LOW LATENCY SCALABLE POINT CLOUD COMMUNICATION

Anique Akhtar, Birendra Kathariya, Zhu Li

University of Missouri-Kansas City
email: {aniqueakhtar, bkkvh8}@mail.umkc.edu, lizhu@umkc.edu

ABSTRACT

Mobile edge and V2V Low latency streaming of 3D information is a crucial technology for smart city and autonomous driving. In this paper, we propose a joint source-channel coding framework for transmitting 3D point cloud data to different quality-of-service devices by creating a scalable representation of point cloud. We employ a scalable Binary Tree embedded Quad Tree point cloud encoder with adaptive modulation and coding schemes to guarantee the latency as well as the quality requirement of each user. We perform link level simulations using outdoor 3D point cloud dataset from LiDAR scans for auto-driving. The scalability of our encoded point cloud provides a trade-off in the received voxel size/quality vs channel condition under a hard latency constraint. The users with good channel conditions receive a near lossless point cloud whereas users with bad channel conditions are still able to receive at least the base layer point cloud.

Index Terms— Point Cloud, Auto-driving, JSCC, Source Encoding, Channel Coding

1. INTRODUCTION

Point Cloud is a 3D data representation that is becoming increasingly popular due to the advent of various depth scanning sensors like LiDAR. For robots and smart devices to work in an unstructured environment, they need to perceive the world. In the future, we can expect most robots and auto-driving cars to be able to "see" the world [1] by receiving real-time point cloud data. With the advances in 5G New Radio technology [2] and the introduction of edge computing [3], point cloud communication for autonomous vehicles has become feasible. The quality of point cloud streaming is affected by both the source-coding accuracy as well as the amount of redundancy introduced by forward error correction (FEC) channel coding to protect the compressed point cloud over the channel. Joint Source Channel Coding (JSCC) is well-studied for 2-D videos but not much work is done on it in 3D point clouds. Transmitting point cloud is challenging since point cloud generates a large amount of data that creates a bottleneck in real-time low latency point cloud streaming. Being able to broadcast point cloud is particularly useful in applications like auto-driving where the vehicles would be able to "see" an area even if it's not in the line-of-sight of the car and hence be able to make more "intelligent" decisions. An infrastructure based point cloud provides a better field of view and angles compared to a vehicle-based sensor.

In [4] joint source-channel coding for video is proposed where an unequal error protection method is proposed for video plus depth data over WiMAX communication channels based on unequal power allocation. [5] considers the joint source-channel coding in the delivery of a 3D video with depth maps by using unequal error protection (UEP) at the packet level. In [6], the authors propose a joint source-channel coding and optimization for layered video broadcasting to heterogeneous devices. For point cloud encoding, Octree is widely used in the literature to compress the static point cloud or to intra-code the frame in a dynamic point cloud. Local surface estimation with predictive coding is combined with Octree in [7] and [8] to compress the point cloud and estimate the child cell configurations. The works that encode point clouds using the difference between the two frames are called dynamic point cloud compression techniques. One such technique, [9] encodes the structural difference between the octree structure of frames. Similarly, [10] creates a fixed size block in a voxelized point cloud and computes the motion associated with these blocks in the next frame. Low latency scalable point cloud communication using adaptive random network coding (ARNC) has been proposed in [11].

To the best of our knowledge, this is the first work to provide a scalable source coding with adaptive FEC to deliver real-time 3D point cloud. In this paper, we present a framework for point cloud transmission with joint source-channel coding. We adopt a Binary Tree embedded Quad Tree (BTQT) based point cloud source encoder [12] to add elasticity to the source rate and QoS. Convolutional codes are used as our channel codes. We create a scalable source-channel encoded point cloud representation that uses adaptive Modulation and Coding Schemes (MCS) as well as BTQT encoding to guarantee the latency as well as quality requirements of each user. We carry out exhaustive link level simulations using the dataset from MPEG PCC [13] called the Mitsubishi Electronics Research Lab (MERL) dataset for autonomous driving. The results of the link level simulations can be used in unicast, multicast, as well as broadcast networks for a system level simulations. The simulation results show the scal-

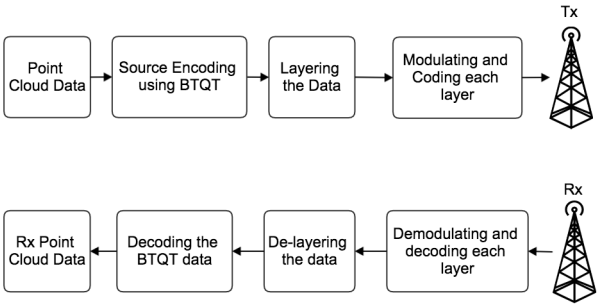
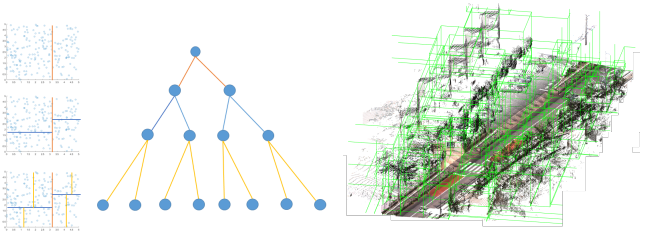
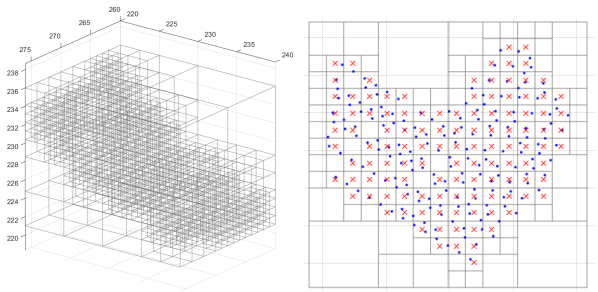


Fig. 1: System Model.



(a) Binary-tree depiction in 2D plane. (b) Block formation using binary tree for MERL dataset.

Fig. 2: Binary Tree.



(a) Octree (b) Quadtree

Fig. 3: Octree and Quadtree visualization.

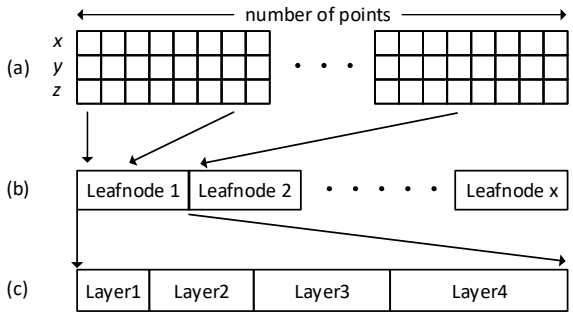


Fig. 4: Bitstream of the encoded BTQT point cloud.

ability of our encoded point cloud provides a trade-off in the received voxel size/quality vs channel condition vs latency.

2. SYSTEM MODEL AND FRAMEWORK

We first encode the point cloud using the Binary Tree embedded Quad Tree (BTQT) encoder. We convert the BTQT encoded data into layers with each layer refining the point cloud by increasing its quality. We use different Modulation and Coding Schemes (MCS) to encode each of the layers of the BTQT encoded point cloud and use convolutional channel coding with code rates of $1/2, 2/3,$ & $3/4$. The system model is shown in Fig. 1.

BTQT characteristics and the working of the binary tree and quad tree together is described in detail in [12]. A voxel is the volume element, defined in 3D space. We try to obtain the smallest voxel size representation of the reconstructed point cloud after BTQT encoding. With BTQT we create multiple representations of the point cloud data. The lowest representation is the base layer and each higher layer is an enhancement layer. Each layer has dependencies across the layers before it, therefore a user would need consecutive layers in order to decode the final layer. We use a two-step approach to encode the point cloud using BTQT. *Step 1:* First we encode the point cloud using binary tree which divides the point cloud into smaller blocks where each block is called a leafnode. In each iteration, Binary Tree (BT) divides one of the dimension ($x, y,$ or z) that has the most variance through its median. The example of how a 2D plane Binary-tree looks like is shown in Fig. 2a. Fig. 2b shows one frame of the MERL dataset divided into leafnodes using Binary-tree. *Step 2:* Once the point cloud is divided into sub-frames using BT, we encode the points in each of the leafnode. If the points in a leafnode exhibit flat surface characteristics and can be projected onto a 2D plane, then they are encoded by Quadtree (QT), otherwise, the points are encoded in the 3D space using Octree (OT). Example of an OT and QT are shown in Fig. 3. Like BT, Octree also works by recursively dividing the 3D block into eight sub-blocks by dividing each dimension into two halves.

The coding process and the final bitstream are shown in Fig. 4. Fig. 4(a) shows the unordered and uncoded point cloud data. We convert it into different leafnodes through BT encoding and each leafnode has multiple layers due to OT/QT encoding. Each layer here is a different quality representation of point cloud where the deeper layers are considerably larger in size compared to the upper layers. In our system, each receiver experiences different channel fading that is dependent on the location, distance between transmitter and receiver, the relative speed of the transmitter and receiver, and the multipath environment. Since we are transmitting multilayer encoded point cloud, our goal is to make sure that the

MCS #	Coding Rate	Modulation Scheme
MCS1	1/2	BPSK
MCS2	1/2	4QAM
MCS3	2/3	4QAM
MCS4	3/4	4QAM
MCS5	1/2	16QAM
MCS6	2/3	16QAM
MCS7	3/4	16QAM
MCS8	1/2	64QAM
MCS9	2/3	64QAM
MCS10	3/4	64QAM

Table 1: Modulation and Coding Schemes

	BT Depth = 10		BT Depth = 12	
	15 FPS	30 FPS	15 FPS	30 FPS
Layer1	0.1	0.2	0.4	0.8
Layer2	0.62	1.24	2.3	4.6
Layer3	2.7	5.4	9.7	19.4
Layer4	9.8	19.6	29.7	59.4

Table 2: Avg. Bitrate (Mbps) of Source Encoding

good channel quality users are able to decode a lot of layers and get the best quality whereas the poor channel quality users are still able to decode some layers and still get a considerable quality of pointcloud. This can be achieved by carefully selecting the number of layers to transmit and adaptively modulating and coding each layer to maximize the network utility.

Layered data is inherently more sensitive to transmission loss, as decoding has dependency across layers. Transmission loss in previous layers affects the decoding of enhancement layers. We use a combination of different coding rates and modulation schemes to create different MCS shown in Table 1. MCS schemes that obtain higher transmission rates also tend to have higher BER. Ideally, we need to use the MCS scheme that gives the highest transmission rate while satisfying the minimum BER requirement for the specific channel SINR value. However, we need to increase the overall utility of our network while making sure that the minimum requirements for each user are satisfied.

3. SIMULATION RESULTS

We use the dataset from MPEG PCC [13] called the MERL dataset provided by Mitsubishi. The dataset provides a typical roadside environment for applications such as auto-driving vehicles and smart devices in the traffic environment. Our goal is not to obtain a system level simulations but to show the effect of different MCS schemes and the number of layers on the quality of the received point cloud for a particular

Mode#	Layer1	Layer2	Layer3	Layer4
Mode1	MCS1	MCS2	MCS3	MCS4
Mode2	MCS4	MCS5	MCS6	MCS7
Mode3	MCS7	MCS8	MCS9	MCS10
Mode4	MCS1	MCS3	MCS5	MCS7
Mode5	MCS1	MCS4	MCS7	MCS10

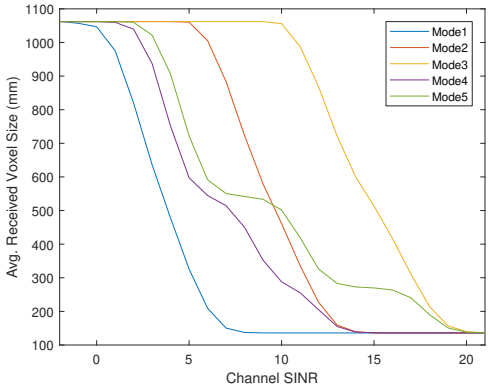
Table 3: Different Modes

		BW = 10 MHz	BW = 20 MHz
Binary Tree Depth = 10	Mode1	83	42
	Mode2	41	21
	Mode3	27	14
Binary Tree Depth = 12	Mode1	269	134
	Mode2	133	67
	Mode3	88	44

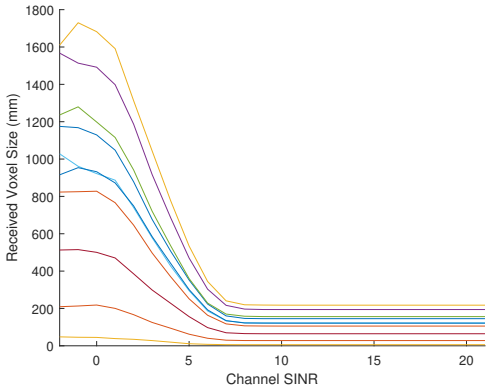
Table 4: Latency per frame (ms)

user. This section only deals with the link level simulation for a single transmitter-single receiver link. Therefore, we perform an exhaustive link level simulations with different channel conditions for a receiver using an additive white gaussian noise channel model. The results of the link level simulations can be used in a system level simulation to model a network level unicast, multicast, or a broadcast system. We show the feasibility of our work and the scalability of our source encoding in transmission. We use Binary tree depth ($B = 12$) and Quad tree depth ($L = 4$) for this particular dataset since it achieved good compression while maintaining fine quality. The average bitrate we achieve from two different settings of the source encoding ($B = 10$ and $B = 12$) and frame rate of 15 FPS and 30 FPS are shown in Table 2. The four layers of the encoded point cloud give us different levels of quality and show that we achieve a considerable amount of compression using our BTQT encoding.

The source encoded layered data is then channel coded using different MCS. For simulations, we only employ five different modes shown in Table 3. The average received voxel size at the receiver is shown in Fig. 5a for MERL dataset. We can observe that the quality of the received point cloud increases with an increase in the Channel SINR. We can notice that for a fair channel condition (5-15 dB), the received voxel size is considerably good and could be used for decision making in applications like autonomous driving. The modes using a lower MCS scheme tends to be able to operate on a lower SINR whereas the modes using higher MCS schemes require a better channel quality. However, as shown in Table 4, the modes using higher MCS have lower latency compared to modes using lower MCS. The latency is calculated assuming that we are using a Nyquist pulse shaping where for prac-

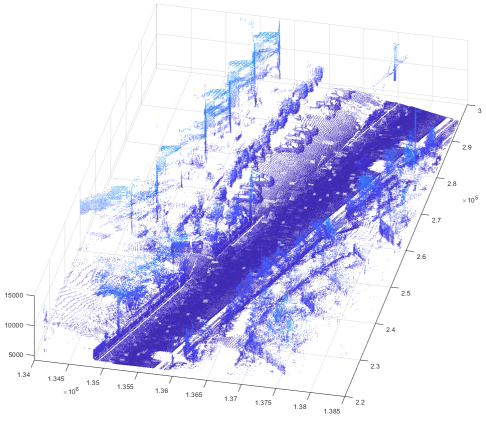


(a) Average received voxel size for different Modes in different channel conditions.

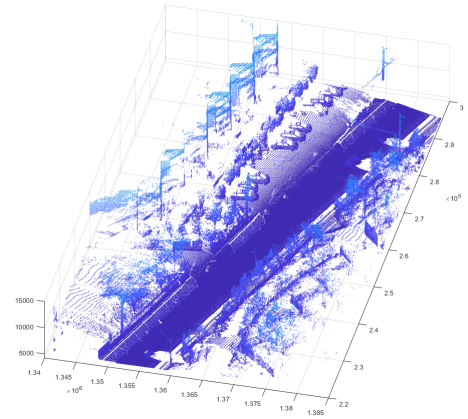


(b) Received voxel size of different frames from MERL dataset in different channel conditions using Mode1.

Fig. 5: Results.



(a) Channel SINR = 5



(b) Channel SINR = 14

Fig. 6: Received Quality for two users using Mode1.

tical purposes the bandwidth is 25% larger than the baud rate. If the bandwidth is low and the mode used to transmit is also low, then the users would face high latency.

Fig. 5b shows the received voxel size for 10 different frames from the MERL dataset for Mode1. We can observe that the difference in the environment changes the geometry of the point cloud and has an effect on the voxel size of that frame. However, all the encoded point cloud frames follow the same pattern of avg. received voxel size vs channel SINR. This shows that point cloud data is a bursty communication and very much dependent on the environment/pointcloud that is being transmitted. Some outdoor environment has a very dense point cloud whereas other environments have sparse point clouds. Depending on the need of the user, the transmitted point cloud could have a very large size or could be considerably smaller in size. Fig. 6 shows a single frame of the received point cloud encoded with Mode1 for two different users with different channel conditions. One user has a channel SINR of 6 whereas the other user has a channel SINR of 14. The user with channel SINR of 14 receives a near lossless reception of the point cloud whereas the user with channel SINR of 6 receives a very lossy reception of point cloud. However, since the point cloud is in layered form, this user is still able to decode the upper layers of encoded source data and still able to get a considerable good representation of the environment.

4. CONCLUSION

In this paper, we propose a framework for scalable 3D point cloud communication. We develop a Binary Tree embedded Quad Tree (BTQT) source encoder that gives us a scalable bitstream representation of the geometry of the point cloud data. We convert the source encoded point cloud into a layered structure where the deeper layers give us a finer representation of the point cloud. Since the layered data is sensitive to transmission losses, we use convolutional codes as our FEC codes to minimize the transmission loss due to channel conditions. We carry out exhaustive link level simulations with MERL point cloud dataset from MPEG PCC. We show the scalability of our framework and how different channel conditions and parameters behave and impact the point cloud streaming system.

5. REFERENCES

[1] Radu Bogdan Rusu and Steve Cousins, "3D is here: Point Cloud Library (PCL)," in *Robotics and automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[2] Salah Eddine Elayoubi, Mikael Fallgren, Panagiotis Spapis, Gerd Zimmermann, David Martín-Sacristán, Changqing Yang, Sébastien Jeux, Patrick Agyapong, Luis Campoy, Yinan Qi, et al., "5G service requirements and operational use cases: Analysis and METIS II vision," in *Networks and Communications (EuCNC), 2016 European Conference on*. IEEE, 2016, pp. 158–162.

[3] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[4] Chaminda TER Hewage, Zaheer Ahmad, Stewart T Worrall, Safak Dogan, Warnakulasuriya Anil Chandana Fernando, and A Kondoz, "Unequal error protection for backward compatible 3-D video transmission over WiMAX," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 125–128.

[5] A. Vosoughi, P. C. Cosman, and L. B. Milstein, "Joint Source-Channel Coding and Unequal Error Protection for Video Plus Depth," *IEEE Signal Processing Letters*, vol. 22, no. 1, pp. 31–34, Jan 2015.

[6] Wen Ji, Zhu Li, and Yiqiang Chen, "Joint source-channel coding and optimization for layered video broadcasting to heterogeneous devices," *IEEE Transactions on Multimedia*, vol. 14, no. 2, pp. 443–455, 2012.

[7] Rufael Mekuria, Kees Blom, and Pablo Cesar, "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, 2017.

[8] Ruwen Schnabel and Reinhard Klein, "Octree-based Point-Cloud Compression.," *Spbg*, vol. 6, pp. 111–120, 2006.

[9] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach, "Real-time compression of point cloud streams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 778–785.

[10] Ricardo L de Queiroz and Philip A Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3886–3895, 2017.

[11] Anique Akhtar, Junchao Ma, Rubayet Shafin, Jianan Bai, Lianjun Li, Zhu Li, and Lingjia Liu, "Low Latency Scalable Point Cloud Communication in VANETs using V2I Communication," in *2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019.

[12] Birendra Kathariya, Li Li, Zhu Li, Jose R. Alvarez, and Jianle Chen, "Scalable point cloud geometry coding with binary tree embedded quadtree," in *Multimedia and Expo (ICME), 2018 IEEE International Conference on*. IEEE, 2018.

[13] "MPEG PCC Datasets," http://mpegfs.int-evry.fr/MPEG/PCC/DataSets/pointCloud/CfP/datasets/.