Taming the beast: Free and open-source massive point cloud web visualization

Oscar Martinez-Rubi[1,*], Stefan Verhoeven[1], Maarten van Meersbergen[1], Markus Schütz[2], Peter van Oosterom[3], Romulo Gonçalves[1], and Theo Tijssen[3]

[1]Netherlands eScience Center, Amsterdam, The Netherlands
[2]Institute of Computer Graphics, Vienna University of Technology, Austria
[3]Section GIS technology, Department OTB, Faculty of Architecture and the Built Environment, TU Delft, The Netherlands
[*]o.rubi@esciencecenter.nl

ABSTRACT

Powered by WebGL, some renderers have recently become available for the visualization of point cloud data over the web, for example Plasio or Potree. We have extended Potree to be able to visualize massive point clouds and we have successfully used it with the second national Lidar survey of the Netherlands, AHN2, with 640 billion points. In addition to the visualization, the publicly available service at http://ahn2.pointclouds.nl/ also features a multi-resolution download tool, a geographic name search bar, a measurement toolkit, a 2D orientation map with field of view depiction, a demo mode and the tuning of the visualization parameters. Potree relies on reorganizing the point cloud data into an multi-resolution octree data structure. However, this reorganization is very time consuming for massive data sets. Hence, we have used a divide and conquer approach to decrease the octree creation time. To achieve such performance improvement we divided the entire space into smaller cells, generated an octree for each of them in a distributed manner and then we merged them into a single massive octree. The merging is possible because the extent of all the nodes of the octrees is known and fixed. All the developed tools are free and open-source (FOSS) and they can be used to visualize over the web other massive point clouds.

1 Introduction

The popularity and usage of point cloud data is growing exponentially. Governmental institutions and companies are collecting large point clouds of the surfaces of countries and cities using Lidar, photogrammetry or other survey technologies. The Netherlands has a national program called Actueel Hoogtebestand Nederland[1] (AHN) that is in charge of periodically collecting point clouds of the country surface. The data set that was produced from the second scanning campaign, AHN2, has 640 billion points and it is open data. Currently the third scanning campaign is in progress. These data sets are widely applied, e.g. to monitor the dike infrastructure that protects the country from possible floods, in 3D city models used, in climate simulations, etc.. In many of the applications a common desired functionality is the interactive visualization.

The visualization of massive data sets is complicated due to their sizes which are usually larger than the main memory of the systems that have to handle them. For example the size of AHN2 is 1.6 TB with compressed and sorted LAZ.[2] In order to avoid sub-sampling the data (decreasing point density) or limiting their visualization to only small portions at once, multi-resolution data structures are used. In previous work[3] we identified and benchmarked the generic point cloud data management solutions available on the market, but these systems are oriented to analytic purposes and do not offer efficient support for the data structures used for point cloud visualization. Thus, specific data management solutions are required for visualization.

A few years ago point cloud visualization was limited to desktop-based solutions but after the introduction of WebGL several web renderers have become available. One of them is Potree[4] which uses a multi-resolution octree data structure to deal with large data sets. However, its usability is limited to few billions points due to amount of computation required to create the multi-resolution octree data structure. In this paper we present an algorithm for speeding up the creation of massive multi-resolution octree data structures based on a divide and conquer approach that splits the task into smaller pieces that can be combined. This enables Potree to visualize country-wise massive point cloud data sets in most modern web browsers. The algorithm is used for the AHN2 data set and, in this paper, we also present the details of a public web service[5] where it can be visualized online. Figure 1 depicts a snapshot of the the web service. Moreover, we have extended the Potree functionality, which already allowed the tuning of the visualization parameters and also contained a measurement toolkit, with a geographic name search bar, a 2D orientation map with field of view depiction, a configurable demo mode and a multi-resolution download tool which requires the usage of a back-end database with additional information. Like Potree, all the developed software is

free and open source (FOSS) and can be found in public repositories.[6–8]

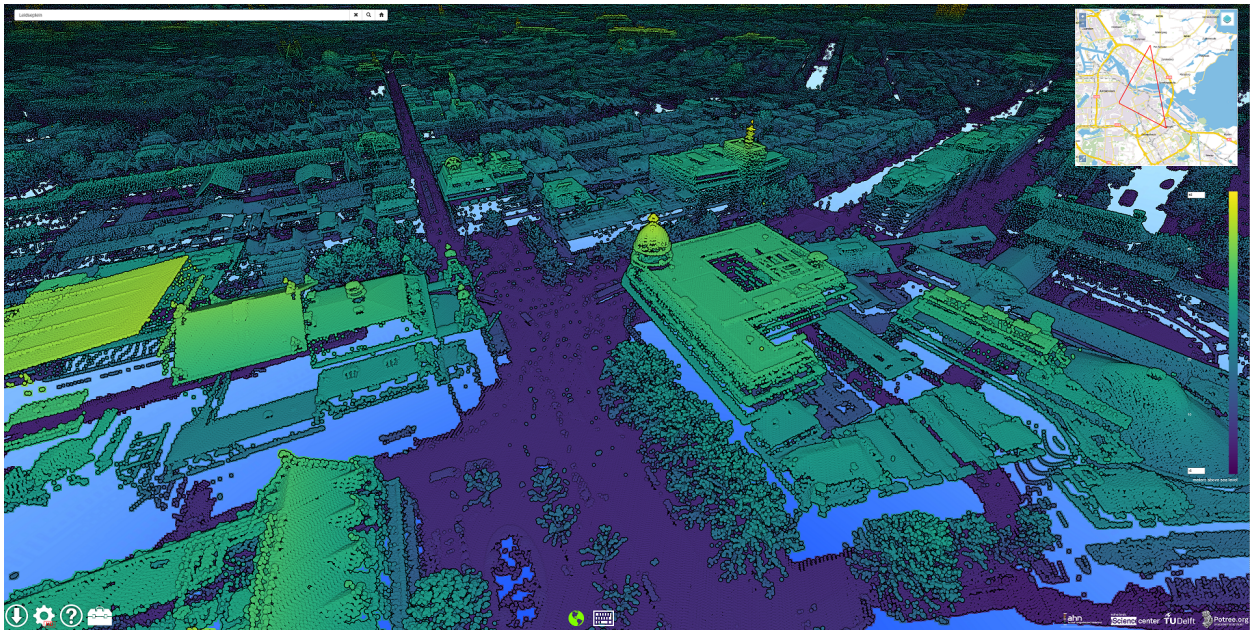

Figure 1. Snapshot of the AHN2 3D web viewer and download tool. This street-level image, which is centered in the Leidseplein in Amsterdam, allows to notice the AHN2 point density which is of 6-10 points per square meter.

The remainder of the paper has the following structure: Section 2 describes the related work. An overview on Potree and its multi-resolution octree data structure is given in Section 3, while in Section 4 we describe the algorithm used for the creation of massive multi-resolution octree data structures. This was used for the AHN2 data set as described in section 5. Section 6 contains information regarding the back-end database and the public web service that provides the 3D web visualization of the Netherlands. Finally Section 7 presents the conclusions and the future work.

2 Related work

Since the introduction of points as rendering primitives[9,10] a large amount of research have been done and many solutions have been proposed. We are going to focus on the related work that deals with large data sets. For a broader overview on the topic we refer the reader to survey literature.[11,12]

QSplat[13] was the first solution to render large point sampled models that do not fit in main memory. It builds a hierarchical point-per-node data structure with level-of-detail (LoD) selection capabilities. In addition, it requires additional data pre-processing such as normal vectors computation[14] and it is based on point-sampling meshes. In Layered Point Clouds (LPC)[15] several points are stored per node which allows to use GPU's to boost the performance. However, it assumes uniform sampling density. The first point renderer that did not assume any sampling distribution and did not require normal computations or other pre-processing steps was Instant Points[16] which was part of the SCANOPY project.[17] This approach uses an octree data structure where the hierarchical nodes also contain multiple points. A different data structure, a multi-way (balanced) kd-tree, is used in the solution implemented by Goswani et al.[18] A similar one, also based on kd-trees, is used by Richter et al.[19,20] in a system where the possibility to display classified points is implemented. Gunther et al.[21] explored techniques on how to further boost the GPU capabilities for point rendering. Lidar visualizations for immersive environments have been proposed and implemented by Kreylos et al.[22,23] De Haan[24] created a visualization of a small part of the AHN2 data set with OpenSceneGraph.[25] This system has been extended by Kehl et al. using a Rendering-on-Budget approach for larger areas,[26,27] while still falling short in visualizing the full AHN2 scale.

Commercial solutions such as products from Bentley[28] or ESRI[29] also include options for point cloud visualization. Also the FOSS Meshlab[30] can be used to visualize point cloud data. There are libraries that can be used to create visualizations of point cloud data such as PCLVisualizer.[31] However, all these solutions work for relatively small data sets. The commercial solution Euclideon[32] can deal with large data sets and offers an excellent point cloud visualization experience.

Even though some of the previous point cloud rendering solutions are very efficient, all of them are desktop-based systems. With the growth of point cloud popularity many more software solutions for desktop-based point cloud visualization have

been developed.[33] However, the rising popularity of WebGL and the availability of cloud computing/storage resources[34] are changing the way in which point cloud data are consumed.

For the FOSS web point cloud visualization Potree is not the only option, Plasio[35] is a web-based point cloud viewer which supports the de-facto standard LAS[36] and its compressed version LAZ. In addition to be used for displaying single files, its development team is currently working on a multi-resolution indexing and rendering stack for large point clouds which is based on Entwine, Greyhound and Plasio. Entwine is an indexing library that uses a multi-resolution data structure tree that responds to cubic octree queries, but packs in a similar way to a quadtree. It is powered by the Point Data Abstraction Library (PDAL),[37] a popular library for manipulating point cloud data which is used, in general, as an abstraction layer on management operations. Hence, Entwine can index any file types supported by PDAL. It is still on beta phase and it is expected to become FOSS in the near future. Greyhound[38] is a RESTful HTTP server that links with the Entwine library to stream the indexed data to a client. Plasio, the web-based client, has been adapted to be compatible with Entwine and Greyhound.

There are several commercial options for point cloud web visualization: PointCloudViz[39] has a free Desktop Lidar viewer and a commercial platform to upload data sets for their web visualization; glob3mobile[40] is a mobile mapping framework with point cloud support; ShareLiDAR[41] is the first solution that supports normal vectors (illumination is possible) but that requires pre-processing; and New Spin[42] is a tool for point cloud web visualization with an upload service, semantic functionality and other interesting features.

Other types of 3D data such as meshes are also possible to be visualized online for example with the FOSS 3DHop[43] which also uses a multi-resolution data structure for the mesh data called Nexus.[44]

3 Potree: multi-resolution octree

The Potree[4,45] renderer is based on the previous work on Instant Points,[16] which was further extended to a mature state by its research group,[46] to render point clouds in web browsers. Currently Potree is being continued under the Harvest4D project.[47]

The PotreeConverter tool[48] is used to convert an input set of point cloud files (in LAS, LAZ, PTX or PLY format) to the required multi-resolution octree data structure required by the Potree renderer, where each node of the octree is stored in a file. The format of the octree files can be a specific binary file format, LAS or LAZ. In order to reduce initial load times, the whole octree hierarchy is stored in multiple auxiliary files. The tool has several input parameters that drive the creation of the octree. The *spacing* parameter defines the resolution, it specifies the minimum distance between points at the root level of the generated octree. Each subsequent level halves the spacing, thus containing data with increased resolution (doubled compared to the previous level). A default value of the *spacing* parameter is derived from the cubic axis-aligned bounding box (CAABB) of the point cloud but the user may chose another value. The used CAABB can also be changed in one of the tool parameters (*aabb*). The *levels* parameters defines the number of levels, i.e. the depth of the octree.

The steps performed by the tool are depicted in Figure 2. First the CAABB is computed if not specified by the user. Second, from the CAABB and the user-defined depth the spatial extents of the octree nodes at the various levels are defined. Finally, the point-node computation is performed using the *spacing* parameter. This is the most expensive step, PotreeConverter computes in which node and level each point has to be stored. For each point, if the distance to any other point inside the root node is larger than the *spacing*, the point is added to the root node. If there is already another point in close proximity, it is passed to the next level and the same checking is repeated with half the spacing. This process is repeated until the point has been added to a node or the octree depth has been exceeded. In the latter case, the point is discarded and will not be added to any node. For more information see the Potree documentation.[49]

The multi-resolution octree data structure allows for efficient view frustum culling and level of detail calculations (perspective view selections) which are performed by the Potree renderer. This means that nodes outside the visible region are not rendered at all and nodes close to the viewer are favored over nodes that are far away. This is illustrated in Figure 3. The renderer implementation uses standard web technologies such as WebGL, three.js[50] and Javascript, and does not require additional plugins. This allows to combine it with other web applications, as done for example in the archaeological project Mapping the Via Appia in 3D.[51] Potree features high-quality rendering and various point attributes can be visualized (RGB, Intensity, Classification, etc.). It has distance and area measuring and height-profiling tools.

4 Distributed multi-resolution octree creation

PotreeConverter is implemented as a single-process tool and for very large data sets its computing time is too large. For example, we use a HP DL380p Gen8 server with 128 GB RAM and 2 x 8 Intel Xeon processors E5-2690 at 2.9 GHz, RHEL 6 as operating system and different disks directly attached including 400 GB SSD, 5 TB SAS 15 K rpm in RAID 5 configuration (internal), and 2 x 41 TB SATA 7200 rpm in RAID 5 configuration (in Yotta disk cabinet). In such a system the performance we can get from PotreeConverter is around 250 million converted points per hour. Thus, in order to convert a massive point cloud such as the AHN2 (640 billion points) we would need over 100 days. One alternative to improve the performance would

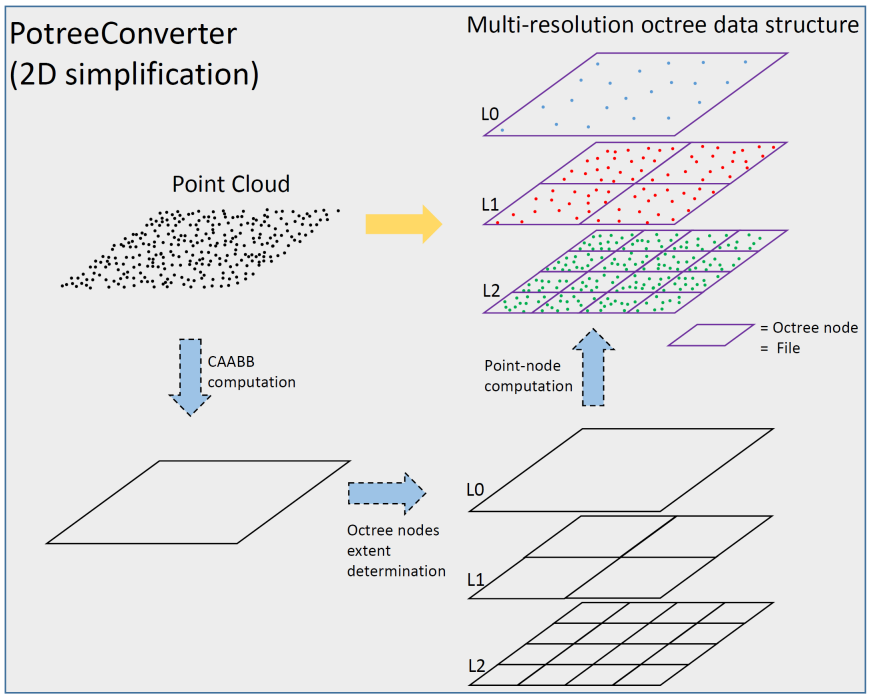

Figure 2. 2D simplification of the PotreeConverter steps to create a 3 levels multi-resolution octree. Note that the 2D simplification of the octree is, in fact, a quadtree

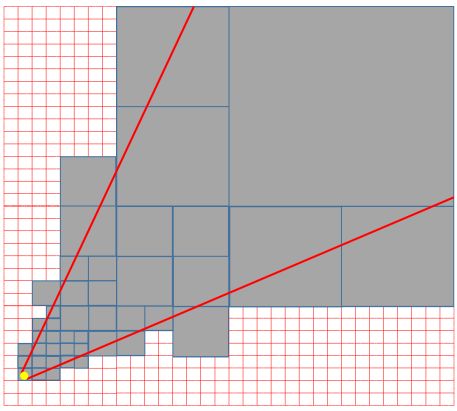

Figure 3. 2D simplification of the view frustum culling and level of detail calculations (perspective view selections) which are possible with multi-resolution data structures. Low-resolution data from far away and gradually higher resolution data when closer

be to add multi-processing in PotreeConverter. This would indeed help and it is a improvement that we also recommend for future work. However, the processing is quite IO-bonded, hence the gain in performance would not scale linearly with the amount of cores. The solution that we propose is called Massive-PotreeConverter[6] and it divides the creation of the octree into completely independent tasks which can run in parallel in the same system (thus in practice already doing a multi-processing approach) but also, and more importantly, on other systems with dedicated disks and memory. These independent tasks produce smaller octrees that can be merged into a large octree covering the whole region.

Figure 4 depicts the divide and conquer approach of the Massive-PotreeConverter. First, the CAABB is computed which determines the spatial extent of the multi-resolution octree nodes. Next we run a 2D tiling operation which is based on the CAABB and the spatial extent of the octree nodes. Note that for the 2D tiling we are essentially flattening the octree into a quadtree. This generates 2D tiles that perfectly fit the 2D spatial extent of the 3D octree nodes at certain level. Which exact level depends on the amount of independent tasks that we want to obtain (in the example in the figure we use L1 which yields four independent tasks / tiles). Next, each independent task is executed with a regular PotreeConverter but using the CAABB of

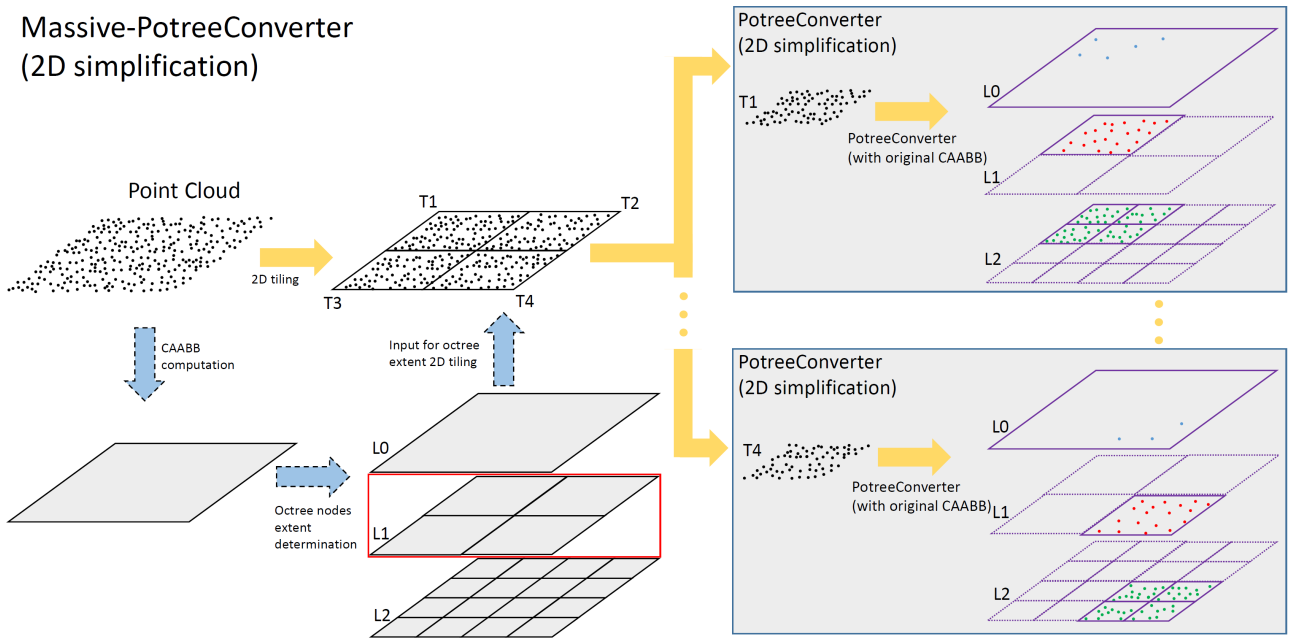

Figure 4. 2D simplification with only 3 levels of the Massive-PotreeConverter approach. The data set is split into four tiles which are converted separately.

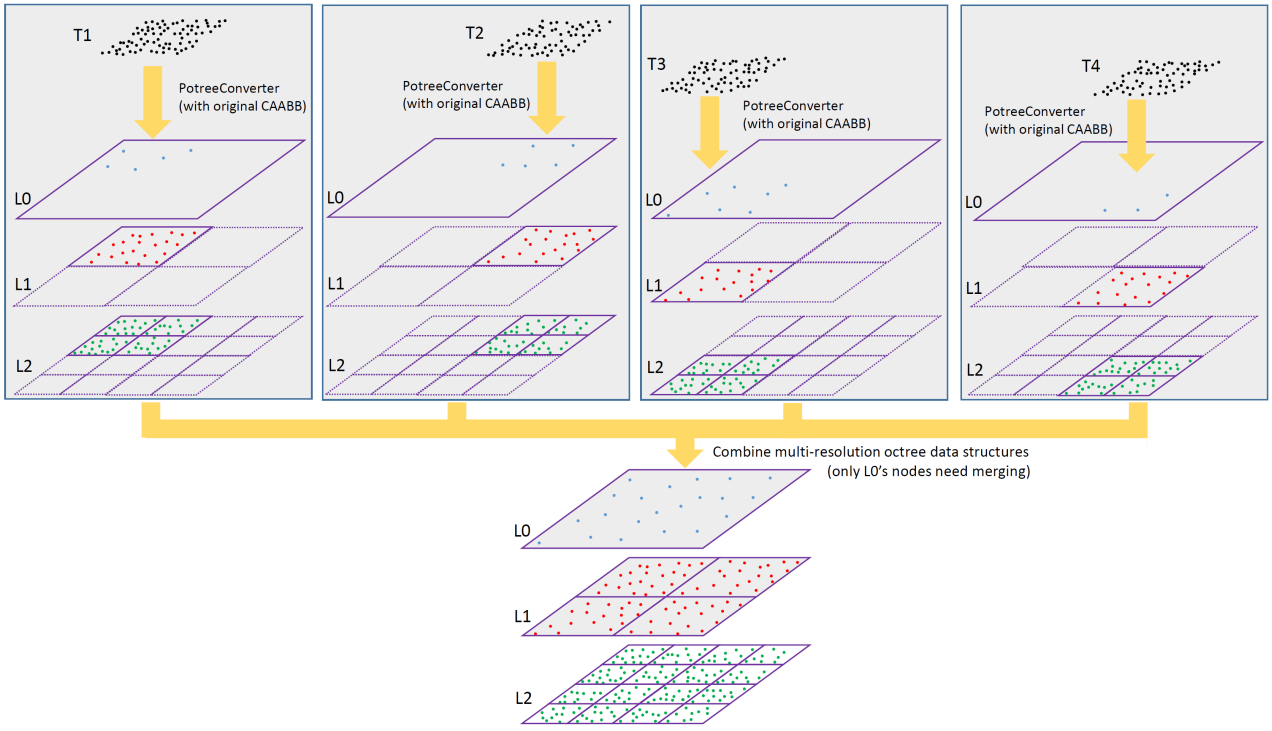

Figure 5. In the final step, the individually generated smaller octrees are merged into a single large one by the Massive-PotreeConverter.

the entire point cloud. Finally, and as depicted in Figure 5, the different octrees generated from the various independent tasks are merged. In this process only the files from the levels above the one used in the 2D tiling operation (L0 in the example) need actual combination. This merging process is fast since it consists on moving files and combining very few files.

Similar algorithms can be applied in other tree structures as long as the extent of the nodes is always pre-determined. For

instance, it can also be applied for quadtree structures. However, other popular choices such as kd-trees can not benefit from the presented algorithm because the extent of its nodes depends on the distribution of points in the space.

5 AHN2 conversion

The Massive-PotreeConverter is executed for the AHN2 data set and several versions (re-organizational stages) of the data set are handled as depicted in Figure 6.

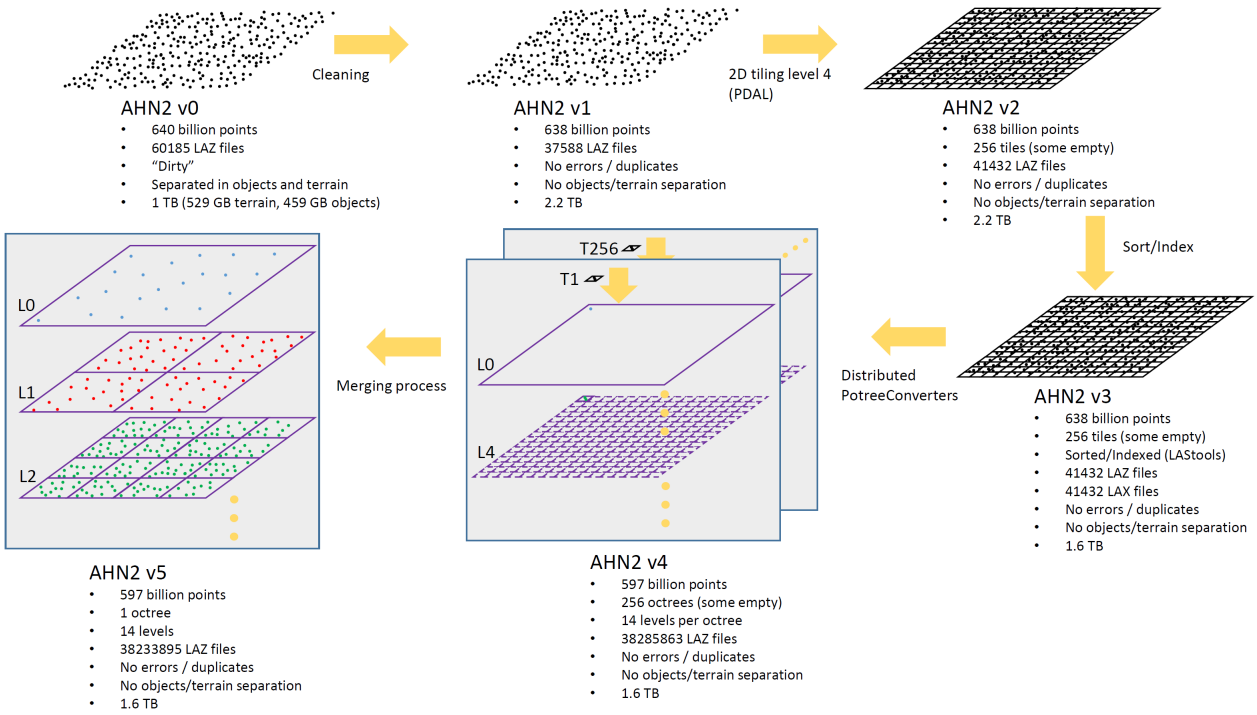

Figure 6. 2D simplification of the various steps and versions required to convert the AHN2 data set into a multi-resolution octree data structure

AHN2 v0 is the raw data set which contains 640 billion points stored in 60,185 LAZ files and has a total size of 1 TB. There are two types of files, one for terrain points (529 GB) and one for objects points (459 GB). This data set is cleaned, i.e. the duplicated and erroneous points are deleted. *AHN2 v1* is the cleaned data set and has 638 billion points stored in 37,588 LAZ files and its total size is 2.2 TB. Note the difference in the sizes. This is due to the fact that for the cleaning process the points from all the files of *AHN2 v0* need to be combined and resorted in order to detect duplicates and that affects dramatically the compression performance of LAZ. *AHN2 v1* is used as input for the Massive-PotreeConverter. First we compute the CAABB of *AHN2 v1* and this is used in a 2D tiling operation that produces the *AHN2 v2*. For the 2D-tiling operation we use the level 4 of the octree which nodes extents can be entirely pre-computed once the CAABB is obtained, i.e. 256 parts/tiles are generated. The 2D tiling operation is done by processing the files of *AHN2 v1* in the following manner: if the 2D extent of the file is completely within the extent of a tile we copy the file to the tile folder, if it is not we use PDAL to split the file in different pieces that overlap one tile each. After the tiling operation there are 41,432 LAZ files split in the 256 tiles. Using LAStools we sort and index the files in the tiles to produce *AHN2 v3*, the indexing produces additional 41,432 LAX files. The last step is not strictly required as part of the Massive-PotreeConverter. However, we do it to decrease the storage size to 1.6 TB and because this version of the data set will be used by the download tool of the web service described in the next section. Sorted and indexed data will increase the performance of clipping operations (also using LAStools) done by the download tool. The next step of the Massive-PotreeConverter is to process the tiles. For each tile, a separate PotreeConverter process is spawned, therefore creating as many octrees as tiles. The tiles can be processed independently using various systems and multiple cores, we used the previously described HP DL380p Gen8 server and the DAS-4 cluster.[52] This decreases the required time enormously, in our case from 100 to 15 days. *AHN2 v4* contains the octrees of all the tiles generated by the various PotreeConverter executions. Finally, these are merged into a single multi-resolution octree data structure, *AHN2 v5*. In Table 1 we show an overview of *AHN2 v5* with the number of LAZ files and points per level, and the ratios of these for consecutive levels. Due to the flat nature of this point cloud (the highest point in the Netherlands is at 380 meters) this octree is more similar to a quadtree structure (the

ratios are in most cases similar to 4). One could claim that for large point clouds a quadtree should always be used because it will be cheaper to generate and it deliver the same performance but in mountainous terrains it would be disadvantageous. An octree is suitable for a broader range of point cloud typology. In addition, in the used octree when a node is empty no children nodes are generated.

level	#files	files_fact	#points	points_fact
0	1		34,045	
1	4	4,00	134,786	3,96
2	14	3,50	541,973	4,02
3	41	2,93	2,205,484	4,07
4	143	3,49	8,833,283	4,01
5	499	3,49	36,081,908	4,08
6	1,804	3,62	155,411,383	4,31
7	6,767	3,75	668,597,511	4,30
8	25,939	3,83	2,834,989,373	4,24
9	101,057	3,90	11,355,433,955	4,01
10	398,423	3,94	39,911,483,676	3,51
11	1,584,598	3,98	112,993,998,398	2,83
12	6,671,815	4,21	259,014,500,658	2,29
13	29,442,790	4,41	170,207,571,211	0,66
Total	38,233,895		597,189,817,644	

Table 1. Overview of *AHN2 v5*, the merged multi-resolution octree data structure

AHN2 v5 has 38,233,895 LAZ files (octree nodes). Note that the total number of points of *AHN2 v5* (and also *AHN2 v4*) is not 638 billion points, 6.5 % of the points are dropped because of the way the tree is created. This can be decreased by tuning the tree creation parameters but the processing time would increase as well. Future work is also expected on trees that contain all the points.

6 Web service: AHN2 3D viewer and download tool

We have created a web service[5] for the 3D web visualization of the AHN2 data set, see a snapshot in Figure 7. In addition to the visualization, it has the following features: a multi-resolution download tool, a measurement toolkit, the tuning of visualization parameters which includes four visualization quality options (low, standard, high and ultra), a geographic name search bar which uses the Bing Geocoding service to convert place names into geographical coordinates, a 2D orientation mini-map which also depicts the field of view of the user, a user-configurable height range and a demo mode where the user can upload/download his own demo paths.

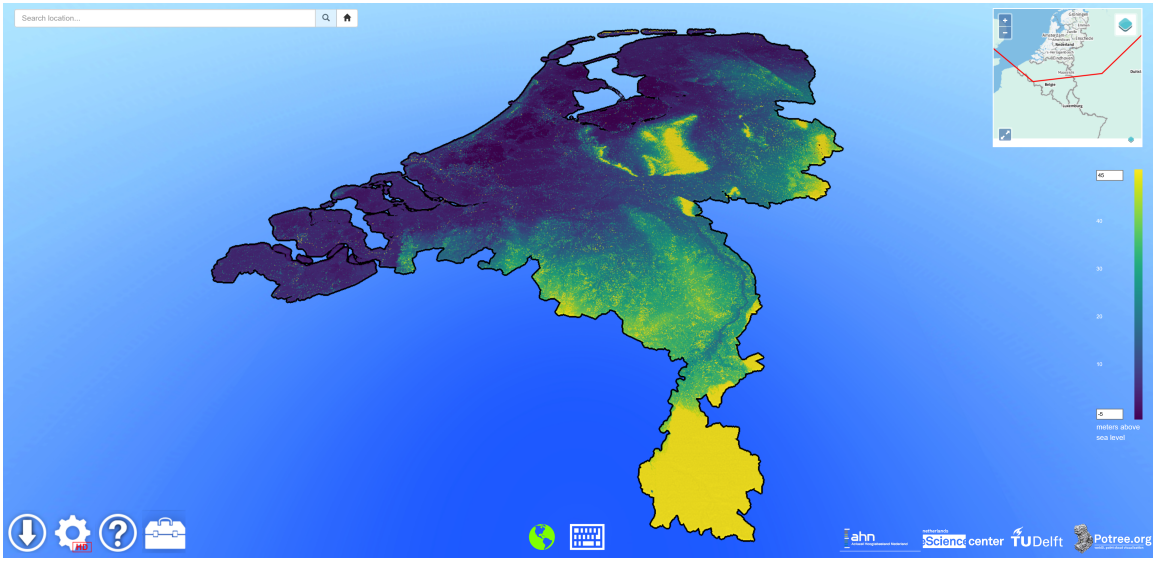

Figure 7. Snapshot of the AHN2 3D viewer and download tool with ultra quality

For the service we use *AHN2 v5* (i.e. the merged multi-resolution octree) and *AHN2 v3* (i.e. the tiled, sorted and indexed version of AHN2). *AHN2 v5* is used by the 3D visualization which is based on the Potree WebGL point cloud renderer.[53] Both,

AHN2 v3 and *AHN2 v5*, are used by the download tool. In the same server where *AHN2 v3* and *AHN2 v5* are stored there is a Postgres-PostGIS database with two tables. The first one contains for each node/file in *AHN2 v5* a row with the file path, the spatial extent (a PostGIS Geometry data type), the number of points, the level and the minimum and maximum elevation. The second table contains for each file in *AHN2 v3* the same information as in the first table except the level, which does not apply in this case.

Regarding the multi-resolution download tool, Figure 8 depicts its functioning. First, the user selects an area of interest, this can be done directly filling-in the text boxes in the download tool or through a selection in the 3D scene or the 2D mini-map. Second, the user has to press the *Count Points* button which computes an estimate of the points in the selected area. *Count points* runs a database query that sums the number of points of the files in *AHN2 v3* that overlap the selected area. Finally, if an email address is provided and the *Submit* button is pressed, the tool schedules a job in the server to generate a file with the points in the selected area. After the user file has been created, a message with a link to the file (to be downloaded within 24 hours) is sent to the user. The file in LAZ format is generated with the LAStools lasmerge tool. However, in order to improve the performance of LAStools a small list of files is given as input to the tool. The list is computed with a database query that selects the files that overlap the selected region. For small selections (lower than a certain threshold) the table with *AHN2 v3* information is used. For large selections the table with *AHN2 v5* information is used, but only the files of a certain level. Which level is used depends on how large the selected area is and it is decided using the known point density in the various *AHN2 v5* octree levels. Hence, if the user selects a small region all the points (from *AHN2 v3*) in the area are available for download. On the other hand if the user selects a large region, then the points are acquired from a certain level of the *AHN2 v5* octree, i.e. only a percentage of all the points in the area are available for download. The field *Coverage* indicates the approximated percentage of the data that the user is allowed to download taking into account the previous consideration.

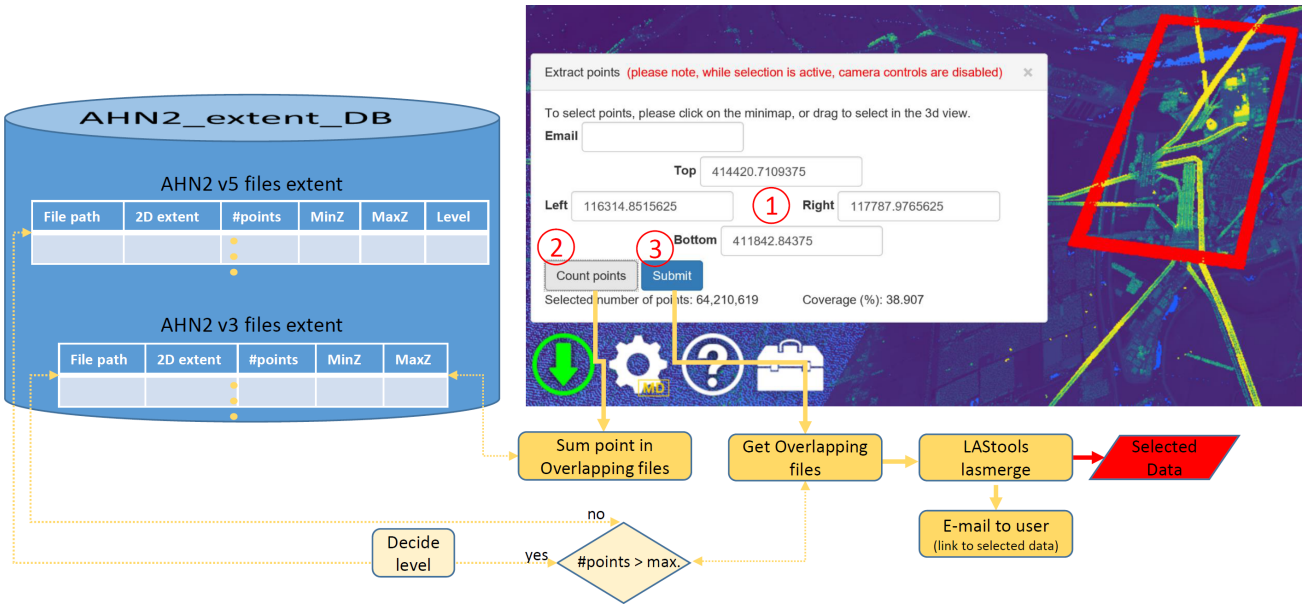

Figure 8. Steps performed by the download tool

7 Conclusions and future work

We have presented an algorithm to create massive multi-resolution data structures for point cloud data based on dividing the process into smaller independent tasks which results can be combined. This aids the reorganization process required by the Potree web viewer which now is able to visualize massive data sets such as the AHN2 with 640 billion points. We have developed a public web service for the AHN2 3D web visualization. We have extended the Potree renderer functionality with a multi-resolution download tool among other features. The work presented in this paper aims at demonstrating a Proof-of-Principle that massive point cloud data sets can be visualized over the web with FOSS software technologies. Future lines of work include:

- The raw AHN2 data set has explicit classification information as data is split in two folder, objects and terrain. This division improves the LAZ compression ratio but the explicit classification information is lost in the cleaning process which is required to delete erroneous and duplicated points. In future work this explicit classification information should

be included, for instance by using the classification attribute of LAS format. Hence, it could be visualized by the current viewer based on the Potree renderer which already supports coloring by attributes others than RGB or height.

- As mentioned earlier, the AHN2 data set is part of a national scanning campaign that collects point clouds of the country surface periodically. The service should be adapted to deal with different time-snapshots of the country and to allow visualizing differences between time-scans.
- We will explore the possibility to merge the point cloud visualization with imagery tiles in order to add RGB colors to the visualized points.
- Currently some points are dropped in the reorganization process required by Potree. This should be improved in order to be able to visualize all the data. In addition we will explore different alternatives for computing the LoD of the points. The continuous levels of detail as suggested in van Oosterom et al.[3] will also be explored. This is based on the vario-scale LoD research.[54]
- In practice our algorithm can be used for both vertical and horizontal scalability. However, due to its relatively complicated functioning, its usage is only recommended for very large data sets. For other data sets we recommend using the normal PotreeConverter which is more easy to use. However, we should assess the benefits of vertical scalability (multi-core processing) on the PotreeConverter itself and, if promissing, implement it in a way that the tool can exploit parallel processing in a transparent way for the user.
- PotreeConverter generates a file for each octree node. In very large data set this means that millions of files are generated which may create problems depending on the used file system. This could be solved by storing the octree nodes in database systems or key-value stores such as MemcacheDB. Another alternative is to join different octree nodes in single files and use HTTP Range Retrieval Requests when accessing the data.
- Currently the multi-resolution download tool only works with 2D selections. The used database already contains 3D information for each octree node / tile file. Future work includes updating the download tool in order to handle 3D selections.
- Even though effort is being undertaken to improve the point cloud data management support in database systems,[55] nowadays still specific solutions are required for the visualization of large point clouds. It is expected than in the future the generic solutions will offer efficient support for the multi-resolution data structures that are required to deal with large point clouds.
- A global service is envisioned. It should be generic enough to add new point clouds, for example more countries in addition to the Netherlands, delivering a service similar to OpenStreetMap[56] but with point cloud data. A suggested name is *OpenPointCloudMap*. It should be able to scale to handle point clouds of continents or even the entire planet. Such service should be flexible to integrate new/updated versions of LiDAR data.
- Several activities for point cloud standardization have been recently initiated. The OGC (Open Geospatial Consortium) has created a Point Cloud Domain Working Group (DWG) to address issues on the interoperability when sharing and processing point cloud data. There have been discussions on a possible cooperation with ISO TC211, the ISO technical committee for Geographic information/Geomatics. In parallel the OSGEO PointDown initiative was started with the aim of creating an overview on the usage of point cloud data over the web in order to provide a generic service definition. It is recommended that the active players in point cloud data handling synchronize their activities with such of the ongoing standardization efforts.

References

1. Actueel Hoogtebestand Nederland (AHN). http://www.ahn.nl/ (Accessed 2015-10-23).
2. rapidlasso GmbH LASzip - free and lossless LiDAR compression. http://www.laszip.org/ (Accessed 2015-10-23).
3. van Oosterom, P. *et al.* Massive point cloud data management: Design, implementation and execution of a point cloud benchmark. *Computers and Graphics* **49**, 92 – 125 (2015). URL http://www.sciencedirect.com/science/article/pii/S0097849315000084.
4. Schütz, M. potree. http://potree.org/ (Accessed 2015-10-23).

5. AHN2 3D viewer and download tool. http://ahn2.pointclouds.nl/ (Accessed 2015-10-23).
6. Massive PotreeConverter. https://github.com/NLeSC/Massive-PotreeConverter (Accessed 2015-10-23).
7. AHN pointcloud viewer. https://github.com/NLeSC/ahn-pointcloud-viewer (Accessed 2015-10-23).
8. AHN pointcloud viewer web service. https://github.com/NLeSC/ahn-pointcloud-viewer-ws (Accessed 2015-10-23).
9. Levoy, M. & Whitted, T. The use of points as display primitives. Tech. Rep. Technical Report 85-022, University of North Carolina at Chapel Hill (1985).
10. Grossman, J. & Dally, W. Point sample rendering. In Drettakis, G. & Max, N. (eds.) *Rendering Techniques '98*, Eurographics, 181–192 (Springer Vienna, 1998). URL http://dx.doi.org/10.1007/978-3-7091-6453-2_17.
11. Kobbelt, L. & Botsch, M. A survey of point-based techniques in computer graphics. *Computers and Graphics* **28**, 801–814 (2004).
12. Gross, M. & Pfister, H. *Point-Based Graphics* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007).
13. Rusinkiewicz, S. & Levoy, M. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, 343–352 (ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000). URL http://dx.doi.org/10.1145/344779.344940.
14. Dey, T. K., Li, G. & Sun, J. Normal estimation for point clouds: A comparison study for a voronoi based method. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'05, 39–46 (Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2005). URL http://dx.doi.org/10.2312/SPBG/SPBG05/039-046.
15. Gobbetti, E. & Marton, F. Layered point clouds: A simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Comput. Graph.* **28**, 815–826 (2004). URL http://dx.doi.org/10.1016/j.cag.2004.08.010.
16. Wimmer, M. & Scheiblauer, C. Instant points: Fast rendering of unprocessed point clouds. In *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'06, 129–137 (Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006). URL http://dx.doi.org/10.2312/SPBG/SPBG06/129-136.
17. SCANOPY project. https://www.cg.tuwien.ac.at/research/projects/Scanopy/ (Accessed 2015-10-23).
18. Goswami, P., Zhang, Y., Pajarola, R. & Gobbetti, E. High quality interactive rendering of massive point models using multi-way kd-trees. In *Proceedings Pacific Graphics Poster Papers* (2010).
19. Richter, R. & Döllner, J. Out-of-core real-time visualization of massive 3d point clouds. In *Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '10, 121–128 (ACM, New York, NY, USA, 2010). URL http://doi.acm.org/10.1145/1811158.1811178.
20. Richter, R., Discher, S. & Döllner, J. Out-of-core visualization of classified 3d point clouds. In Breunig, M. *et al.* (eds.) *3D Geoinformation Science*, Lecture Notes in Geoinformation and Cartography, 227–242 (Springer International Publishing, 2015). URL http://dx.doi.org/10.1007/978-3-319-12181-9_14.
21. Günther, C., Kanzok, T., Linsen, L. & Rosenthal, P. A gpgpu-based pipeline for accelerated rendering of point clouds. *Journal of WSCG* **21**, 153–161 (2013). URL http://www.paul-rosenthal.de/wp-content/uploads/2013/06/guenther-wscg-2013.pdfhttp://www.paul-rosenthal.de/wp-content/uploads/2013/06/guenther-wscg-2013.png.
22. Kreylos, O., Bawden, G. & Kellogg, L. Immersive visualization and analysis of lidar data. In Bebis, G. *et al.* (eds.) *Advances in Visual Computing*, vol. 5358 of *Lecture Notes in Computer Science*, 846–855 (Springer Berlin Heidelberg, 2008). URL http://dx.doi.org/10.1007/978-3-540-89639-5_81.
23. Lidar Viewer. http://idav.ucdavis.edu/~okreylos/ResDev/LiDAR/ (Accessed 2015-10-23).
24. de Haan, G. Scalable visualization of massive point clouds. In van Oosterom, P., Vosselman, G., van Dijk, T. & Uitentuis, M. (eds.) *Management of massive point cloud data: wet and dry*, Green Series, 59–68 (Nederlandse Commissie voor Geodesie, 2010). URL http://www.ncgeo.nl/phocadownload/49NCGGroenPointClouds.pdf.
25. OpenSceneGraph. http://www.openscenegraph.org/ (Accessed 2015-10-23).

26. Kehl, C. & Haan, G. *Intelligent Systems for Crisis Management: Geo-information for Disaster Management (Gi4DM) 2012*, chap. Interactive Simulation and Visualisation of Realistic Flooding Scenarios, 79–93 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2013). URL http://dx.doi.org/10.1007/978-3-642-33218-0_7.

27. Kehl, C., Tutenel, T. & Eisemann, E. Smooth, interactive rendering techniques on large-scale, geospatial data in flood visualisations. In *Proceedings of ICT Open* (2013). URL http://graphics.tudelft.nl/Publications-new/2013/KTE13.

28. Bentley Pointools. http://www.bentley.com/en-US/Promo/Pointools/pointools.htm (Accessed 2015-10-23).

29. ESRI ArcGIS. http://www.esri.com/software/arcgis (Accessed 2015-10-23).

30. Meshlab. http://www.meshlab.org/ (Accessed 2015-10-23).

31. PCL CloudViewer. http://pointclouds.org/documentation/tutorials/cloud_viewer.php (Accessed 2015-10-23).

32. Euclideon. http://euclideon.com/ (Accessed 2015-10-23).

33. Tobias, W. A short comparison of freely available point cloud viewers for windows. Tech. Rep., GeoNext BV (2015). URL http://www.geonext.nl/wp-content/uploads/2014/05/Point-Cloud-Viewers.pdf. Accessed 2015-10-23.

34. Kodde, M. The art of collecting and disseminating point clouds. In van Oosterom, P., Vosselman, G., van Dijk, T. & Uitentuis, M. (eds.) *Management of massive point cloud data: wet and dry*, Green Series, 9–15 (Nederlandse Commissie voor Geodesie, 2010). URL http://www.ncgeo.nl/phocadownload/49NCGGroenPointClouds.pdf.

35. plasio. http://plas.io/ (Accessed 2015-10-23).

36. ISPRS. LAS 1.4 Format Specification. Tech. Rep., The American Society for Photogrammetry & Remote Sensing (2011). http://www.asprs.org/a/society/committees/standards/LAS_1_4_r13.pdf.

37. PDAL. http://www.pdal.io/ (Accessed 2015-10-23).

38. Butler, H. Greyhound. https://github.com/hobu/greyhound (Accessed 2015-10-23).

39. PointCloudViz. www.pointcloudviz.com/ (Accessed 2015-10-23).

40. glob3mobile inc. glob3mobile framework. http://www.glob3mobile.com/ (Accessed 2015-10-23).

41. Sharelidar laser scanning sharing platform. http://www.sharelidar.com/ (Accessed 2015-10-23).

42. New Spin. https://newspin.squarespace.com/ (Accessed 2015-10-23).

43. 3DHop. http://www.3dhop.net/ (Accessed 2015-10-23).

44. Nexus format. http://vcg.isti.cnr.it/nexus/ (Accessed 2015-10-23).

45. Schütz, M. & Wimmer, M. Rendering large point clouds in web browsers. In Wimmer, M., Hladuvka, J. & Ilcik, M. (eds.) *Proceedings of CESCG 2015: The 19th Central European Seminar on Computer Graphics*, CESCG 2015, 83–90 (Vienna University of Technology. Institute of Computer Graphics and Algorithms, Favoritenstrasse 9-11/186, 1040 Vienna, Austria, 2015). URL http://www.cescg.org/CESCG-2015/proceedings/CESCG-2015.pdf.

46. Scheiblauer, C. *Interactions with Gigantic Point Clouds*. Ph.D. thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria (2014). URL http://www.cg.tuwien.ac.at/research/publications/2014/scheiblauer-thesis/.

47. Harvest 4D. https://harvest4d.org/ (Accessed 2015-10-23).

48. PotreeConverter. https://github.com/potree/PotreeConverter (Accessed 2015-10-23).

49. Potree file format. https://github.com/potree/potree/blob/master/docs/file_format.md (Accessed 2015-10-23).

50. three.js - Javascript 3D library. http://threejs.org/ (Accessed 2015-10-23).

51. Mapping the Via Appia in 3D. http://mappingtheviaappia.nl/4dgis/ (Accessed 2015-10-23).

52. Distributed ASCI Supercomputer 4 (DAS-4). http://www.cs.vu.nl/das4/ (Accessed 2015-10-23).

53. Potree WebGL point cloud viewer. https://github.com/potree/potree (Accessed 2015-10-23).

54. van Oosterom, P. & Meijers, M. Vario-scale data structures supporting smooth zoom and progressive transfer of 2d and 3d data. *International Journal of Geographical Information Systems* **28**, 455–478 (2014).

55. Martinez-Rubi, O. *et al.* Benchmarking and improving point cloud data management in monetdb. *SIGSPATIAL Special* **6**, 11–18 (2015). URL http://doi.acm.org/10.1145/2744700.2744702.

56. OpenStreetMap. https://www.openstreetmap.org/ (Accessed 2015-10-23).

Acknowledgements

We thank all the members of the project Massive Point Clouds for eSciences, which is supported in part by the Netherlands eScience Center under project code 027.012.101. In addition we thank the Mapping the Via Appia project members (project code 027.013.901) which triggered the NLeSC involvement in point cloud visualization that has made this work possible. We also want to take the chance to thank Harvest4D project (EU FP7 no. 323567) members and other sponsors which made the fantastic Potree possible.