TECHNICAL REPORT: REDUCTION AND COMPRESSION USING OCTREES — 3DTK'S ENTRY TO THE ICIP 2019 CHALLENGE ON POINT CLOUD CODING

Andreas Nüchter, Johannes Schauer, Dorit Borrmann

Informatics VII – Robotics and Telematics
Julius Maximilian University of Würzburg, Am Hubland, 97074 Würzburg, Germany

ABSTRACT

[1]Modern photogrammetric methods as well as laser measurement systems make it easy to collect large 3D point clouds that sample objects or environments. Several mechanisms are available in literature for storing and compressing point clouds, e.g., applying conventional image based compression methods to 3D point clouds. In this challenge entry however, we make use of octree subsampling and compression, which is nicely suited for unstructured, registered 3D point clouds of arbitrarily shaped objects.

***Index Terms*—** 3D Point Cloud Reduction, 3D Point Cloud Compression, Octrees.

1. INTRODUCTION AND STATE OF THE ART

There are two basic principles for obtaining range measurements: Triangulation and Time-of-Flight (ToF). Triangulation is the underlying principle in stereo vision and structure from motion, where 3D point clouds are generated by finding or matching corresponding image points. But also sensors like the Microsoft Kinect (version 1) and other structured light scanners use triangulation as basic measurement principle. In contrast, ToF is often used by laser scanners, which emit a laser beam and measure the reflected light. While sensors measuring the phase shift of a modulated light source calculate the ToF from the measured shift, pulsed systems determine the ToF directly, sometimes including a full-wave-analysis [1]. LiDAR (light detection and ranging) systems, sometimes also called Ladar (laser detection and ranging) include a mechanism for steering the laser over the object of interest. Sometimes, the mechanism includes mobile vehicles or even drones and aircrafts. In this case, one talks about mobile or airborne mapping. The amount of data such LiDAR systems acquire is huge. Typical measurement rates in mobile or airborne measurement campains are in the order of 100k to 1M points per second which need to stored and processed.

The majority of the sensors work in a spherical way, i.e., the measurement is done from a central source. It is important to note, that this applies to triangulation based systems as well as to ToF systems. The result is, that objects closer to the sensor are gauged in a higher resolution than objects further away, where resolution is defined as measurement points per volume. This fact implies a lot of potential for point cloud reduction and compression.

Definition: 3D Point Cloud Reduction means reducing the amount of 3D points in a point cloud, i.e., the resulting point cloud contains less points. Thus, reduction is a form of subsampling.

Definition: 3D Point Cloud Compression consists of encoding and decoding a 3D point cloud in terms of bit-rate reduction which means the point cloud is represented with fewer bits than the original representation. Compression is either lossy or lossless and therefore, in the lossy case, 3D point cloud reduction is a form of point cloud compression.

Driven by the huge amount of airborne laser scan data, the American Society of Photogrammetry and Remote Sensing (ASPRS) created a simple binary exchange format, namely the LAS format [2]. The The LAZ format, a compressor for LAS, is a widely-used lossless, non-progressive, order-preserving compressor for LiDAR measurements. Data stored in LAS format as presented in [3] and Mongus et al. [4] shows predictive coding, a variable-length coding and an arithmetic coding for compression of LiDAR LAS files.

Other approaches employ special data structures such as k-d trees [5, 6] and octrees [7, 8, 9, 10]. Both data structures are well-suited for 3D point cloud processing, as they also support other tasks like nearest-neighbor search, which is often needed for registration.

Point clouds can be encoded with images. For example Neci et al. present a method that exploits H.264 compression to reduce the size of the data stream from sensors such as the Kinect [11]. Depending on the sensors, the images might be panorama images [12] which store range information as pixel values, forming so-called range images. There is a one-to-one correspondence between range images and the resulting 3D point clouds. The images can be down-scaled and thus the point clouds are reduced. Additionally, conventional image compression methods are applicable and in case of lossy compression combined with filtering. There is a reduction as well [12], especially in combination with resizing the range image. The different compression schemes have an impact on the 3D points [13], e.g., using the 265/HEVC video compression results showed good visual quality with lossless video compression, while lossy compression introduced additional noise in projection images, resulting a in lower visual quality of the 3D point cloud.

In this challenge, we have applied our octree based compression [14] to the given data sets and this paper reports our results. The input data are –compared to what current scanners yield– small point clouds, which probably have been sampled from 3D meshes. We want to emphasize that we do not see point cloud processing as a means for working with meshes, since the corresponding community has its own algorithms for mesh simplification and compression [15, 16].

2. OCTREE FOR REPRESENTING 3D POINT CLOUDS

In general, an octree is a tree data structure that is used for indexing 3D data. An octree node has up to eight children, each corre-

Thanks to our former colleague Jan Elseberg for implementing the octree in *3DTK – The 3D Toolkit*, cf. `http://www.threedtk.de`

[1]This text was the authors entry to the ICIP 2019 challenge, which was canceled to a lack of participants. Thus, it is published as technical report.

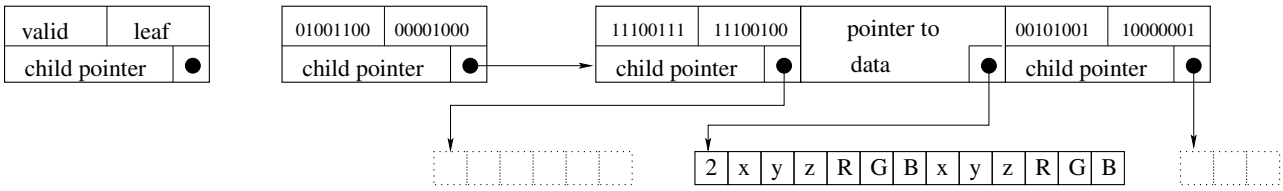

Fig. 1. The encoding of an octree node optimized for memory efficiency. Left: The child pointer as the relative pointer is the largest part of an octree node, in our implementation for 64 bit systems, it is 48 bit. `valid` and `leaf` are 8 bit large and denote whether the subtree is pruned or a leaf node containing a pointer to the points. Right: An example of a simple octree as it is stored using the proposed encoding. The node in the upper left has three valid children, one of which is a leaf. Therefore, the child pointer only points to 3 nodes stored consecutively in memory. The leaf node in this example is a simple pointer to an array which stores both the number of points and the points with all their attributes.

sponding to one octant of the associated parent volume/node. A node having no children usually implies that the corresponding volume is empty and can be pruned. For 3D point clouds this is often the case, as large volumes are simply empty space and typically most octree nodes will only have few children. We define the smallest possible leaf size, also referred to as the voxel size, as a stopping criteria. If the voxel size falls below a a given limit, a leaf node with the list of points in the corresponding enclosing volume is created. Volumes containing only a single point are not further subdivided. Alternatively, the maximal depth serves as stopping criteria. This leads to strongly varying leaf volumes, depending on the volume covered by the point cloud.

2.1. Memory efficient encoding of an octree

Octrees with pointers are often sparse voxel octrees in the computer graphics community, where an octree structure is used to efficiently access otherwise large amounts of data. In contrast, there also exist serialized pointer-free encodings. These have the highest potential for memory efficiency, since they do not need to store the actual octree structure. One such encoding is given by [17] and is implemented in the point cloud comparison software CloudCompare [18], which is interestingly used as a viewer for this ICIP challenge. Cloud Compare employs the Morton order to store only the leaf level of an octree at 16 bytes per leaf. The Morton order or Z-order is an ordering of, in this case, 3-dimensional data. A drawback of this approach is that traversing the tree is not possible. Due to its encoding of the index in a 64-bit integer, CloudCompare's octree has a maximum depth of 20.

Serialization is a very useful tool when storing the data for later use or when communicating over channels with limited bandwidth. [7] combine a serialized octree with arithmetic coding to provide superior point cloud compression. Modification, i.e. adding or deleting points from a serialized octree involves shifting the entire region before or after the position where the modification takes place.

We opt for a pointer based octree since it allows for several operations and applications which are not feasible with the above designs. We create an efficient octree implementation that is free of redundancies and is nevertheless capable of fast access operations. Our implementation allows for access operations with a time complexity of $O(\log n)$, where n being the number of points stored in the octree.

For the sake of memory efficiency, we omit any information that is computable by traversing the tree. Our implementation uses a single byte, where each of its eight bits corresponds to one octant of the node and denotes, whether the node is present or pruned. We use another byte, where each bit signals whether the corresponding octant is a leaf node. This allows the removal of the point information that is unnecessary in inner nodes. The encoding that results from these considerations is presented in Fig. 1, left.

Our encoding consists of three parts. The child pointer is the largest part of each node and is implemented as a relative pointer to the first child. All other valid children are arranged linearly in memory as shown in Figure 1. For 64 bit architectures we have chosen 6 bytes. The AMD64 architecture defines a 64-bit virtual address format, of which only the low-order 48 bits are used in current implementations. This allows up to 256 TB (2^{48} bytes) of address space. The octree is stored in a linear array in breadth first order, with each child pointer simply indexing the array. `valid` and `leaf` are each a single byte large, one bit for each subvolume. `valid` bits signal whether the corresponding octant is present, while `leaf` bits signal whether the corresponding child is a leaf node.

Our implementation stores points in the leaf nodes, thus they need to be represented differently from inner nodes. In Figure 1, right, leaf nodes are pointers to arrays of points. The first entry is always the total number of points, then sequentially the information for each point, i.e., the coordinates and additional attributes such as reflectance values or, for this challenge, RGB values.

2.2. Octree based reduction of 3D point clouds

Given a large number of points from a laser scan we propose to uniformly subsample the entire point cloud to reduce the number of points. This is achieved by first binning the point cloud in a regular 3D grid and then randomly selecting a fixed number of points in each voxel. Both, the number of points and the side length of a voxel, may be adjusted to allow for many different point densities. An additional advantage of the uniformity of the subsampling is that differences in density caused by the data acquisition process are reduced. Surfaces closer to the scanner are more densely sampled than surfaces further away. Selecting a fixed number of points from each voxel will remove more points in voxels close to the scanner than in the voxels further from the scanner. After reduction the points will be uniformly distributed across the scanned object or environment.

2.3. Octree based compression of 3D point clouds

The octree encoding drastically decreases the overhead for obtaining the data structure itself [14]. Next, we can compress the point list as well. For a simple technical reason we like to store each point coordinate using only two bytes. Two bytes are exactly the resolution at which most laser scanners measure additional point attributes, such as reflectance and deviation. To store floating point coordinates in only two bytes without significant loss of precision, we use each bit of the two byte coordinate as $s/2^{16}$ increments to the lower left front corner of the rectangular cuboid of the leaf node, where s is the side

Table 1. Reduction results for the data sets of the challenge. Achieved vs. requested compression rates are shown.

dataset	R1 = Rmax		R2		R3		R4	
amphoriskos	0.292 94	0.30	0.749 50	0.75	1.248 06	1.25	1.991 68	2.00
head	0.307 15	0.30	0.994 75	1.00	2.495 41	2.50	3.991 22	4.00
plane	0.298 20	0.30	1.000 26	1.00	2.498 90	2.50	3.999 96	4.00
romanoillamp	0.300 14	0.30	0.996 73	1.00	2.502 85	2.50	3.999 36	4.00
longdress	0.349 25	0.35	0.752 06	0.75	1.501 29	1.50	2.999 01	3.00
loot	0.349 65	0.35	0.750 48	0.75	1.502 63	1.50	2.998 67	3.00
the20smaria	0.350 34	0.35	0.750 40	0.75	1.504 01	1.50	2.995 67	3.00
ulliwegner	0.347 75	0.35	0.751 08	0.75	1.496 64	1.50	2.999 97	3.00

length of the cuboid. This is similar to color quantization as used for example by [19] and a type of lossless compression.

Terrestrial laser scanners have a high dynamic range. Typical measurement ranges to 500 m with precision and accuracies in the millimeter range. A four byte floating point value has a precision of approx. 100 μm (100 micrometer) at the maximal distance of 500 m. At a smaller distance, e.g., at 1.5 m, the precision increases to 1 μm. To achieve the same 1 μm precision the smallest volume in the octree must have a side length of 6.5 cm. Assuming a desired precision of 10 μm, which is still orders of magnitude smaller than typical specified measurement precisions, the largest node is allowed to have a side length of 65 cm. At this voxel size the octree overhead is minimal even for large scans.

3. RESULTS AND DISCUSSION

As already stated, the data sets provided in the ICIP 2019 Challenge on Point Cloud Coding are small point clouds, from objects and probably meshes of the provided point clouds exists. In our understanding 3D point clouds are more basic and close to sensor data than data from meshes.

Table 1 presents the requested results for compressing each of the reference contents using the five target bitrates, namely, "R1", "R2", "R3", "R4", and "Rmax" that are expressed in target bits-per-point (bpp). Please note that we assign R1 to Rmax, but higher reductions are easily possible. The target bpp is computed as the total number of bits at the output of the encoder divided by the number of points of its corresponding original version. Figure 3 showcases compressed version for each pointcloud as submitted to the challenge. For rendering, we use the program `show` that is included in *3DTK – The 3D Toolkit*. The submitted screenshots have been rendered as colored 3D point clouds using OpenGL `GL_POINT` with a fixed point size and antialiasing, however also other renderings are possible. The octree allows for efficient frustum culling. Frustum Culling is a highly important feature for navigation within 3D point clouds [20], but is negligible for object centered point cloud views. Level of detail rendering in `show` renders only single vertices in any octree volume that falls below a level-of-detail threshold (number of pixels on screen). The size of the rendered vertex is then adjusted accordingly, cf. Fig 2. This trades resolution for speed and appealing views and meshing is often not needed [21]. However, to the given data sets these issues do not matter.

Further compression experiments have been carried out using data acquired with a Riegl VZ400 laser scanner. Fig. 4 shows a 3D scan of the 2019 class on "3D Point Cloud Processing" in front of the Maria-Schmerz-Chappel in Randersacker in Spring 2019. Comparing the compression results using panorama images [12, 13], where we project the points into a panorama image and apply conventional image resizing techniques, one notices that image-based techniques produce nice results as long as the viewing pose of the point cloud is close to the position, where the scan was acquired. This is clear, as the OpenGL-based 3D viewer does also some projection to an image plane. However, inspecting other views at the point cloud reveals that the point densities are not uniform and surfaces far away from the scanner contain only a few 3D points.

4. SUMMARY AND CONCLUSIONS

The paper has presented the results of *3DTK – The 3D Toolkit* to the ICIP 2019 Challenge on Point Cloud Coding from University of Wrzburg. While 3DTK offers two main lines for compression, namely image-based and octree-based, we have decided to apply the octree-based method here. The compression was mainly performed by reducing the number of data points. The 3DTK viewer provides a performant 3D point cloud inspection, where the octree is used to support frustum culling and rendering.

Point Cloud Coding is an extremely interesting topic. However, 3D point clouds are a data type which is closely related to the hardware acquiring the data. There is a large need for compression of 3D point clouds, e.g., in robotics and telematics, where remote machinery must be operated using point cloud sensor data. Therefore, point cloud coding should not focus on mesh-like data sets.

5. FUTURE CHALLENGES

Needless to say, a lot of work remains to be done. In future work, we will focus of enhancing our octree-based compression methods. Instead of just using 16 bits in each voxel to describe the offset of the point coordinate, one could use more sophisticated lossless compression to the points stored in an actree. One could also fit local primitives to the points in each voxel, similarly to the 3D normal distribution transform [22] to enhance subsampling.

Further future work will also address compression using image based techniques, where the 3D acquisition methods are emphasized. A combination of the octree with projections would yield a

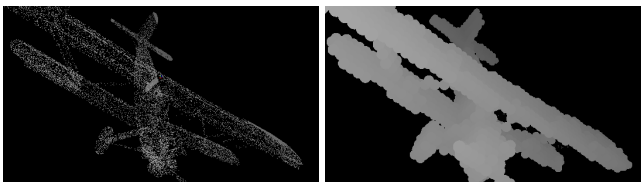

Fig. 2. Level of Detail rendering in `show`. Left: 3D point cloud with all points using pointsize 1.0. Right: LoD with larger point size.

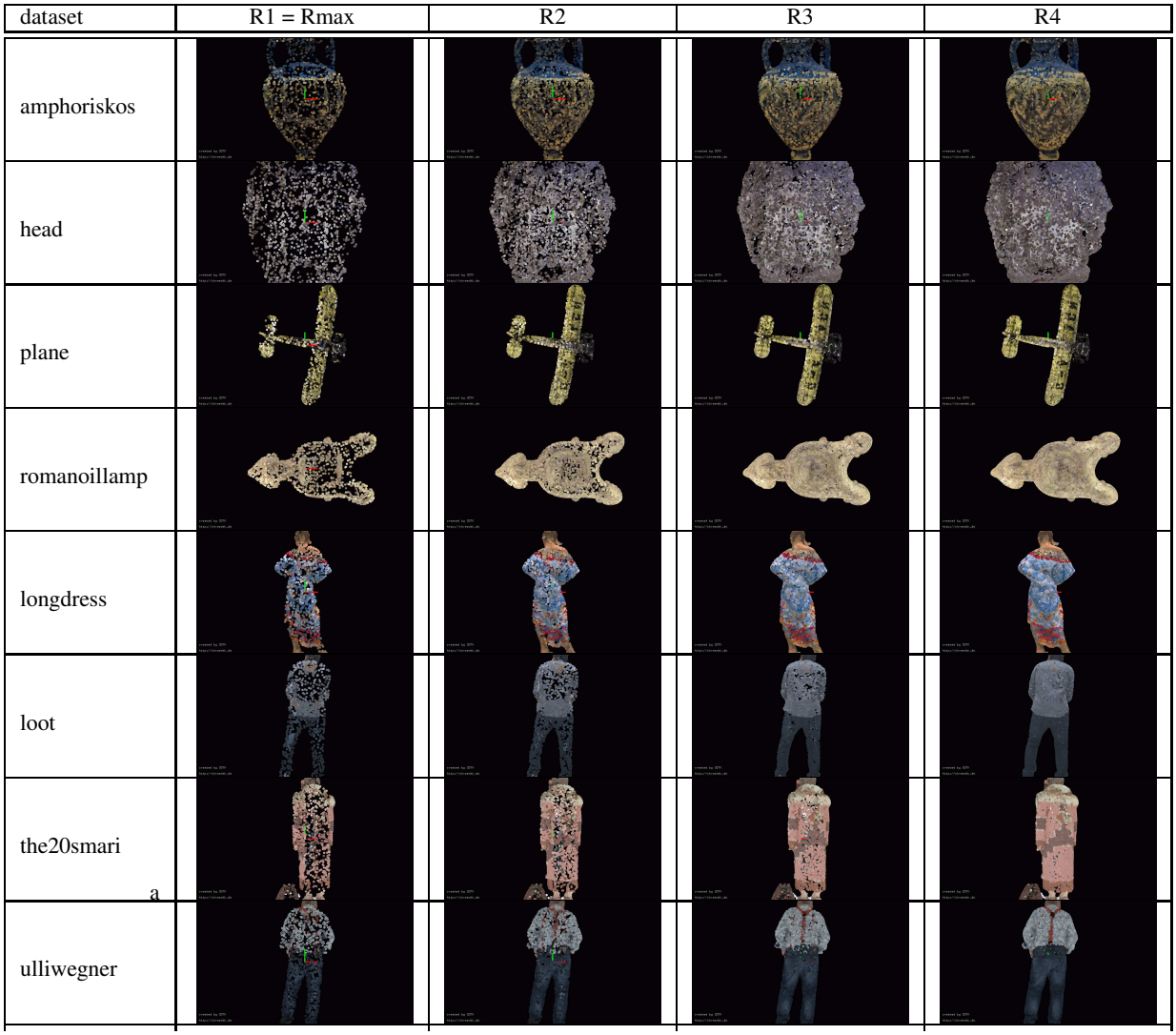

Fig. 3. Compressed versions of each pointcloud.

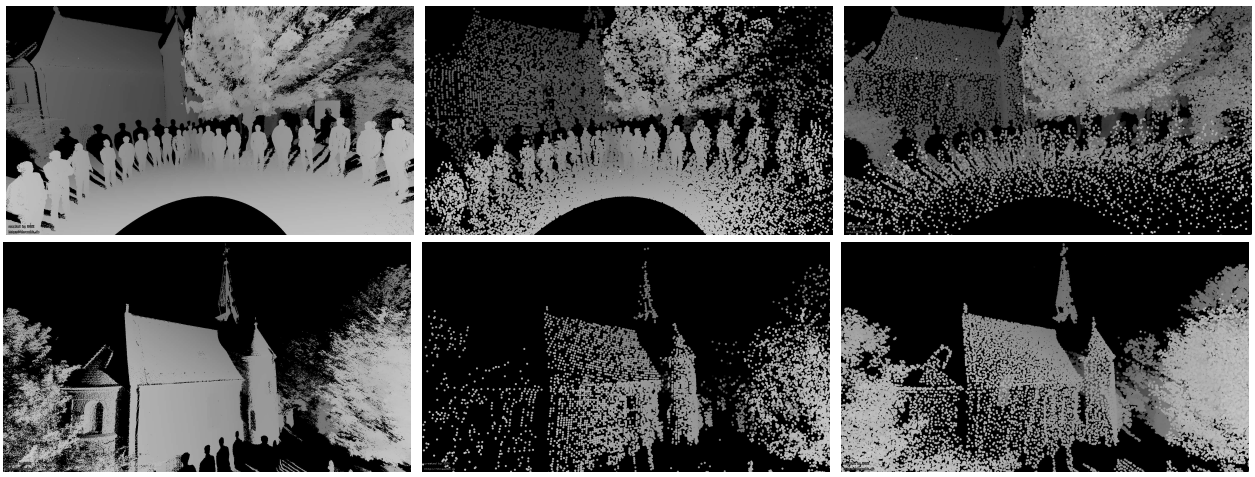

Fig. 4. Visual inspection of a compressed 3D scan. Left: Original data. Middle: Image-based compression. Right: Octree-based. Above: View from the scanner position. Below: As seen from a distance away from the scanning position. For reduction we used an octree voxel size of 10 cm, yielding a reduction from 18543615 points to 300645. Parameters for the projection-based reduction, were chosen to yield a similar compression.

spherical quadtree, similarly to the one presented in [23]. This will be used as a basis for new compression schemes.

6. REFERENCES

[1] J. Elseberg, D. Borrmann, and A. Nüchter, "Full Wave Analysis in 3D Laser Scans for Vegetation Detection in Urban Environments," in *Proceedings of the XXIII International Symposium on Information, Communication and Automation Technologies (ICAT '11)*, Sarajevo, Bosnia, October 2011, IEEE Xplore.

[2] American Society for Photogrammetry and Remote Sensing, "Common lidar data exchange format," `http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html` (last accessed May 19, 2019), February 2011.

[3] Martin Isenburg, "LASzip: lossless compression of LiDAR data," European LiDAR Mapping Forum, 2012.

[4] Domen Mongus and Borut Zalik, "Efficient method for lossless lidar data compression," *International Journal Remote Sensing*, vol. 32, no. 9, pp. 2507–2518, May 2011.

[5] Olivier Devillers and Pierre Marie Gandoin, "Geometric compression for interactive transmission," in *IEEE Visualization*, 2000, pp. 319 – 326.

[6] Pierre-Marie Gandoin and Olivier Devillers, "Progressive lossless compression of arbitrary simplicial complexes," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 372 – 379, July 2002.

[7] Ruwen Schnabel and Reinhard Klein, "Octree-based point-cloud compression," in *Symposium on Point-Based Graphics 2006*, July 2006.

[8] Jingliang Peng and C.-C. Jay Kuo, "Octree-based progressive geometry encoder," in *In Internet Multimedia Management Systems IV. Edited by Smith, John R.; Panchanathan, Sethuraman; Zhang, Tong. Proceedings of the SPIE*, 2003, pp. 301 – 311.

[9] Yan Huang, Jingliang Peng, C.-C. Jay Kuo, and M. Gopi, "Octree-based progressive geometry coding of point clouds," in *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, 2006, pp. 103 – 110.

[10] Armin Hornung, Kai M. Wurn, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," in *Autonomous Robots*, 2013.

[11] F. Nenci, L. Spinello, and C. Stachniss, "Effective compression of range data streams for remote robot operations using h.264," in *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

[12] H. Houshiar and A. Nüchter, "3D Point Cloud Compression using Conventional Image Compression for Efficient Data Transmission," in *Proceedings of the XXV International Symposium on Information, Communication and Automation Technologies (ICAT '15)*, Sarajevo, Bosnia, October 2015, pp. 1–8, IEEE Xplore.

[13] E. Dumic, A. Bjelopera, and A. Nüchter, "Projection based dynamic point cloud compression using 3DTK toolkit and H.265/HEVC," in *Proccedings of the 2nd International Colloquium on Smart Grid Metrology (SmaGriMet)*, Split, Croatia, April 2019, pp. 1–4, IEEE Xplore.

[14] J. Elseberg, D. Borrmann, and A. Nüchter, "One Billion Points in the Cloud — An Octree for Efficient Processing of 3D Laser Scans," *ISPRS Journal of Photogrammetry & Remote Sensing (JPRS), Special issue on terrestrial 3D modelling*, vol. 76, pp. 76–88, February 2013.

[15] A. Maglo, G. Lavou, F. Dupont, and C. Hudelot, "3D mesh compression: survey, comparisons and emerging trends," *ACM Computing Surveys*, vol. 9, no. 4, pp. 39:1–39:40, September 2013.

[16] J. Peng, C. Kim, and C.-C. Kuo, "Technologies for 3D mesh compression: A survey," *Journal of Visual Communication and Image Representation*, vol. 16, no. 6, pp. 688–733, 2005.

[17] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault, "Change detection on points cloud data acquired with a ground laser scanner," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. Part 3/W19, pp. 30–35, 2005.

[18] D. Girardeau-Montaut, "Cloudcompare," `http://www.danielgm.net/cc` (last accessed May 16, 2019), January 2013.

[19] M. Gervauts and W. Purgathofer, "A simple method for color quantization: Octree quantization," in *Graphics Gems*, A. S. Glassner, Ed., pp. 287–293. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[20] R. Richter and J. Döllner, "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*, New York, NY, USA, 2010, AFRIGRAPH '10, pp. 121–128, ACM.

[21] J. Boehm and M. Pateraki, "From point samples to surfaces: On meshing and alternatives," in *Proceedings ISPRS Image Engineering and Vision Metrology*, Dresden, Germany, 2006, vol. XXXVI, pp. 50–56.

[22] Martin Magnusson, *The Three-Dimensional Normal-Distributions Transform – an Efficient Representation for Registration, Surface Analysis, and Loop Detection*, Ph.D. thesis, Örebro University, Örebro, Sweden, 2009.

[23] Johannes Schauer and Andreas Nüchter, "Removing non-static objects from 3d laser scan data," *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, vol. 143, pp. 15–38, 2018.