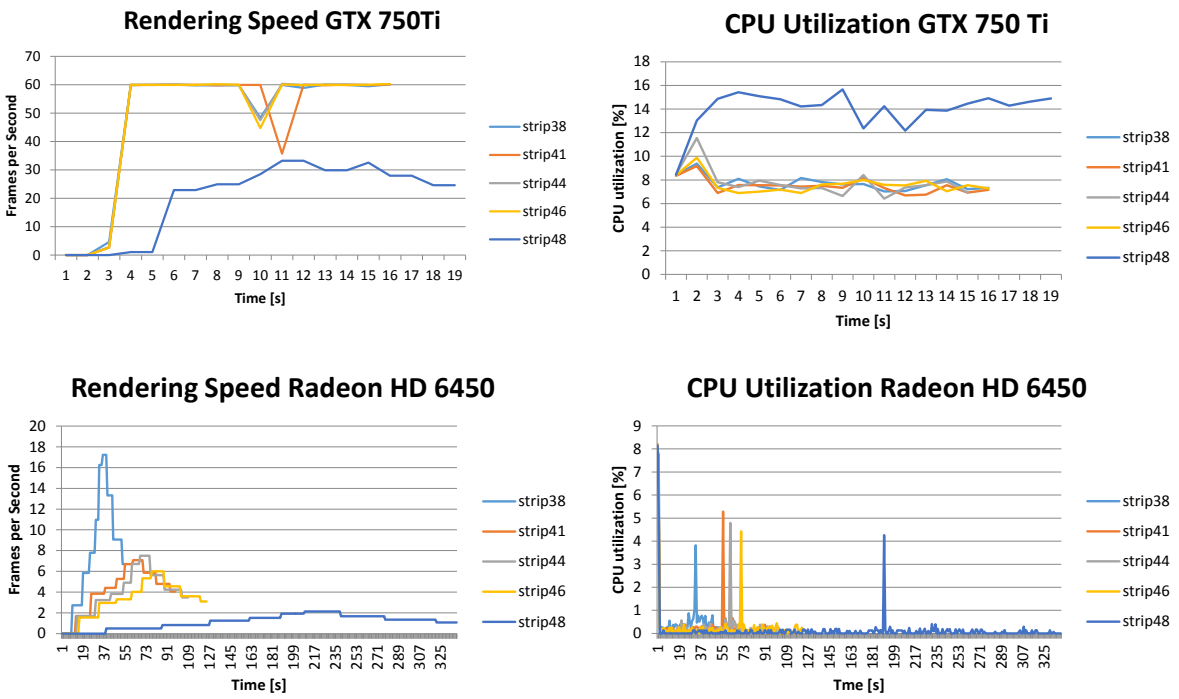

FIGURE 9: EXAMPLE OF CURRENT BENCHMARK RESULTS FOR MID-RANGE NVIDIA CARD (TOP) AND LOW-END AMD CARD (BOTTOM).

The spikes in frame rate of the measurement for the NV GTX 750 Ti are attributed to the intermediate screenshot during animation. We have not disabled V-Sync for the initial benchmarking experiments, this will be changed for the final tests. As can be seen in the graphs for the (low-end) AMD HD 6450, the rendering in this setup is GPU-limited. We also have to state that the current implementation of the animation sequence is synchronized on a per-frame basis, which can lead to problems in comparing results on different hardware configurations. Therefore, we were not able to provided overall performance testing results yet.

5.2 VISUALIZATION PRE-PROCESSING

The development of the visualization pre-processing service for remote data has only started in the current project year and is still under continuous development. For benchmarking the service, we adapt the available methodology for scalability testing of the processing services. However, we need to extend the available system, as the visualization pre-processing combines different processing steps into one service, which should be measured individually. This is still work in progress, as the service itself is under continuous development and optimization.

6 VISUALIZATION AND INTERACTION

6.1 VISUALIZATION FOR SHOW CASES

Initial developments to visualize the results of the show cases defined in the project were reported at the end of the second project period. These developments have been extended and new show-case-specific visualizations have been developed during the third project period. The following sections provide an overview of the work performed for each supported show case.

MS4: Measuring submarine dune migration

The visualization of submarine dune migration utilizes GPU-generated (tessellated) arrow glyphs to illustrate the vector field of submarine dune displacements [GBS*2015]. While the generation of the glyphs via tessellation shaders remains unchanged compared to first visualization prototype, several glyph properties were made adjustable and accessible by the user. These properties include

- The glyph spacing strategy
- Length and scale of the glyphs
- Material and lighting settings
- Level-of-detail of the tessellation

Furthermore, the visualization has been extended to support colour mapping of scalar attributes attached to the glyphs. Figure 10 shows the current MS4 visualization of a workflow result downloaded from the *IQmulus Data Access Service*.

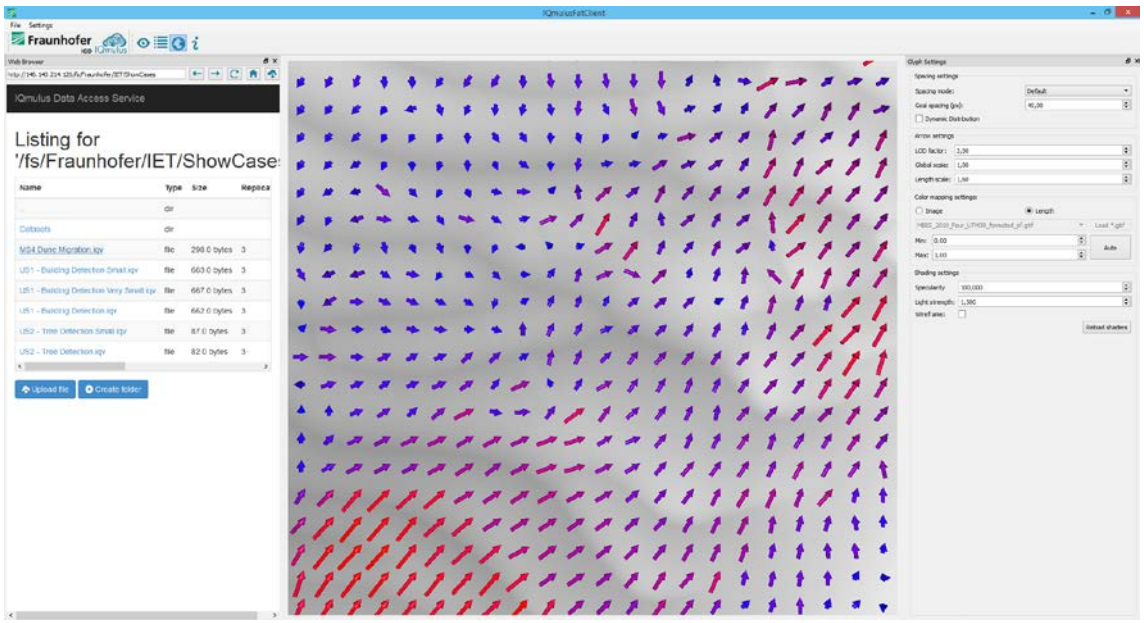

FIGURE 10: CURRENT VISUALIZATION OF THE DUNE MIGRATION SHOW CASE (MS4)

US1: Detection of buildings for monitoring and cadastral mapping

The visualization of the results of a building detection workflow is comprised of an image of the area under consideration and several shape files representing different classes of detected buildings (new, changed, removed, etc.). During the third project period we developed a layer view concept which interprets each for the aforementioned data sets as a layer. The final visualization is generated by composing these layers (currently via sequential rendering to the z-buffer). The visibility of layers may be changed using the list style user interface. Work on generalizing the usability of the layer concept by providing different rendering methods for different layers has started. An example of the current US1 visualization of a workflow result provided via the *IQmulus Data Access Service* is depicted in Figure 11.

FIGURE 11: CURRENT VISUALIZATION OF THE BUILDING MONITORING SHOW CASE (US1)

US2: Individual tree extraction from urban LMMS data

The individual tree extraction workflow detects trees in a laser scanned point cloud and assigns a unique identifier to all points associated with a single tree. For the purpose of visualization of the resulting dataset we developed a category viewer which performs a distinct colour mapping of the points of different trees. Trees may be highlighted based on their ID and colours as well as point rendering parameters may be manually specified. The colour map is currently randomly generated, but we are working on a perceptual uniform colour distribution. The current visualization for US2 results provided through the *IQmulus Data Access Service* is shown in Figure 12.

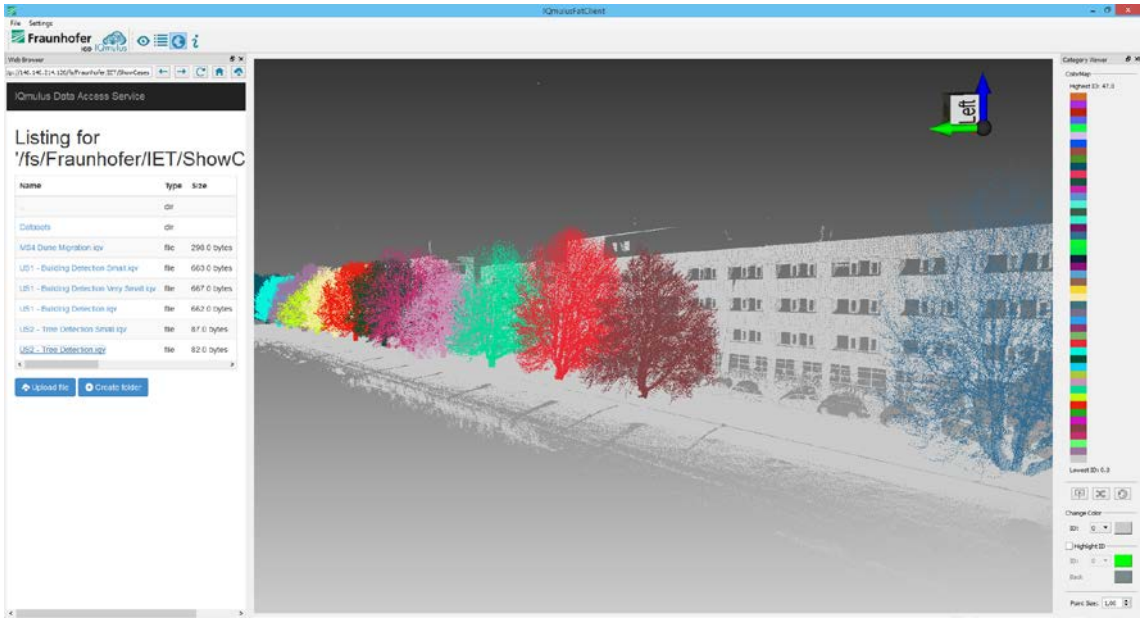

FIGURE 12: CURRENT VISUALIZATION OF THE TREE EXTRACTION SHOW CASE (US2)

MS1/MS2: LR-Spline Surfaces and distance fields

The LR B-spline surfaces are capable of accurately representing shapes with local details, but there will, in a typical geographical point cloud, be points that are not resolved by the surface. It can be outliers, points representing vegetation or the surface is created with a lower accuracy. Moreover, several data sets can be acquired from the same area at different points in time and the bathymetry may have changed in the meantime. This is typical for sand dunes. Also, different data sets representing the same area can give conflicting information due to acquisition with different equipment and accuracy or problems in registration. Visualization of distance fields is useful for capturing such effects. The points can be coloured according to their distances to the surface and moved in or out of view selected by distance and user interaction. It is also possible to colour points close to the surface different from the points further away. If this, for instance, results in one or a few points with large distances to the surface in an area where the majority of the points lie within an acceptable distance, this indicates that the distant points are outliers. The tailored visualization becomes an important tool in the data analysis. In the figures below a detail of a set of surfaces approximating several overlapping data sets is shown. Since the different data sets not necessarily yield a completely consistent picture, one data set is selected for further analysis.

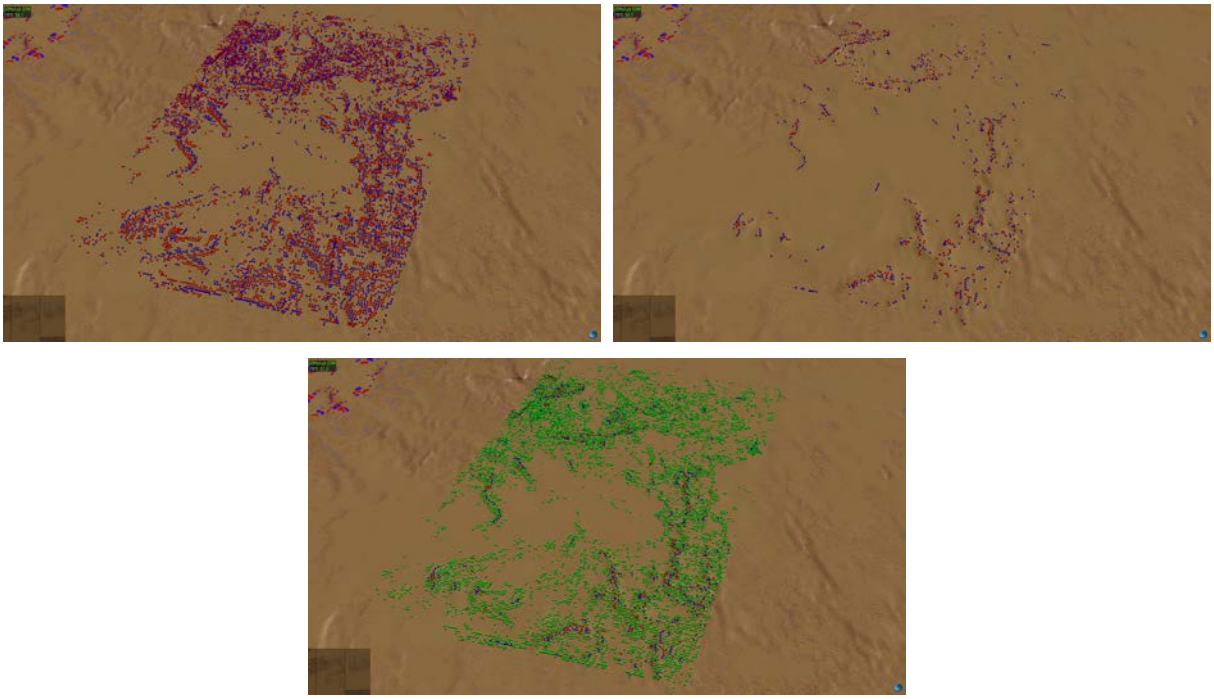

FIGURE 13: (TOPLEFT) ONE OF THE INITIAL POINT CLOUDS IS HIGLIGHTED IN A DETAIL OF THE SCENE. RED POINTS LIE BELOW THE SURFACE AND BLUE POINTS ABOVE. POINTS THAT LIE ON THE SURFACE ARE HIDDEN, AS THEY ARE WELL REPRESENTED BY THE SURFACE. (TOPRIGHT) THE MOST DISTANT POINTS ARE SHOWN EMPHASIZING WHERE THE SURFACE DOES NOT REPRODUCE THIS POINT SET ACCURATELY. THESE POINTS CAN REPRESENT SHARP FEATURES THAT ARE NOT WELL REPRESENTED BY LR-B SPLINES OR OUTLIERS IN THE SURVEY. (BOTTOM) THE POINTS CLOSE TO THE SURFACE ARE HERE SHOWN IN GREEN. AS THE MAJORITY OF THE POINTS ARE GREEN AND THE REMAINING POINTS LIE ON BOTH SIDES OF THE SURFACE, WE CAN, FROM THE VIEW, CONCLUDE THAT THE SURFACE CAPTURES THE MAIN FEATURES OF THE POINTS WELL, BUT THE FINER DETAILS ARE SMOOTHED OUT.

The purpose of the visualization is not necessarily data analysis. The aim can be to view the shape of the landscape or sea bottom. Since the surface represents the same topography or bathymetry as the point cloud, the points being most distant to the surface are the only ones that need to be visualized in addition to the surfaces. We can then reduce the data size for visualization by setting a minimum threshold for the absolute distance, removing points that are close to the surfaces. An example of this functionality is shown in the pictures below. The threshold value can be dynamically altered, although this is a costly operation since it means that the point clouds need to be reparsed and the data sent to the GPU.

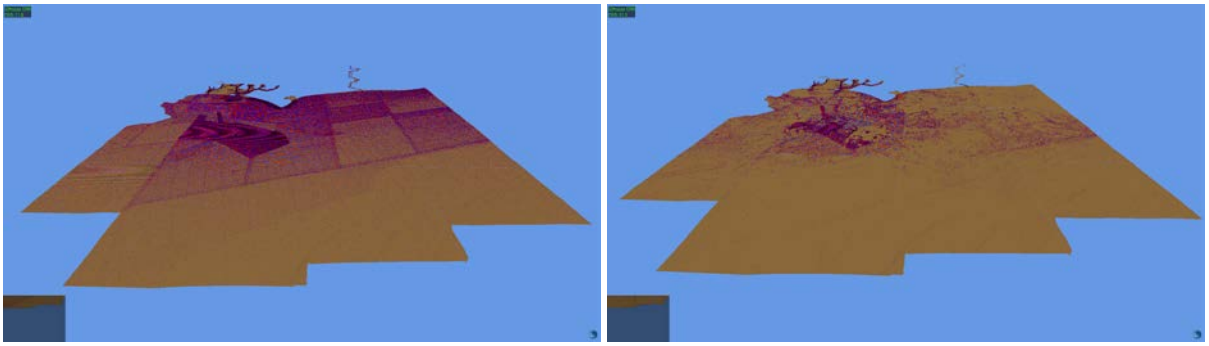

FIGURE 14: A NUMBER OF POINT CLOUDS ARE SHOWN TOGETHER WITH SURFACES APRPOXIMATING THE POINTS. (LEFT) ALL THE POINTS ARE SHOWN, RED POINTS BELOW THE SURFACE AND BLUE POINTS ABOVE. (RIGHT) THE POINTS LYING CLOSE TO THEIR RESPECTIVE SURFACE ARE DISCARDED WHEN READING FROM THE FILE

6.2 POINT-BASED RENDERING: AUTOSPLATS

In the previous report we described our approach to advanced point-based rendering with an adapted AutoSplats [PJW2012] algorithm. One goal of this work is to increase the visual quality for investigation of raw point dataset without the requirement of a pre-processing step. The algorithm requires as input only the point coordinates (it may utilize normals and colours if available, but this is optional). No derived information like local point density, splat size, or splat orientation is required. Splat sizes and orientations are calculated on-the-fly on the GPU.

Our previous implementation was developed as a stand-alone OpenGL prototype. This work has been integrated into the *IQmulus Fat Client* by developing an *OpenSG* Stage node, which performs the rendering by utilizing off-screen buffers and geometry shaders. Furthermore, the initial developments were enhanced to support vertex attributes (e.g. per vertex colours) and advanced lighting calculations. Figure 15 demonstrates the current state of these developments with a tile of the Toulouse data set (2.2 million points, filename Tiles3/1093-1197.ply). A standard point rendering is shown in the left column, the results of AutoSplats rendering is shown in the right column. As can be seen, the structure of the point cloud is much more visible in the splatted reconstruction of the surface.

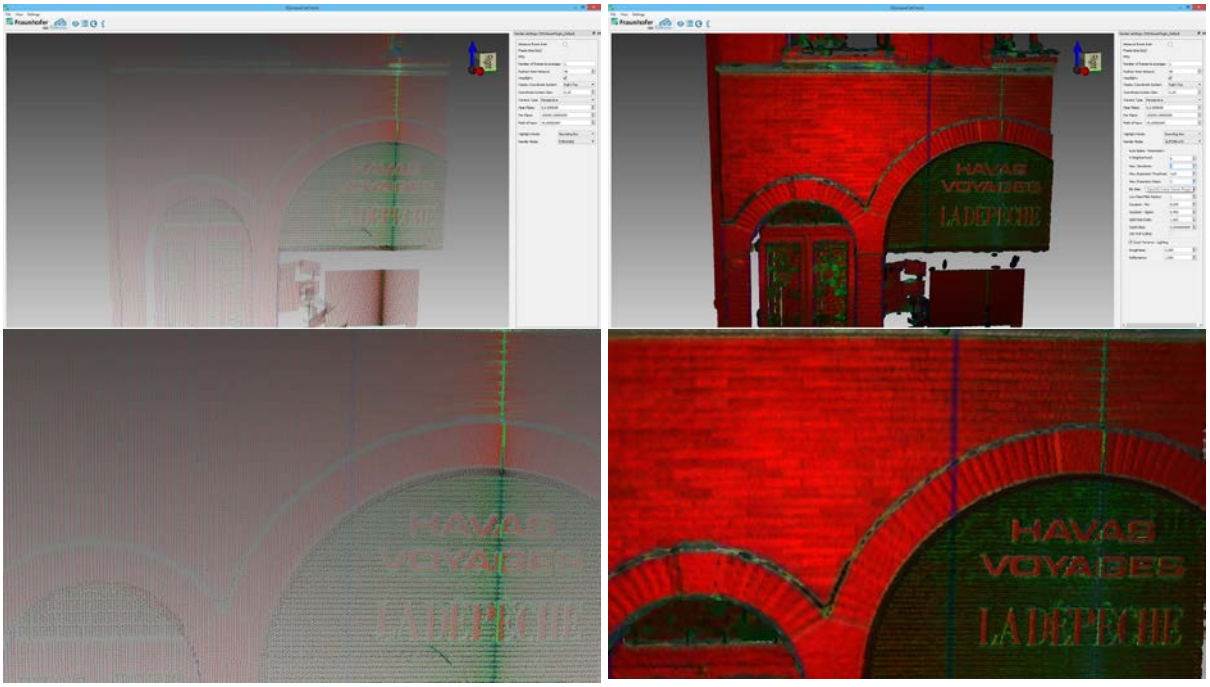

FIGURE 15: STANDARD POINT RENDERING (LEFT) AND AUTOSPLATS RENDERING (RIGHT) OF A TILE OF THE TOULOUSE DATA SET.

However, the increased visual quality comes at the price of increased computational complexity and decreased frame rates. The run-time requirements of the algorithm depend on a variety of factors including size of the point cloud, screen size, splat size, and number of neighbours and search radii of a screen-space nearest neighbour search. These dependencies make it difficult to state the general performance of this rendering method. The images above were generated on a Laptop (Core-i7, 16GB RAM, NVIDIA GeForce GTX 860M, 4GB GPU RAM). Depending on the parameter settings the rendering performance on a Full-HD monitor was between 1 and 20 frames per second, which indicates that this rendering method requires a strong GPU to deliver interactive frame rates.

We are currently developing a client-server rendering architecture for visualizing large point clouds and triangle meshes by point-sampling the data set on the server-side and using these point samples together with the splatting algorithm to reconstruct a surfaces on the client. This approach should allow a user to analyse data sets which are too large to be transmitted, stored, and visualized on the client machine. Compared to a pure image-based visualization approach where the server generates an image and transmits it to the client, our approach allows for local navigation on the client without per frame communication with the server. Of course a capable server is required for the point sampling process. This approach is work in progress, we hope to provide first results in time for the review.

6.3 EXTENDED DATA SET IMPORT

The data import facilities of the *IQmulus Fat Client* have been extended during the third project period to support user defined data. The following extensions to import modules were implemented:

PLY Import

The importing capabilities of the PLY Loader were extended to be able to handle non-standard attributes manually as well as automatically. The loader recognizes the most common attribute names for:

- Positions (x, y, z)
- Colours (r, g, b, red, green, blue)
- Normals (normalx, normaly, normalz, nx, ny, nz)
- A texture file
- Texture coordinates

and loads them into the corresponding slots (vertex attributes) in the geometry. If the loader finds any non-standard or unknown attribute it presents the user a dialog, where the preferred slots can be selected (Figure 16 left).

FIGURE 16: PLY LOADER: (LEFT) CUSTOM ATTRIBUTE DIALOG AND (RIGHT) JSON-FILE WITH ATTRIBUTE MAP

Furthermore, there is the possibility to describe the custom attributes in an additional JSON-file which should have the same name as the PLY-File with the suffix „-attributemap“ (e.g. filename.ply, filename-attributemap.json). In this file one can specify the specific slot for each custom property (Figure 16 right). Note that in this file only non-standard attributes are

described, as the loader already knows the specific slots for the standard ones. The loader recognizes this file and loads the attributes accordingly.

XYZ Import

The XYZ Loader was also extended to handle more than the standard position (x, y, z) attributes. First of all, if the file has more than three values (4 or 6) per vertex it will load these attributes automatically into the TexCoord0 (X or X, Y, Z) slots. Other configurations of custom attributes (e.g. 5 or more than 6) will be discarded. Additionally, there is the possibility to describe any number of custom attributes by a JSON-file (with the same name as the loaded file, e.g. filename.xyz and filename.json). This file contains the names of the corresponding values in the XYZ-File (Figure 17 right). The loader recognizes this file and presents the user a dialog (Figure 17 left), similar to the PLY custom dialog) where the corresponding slots can be specified.

Finally, there is also the ability to automate the process of loading custom elements into corresponding slots by using an "attributemap" JSON-file (Figure 16 right), similar to the PLY Loader (e.g. filename-attributemap.json). This file contains only the additional values and the corresponding destination slots. The loader automatically recognizes this file and loads the values accordingly.

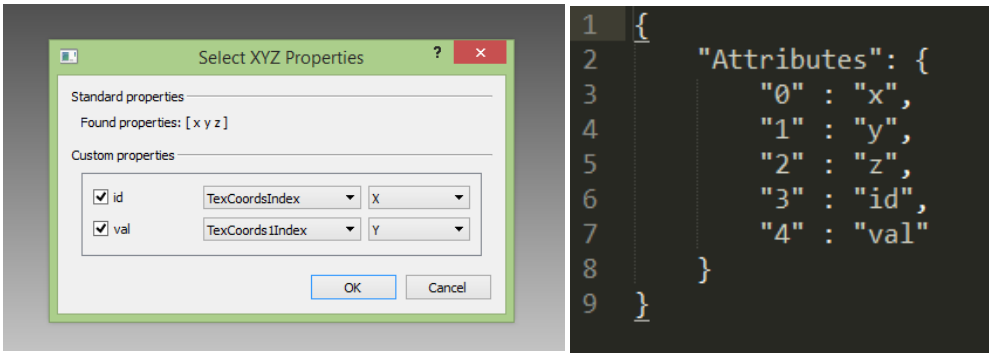

FIGURE 17: XYZ LOADER: (LEFT) CUSTOM ATTRIBUTE DIALOG AND (RIGHT) JSON-FILE WITH ATTRIBUTE DESCRIPTION

7 REFERENCES

[Potree] http://www.potree.org

[BMP*2015] Johannes Behr, Christophe Mouton, Samuel Parfouru, Julien Champeau, Clotilde Jeulin, Maik Thöner, Christian Stein, Michael Schmitt, Max Limper, Miguel de Sousa, Tobias Alexander Franke, and Gerrit Voss. 2015. webVis/instant3DHub: visual computing as a service infrastructure to deliver adaptive, secure and scalable user centric data visualisation. InProceedings of the 20th International Conference on 3D Web Technology (Web3D '15). ACM, New York, NY, USA, 39-47. DOI=http://dx.doi.org/10.1145/2775292.2775299

[GBS*2015] Thomas Gierlinger, Andre R. Brodtkorb, Andre Stumpf, Marcel Weiler and Frank Michel. Visualization of Marine Sand Dune Displacements utilizing Modern GPU Techniques. ISPRS Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XL-3/W3.

[PJW2012] Reinhold Preiner, Stefan Jeschke, Michael Wimmer. Auto Splats: Dynamic Point Cloud Visualization on the GPU. In Proceedings of Eurographics Symposium on Parallel Graphics and Visualization, May 2012