MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Representation and Coding of Large-Scale 3D Dynamic Maps

Cohen, R.A.; Tian, D.; Krivokuca, M.; Sugimoto, K.; Vetro, A.; Wakimoto, K.; Sekiguchi, S.-I.

TR2016-116 September 2016

Abstract

Large-scale 3D maps of indoor and outdoor environments can be created using devices that provide localization combined with depth and color measurements of the surrounding environment. Localization could be achieved with GPS, inertial measurement units (IMU), cameras, or combinations of these and other devices, while the depth measurements could be achieved with time-of-flight, radar or laser scanning systems. The resulting 3D maps, which are composed of 3D point clouds with various attributes, could be used for a variety of applications, including finding your way around indoor spaces, navigating vehicles around a city, space planning, topographical surveying or public surveying of infrastructure and roads, augmented reality, immersive online experiences, and much more. This paper discusses application requirements related to the representation and coding of large-scale 3D dynamic maps. In particular, we address requirements related to different types of acquisition environments, scalability in terms of progressive transmission and efficiently rendering different levels of details, as well as key attributes to be included in the representation. Additionally, an overview of recently developed coding techniques is presented, including an assessment of current performance. Finally, technical challenges and needs for future standardization are discussed.

SPIE Conference on Applications of Digital Image Processing

Representation and Coding of Large-Scale 3D Dynamic Maps

Robert A. Cohen[a], Dong Tian[a], Maja Krivokuća[a], Kazuo Sugimoto[a], Anthony Vetro[*a],
Koji Wakimoto[b], Shunichi Sekiguchi[b]
[a]MERL – Mitsubishi Electric Research Labs, 201 Broadway, Cambridge, MA USA 02139
[b]Mitsubishi Electric Corporation, Information Technology R&D Center,
5-1-1, Ofuna, Kamakura City, 247-8501, Japan

ABSTRACT

Large-scale 3D maps of indoor and outdoor environments can be created using devices that provide localization combined with depth and color measurements of the surrounding environment. Localization could be achieved with GPS, inertial measurement units (IMU), cameras, or combinations of these and other devices, while the depth measurements could be achieved with time-of-flight, radar or laser scanning systems. The resulting 3D maps, which are composed of 3D point clouds with various attributes, could be used for a variety of applications, including finding your way around indoor spaces, navigating vehicles around a city, space planning, topographical surveying or public surveying of infrastructure and roads, augmented reality, immersive online experiences, and much more. This paper discusses application requirements related to the representation and coding of large-scale 3D dynamic maps. In particular, we address requirements related to different types of acquisition environments, scalability in terms of progressive transmission and efficiently rendering different levels of details, as well as key attributes to be included in the representation. Additionally, an overview of recently developed coding techniques is presented, including an assessment of current performance. Finally, technical challenges and needs for future standardization are discussed.

Keywords: compression, 3D point clouds, 3D maps

1. INTRODUCTION

3D maps are often used as means for machines to navigate indoor or outdoor environments in a semi-autonomous or autonomous manner. The maps themselves can be created using devices that provide localization combined with depth and color measurements. The location information may be obtained through GPS, inertial measurement units (IMU), cameras, or combinations of these and other devices, while depth information is typically obtained through time-of-flight, radar or laser scanning systems. Example mapping systems are already commercially available and come in various forms such as the high-end mobile mapping system shown in Figure 1(a) [1]. There also exist lightweight mobile platforms based on sensors mounted on drones or in mobile phones/tablets.

Irrespective of the specific platform that is used to acquire the measurement data, it is possible to generate a 3D map by combining the depth measurements, such as the high-density laser-scanned point cloud in Figures 1(b) [1], with camera images. This combination of point cloud data with camera images to generate a 3D map is illustrated in Figure 1(c) [1]. These maps can further be combined with road markings such as lane information and road signs to create maps to enable autonomous navigation of vehicles around a city as shown in Figure 1(d) [2]. Some examples of methods for detecting these kinds of road markings from camera-captured images and LiDAR point clouds are discussed in [3].

Multiple map layers will be stored and exchanged across the network, including static maps that do not change very frequently and dynamic maps that include real-time information about dynamic objects in the scene such as vehicles or pedestrians. It is anticipated that dynamic map information should be communicated with very low delay. To facilitate autonomous navigation, an example of an algorithm unifying long-term map updates with dynamic object tracking is presented in [4]. An example of a layered data organization paradigm that includes static and dynamic mapping data is the Local Dynamic Map (LDM) [5]. An implementation and study of a collision detection system using LDM is presented in [6].

The amounts of data that are captured for static and dynamic maps can be massive, so efficient representation and compression schemes are needed to facilitate the storage and transmission of scanned data. Commonly-used storage formats for 3D scanned data include the ASTM E57 [7] and LAS [8] file formats. A lossless compression scheme for compressing LAS files is LASzip [9], which is designed to compress points such as LiDAR data that are stored in the

*Corresponding author information. email: avetro@merl.com; phone +1-617-621-7591; web: http://www.merl.com.

same order in which they are captured. The choice of representation format and compression technique depends upon the type of data that is captured and the application requirements.

The rest of this paper is organized as follows. Section 2 outlines the application requirements for representing and compressing large-scale 3D dynamic maps. Section 3 provides a summary of select techniques that have been developed for the compression of point cloud attributes, including prediction techniques, 3D shape-adaptive transforms and 3D graph transform methods for compressing blocks and residuals. Section 4 reviews approaches for compressing 3D point cloud locations, including the well-established octree method and a new method that uses fitted surfaces to approximate the locations. Concluding remarks are given in Section 5.

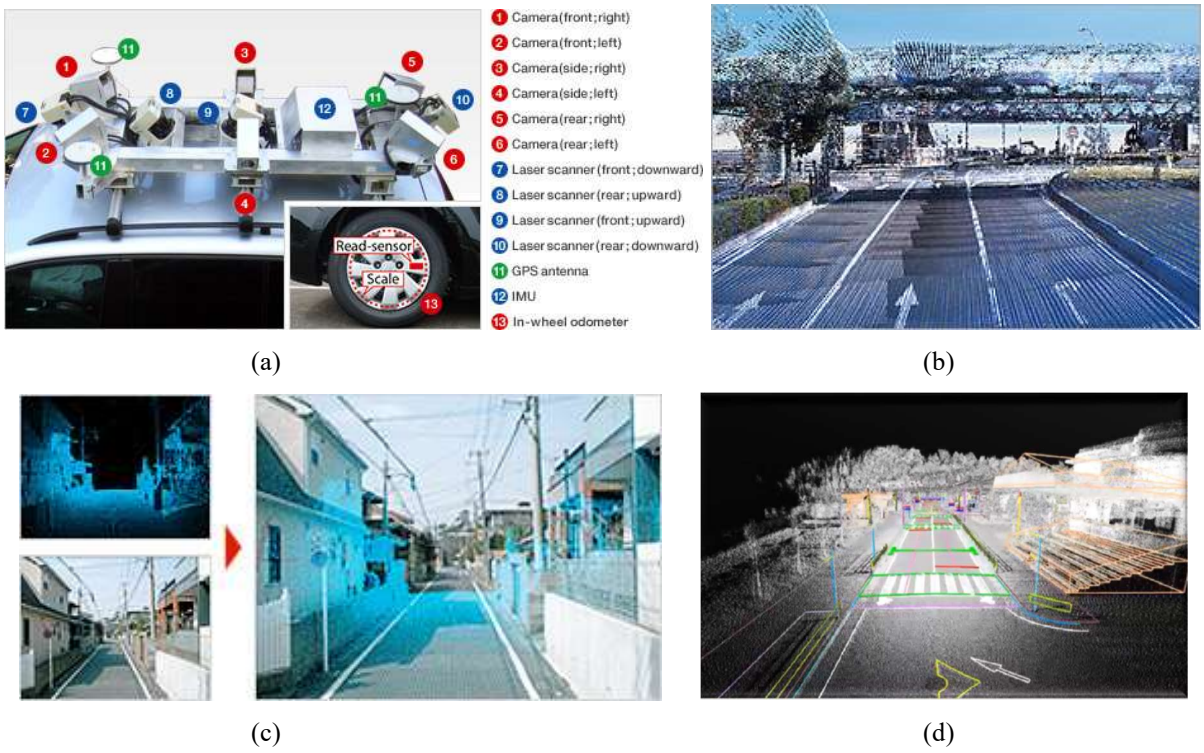

Figure 1. Illustrations of large-scale 3D mapping equipment and data. (a) Mobile mapping system for data acquisition, (b) sample of a high-density laser-scanned point cloud, (c) map creation where images are projected by superimposing laser point cloud data onto camera images, and (d) sample of map with roadside features added. The images of (a)-(c) are from [1], while (d) is from [2].

2. 3D DYNAMIC MAP APPLICATION REQUIREMENTS

In order to study and develop methods for representing and compressing point clouds for mobile mapping applications, it is useful to understand the requirements and capabilities associated with these applications. In this section, we look at typical application scenarios and what capabilities are needed to represent and compress the point cloud data used by these applications.

2.1 Application scenarios

Typical mobile mapping systems gather about 50,000 points every second through laser sensors. If a mobile mapping system runs for 10 minutes while traveling along a road at 60 km/h, it gathers 30,000,000 points corresponding to the 10 kilometer road section. If we plan to measure all interstate highways in the US, the number of points would be 450,000,000,000, which is huge amount of data, assuming we measure each direction separately.

After the point cloud data is gathered, operators visually inspect the data and annotate geographic objects such as lanes, road signs or buildings. The annotated point cloud data forms the 3D map database. Once the database is built, any

portion of the 3D map database can be retrieved according to specified positions and scales. The retrieved map data can be used for several purposes such as autonomous vehicle navigation or road maintenance. The retrieved map data should be less than several hundred megabytes to enable light-weight terminals to handle the map.

To keep up with changes in the roads over the years, data gathering should be periodically performed in order to update the database. To make the update process easy, old point clouds and new point clouds can be automatically compared to determine which portion of the map has been changed. The change detection capability is also useful for finding portions of the roads that need repairs, such as potholes or rutting.

2.2 Representation Format

In order to store and process point clouds, a common understanding is needed as to what kind of data comprises a point cloud and how it is structured. Generally, we consider a point cloud as being a collection of 3D positions (X, Y, Z) of known precision and dynamic range. It is expected that an accuracy of 10-30 cm will be required for the 3D map data. Each position or point can have associated attributes such as color, reflectance, normal vectors, transparency, and many others. Generic attributes such as arbitrary data fields can also be associated with each 3D position.

For some applications, an object may be scanned repeatedly from many different distances or angles, such is the case when performing a scan from a moving vehicle. In this case, it may be desirable to have multiple values of a particular attribute for a given point, where the attribute is dependent upon the view distance and angle when visualizing the point cloud.

While many point clouds may represent the physical world at one point in time, or perhaps the duration of time it took to perform the scan, some scenarios may need representations that support time-varying point clouds. Such representations would be especially useful for dynamic map updating having short time intervals, such as on the order of seconds or milliseconds.

2.3 Compression Format

Compression algorithms can generally be classified as being lossy or lossless, where lossy compression allows the reconstructed or rendered point cloud to be different from the original (uncompressed) point cloud. The distortion resulting from lossy compression can be either geometric distortion, i.e. the reconstructed point positions are different from the original point positions; or attribute distortion, in which one or more of the attributes are reconstructed with some noise or distortion. For lossy compression, a method for controlling the bit-rate or bits per point in the compressed point cloud is needed. For lossless compression, the reconstructed data is mathematically equivalent to the original data.

For time-varying point clouds, the compression method should be capable of leveraging redundancies in temporal variations to achieve greater compression efficiency as compared to separately compressing a set of point clouds captured at different times. Given the large amount of data that can be contained in both time-varying and static point clouds, it is also desirable for the compression method to support progressive or scalable coding. In these cases, a coarse point cloud can be decoded first, and then it can subsequently be refined by decoding additional data.

For view-dependent coding, it would be practical to be able to first decode the point cloud corresponding to a particular region, and after that continue to decode other regions if desired. Similarly, it would be useful to be able to access different regions in the point cloud directly from the compressed data, i.e. have compressed data that supports random access.

In addition to the functional capabilities and requirements mentioned above for point cloud compression formats, some requirements related to implementation should be considered as well. For example, the compression method should support low-complexity or real-time encoding and decoding. These capabilities would be facilitated with compression formats supporting parallel encoding and decoding architectures.

3. ATTRIBUTE COMPRESSION METHODS

Many of the techniques used for the compression of pixel-based images and video sequences can be extended for compressing point cloud attributes. Compression can be achieved in reducing or eliminating redundancies among attributes of neighboring points in a point cloud. This section summarizes selected techniques that use prediction and transforms for compressing attributes.

3.1 Intra prediction of 3D point cloud blocks

Using prediction among blocks to reduce redundancy is a common technique in most video coding standards. Neighboring decoded blocks are used to predict pixels in the current block, and then the prediction error or residuals are optionally transformed and then are coded in a bit-stream. In [11], a block prediction scheme for 3D point cloud data was introduced. As shown in Figure 2(a), points in the current block can be predicted from points contained in non-empty neighboring blocks, when adjacent neighboring blocks are available. The point cloud encoder performs prediction in the x, y, and z directions and chooses the prediction direction that yields the least distortion. Coding the current block without prediction from neighboring blocks is also considered if it yields lower distortion. Therefore, the current block has the option of being coded with or without prediction.

To perform the prediction itself, the method uses a multivariate interpolation/extrapolation to compute a projection of the attribute values in the neighboring block onto the adjacent edge plane of the current block. An example of this step is shown in Figure 2(b). Here, data from known points are used to compute an interpolation or prediction located at an arbitrary point, in this case, along the boundary between the previous block and the current block.

Once the attribute values along the boundary plane are estimated, these values are then projected or replicated into the current block along the direction of prediction, similar to how prediction values are projected into the current block for the directional intra prediction used in standards such as H.264/AVC and HEVC. These projected and replicated values are then used to predict attributes for points present in the current block.

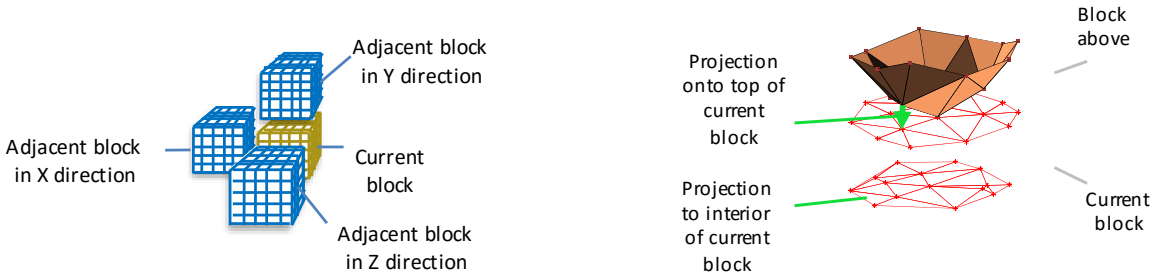

Figure 2. 3D block-based prediction. (a) Available prediction directions; (b) Multivariate interpolation/projection.

3.2 Transforms for 3D block data

After the prediction process, a 3D block containing prediction residuals for each point in the current block, or the current block itself if it yields lower coding distortion, is transformed. Recall that not all the positions in the block may be occupied by a point. The transform must therefore be designed so it will work on these potentially sparse blocks. Two types of transforms have been studied in [11]: a new variant of the shape-adaptive discrete cosine transform (SA-DCT) designed for 3D point cloud attribute compression, and a 3D graph transform.

The shape-adaptive DCT (SA-DCT) [10] is a well-known transform designed to code arbitrarily shaped regions in images. This concept is extended for 3D point cloud data such that positions in the block that do not contain points are considered as being outside the region. The modified SA-DCT process is shown in Figure 3. Given a 3D block of attribute values or prediction residual values, the points present in the block are shifted line by line along dimension 1 toward the border so that there are no empty positions in the block along that border, except for empty lines. One-dimensional DCTs are applied along each occupied column of data along dimension 1, starting at the block border and ending at the last data point present in the block for each column. If there are empty positions between the first and last points in the column, we insert filler values, e.g. zero. An alternative method that will be considered for future work would be to shift the remaining data in the column into those empty positions, thus reducing the lengths of the DCTs. After the DCTs are applied along dimension 1, the shift and transform process is repeated on the transform coefficients along dimension 2. Finally, the process is applied along the third dimension, resulting in one DC and one or more AC coefficients. Compression is achieved by quantizing the coefficients.

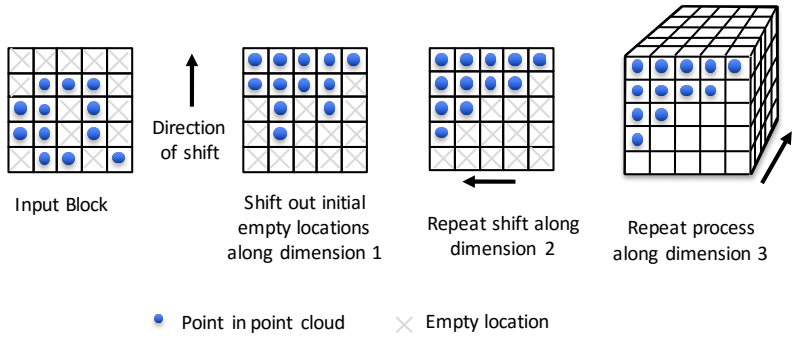

Figure 3. Modified Shape-Adaptive DCT for 3D point cloud data.

The basic idea behind the graph transform is illustrated in Figure 4. A graph is formed by connecting adjacent points present in the 3D block. Two points are considered adjacent if they are at most one position apart in any dimension. Graph weights are assigned to each connection between points, also referred to as a graph edge. The weights of a graph edge are inversely proportional to the distance between the two connected points. An adjacency matrix is populated by the weights, from which a graph Laplacian matrix is computed. The eigenvector matrix of the graph Laplacian matrix is used as a transform for the attribute values. After the transform is applied, each connected sub-graph has the equivalent of one DC coefficient and one or more AC coefficients. Therefore, in the example of Figure 4(a), the graph is composed of two disjoint sub-graphs, so the resulting graph transform will produce two DC coefficients and two corresponding sets of AC coefficients. In [ICIP16], the benefits of expanding the connections between points to the *k* nearest neighbors was studied, hence the graph in Figure 4(c) will only produce one DC coefficient.

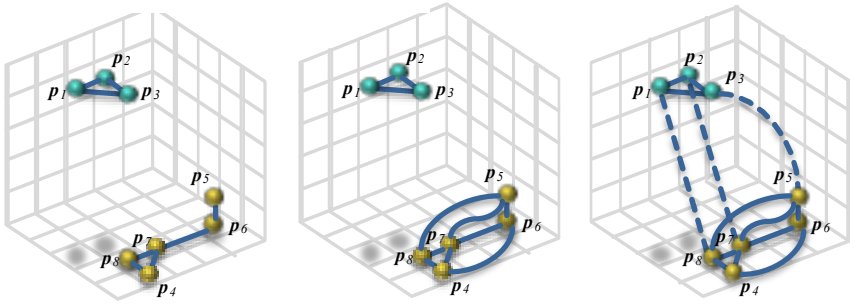

Figure 4. Example of block containing two disjoint sets of connected points. (a) Graph connecting nearest neighbors for each point; (b)(c) graphs connecting points with greater numbers of nearest neighbors.

3.3 Experimental Results

As described in [11], the experiments reported here use an input point cloud (*Statue Klimt PointCloud.ply*) that has been preprocessed so that the points lie on a uniform grid. This point cloud contains approximately 500k points, with each point having a floating-point (x, y, z) location with a corresponding RGB color attribute. We convert the RGB color attributes to YCbCr luminance and chrominance values and use the 8-bit luminance value Y as the attribute used in all experiments. The octree resolution used for preprocessing is r = 1.0. Coding performance is as luminance PSNR vs. the entropy in bits needed to represent the attribute, assuming a sign-magnitude representation of the coefficients, is shown in Figure 5(a). Coding performance results for block sizes k = 5 and k = 10 are shown, along with results for when one large block is used to contain the entire point cloud. Generally, the modified SA-DCT outperformed the graph transform approach. Part of this difference in performance is due to the graph transform producing a DC coefficient that was significantly larger than the AC coefficients. Also, for sparse blocks containing few connected points, more DC coefficients are present in each block when the graph transform is used. More bits are therefore needed to represent that larger number of DC coefficients.

With the aim to overcome some of these issues, as reported in [12], Figure 5(b) illustrates the results of the graph transform with a higher number of connected neighbors. The results show that greater connectivity generally has a benefit at lower bit rates. As the rate increases, the performance of the graph transform with less connectivity is better. The rate at which this crossover point occurs increases as the partition resolution or block size decreases.

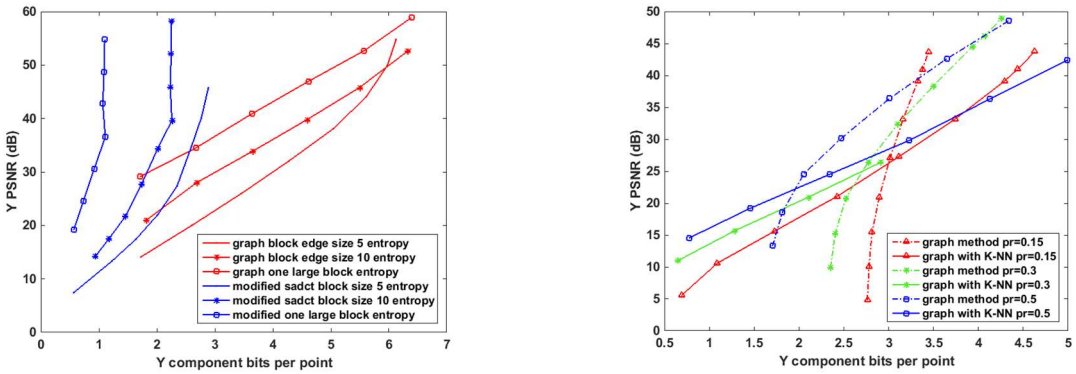

Figure 5. Simulation Results. (a) Comparison of modified SA-DCT and graph transforms; (b) Comparison of coding performance of graph transform with varying number of nearest neighbors.

4. POINT LOCATION COMPRESSION METHODS

Many techniques exist for compressing the point locations or (x,y,z) coordinates of each point in a point cloud. The octree-based method used for mesh representations in [13] was extended in [14] for coding point clouds for which a mesh is not defined. The latter work was implemented using the open-source Point Cloud Library (PCL) [15]. The octree-based method hierarchically partitions a 3D bounding box for the point cloud into leaf nodes or voxels, where each node contains one or more points. The point locations can then be downsampled by representing all the points within each leaf node as the centroid of their corresponding node; or all point locations can be preserved by signaling the difference between each point and the coordinates of the origin of each octree leaf node.

Some additional methods for 3D point cloud compression make use of surface fitting techniques to approximate the point locations. These methods are usually either based on fitting planes to the input point cloud as described in [16], in which approximation capabilities are limited to points that lie on or close to a plane; or else they require a prior decomposition of the point cloud into an octree and then they approximate the points in each octree cell by a localized surface patch [17][18]. The idea of fitting Bézier or other B-Spline surfaces to compactly represent 3D point clouds has been explored in [19] and [20]; however, these applications do not aim to be able to reconstruct the original point locations, but to create a new representation of the point cloud surface, which is an approximation of the original point cloud geometry.

In this section, we present a method for point location compression, which is based on fitting Bézier surface patches to a hierarchical decomposition of an organized point cloud. These surface patches are used as prediction models for the point cloud geometry. We achieve compression by encoding the model parameters, as well as quantizing and encoding the residual vectors that represent the fitting errors between the points on the model and each corresponding input point location.

The input to our method is a point cloud organized in a raster-grid structure, where each (x,y,z) coordinate of a point is associated with a cell on a grid whose width × height equals the total number of points in the point cloud. Note that this does not mean that the (x,y,z) coordinates themselves are necessarily aligned to a 3D spatial grid. We adaptively subdivide this input into a set of cubic Bézier surface patches, according to a predefined fitting error threshold. The subdivision process is hierarchical: it starts with one large patch (rectangle) that covers the entire point cloud; if the fitting error for this patch is too large, then the patch is divided into two patches. This process is repeated hierarchically on any patches having a sufficient number of points, until we have a set of patches that approximate the input point cloud as well as possible given the prescribed fitting error threshold. Each of these Bézier surface patches is a 4 x 4 grid of 16 control points and corresponds to a rectangular sub-domain in the input data parameter domain. These patches, i.e.

their control points, constitute our model that generates a prediction of the point cloud geometry. The residual or difference between each fitted point and its corresponding point in the original point cloud is then quantized and signaled along with the control points. An illustration of a model having four patches is shown in Figure 6. A binary tree representing the splitting process is also signaled.

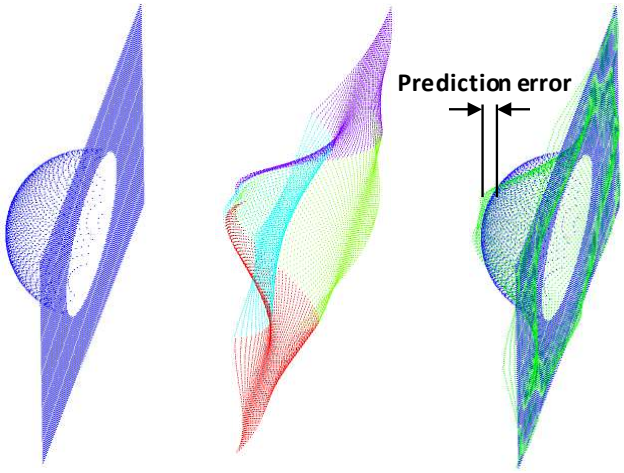

Figure 6. Illustration of point cloud, a predictor having four patches and the prediction error

In Figure 7 we compare the compression performance between the octree-based method of [14] and the patch-fitting method described above. For the patch-fitting method, the initial point at approximately 40 dB was achieved without signaling the residuals; only the control points and associated hierarchy information were signaled, using a total of approximately 0.1 bits per input point. For the remaining points, the residuals with successively finer levels of quantization were additionally coded.

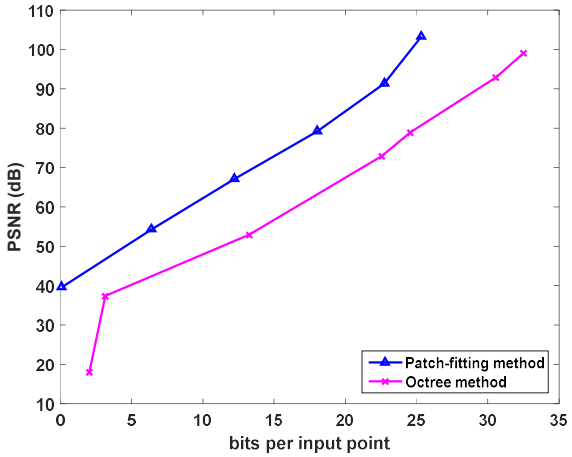

Figure 7. Compression performance for octree and Bézier-fitting methods

5. CONCLUDING REMARKS

In this paper, we outlined the application requirements for representing and compressing large-scale 3D dynamic maps, with a focus on point cloud representations. Examples of recently developed methods for compressing point cloud attributes and geometries were also presented. With the recent increased proliferation of 3D mobile mapping scanner systems used to capture both static and dynamic map data, the need to develop standards for representing 3D dynamic map data is becoming more urgent. For storing and transmitting these increasing amounts of data, efficient compression schemes are needed as well.

REFERENCES

[1] Mobile Mapping System – High-accuracy GPS Mobile Measuring Equipment, Sep. 2015 [Online]. Available: http://www.mitsubishielectric.com/bu/mms/features/index.html

[2] H. Koyama, "Activity Plan of Dynamic Map Study for SIP-adus," 2nd SIP-adus Workshop on Connected and Automated Driving Systems, October 27, 2015 [Online]. Available: http://www.sip-adus.jp/workshop/program/speaker/profile/dm/koyama.pdf

[3] J. Landa and D. Prochazka, "Automatic road inventory using LiDAR," *Procedia Economics and Finance*, vol. 12, pp. 363-370, Sep. 2014,

[4] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale and R. Siegwart, "Long-term 3D map maintenance in dynamic environments," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014, pp. 3712-3719.

[5] ETSI TR 102 863 V1.1.1, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and guidance on standardization," June 2011.

[6] H. Shimada, A. Yamaguchi, H. Takada, K. Sato, "Implementation and Evaluation of Local Dynamic Map in Safety Driving Systems," *Journal of Transportation Technologies*, pp. 102-112, May 2015.

[7] D. Huber, "The ASTM E57 File Format for 3D Imaging Data Exchange," *Proceedings of the SPIE Vol. 7864A, Electronics Imaging Science and Technology Conference (IS&T), 3D Imaging Metrology*, Jan. 2011.

[8] The American Society for Photogrammetry & Remote Sensing, "LAS specification version 1.4 – R13," Jul. 2013. [Online]. Available: http://www.asprs.org/LAS_Specification

[9] M. Isenburg, "LASzip: lossless compression of LiDAR data," *Photogrammetric Engineering & Remote Sensing*, no. 2, Feb. 2013, pp. 209-217.

[10] T. Sikora T, and B. Makai, "Shape-adaptive DCT for generic coding of video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 1, pp. 59–62, Feb. 1995.

[11] R. A. Cohen, D. Tian, and A. Vetro, "Point Cloud Attribute Compression using 3-D Intra Prediction and Shape-Adaptive Transforms," *Proc. Data Compression Conference*, Snowbird, UT, April 2016.

[12] R. A. Cohen, D. Tian, and A. Vetro, "Attribute Compression for Sparse Point Clouds using Graph Transforms," *Proc. IEEE International Conference on Image Processing*, Phoenix, AZ, September 2016.

[13] J. Peng and C.-C. Jay Kuo, "Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 609–616, July 2005.

[14] J. Kammerl, N. Blodow, R.B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in Robotics and Automation (ICRA), 2012 IEEE International Conference on, May 2012, pp. 778–785.

[15] R. B. Rusu and S Cousins, "3D is here: Point Cloud Library (PCL)," in IEEE International Conference on Robotics and Automation (ICRA), Shanghai, CN, May 2011.

[16] V. Morell, S. Orts, M. Cazorla, and J. Garcia-Rodriguez, "Geometric 3D point cloud compression", *Pattern Recognition Letters*, vol. 50, pp.55-62, Jun. 2014.

[17] S.-B. Park and S.-U. Lee, "Multiscale Representation and Compression of 3-D Point Data", *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 177-183, Jan. 2009.

[18] J. Smith, G. Petrova, and S. Schaefer, "Progressive encoding and compression of surfaces generated from point cloud data", *Computers & Graphics*, vol. 36, pp. 341-348, Mar. 2012.

[19] F. Remondino, "From point cloud to surface: the modeling and visualization problem", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, no. 5, p. W10, Feb. 2003.

[20] B. Li and S. Yi, "Parametric 3D Object Representation and Applications", *Proc. International Conference on Computer Graphics, Imaging and Visualization*, Bangkok, Thailand, Aug. 2007.