Estudo de Perdas para Codificação de Geometria de Nuvens de Pontos por Decomposição Diádica

Davi R. Freitas, Eduardo Peixoto, Ricardo de Queiroz e Edil Medeiros

***Resumo*— Este artigo propõe um estudo sobre a introdução de perdas em um codificador intra-frame de geometria de nuvens de pontos (PC) voxelizadas. Usando uma abordagem de representação alternativa, esse método de codificação representa a PC como um arranjo de imagens binárias. Além disso, procurou-se também aferir a qualidade da PC decodificada com diferentes técnicas de reconstrução para uma determinada taxa de bits. Verificou-se que apenas a utilização do método de perdas proposto não permite alcançar taxas de bits abaixo de 0,2 bpov, o que sugere a combinação desse método com outra soluções de introdução de perdas em estudos futuros.**

***Palavras-Chave*— Nuvens de pontos, codificação com perdas, compressão de geometria**

***Abstract*— This paper proposes a study about the introduction of lossy techniques over an intra-frame coder of the geometry information of voxelized point clouds (PC). Using an alternative representation approach, this method represents the PC as an array of binary images. Moreover, we sought to assess the quality of the decoded PC with different reconstruction techniques for a given bit rate. We observed that the use of the lossy proposed method alone wasn't enough to reach bitrates values below 0.2 bpov, which suggests the combination of this method with other lossy solutions in future studies.**

***Keywords*— Point clouds, lossy coding, geometry compression**

I. Introdução

Nuvens de pontos (PC) são estruturas de dados importantes no contexto de aplicações de realidade virtual e aumentada, uma vez que proveem um meio de representação de objetos de geometria complexa de uma maneira flexível. Uma PC pode ser definida como um conjunto de pontos no espaço 3D, representado por coordenadas (x, y, z), e atributos opcionais - geralmente cor. Apesar do volume de dados de uma nuvem de pontos se apresentar menor do que aquele observado em malhas poligonais [1], ele ainda é significante e, por conseguinte, estimula o desenvolvimento de algoritmos de compressão eficientes. Enquanto existem trabalhos envolvendo a compressão de nuvens de pontos não-voxelizadas (contínuas) [2], [3], [4], o escopo desse trabalho concentra-se em PCs voxelizadas.

Um grande número de *codecs* usam a representação por *octree* [5] como um meio de estruturar nuvens de pontos e explorá-las para comprimir seus dados, tanto para codificação intra [6] e inter-frames [7]. Métodos de compressão sem perdas atingem reduções de taxas de bits significativas, mas economias adicionais podem ser alcançadas pela introdução de mecanismos de perda no esquema de compressão. Kammerl et al. [8] propõem uma abordagem de geometria com perdas baseada em um esquema de *buffering* duplo para codificação inter-frame. Mekuria et al. [9] abordam o problema de compressão de geometria com perdas pela construção da *octree* até um certo nível de detalhe. Oliveira et al. [10] definem a representação por *octree* como uma camada base e uma transformada baseada em grafos como uma camada de refinamento para alcançar uma compressão com perdas. Quach et al. [11] propõem um método de codificação de geometria com perdas baseado em um autoencoder convolucional.

Pode-se notar que representar nuvens de pontos por *octrees* é uma abordagem popular. Todavia, soluções usando métodos diferentes de representação também tem sido propostas. Daribo et al. [12] modelam a PC como uma curva de espaço 3D e comprime com perdas a geometria por meio de codificação aritmética. Zhu et al. [13] propõem um esquema de codificação intra-frame de geometria sem perdas pelo uso de um particionamento por árvores binárias. Milani et al. [14] introduzem um algoritmo intra-frame com perdas para geometria de nuvens de pontos baseado em uma transformada reversível de blocos por autômato celular. Peixoto [15] representa uma PC como uma sequência de imagens de silhueta em um cubo 3D e as codifica usando decomposição diádica em um algoritmo sem perdas.

O MPEG G-PCC *TMC13* v7.0 [16] é um *codec* baseado em geometria que originou das atuais propostas pelo grupo MPEG. Em adição à compressão de geometria sem perdas, o codificador *TMC13* possui dois métodos para comprimir a geometria com perdas [17]. O primeiro é uma quantização direta da geometria, a qual será referida nesse trabalho como *TMC13 Octree*. Essa técnica funciona pela quantização das diferenças entre cada ponto da nuvem de pontos e as coordenadas mínimas ao longo de seus três eixos. O outro é a aproximação por superfícies triangulares, referida aqui como *TMC13 Trisoup*. Essa técnica é implementada pela representação de cada voxel como uma superfície que intersecta cada uma de suas arestas. No lado do decodificador, esse método constrói triângulos a partir de um conjunto de vértices, os quais são caracterizados pelas interseções entre a superfície e o voxel, e os voxels perdidos são estimados pela extração de vértices refinados provindos dos triângulos construídos.

Embora o algoritmo sem perdas proposto por Peixoto [15] apresente um desempenho superior ao modo sem perdas do *TMC13*, ele não possui uma técnica de compressão com perdas. Com base nesse trabalho anterior, procurou-se realizar um estudo a fim de investigar o comportamento desse codificador com a inserção de um método que introduz perdas na geometria, em uma tentativa de alcançar reduções ainda maiores

Davi R. Freitas, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília-DF, email: rabbouni.davi@image.unb.br; Eduardo Peixoto, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília-DF, email: eduardopeixoto@ieee.org; Ricardo de Queiroz, Departamento de Ciência da Computação, Universidade de Brasília, Brasília-DF, email: queiroz@ieee.org; Edil Medeiros, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília-DF, email: j.edil@ene.unb.br. Este trabalho foi parcialmente financiado por CNPq (301647/2018-6).

nas taxas de bits transmitidos. Além disso, esse método de introdução de perdas foi implementado com diferentes técnicas de reconstrução com o objetivo de aferir seus efeitos sobre a qualidade da PC para uma dada taxa de bits.

II. Decomposição Diádica

Considere uma nuvem de pontos voxelizada de dimensões $N \times N \times N$ representada como um arranjo (do inglês, *array*) de ocupação 3D booleano $\mathbf{G}(x, y, z)$. A ideia central do método de decomposição diádica [15] é de fatiar o arranjo 3D $\mathbf{G}$ recursivamente ao longo de um eixo escolhido. Em cada iteração, uma imagem de silhueta $N \times N$ é obtida pela projeção no eixo escolhido de todos os voxels ocupados na região sendo processada. As imagens são então comprimidas por um codificador aritmético.

A natureza recursiva do algoritmo funciona pela divisão de um intervalo de fatias em dois intervalos menores de mesmo tamanho, implicando, assim, um percurso em árvore binária. Cada nó da árvore binária diádica possui uma silhueta para toda a região 3D que ela representa. A lógica por trás dessa decomposição diádica é de que, a partir da imagem de silhueta para um nó arbitrário, pode-se inferir um grande número de voxels não ocupados: todos os pixels em branco na imagem de silhueta são vazios pois estavam vazios em todas as fatias ao longo daquele eixo, o que significa que todas as sub-árvores daquele nó vão apresentar pelo menos aqueles mesmo voxels não ocupados. O algoritmo usufrui das similaridades entre a silhueta do nó atual e a do seu pai para omitir informações que podem ser inferidas no decodificador. Ele também usa informações dos nós vizinhos para construir contextos para um codificador aritmético inspirado em JBIG [18]. O processo de fatiamento continua até que cada folha da árvore binária tenha um intervalo de fatia atômico.

Experimentos demonstram que fatias próximas das folhas da árvore binária diádica não são comprimidas com a mesma eficiência que fatias próximas da raíz. Por conseguinte, além do método de decomposição diádica, uma outra abordagem chamada codificação em *modo único* também foi proposta no trabalho anterior. O algoritmo adiciona um parâmetro $P \in [1, N]$ e prossegue com a decomposição diádica até que a região de processamento compreenda um volume de $k <= P$ fatias. A partir desse ponto, a codificação em *modo único* "pula" o fatiamento diádico e transmite todas as k fatias restantes como folhas da árvore, usando a silhueta correspondente a esse intervalo como máscara para codificação de cada folha. Para a implementação em [15], os dois métodos de codificação são testados e o método que produz a taxa de bits mais baixa é escolhida para compor o fluxo de bits (do inglês, *bitstream*) de saída.

A. Perda de informação para melhoria da compressão

Neste artigo, propõe-se a estudar os efeitos da introdução de mecanismos de perda no codificador de decomposição diádica para alcançar taxas de bits mais baixas sem apresentar uma degradação significativa da geometria. Somente a codificação em *modo único* é realizada usando um valor fixo de $P = 64$ fatias. Ou seja, procede-se com a decomposição diádica com uma codificação sem perdas das silhuetas até que, e incluindo, profundidade $\log_2 N - 5$. O algoritmo então interrompe o fatiamento diádico da árvore e pula para as imagens de folha, as quais serão comprimidas com perdas. A Fig. 1 apresenta uma representação esquemática dessa construção. A escolha do parâmetro $P = 64$ foi determinada empiricamente entre um conjunto específico de valores com base em resultados de taxa e distorção e uma maior afinação desse parâmetro é considerada como trabalho futuro.

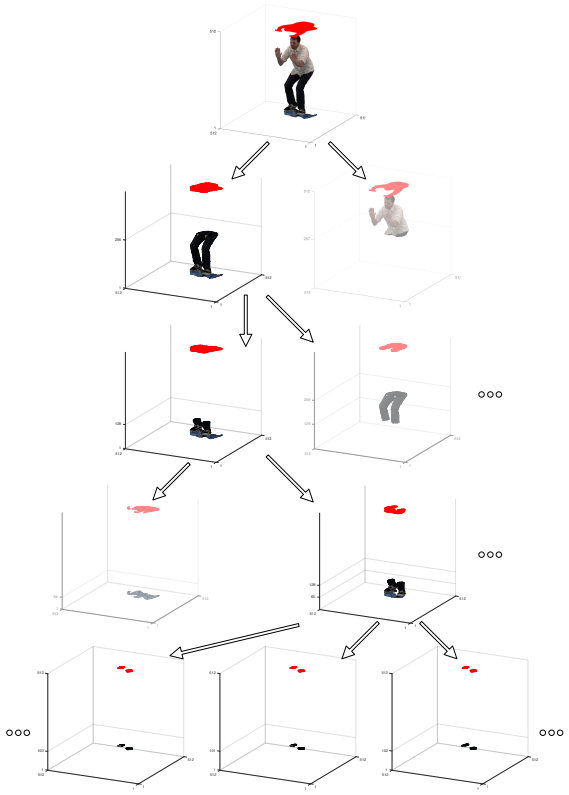
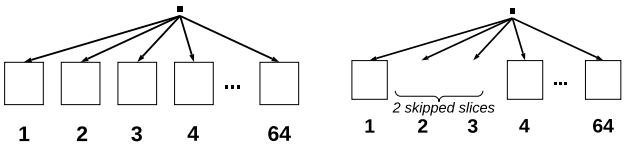


Fig. 1: Diagrama demonstrando a decomposição diádica e a codificação fixa em modo único com P = 64.



(a) (b)

Fig. 2: Codificação em modo único: (a) sem perdas; (b) com perdas (*step* = 3)

A fim de comprimir a geometria na codificação em modo único, um método de introdução de perdas foi implementado. Para isso, ao invés de se transmitir todas as imagens (folhas) em um dado intervalo, procurou-se pular um certo número de fatias consecutivas, as quais são interpoladas no lado do decodificador. Esse método é ilustrado na Fig. 2. Para o resto desse artigo, essa técnica será referida simplesmente como *step*, onde um valor de *step* de k significa que $k - 1$ fatias sucessivas são puladas.

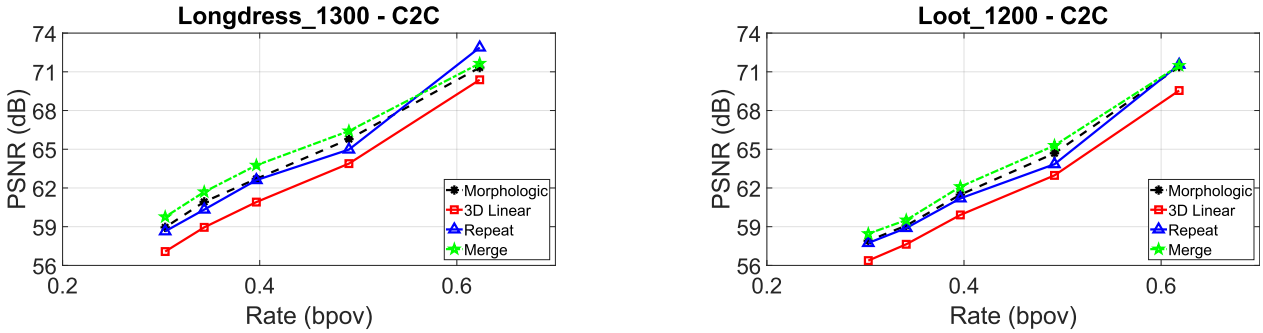


Fig. 3: Comparação em RD dos métodos de reconstrução propostos para o a) Frame 1300 de Longdress; b) Frame 1200 de Loot

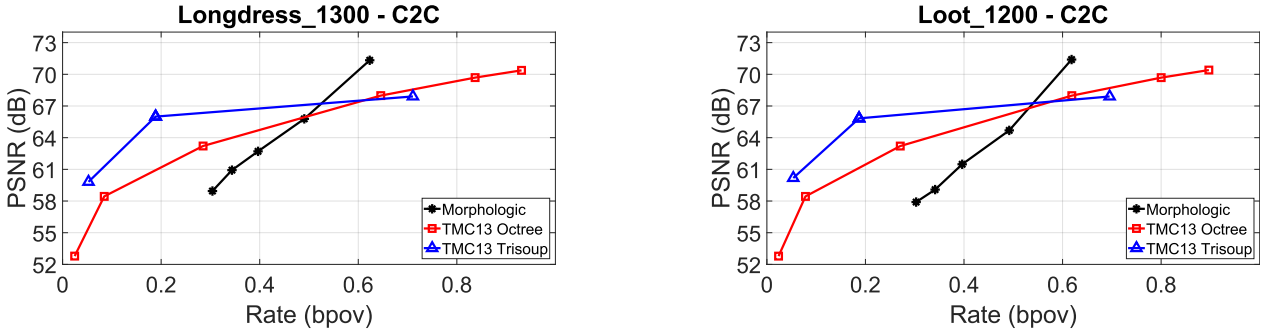


Fig. 4: Comparação em RD dos métodos de reconstrução morfológico com os métodos de perda do TMC13 v7.0 para: a) Frame 1300 de Longdress; b) Frame 1200 de Loot

B. Reconstrução para a minimização de perdas

Para a técnica *step* de introdução de perdas, foram analisados quatro diferentes algoritmos de reconstrução das imagens descartadas para reduzir o custo da transmissão. Dessa forma, permite-se efetuar uma análise comparativa dos algoritmos com base no nível de distorção gerado por cada um, uma vez que a taxa de bits transmitida é a mesma para todos os casos.

1) Interpolação Linear: baseado na aplicação de uma interpolação linear tridimensional para calcular os valores de cada ponto de interesse.

2) Algoritmo morfológico: um algoritmo de estimação de movimento entre os frames que não foram pulados (referidos aqui como frames *anterior* e *posterior*) é executado para encontrar o melhor ajuste entre eles - referido aqui como t. Uma vez que esse valor t é encontrado, as fatias intermediárias são geradas pela translação do frame antecedente por frações de t. Como exemplo, suponha que o melhor valor de ajuste entre dois frames é $t = 1$. Para um valor de *step* de 4, os três frames intermediários são gerados pela translação do frame antecedente por valores de $\frac{1}{4}$, $\frac{2}{4}$ e $\frac{3}{4}$. Além disso, operações de fechamento morfológico e o uso de Y_S como máscara são aplicados para enriquecer ainda mais as fatias interpoladas.

3) Merge: o método de reconstrução *merge* consiste em substituir as imagens descartadas pela operação binária de *OU* entre os frames *anterior* e *posterior.*

4) Repetição: consiste na substituição do frames descartado pela imagem transmitida "mais próxima" a ele. Como exemplo, considere um caso em que o número de imagens consecutivas descartadas é igual a 5 (valor de *step* de 6). Esses 5 frames seriam substituídos na reconstrução por: *Anterior*, *Anterior*, *Posterior*, *Posterior*, *Posterior*.

III. Resultados Experimentais

Em 2017, o *Moving Picture Experts Group* (MPEG) emitiu um convite à apresentação de propostas [19], no qual uma avaliação de desempenho objetiva para soluções de compressão de PCs foi delineada. Essa avaliação é baseada em desempenho de taxa-distorção (do inglês, *rate-distortion*), no qual a taxa é reportada como bits por voxel ocupado (bpov) e a distorção pode ser medida em termos de relação sinal-ruído de pico (PSNR) por duas diferentes métricas [20]: o erro ponto a ponto (C2C) é obtido pelo cálculo do erro quadrático médio (MSE) entre os pontos reconstruídos e seus vizinhos mais próximo na nuvem de pontos de referência. A segunda métrica é o erro ponto a plano (C2P), o qual é computado tendo em conta os planos de superfície ao invés dos vizinhos mais próximos [21].

Os experimentos foram realizados na base de dados pública JPEG Pleno [22], em particular na base 8i Voxelized Full Bodies [23]. Essa consiste de 4 sequências com resoluções de 10 bits: *LongDress*, *Loot*, *RedAndBlack* e *Soldier*.

Para a avaliação de taxa-distorção das quatro técnicas de reconstrução propostas, foram utilizados cinco diferentes valores do método de *step*: $2, 3, 4, 5$ e 6, correspondendo a 1 a 5 imagens consecutivas descartadas, respectivamente.

A. Avaliação de RD para os métodos de reconstrução

As curvas de taxa-distorção dos quatro métodos de reconstrução foram avaliadas quanto ao erro C2C. A Fig. 3 apresenta os resultados obtidos para o frame 1300 da sequência

Fig. 5: Comparação da vista frontal para o frame 1550 da sequência RedAndBlack para valores *step* (acima) de a) 1 (PC original) b) 2 c) 3 d) 4 e) 5 f) 6 e para as taxas do TMC13 (abaixo) g) 5 h) 4 i) 3 j) 2 k) 1

Sequência	*C2C*		
	Morfológico	**Merge**	**Repetição**
LongDress	1,76	2,50	1,50
Loot	1,64	2,15	1,21
RedAndBlack	1,98	2,49	1,43
Soldier	1,73	2,26	1,30
Average	1,78	2,35	1,36

TABELA I: Comparações em BD-PSNR (em dB) entre os quatro métodos de reconstrução propostos. Resultados do método de reconstrução por interpolação linear 3D foram usados como referência.

Sequência	*Porcentagem de pontos decodificados*			
	Morfológico	**Merge**	**Repetição**	**Interpolação Linear 3D**
LongDress	102	157	100	44
Loot	100	157	100	44
RedAndBlack	104	158	100	43
Soldier	100	158	100	42
Average	101	157	100	43

TABELA II: Comparações em termos de números de pontos decodificados (em % em relação ao número original de pontos) entre os quatro métodos de reconstrução propostos.

Longdress e para o frame 1200 da sequência *Loot*. A Tab. I apresenta os resultados em termos de valores BD-PSNR [24] das quatro curvas para as quatro sequências testadas, tomando como âncora o algoritmo de reconstrução por interpolação linear tridimensional.

Pode-se observar que o algoritmo de *merge* apresentou um resultado superior que os outros três métodos, com uma média de $2,35$dB acima de método por interpolação linear e $0,57$dB acima do algoritmo morfológico. Todavia, como a métrica C2C é dependente do MSE, seu valor é sensível à quantidade de pontos na nuvem de pontos. Dessa forma, uma PC com uma quantidade significativamente maior de pontos pode apresentar um valor de PSNR maior mesmo se possuir um erro absoluto - e, portanto, uma distorção - maior.

Por conseguinte, a Tab. II apresenta a porcentagem de pontos decodificados para cada método de reconstrução. Esse valor foi calculado pela razão entre a média de pontos geradas pelas 5 taxas de compressão testadas e a quantidade de pontos original da sequência. Pode-se perceber para o método de *merge* um aumento médio de 57% do número de pontos, o que justifica a ocorrência dos seus valores de PSNR mais altos que o dos outros métodos. Por outro lado, o método de interpolação linear tridimensional apresentou uma redução

média de 57% em relação ao número de pontos original. Enquanto um número significativamente menor de pontos não é desejável pela diminuição da densidade da PC e pela perda significativa de informação, um número sensivelmente maior também não é desejável pela introdução de informações inexistentes e pelo aumento de complexidade da PC. Dessa forma, o método morfológico apresentou um melhor balanço entre o número de pontos decodificados e valores de PSNR.

B. Comparação com estado da arte

O método de construção morfológico foi avaliado com relação ao *codec TMC13* v7.0, a fim de comparar o comportamento do algoritmo proposto com soluções do estado da arte. A Fig. 4 apresenta os resultados de RD nos mesmos dois frames da Fig. 3 para a solução proposta com relação aos dois modos de compressão presentes no *TMC13*: *Trisoup* e *Octree*.

Com base nas Figs. 3 e 4, a primeira observação que pode ser feita se refere ao limite das taxas de transmissão: a introdução de perdas feita somente pelo descarte de imagens consecutivas não consegue alcançar taxas de bits tão baixas quanto às comparações do estado da arte. Além disso, para essas taxas que estão sendo trabalhadas, pode-se notar na Fig. 4 uma redução de qualidade mais considerável para a solução proposta em valores de *step* acima de 3.

C. Avaliação subjetiva dos resultados

A Fig. 5 apresenta a vista frontal do frame 1550 da sequência *RedAndBlack* para os valores de *step* de 1 (equivalente à PC original) a 6, usando como método de reconstrução o algoritmo morfológico. As cores foram interpoladas por uma média ponderada baseada na distância dos oito vizinhos mais próximos de cada ponto. Como foi observado na Seção III-B, pode-se notar uma degradação mais sensível a partir de uma valor de *step* de 3, com um aparecimento progressivo de buracos e efeitos de blocagem na estrutura. Comparando com as cinco taxas do TMC13 (parte inferior da Fig. 5), pode-se notar que os buracos na nuvem reconstruída aparecem em todas as taxas com perdas do TMC13.

IV. Conclusão

Este trabalho apresenta um estudo quanto à inserção de um método que introduz perdas na geometria de nuvens de pontos em um codificador com compressão intra sem perdas, o qual utiliza uma representação alternativa à popular estrutura por *octree*. Além disso, foram propostas para esse método de introdução de perdas diferentes técnicas de reconstrução para aferir seus efeitos sobre a distorção para uma dada taxa de bits. Verificou-se que somente a utilização do método proposto não é o suficiente para alcançar taxas de bits mais baixas - abaixo de 0,2 bpov. Além disso, notou-se tanto subjetivamente como pela comparação com codificadores com perdas do estado da arte que o método proposto apresenta a partir de certas taxas um aumento significativo da distorção para pequenas diminuições de taxas. Com isso, sugere-se a combinação do método proposto com outras soluções que introduzam perdas, como a aplicação de subamostragem nas imagens enviadas pela codificação em *modo único*, por exemplo.

Referências

[1] S. Gebhardt, E. Payzer, L. Salemann, A. Fettinger, E. Rotenberg, and C. Seher, "Polygons, point-clouds and voxels: A comparison of high-fidelity terrain representations," in *Simulation Interoperability Workshop and Special Workshop on Reuse of Environmental Data for Simulation—Processes, Standards, and Lessons Learned*, 2009, p. 9.

[2] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Continuous Point Cloud Data Compression Using SLAM Based Prediction," *2017 IEEE Intell. Vehicles Symp. (IV)*, , no. Iv, pp. 1744–1751, 2017.

[3] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Compressing Continuous Point Cloud Data Using Image Compression Methods," *2016 IEEE 19th Int. Conf. ITSC*, pp. 1712–1719, 2016.

[4] X. Sun, H. Ma, Y. Sun, and M. Liu, "A Novel Point Cloud Compression Algorithm Based on Clustering," *IEEE Robot. and Automat. Letters*, vol. 4, no. 2, pp. 2132–2139, 2019.

[5] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129 – 147, 1982.

[6] D. C. Garcia and R. L. de Queiroz, "Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry," in *2018 25th IEEE Int. Conf. on Image Process. (ICIP)*, Oct 2018, pp. 1807–1811.

[7] D. C. Garcia and R. L. de Queiroz, "Context-based octree coding for point-cloud video," in *2017 IEEE Int. Conf. on Image Process. (ICIP)*, Sep. 2017, pp. 1412–1416.

[8] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE Int. Conf. on Robot. and Automat.*, May 2012, pp. 778–785.

[9] R. Mekuria, K. Blom, and P. Cesar, "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," *IEEE Trans. on Circuits and Syst. for Video Technol.*, vol. 27, no. 4, pp. 828–842, April 2017.

[10] P. de Oliveira Rente, C. Brites, J. Ascenso, and F. Pereira, "Graph-Based Static 3D Point Clouds Geometry Coding," *IEEE Trans. on Multimedia*, vol. 21, no. 2, pp. 284–299, Feb 2019.

[11] M. Quach, G. Valenzise, and F. Dufaux, "Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression," *CoRR*, vol. abs/1903.08548, 2019.

[12] I. Daribo, R. Furukawa, R. Sagawa, and H. Kawasaki, "Adaptive arithmetic coding for point cloud compression," in *2012 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, Oct 2012, pp. 1–4.

[13] W. Zhu, Y. Xu, L. Li, and Z. Li, "Lossless point cloud geometry compression via binary tree partition and intra prediction," in *2017 IEEE 19th Int. Workshop on Multimedia Signal Process. (MMSP)*, Oct 2017, pp. 1–6.

[14] S. Milani, "Fast point cloud compression via reversible cellular automata block transform," in *2017 IEEE Int. Conf. on Image Process. (ICIP)*, Sep. 2017, pp. 4013–4017.

[15] E. Peixoto, "Intra-Frame Compression of Point Cloud Geometry using Dyadic Decomposition," *IEEE Signal Process. Letters*, pp. 1–1, 2020.

[16] 3DG, "G-PCC codec description v4," Tech. Rep., ISO/IEC JTC1/SC29/WG11/W18673, 2019.

[17] H. Liu, H. Yuan, Q. Liu, J. Hou, and J. Liu, "A Comprehensive Study and Comparison of Core Technologies for MPEG 3-D Point Cloud Compression," *IEEE Trans. on Broadcasting*, vol. PP, pp. 1–17, 2019.

[18] ISO/IEC JTC 1 / SC 29 /WG 1 (ITU–T SG8), "Coding of Still Pictures," Tech. Rep. N 1359, ISO/IEC, July 1999.

[19] 3DG, "Call for Proposals for Point Cloud Compression (v2)," Tech. Rep., ISO/IEC JTC1/SC29/WG11/N16763, Hobart, Australia, Apr. 2017.

[20] S. Schwarz, G. Cocher, D. Flynn, and M. Budagavi, "Common test conditions for point cloud compression," Tech. Rep., ISO/IEC JTC1/SC29/WG11/N17766 ISO/IEC JTC1/SC29/WG1 M72012, Jul. 2018.

[21] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE Int. Conf. on Image Process. (ICIP)*, Sep. 2017, pp. 3460–3464.

[22] "JPEG Pleno Database," `https://jpeg.org/plenodb/`, [Online].

[23] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, "Microsoft Voxelized Upper Bodies – A Voxelized Point Cloud Dataset," Tech. Rep., ISO/IEC JTC1/SC29/WG11 m38673 ISO/IEC JTC1/SC29/WG1 M72012, Geneva, Switzerland, Jan. 2017.

[24] Gisle Bjøntegaard, "Improvements of the BD-PSNR Model," Tech. Rep., VCEG-AI11, ITU-T SG16/Q6, Berlin, Germany, Jul. 2008.