Point Cloud Compression and Transmission for Remote Handling Applications

Salvador Pacheco-Gutierrez*, Ipek Caliskanelli, Robert Skilton

UK Atomic Energy Authority, Remote Applications in Challenging Environments, Culham Science Centre, Abingdon, Oxfordshire OX14 3DB, United Kingdom.

* Corresponding author. Tel.: +44 (0)1235 467107; email: salvador.pacheco-gutierrez@ukaea.uk

Manuscript submitted August 25, 2020; accepted October 10, 2020.
doi: 10.17706/jsw.16.1.14-23

Abstract: Remote handling systems are commonly used for decommissioning and maintenance of hazardous environments, especially in the nuclear sector. The necessity for a more realistic and accurate user interaction with the remote environment has led research towards the usage of immersive technologies such as augmented and virtual reality. In order for this to succeed, the state of the remote environment needs to be known accurately at all times. Information gathered using RGB-D cameras can serve this purpose. The high accuracy and density of data retrieved by these devices provide an extraordinary insight of the remote environment but can represent a burden on the communication channels. This paper addresses two point cloud compression techniques based on kd-trees and octrees for point cloud data transmission within a Robot Operative System (ROS) communications middleware.

Key words: Kd-tree, octree, point cloud compression, remote handling, ROS.

1. Introduction

In the nuclear sector, inspection and maintenance of the facilities are carried out by remote operations (i.e. using teleoperation systems). Such remote operations involve handling of devices and objects in hazardous environments such as nuclear reactor vessels or contaminated gloveboxes. Therefore, the utilisation of technologies that guarantee safety of the human operator and protection of the facility are required e.g. robotic systems.

Although the use of teleoperated systems in nuclear industry can be considered as the standard tool nowadays, there is a big push towards the integration of novel solutions in order to assist and guide the human operator, e.g. viewing systems. This aim is to improve safety while increasing efficiency and reducing operational costs.

Given the fact that teleoperated systems heavily depend on in-situ viewing systems, research focused on the recreation of a more realistic, real-time and accurate visualisation of environment is required. Furthermore, improvements on viewing systems are more likely to reduce the mental burden on the human operator whilst providing guidance on specific tasks in the assistive manner.

Virtual reality (VR), a fully immersive visual technology, frequently used in the recreation of 3D environments that the user can interact with, was first introduced as a simulation tool in 1960s [1], and was rapidly adopted for remote handling operation in JET (Joint European Torus) at UKAEA [2].

Subsequent technologies such as augmented reality (AR), which is a human-machine interaction tool that superpose 3D information on the real scene, appear suitable to remote handling applications. However, problems related to depth perception, fidelity, luminescence [3] and particularly latency create a gap for remote nuclear handling tasks.

For the integration of AR in remote handling applications, it is necessary to maintain an up-to-date model of the environment to guarantee an accurate virtual representation of the scene. Therefore, it is necessary to accurately measure the state of the remote environment, objects and robotic systems in real-time.

Off-the-shelve RGBD cameras can be used for retrieving 3D information in low radiation environments (e.g. gloveboxes). These cameras can provide high-resolution point clouds[1] ranging from a few thousands to several million points. This amount of data, although beneficial in terms of accuracy and resolution, can represent a burden for the communication channels and transport of data, especially in scenarios where the operator is far away from the location of interest. Having said that, the inspiration for this work arise due to absence of benchmarked compression and transmission techniques for point cloud data.

This paper assesses two techniques for compression of point clouds: an octree-based technique from PCL [4] and a kd-tree-based technique from the recently developed Google's DRACO library [5]. The communication ecosystem was based on the middleware ROS (Robotic Operative System) Melodic, for which a new message type for compressed point cloud was developed. For comparison purposes, two RGBD sensors with different resolutions were used to gather 3D point cloud data: Kinect2 and Zivid One+ Small. Codification and decodification of point cloud data streams is performed in different processing units communicating through a local area network, this in order to mimic a remote data collection, compression and transmission (see Fig. 3).

This paper is organised as follows: section 2 provides an insight of the latest developments in point cloud compression techniques. Section 3 discusses the implementation of the compression techniques analysed, while section 4 discusses the results obtained. Conclusions can be found in section 5.

2. Literature Review

The transmission of high-density point clouds requires large transmission bitrates. According to [6], for a set of ~1M points transmitted at 30 frames per second (fps), results in a total bandwidth of ~3.6Gbps, this for the solely purpose or having a visually pleasant and realistic representation of 3D point cloud data. This clearly exemplifies the needs for point cloud compression.

Similar to video or audio, point cloud compression (PCC) techniques can also be divided into two: lossy as [4][7] and lossless as [5]. On one hand, lossless compression eliminate redundancy in the data whilst maintaining original information that results in decompressing the data without losing any of it. On the other hand, lossy compression removes *unnecessary* data by means of data quantisation [8].

The need for standardisation of compression techniques brought the attention of the Moving Picture Experts Group (MPEG), which launched a call for proposals [9] resulting in what can be considered another division in compression techniques: video-based compression (V-PPC) and geometry based compression (G-PCC)[6]. The former technique V-PPC converts 3D point clouds into 2D images for video streaming, therefore benefiting from the well-developed video compression techniques. The latter (G-PCC), encodes information directly in 3D space using data structures, such as octrees. As mentioned in [6], G-PCC is more suitable for sparse point clouds, which applies for most real-time data acquisition applications. Standardisation works are still ongoing and are expected to be delivered under ISO/IEC 23090-5 and -9 for V-PCC and G-PCC respectively.

Octree-based data structures, that applies a recursive decomposition principle, date from the early eighties and were developed to produce a representation of 3D objects [10], [11]. Other applications that implement octrees include 3D navigation [12], [13] and shape analysis by means of convolutional neural networks [14]

[1] A 3D point cloud is a set of points in $\mathbb{R}^3$ usually carrying other attributes such as colour and texture.

[15]. In 2006, the first application of octrees for point cloud compression was proposed in [16], where the point cloud is encoded based on the occupied octree cells. For PCC applications, octrees are sheared up to a certain depth level in the tree-like structure, then, points lying within each cell are combined, together with its attributes, to generate a single data point. This technique was soon implemented in the emerging Point Cloud Library for dynamic point cloud coding [7].

Similar to octrees, kd-trees are also a space partitioning data structure that utilises a multidimensional binary search tree [17]. Initially developed for the purpose of fast search, research started to be conducted on applying this technique for mesh compression [18] and its adaptation for point cloud compression [19] and combination with a model-based approach [20].

Within the robotic operative system (ROS) middleware, the handling, transfer and processing of raw point cloud data messages remains as a problem [21]. In other words, large point cloud data sets represent a burden for the ROS communication network. Although ROS carries PCL within its library directory for different point cloud-related processing tasks, there is currently no native message for compressed point clouds nor compression technique natively implemented.

3. Point Cloud Compression Strategy

The compression of the point cloud is performed using two different strategies: octree and kd-tree based codification. For the first, we used an implementation provided by the Point Cloud Library [7], and for the latter a relatively new implementation by Google called DRACO [5]. Each strategy provides different parameters for adjusting the compression to our needs. It is important to mention that these parameters need to be adjusted depending on the type of sensor used, point cloud resolution and density required.

3.1. Octree-Based Point Cloud Compression

An octree is a data structure that allows to perform spatial partitioning to a set of points converting them into a tree-like structure (see Fig. 1). This is done by iteratively dividing the space into eight identical boxes, starting with the initial bounding box, also known as the root.

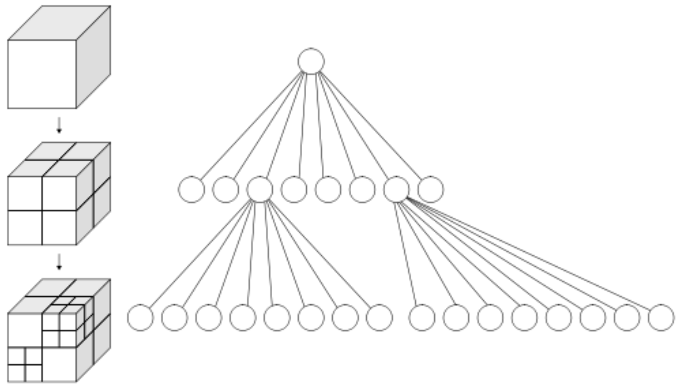

Fig. 1. Octree schematic representation [22].

The construction of an octree is achieved by iteratively traversing the tree from lower to higher depth levels, while assigning the corresponding child leaf to every branch. This is done until each point is added to a list of points corresponding to a leaf node [7]. At the highest depth level, each leaf node can be considered as a Boolean value (bit). Due to the fact that spatial partition is always performed in groups of eight elements a node can be represented as a single byte. Knowing the codification order, the spatial distribution of a set of points can be efficiently represented using a stream of bytes. Other relevant data such as the three depth, can be added to the stream as a header providing information to the decoder. This can then be easily transformed into a ROS message.

PCL [4] provides an octree implementation class with six configuration parameters and 12 pre-defined compression profiles listed in Table 1. The nomenclature for the pre-defined compression profiles is intuitevely defined. For instance, **MRON-C** stands for **M**edium **R**esolution **ON**line **C**olour, whereas **HROFF-NC** means **H**igh **R**esolution **OFF**line **N**o **C**olour. MRON-C is the default compression profile.

Table 1. Octree Configuration Parameters

Compression profile	Compression parameters					
	Point Resolution	Octree Resolution	Downsample?	Frame Rate	Colour Coding	Colour Bit Resolution
LRON-C	0.01	0.01	True	50	True	4
LRON-NC	0.01	0.01	True	50	False	4
LROFF-C	0.01	0.01	True	100	True	4
LROFF-NC	0.01	0.01	True	100	False	4
MRON-C*	0.005	0.01	False	40	True	5
MRON-NC	0.005	0.01	False	40	False	5
MROFF-C	0.005	0.01	False	100	True	5
MROFF-NC	0.005	0.005	True	100	False	5
HRON-C	0.0001	0.01	False	30	True	7
HRON-NC	0.0001	0.01	False	30	False	7
HROFF-C	0.0001	0.01	False	100	True	8
HROFF-NC	0.0001	0.0001	True	100	False	8

*Default profile

The class allows to specify whether colour is included during codification, this is set by the colour coding flag. Downsample the cloud prior to codification is also possible, this can increase compression speed but can significantly reduce the density and quality of the cloud.

3.2. Kd-Tree-Based Point Cloud Compression

Similar to octrees, a kd-tree is a data structure based on spatial partitioning. Conversely to octrees, on which partition is always defined by a set of eight children bounding boxes, kd-trees performs space partition by means of hyperplanes aligned with the dimensional axes. This iterative one-dimensional partition is usually computed by algorithms such as Quick Sort [23], which uses the median as root value and then separates values on its left and right. This is then repeated on the previously obtained divisions until there is only one element in the partition as shown in Fig. 2.

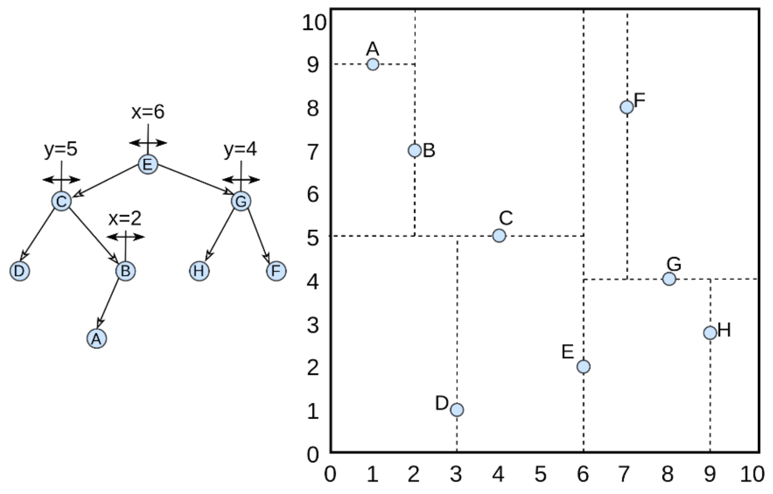

Fig. 2. Kd-tree partition and tree schematic for the set of points: [(1,9), (2,7), (3,1), (4,5), (6,2), (7,8), (8,4), (9,3)].

DRACO is a open-source library developed by Google with aims at improving transmission of 3D graphics [5]. DRACO provides a class for mesh and point cloud compression by means of kd-trees. This class allows to control several compression options such as the number of quantisation bits for all attributes. It also allows the specification and compression of user-defined attributes, which results in high flexibility and adaptation to carry multiple types on point cloud information. A key configuration parameter is the compression level, which allows specification of compression quality. The highest the compression level, the long it takes to compress. For our application, we focus primarily on the compression of geometry and colour attributes listed in Table 2. In terms of coordinate quantisation, the actual precision will depend on the scale of the attribute values, e.g. the scale of the X, Y and Z.

Table 2. Kd-Tree Configuration Parameters

Parameter	**Default value**	**Description**
Position quantification [bits]	10	Precision of the quantised box
Colour quantification [bits]	8	$2^8 = 256$ colours
Compression level (CL)	7	Range = [0-10] 10 – Best compression, slowest speed 0 – Worst compression, fastest speed

In comparison with the octree-based PCC, that holds a consistent and predictable structure over time, kd-trees divide the space at each tree leaf based statistical parameters such as the median. This makes kd-trees highly dependent on the point cloud's bounding box and the data lying within [8]. This bounding box can vary drastically when acquiring data in real-time and by the type of sensor used. Hence, kd-trees are considered efficient for one-off PCC, for multiple frame codification, kd-tree can be expensive to compute.

3.3. ROS Integration

The integration of cameras, compression algorithms and network connection were conducted within the ROS middleware. To achieve compatibility, containerisation was also utilised in order to isolate camera-specific libraries and controllers from the rest of the system. Fig. 3 shows a simplified schematic diagram of the setup.

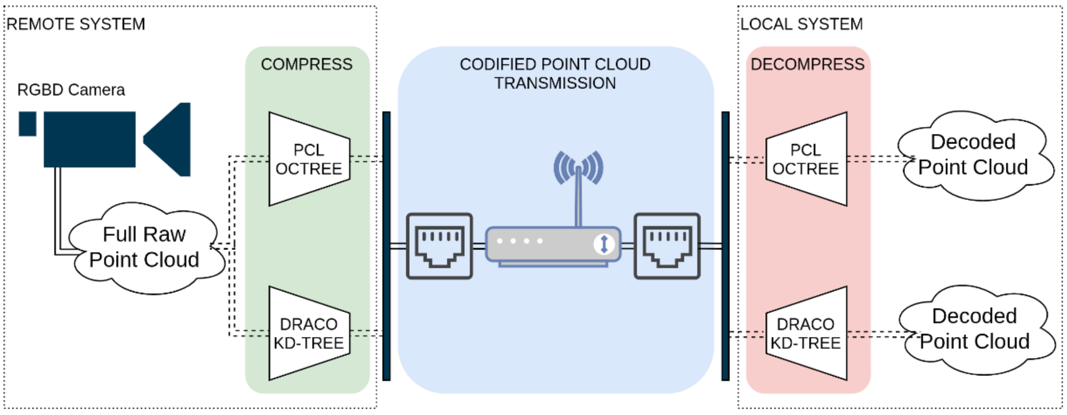

Fig. 3. Schematic diagram of the compression/decompression setup.

The codified point cloud resulting from the compression was encapsulated in the form of a customised ROS message. The message structure contains information such as sequence number, time stamp and reference frame, cloud ID and the *string* of characters corresponding to the codified cloud.

4. Results

The performance of each compression technique is assessed against four key performance metrics:

- Number of points, before compression and after decompression.
- Compression and decompression execution time.

- Size, in kb, of the compressed cloud.
- Transmission time.

Compression and decompression are performed in two different computers simulating the remote and local systems. Both devices have been connected to a standard home router through Ethernet. In order to ensure synchronised time stamps for determining transmission time, computer's clocks were synchronised using network time protocol (NTP). TCP is used for network communications. Other protocols such as UDP may provide faster transmission rates at the cost of potentially less reliable data transfer.

For comparison purposes, we have used two different RGBD cameras with different characteristics: *Kinect2* and *Zivid One+ Small*. Kinect2 is widely used within the community due to its low cost and high-quality point cloud data. Zivid is a sophisticated industrial camera that provides very high resolution and density point clouds. In its SD resolution, Kinect2 can provide circa 220k points per frame, where in contrast with Zivid can deliver over 2M points per frame.

For the PCL's octree compression, the 12 pre-defined profiles shown in Table 1 are compared. For the DRACO's kd-tree codification, the compression level (CL), listed in Table 2, is swept from 0 to 10. For each compression profile, a stream of 100 raw point clouds is gathered within the workspace of each sensor. This is done to avoid NaN values (invalid readings) and to achieve a consistently dense point cloud.

The results obtained using Kinect2 and Zivid camera are shown in Fig. *4* and Fig. *5* respectively. The left and right columns on each figure corresponds to a different compression technique, kd-trees (DRACO) and octrees (PCL) respectively.

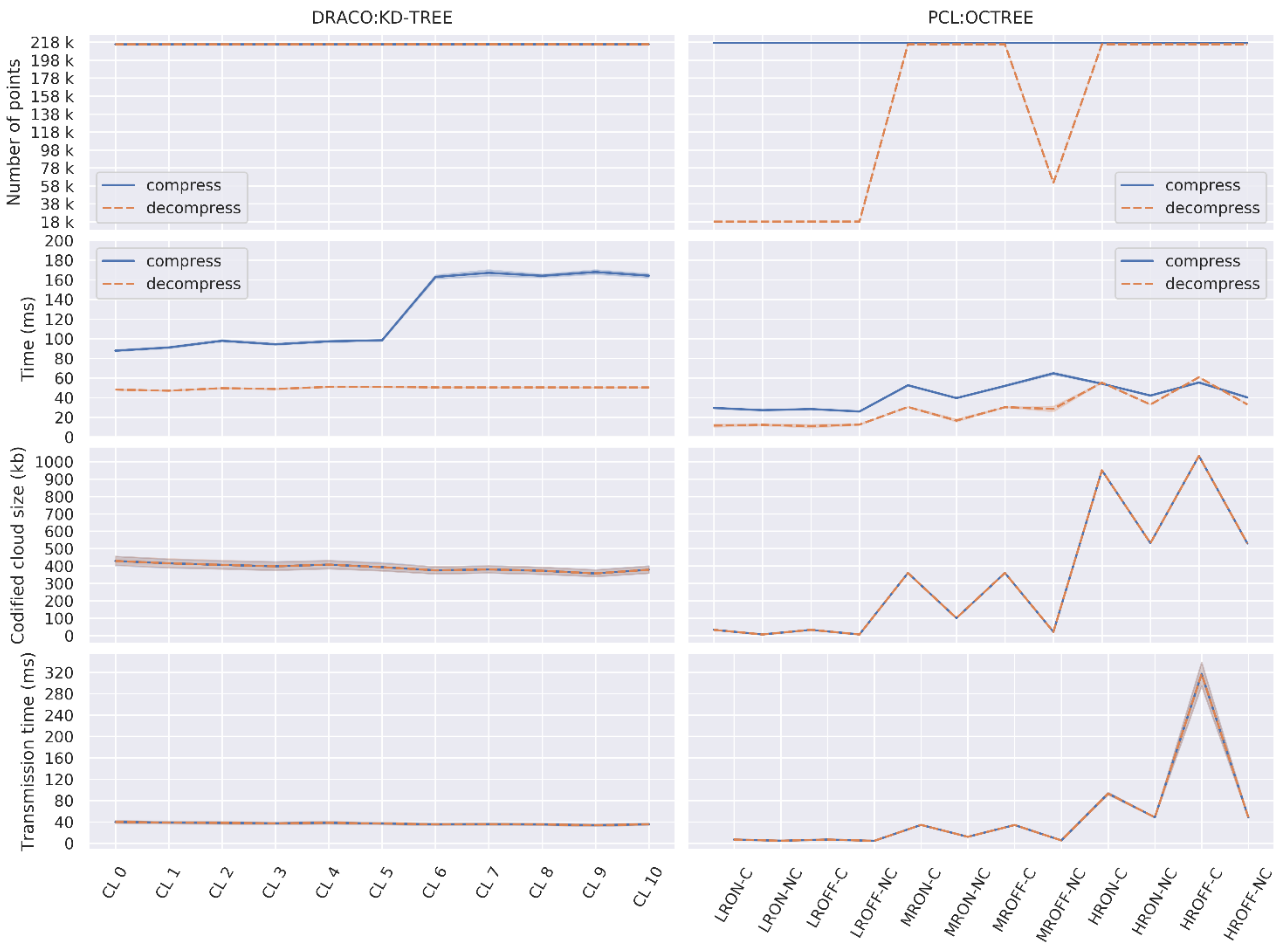

Fig. 4. Compression performance evaluation using Kinect2.

The first row shows the number of points before compression and after decompression. As can be observed, DRACO performs a lossless compression while PCL is lossy, having a loss ratio of up to ~ 92% in its low-resolution compression profiles (LRON-C, LRON-NC, LROFF-C, LROFF-NC).

The second row shows the execution time for compression and decompression. DRACO's kd-tree encoder exhibits a jump in the computational time CL5 and CL6 for compression while decompression time remains steady over all compression levels. PCL's compression and decompression times follow a similar trend varying due to resolution and colour coding.

The codified cloud size, listed in the third row, remains almost the same for all DRACO's compression levels making this parameter invariant to the compression level. This also relates to the invariance on decompression time. PCL's octree cloud size varies drastically when colour is not encoded. For Kinect2 and Zivid point cloud data, DRACO's compression ratio is ~5.25 and ~7.75, respectively, whilst PCL's compression ratio using MRON-C profile is ~6 and ~10.9, respectively.

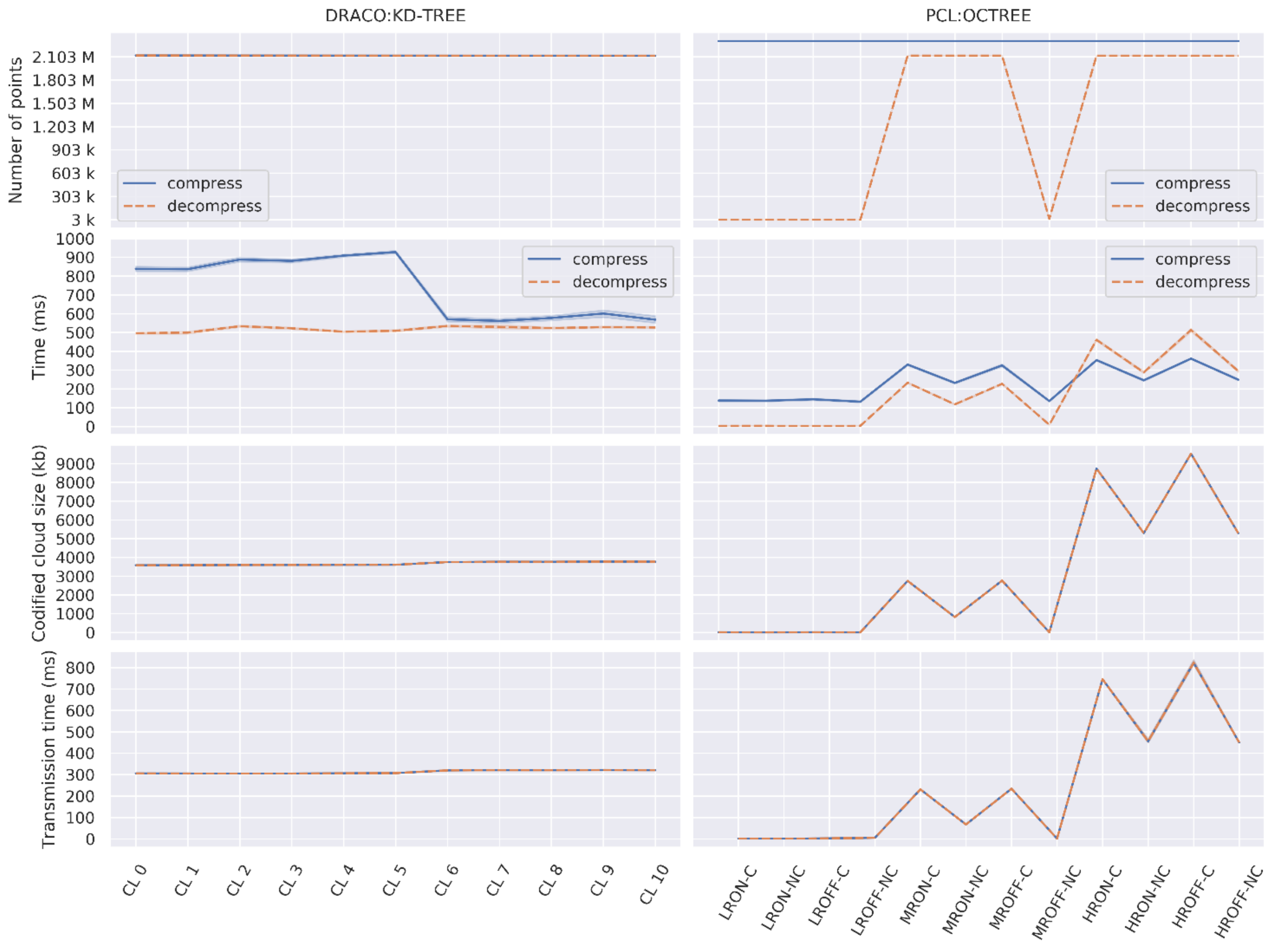

Fig. 5. Compression performance evaluation using Zivid One+ Small camera.

Transmission time, shown in the last row of Fig. 4 and Fig. 5, is dependent on the size of the compressed cloud. For referencing, the transmission of a single raw point cloud frame of Kinect2 and Zivid cameras takes ~1.34 and ~4.5 seconds respectively. Thus, for Kinect2 and Zivid cameras, DRACO provides a transmission ratio improvement of ~33.5 and ~15 respectively, whilst PCL's MRON-C profile provides a transmission ratio improvement of ~35.2 and ~19.5 respectively. The transmission time could be improved by other means, this would include modification on both the software and hardware. To begin with, a different communication protocol could be used, such as UDP. Implementing the same algorithms on a high-end network system together with a dedicated network device could offer a faster transmission rate. Given the importance of high transmission rate in achieving realistic, accurate visualisation systems, we may consider adopting some of these changes in the future to speed up transmission of the data.

5. Conclusions

This paper provides a comparative study between two different point cloud compression techniques, kd-tree and octree, using DRACO and Point Cloud Library, respectively. This study is implemented using the ROS communication middleware. Compressed point cloud data transmits across the network faster, as opposed to transmission of raw data. Empirical results illustrate that the kd-tree implementation exhibits invariable point cloud compression size for different compression levels. Thus, causes variation only on the quality of the compressed cloud. The MRON-C (medium resolution with colour) compression profile used in the octree implementation results in a challenging trade-off between data loss and compressed cloud size, whilst maintaining colour information.

To facilitate integration within the ROS middleware, a new ROS message type is created and used to transmit the codified data across the ROS network. Latency is considered as a big problem within the ROS community, and results demonstrated in this paper show that the point cloud compression improves the latency of the point cloud transmission drastically.

The research presented in this paper lays out the payment towards achieving improvements on viewing systems and we are aiming to extend our implementation to facilitate real-time visualisation of remote environments for object manipulation and grasping using a digital twin in the future.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

SP conducted the research and software development, IC and RS provided guidance, support and feedback; all authors had approved the final version.

Acknowledgment

This work is funded by the UK Engineering \& Physical Sciences Research Council (EPSRC) Grant No. EP/S03286X/1. The authors would like to thank Michael Hellebrand and National Nuclear User Facility for Hot Robotics project (Grant No. EP/T011432/1) for leasing hardware to support this research.

References

[1] Nee, A., & Ong, S. (2013). Virtual and augmented reality applications in manufacturing. *Proceedings of the 7th IFAC Conference on Manufacturing Modelling, Management, and Control*, *IFAC Proceedings Volumes* (pp. 15 – 26).

[2] Sanders, S., & Carman, P. (2006). Colour, design and virtual reality at jet. *Optics & Laser Technology: Vol. 38, No. 4*, (pp. 335 – 342). Colour and Design in the natural and man-made worlds.

[3] Do, T. D., Laviola, J. J., & McMahan, R. P. (2020). The effects of object shape, fidelity, color, and luminance on depth perception in handheld mobile augmented reality. *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

[4] Rusu, R. B., & Cousins, S. (2011). 3D is here: Point cloud library (PCL). *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China.

[5] Google (2020). Draco: 3d data compression. Library for compressing and decompressing 3D geometric meshes and point clouds, Last access February 2020.

[6] Graziosi, D., Nakagami, O., Kuma, S., Zaghetto, A., Suzuki, T., & Tabatabai, A. (2020). An overview of

ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing.*

[7] Kammerl, J., Blodow, N., Rusu, R. B., Gedikli, S., Beetz, M., & Steinbach, E. (2012). Real-time compression of point cloud streams. *Proceedings of the 2012 IEEE International Conference on Robotics and Automation* (pp. 778–785).

[8] Cao, C., Preda, M., & Zaharia, T. (2019), 3d point cloud compression: A survey. *Proceedings of the 24th International Conference on 3D Web Technology* (pp. 1–9).

[9] Motion Picture Experts Group. (2016). Call for proposals for point cloud compression. Retrieved from: https://mpeg.chiariglione.org/standards/mpeg-i/point-cloud-compression/call-proposals-point-cloud-compression

[10] Jackins, C. L., & Tanimoto, S. L. (1980). Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing, 14(3),* 249 – 270.

[11] Meagher, D. (1982). Geometric modelling using octree encoding. *Computer Graphics and Image Processing*, *19(2)*, 129 – 147.

[12] Saona-Vazquez, C., Navazo, I., & Brunet, P. (1999). The visibility octree: A data structure for 3d navigation. *Computers and Graphics, 23(5)*, 635 – 643.

[13] Chen, S. (1990). A spherical model for navigation and spatial reasoning. *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 776–781).

[14] Wang, P. S., Liu, Y., Guo, Y. X., Sun, C. Y., & Tong, X. (2017). O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics*, *36(4)*.

[15] Tatarchenko, M., Dosovitskiy, A., & Brox, T., (2017). Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[16] Schnabel, R., & Klein, R. (2006). Octree-based point-cloud compression. *Proceedings of the Symposium on Point-Based Graphics* (pp. 111–120).

[17] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM, 18,* p. 509517.

[18] Gandoin, P. M., & Devillers, O. (2002). Progressive lossless compression of arbitrary simplicial complexes. *ACM Transactions on Graphics, 21*, 372379.

[19] Waschbüsch, M., Gross, M., Eberhard, F., Lamboray, E., & Würmlin, S. (2004). Progressive compression of point-sampled models. *Proceedings of the First Eurographics Conference on Point-Based Graphics, SPBG'04*. Eurographics Association (p. 95103).

[20] Lien, J. M., Kurillo, G., & Bajcsy, R. (2010). Multi-camera tele-immersion system with real-time model driven data compression. *The Visual Computer*, *26*, 3–15.

[21] Forums, R. D. (2019). Compressed pointcloud2. *ROS discourse forums*. Retrieved from: https://discourse.ros.org/t/compressed-pointcloud2/10616

[22] Wikipedia, W. F. (2010). Schematic drawing of an octree, a data structure of computer science. Retrieved from: https://commons.wikimedia.org/wiki/File:Octree2.svg

[23] Hoare, C. A. R. (1961). Algorithm 64: Quicksort. *Communications of the ACM,* Association for Computing Machinery.

Copyright © 2021 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0)

Salvador Pacheco-Gutierrez was born in Irapuato, Mexico. He studied in the University of Guanajuato, where he obtained his BSc in mechatronics engineering focusing on kinematic analysis of robot manipulators. He then studied a PhD in robotics in the University of Manchester UK, applying control theory and computer vision to mobile robots for nuclear decommissioning.

He conducted post-doctoral research in the University of Manchester in model predictive control for applications in energy management. He then moved to industry to lead the automation and robotics development for Sensor Coating Systems LTD in London. During this time, he delivered several public funded research projects and supervised internships and master students in their industrial placements and thesis. He currently works as Control Systems Software Engineer in the UK Atomic Energy Authority, where he is working in an EPSRC funded project focused on the development of a digital-twin system for remote handling applications in collaboration with The University of Manchester and the Korea Atomic Energy Research Institute. His main interests are robotics, computer vision, machine learning and software development.

Ipek Caliskanelli is a research engineer at RACE. Ipek has granted her PhD in computer science from the University of York in 2014. Her thesis explored resource efficiency and load-distribution of distributed embedded systems.

Ipek's research interests are focused on optimisation, real-time systems-of-systems control, multi-agent systems, cooperation and coordination. Ipek has 10 years of academic and industrial research experience in developing software frameworks and control algorithms for wide range of distributed digital systems including embedded, cyber-physical and robotic systems. Her industry focus primarily span around real-time interoperable systems-of-systems control frameworks for nuclear and other extreme environments.

Robert Skilton graduated with an MSc in cybernetics in 2011, and is currently studying for a PhD in autonomous robotics and machine learning at the Surrey Technology for Autonomous systems and Robotics (STAR) Lab.

He is the head of cybernetics and lead technologist at RACE, a UK centre for Remote Applications in Challenging Environments, where he leads a team specialising in control systems, autonomy, and perception for robotic operation and inspection in hazardous environments. He is a chartered engineer, brings experience in developing robotic systems for hazardous environments and has developed numerous robotic and software platforms for use in nuclear and other extreme environments. Robert has experience from a wide range of roles on industrial engineering and R&D projects including in telerobotics, and is currently leading various related activities including the Robotics and AI in Nuclear (RAIN) work on teleoperation of industrial robots.