# Representação Compressível para Codificação de Point Clouds Plenópticas

Gustavo Sandri, Ricardo de Queiroz e Philip A. Chou

***Resumo*— Point clouds assumem, tipicamente, uma cor por ponto, tornando-as menos realísticas para representar superfícies especulares. Neste trabalho tratamos da compressão de point clouds plenópticas onde é associado as cores tal como visto de diferentes ângulos. Propomos uma representação compressível para incorporar a informação plenóptica, que é então comprimida utilizando a region-adaptive hierarchical transform (RAHT). Até então, o melhor desempenho para a RAHT havia sido obtido combinando-a com um codificador aritmético. Entretanto, observamos que reordenando os coefficientes da RAHT podemos melhorar significativamente seu desempenho usando o codificador RLGR, gastando em média 0,2 e 0,09 bits por voxel ocupado a menos para um passo de quantização de 10 e 40, respectivamente, usando o codificador de entropia RLGR.**

***Palavras-Chave*— point cloud, plenóptica, compressão, realidade aumentada, realidade virtual.**

***Abstract*— Point clouds typically assume one color per point. This makes it less realistic with specular surfaces. In this work, we consider the compression of plenoptic point clouds, wherein is associated colors as seen by different angles. We propose a compressible representation to incorporate the plenoptic information, which is then compressed using the region-adaptive hierarchical transform (RAHT). However, we observed that by reordering the RAHT coefficients we can significantly improve its performance using the RLGR entropy encoder, spending on average 0.2 and 0.09 bits per voxel less for quantization steps of 10 and 40, respectivelly.**

***Keywords*— point cloud, plenoptic, compression, augmented reality, virtual reality.**

## I. Introdução

Uma *point cloud* (PC, em inglês) é uma forma de representar estruturas e cenas 3D por um conjunto de pontos disperso no espaço, cada um com uma cor associado a si, que, em conjunto, formam a superfície das estruturas representadas.

A *region-adaptive hierarchical transform* (RAHT, em inglês) foi introduzida recentemente por Queiroz e Chou [1] para comprimir a informação de cor e outros atributos de PCs. Ela é uma transformada hierárquica em sub-bandas que corresponde a uma variação adaptativa da *wavelet* Haar. A qualidade da RAHT é comparável a de codificadores baseados em *Graph Transform* [2] e *Gaussian Process Model* [3], mas com uma complexidade significativamente menor.

Um codificador de entropia é empregado para codificar os coeficientes da RAHT baseado em um codificador aritmético (AC, do inglês *arithmetic coder*). Os resultados reportados demonstram um desempenho comparável ao codificador baseado na *graph transform* [2], até então o estado-da-arte, mas com uma complexidade muito menor. O codificador *Adaptive run-length Golomb-Rice* (RLGR) [4], que é um codificador entrópico menos complexo, também foi testado com a RAHT.

As PCs são normalmente tratadas de forma discretizada, representadas por voxels, o equivalente 3D dos pixeis, ao invés de pontos, referidas como PC voxelizada. Enquanto que um ponto no espaço é adimensional, um voxel é um cubo de dimensão $1 \times 1 \times 1$. Em uma PC voxelizada, o espaço é definido como um cubo de dimensão $W \times W \times W$, $W$ um número natural, composto por $W^3$ voxels. Em uma PC voxelizada representando uma imagem 3D, a maioria dos voxels são transparentes pois, do contrário, só seríamos capaz de ver as faces do cubo que compõe o espaço. Dizemos que os voxels transparentes estão vazios e não possuem nenhuma cor associada a si. Os outros voxels são ditos ocupados e possuem um atributo de cor associado a si (em RGB ou YUV).

No momento de renderizar a cena, estes voxels atuam como fontes de luz, emitindo a mesma cor em todas as direções. A representação da cena por voxels de uma única cor pode não ser muito realista para superfícies especulares, onde a cor de uma dada posição varia de acordo com o ângulo de vista. O caso extremo seria um espelho, que reflete a sua redondeza. Uma representação mais realística deveria permitir que um voxel mude sua cor de acordo com o ângulo de vista. Para tanto, precisamos atribuir ao voxel a sua cor vista por uma pluralidade de direções. Nos referimos a estes dados como a informação plenóptica pois é baseado na função plenóptica que representa uma cena.

A função plenóptica de sete dimensões (7D) representa a intensidade da luz observada em cada posição, numa dada direção no espaço 3D, num dado intervalo de tempo, para um dado comprimento de onda [5]. Esta função pode ser representada de forma mais conveniente, reduzindo para cinco dimensões (5D), se considerarmos apenas um dado instante de tempo (ignorando a dependência temporal) e retornando a cromaticidade da luz (em RGB, por exemplo), dada por:

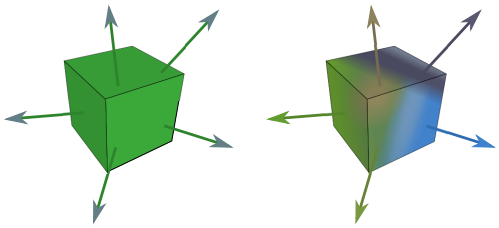
$$P(x, y, z, \theta, \phi), \tag{1}$$

onde $(x, y, z)$ são as coordenadas de um ponto no espaço, $\theta$ o ângulo de azimute e $\phi$ o ângulo de elevação. A informação plenóptica de um voxel é obtida fixando $(x, y, z)$ na posição do voxel e deixando $\theta$ e $\phi$ variar de acordo com o ângulo de vista.

PC plenópticas podem ser obtidas processando os dados capturados por um conjunto de câmeras combinadas com

Gustavo Sandri, Instituto Federal de Brasília, gustavo.sandri@ifb.edu.br
Ricardo de Queiroz, Departamento de Ciência da Computação, Universidade de Brasília, queiroz@ieee.org
Philip A. Chou, 8i, pachou@ieee.org

mapas de profundidade [6], [7], ou por câmeras *light-field* [8]. Desta forma, o número de direções amostradas é determinado pelo número de câmeras empregadas e a informação plenóptica é dada pelas cores tal como vistas por cada uma das câmeras. É mais prático codificar as cores de cada uma das câmeras ao invés da função plenóptica, que é contínua para os ângulos $(\theta, \phi)$. Portando, a informação plenóptica é um vetor de componentes de cor para cada voxel.



(a) Voxel não plenótico (b) Voxel plenótico

Fig. 1. Um voxel não plenótico não possui informação de cor direcional. Tal informação está presente em um voxel plenótico e pode ser usada para representar uma cena de forma mais realista.

Neste trabalho, propomos incorporar a informação plenóptica amostrada em cada voxel subdividindo o voxel em subvoxels. A posição dos subvoxels representa a disposição das câmeras. Assumimos que tanto o codificador quando o decodificador conhecem a geometria original da PC (quais voxels estão ocupados) e a disposição das câmeras (codificado com outro algoritmo) e focamos unicamente na compressão de cor. Este trabalho contrasta com outros métodos, como aqueles empregando *surface light field representation* [9], [10].

## II. Subdivisão do voxel

Considere o voxel na Fig. 2, para o qual as cores são capturadas por 5 câmeras posicionadas nas direções apresentadas.

A informação plenótica amostrada contem não somente a cor, mas também a direção das câmeras. Se dividirmos o voxel em $M$ partições ao longo de cada eixo ($M = 4$ na Fig. 3), obteremos $M^3$ cubos com uma largura $1/M$. Cada um destes cubos resultantes da divisão se assemelham a um voxel e nos referimos a eles como subvoxels.

Podemos utilizar a posição do subvoxel dentro do voxel para representar a direção de vista. Vamos nos restringir apenas aos

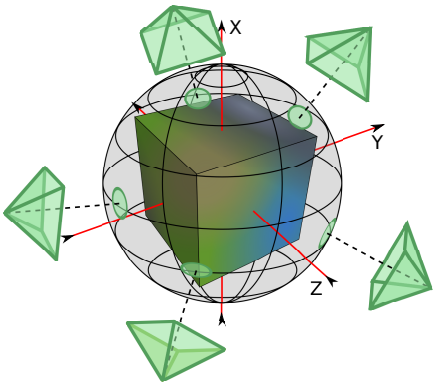



Fig. 2. Captura da informação plenóptica de um voxel.

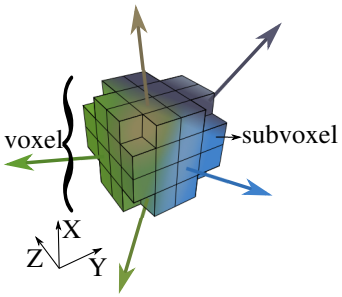



Fig. 3. Um voxel é dividido em subvoxels e seus suboxels são empregados para incorporar a informação plenótica.

subvoxels que são cortados pela superfície esférica tangente as faces do voxel (ver Fig. 3). A linha conectando o centro do voxel à câmera, simplesmente chamada de linha de vista, é então usado para representar a direção de vista. Esta linha cruza um destes subvoxels no caminho e a cor vista nesta direção é então associada a este suboxel (veja Fig. 3). Em outras palavras, a posição do centro do subvoxel relativo ao centro do voxel define a direção de vista. O processo é repetido para todas as outras linhas de vista e os subvoxels que não foram empregados para indicar a direção de nenhuma vista são marcados como desocupados.

Após atribuirmos a informação plenóptica aos subvoxels, podemos aplicar a RAHT na PC de subvoxels. Este processo é transparente para a RAHT pois ela trata os subvoxels como voxels.

## III. Codificador entrópico

Os coeficientes gerados pela RAHT são então quantizados antes de serem enviados ao codificador entrópico. A quantização que aplicamos consiste em dividir os coeficientes por um passo de quantização $Q$ e arredondar o valor resultante para o inteiro mais próximo. Esta etapa do processo introduz perdas, já que parte da informação é perdida pelo arredondamento, mas nos permite controlar a relação taxa em bits versus qualidade da imagem reconstruída através do passo de quantização. Passos de quantização grandes resultam em imagens codificadas com poucos bits, mas com uma baixa qualidade da imagem reconstruída, enquanto que passos de quantização pequenos resultam em muitos bits para as imagens codificadas, mas uma maior qualidade da imagem reconstruída.

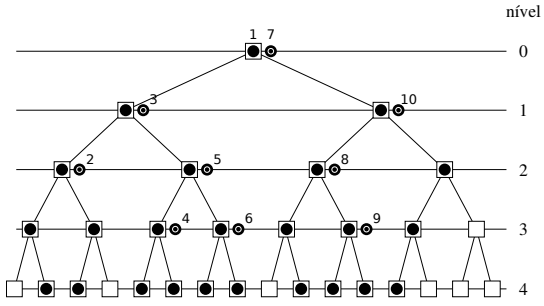
Os coeficientes transformados são então enviados para um codificador entrópico. No trabalho de Queiroz e Chou [1], o AC é empregado para codificar os coeficientes da RAHT (o codificador RLGR também foi testado com a RAHT). Os resultados apresentados mostraram que a codificação dos coeficientes da RAHT com RLGR (RAHT-RLGR) era substancialmente inferior à codificação com AC (RAHT-AC), entretanto com uma complexidade muito menor.

Queremos mostrar que, rearranjando os coeficientes da RAHT podemos melhorar o desempenho do codificador RLGR ao ponto de superar a RAHT-AC.

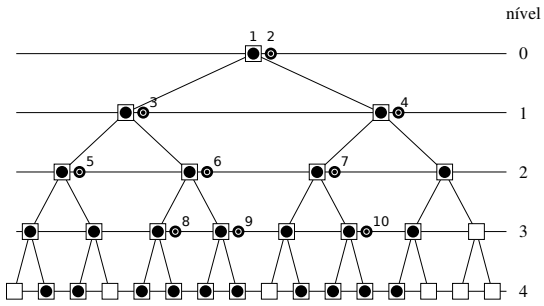
A RAHT é executada seguindo a octree de trás para frente, partindo de voxels individuais até todo o espaço de voxels,

em cada passo recombinando voxels gerando outros maiores até atingir a raiz (veja Fig. 4). Na Fig. 4, começamos no nível 4. Se o voxel vizinho no mesmo galho está ocupado, suas cores são combinadas através de uma transformação linear ([1], Eq. (6)) e promovidos ao nível 3. Se um voxel não possui um vizinho no mesmo galho, ele é diretamente promovido ao nível 3. Este processo é repetido em cada nível até atingir o nível 0, a raiz da octree. Na figura, uma vez que um par de voxels é combinado, é gerado um coeficiente passa-alta que está pronto para ser quantizado e codificado, e um coeficiente passa-baixa que é passado para o próximo nível da árvore. O decodificador necessita do coeficiente DC (raiz da árvore) e todos os coeficientes passa-alta gerados durante a transformada.

Existem várias formas de ordenar os coeficientes. Na implementação original [1], o coeficiente DC é enviado primeiro, como mostrado na Fig. 4. Em seguida, desce-se a árvore pelo caminho dos galhos mais a esquerda até atingir uma folha. Os galhos são então esquadrinhados um a um, sempre começando pelo galho a esquerda, e os coeficientes passa-alta são enviados na ordem que são encontrados, gerando um lista de coeficientes ordenados pela travessia.




(a) Travessia




(b) Profundidade

Fig. 4. Ilustração da transformada RAHT e ordenação dos coeficientes. Após a transformada, a ordem dos coeficientes DC e passa-alta são indicados para a ordenação por (a) travessia e por (b) profundidade.

Neste trabalho, propomos esquadrinhar os coeficientes passa-alta ordenados pela profundidade da árvore. A transformada RAHT produz coeficientes passa-baixa e alta, similar a transformada Haar. Assim, coeficientes gerados mais profundamente na árvore (a partir da raiz) representam frequências mais altas. Para imagens naturais, entretanto, esperamos que os componentes de alta frequência tenham uma amplitude menor do que os de menor frequência. Assim, a quantização dos coeficientes de alta frequência deve levá-los mais facilmente a atingir o valor zero quando comparados aos coeficientes de baixa frequência.

Os coeficientes da RAHT são então ordenados de forma crescente de acordo com sua profundidade associada (profundidade da árvore onde foram gerados) e codificamos entropicamente os coeficientes usando RLGR (veja Fig. 4).

A reordenação dos coeficientes proposta induz a geração de sequências de zeros mais longas. A Fig. 5 compara o *zero-run-length* médio para as 7 PCs testadas por Queiroz e Chou [1] em função do passo de quantização. Podemos observar facilmente que a reordenação pela profundidade produz sequências mais longas de zeros.

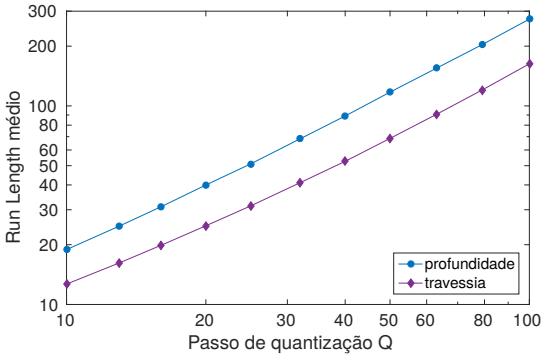



Fig. 5. *Zero-run-length* médio entre as 7 PCs testadas utilizando diferentes formas de ordenamento.

Além disso, o RLGR é um codificador adaptativo que continuamente atualiza seus parâmetros. A Fig. 6 compara os coeficientes no ordenamento original e o proposto neste trabalho para a PC "Phil", onde podemos esperar que o decaimento mais bem comportado dos coeficientes ordenados por profundidade devem levar a uma melhor adaptação dos parâmetros do RLGR ao longo da codificação e uma maior compressão.

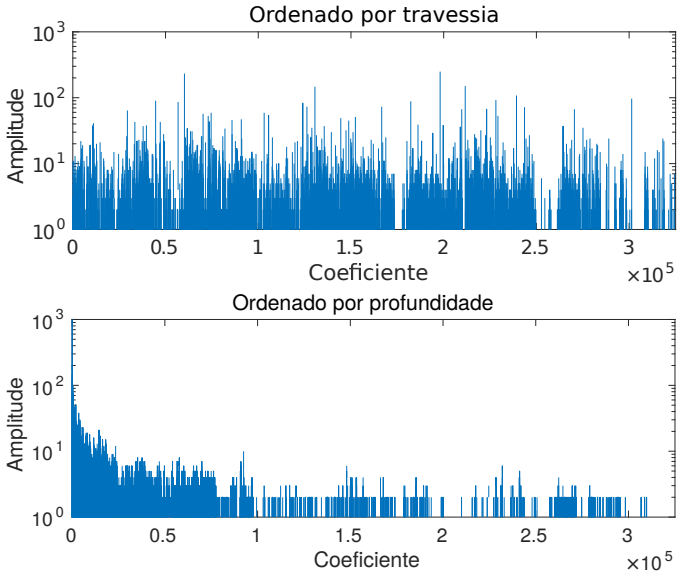



Fig. 6. Evolução da amplitude dos coeficientes para a PC "Phil" ($Q = 20$). Os coeficientes ordenados pela travessia possuem uma evolução que lembra um comportamento randômico, enquanto que os ordenados pela profundidade apresentam um comportamento de decaimento da amplitude.

TABELA I

COMPARAÇÃO DA TAXA DE COMPRESSÃO, EM $bits/voxel\ ocupado$, PARA OS COEFICIENTES DA RAHT ORDENADOS POR TRAVESSIA E PROFUNDIDADE E CODIFICADOS COM RLGR E CODIFICADOS COM AC, RESPECTIVAMENTE.

| | **Bits por voxel ocupado** | | | | | |
|---|---|---|---|---|---|---|
| | $Q = 10$ | | | $Q = 40$ | | |
| **Point cloud** | Travessia | Profundidade | Aritmético | Travessia | Profundidade | Aritmético |
| Man | 3,1046 | **2,0581** | 2,2759 | 0,7638 | **0,5345** | 0,6143 |
| Andrew | 4,4661 | **3,3881** | 3,6048 | 1,0225 | **0,7437** | 0,8111 |
| Phil | 5,0985 | **3,7117** | 3,7363 | 1,0416 | **0,6904** | 0,7124 |
| Ricardo | 1,5040 | **0,8993** | 1,1266 | 0,3395 | **0,2074** | 0,3049 |
| Sarah | 1,9500 | **1,1003** | 1,4549 | 0,4474 | **0,2455** | 0,3517 |
| Skier | 4,7303 | **2,8717** | 3,2023 | 1,3744 | **0,9071** | 0,9844 |
| Objects | 4,2951 | **3,3085** | 3,4167 | 0,8551 | **0,6692** | 0,8142 |

A Tabela I compara o desempenho obtido em $bits/voxel\ \ ocupado$ para os coeficientes ordenados por travessia e ordenados por profundidade, assim como o desempenho da RAHT-AC proposta originalmente por Queiroz e Chou. Pela tabela podemos observar que reordenar os coeficientes por profundidade e codificar com RLGR possui o melhor desempenho, além de o codificador RLGR possuir um baixo custo computacional. Riscamos afirmar que estes resultados fazem com que a RAHT-RLGR com coeficientes ordenados por profundidade seja o novo estado-da-arte em codificação de PCs.

## IV. EXPERIMENTOS

Executamos testes em 5 cenas realísticas capturadas em tempo real. Elas foram capturadas com 12 ou 13 câmeras e aproximadamente 3 milhões de pontos (veja Tabela II e Fig. 7). Estas imagens foram voxelizadas usando 11 bits de resolução espacial (octree com um profundidade $L = 11$), resultando em aproximadamente 2 milhões de voxels.

TABELA II

BASE DE DADOS

| | **Número de** | |
|---|---|---|
| **PC** | **Voxels ocupados** | **Câmeras** |
| boxer | 2056256 | 13 |
| longdress | 1860104 | 12 |
| loot | 1858707 | 13 |
| redandblack | 1467981 | 12 |
| soldier | 2365732 | 13 |

As cores são representadas no espaço YUV. Em nossos experimentos, o passo de quantização foi variado entre 15 e 500. Comparamos nosso método com a RAHT-RLGR aplicada independentemente para cada câmera, referida simplesmente como 'independente'. Os resultados em termos das curvas de taxa-distorção (TD) são mostrados nas Fig. 8 e 9 onde a PSNR foi calculada usando todos os componentes de cor.

Pelas Fig. 8 and 9 podemos observar que o método proposto possui um desempenho susbtancialmente superior à RAHT quando aplicado independentemente para cada câmera, com excessão da PC "boxer" para taxas acima de 0.2 $bits/voxel\ ocupado/camera$.


Fig. 7. Imagens renderizadas. Point clouds são uma cortesia de 8i®.

## V. CONCLUSÃO

Neste trabalho foi apresentado um método para comprimir a informação plenóptica de *point clouds*. O método consiste em subidividir os voxels em subvoxels e utilizar a posição dos subvoxels para representar as direções de vista. Este processo gera uma *point cloud* derivada, onde cada subvoxel possui apenas uma cor associada a sí, e permite a utilização de algoritmos já testados para a compressão de *point clouds* não plenópticas. Em particular, escolhemos a RAHT para a compressão dos dados. O resultados mostraram que o método consegue reduzir signifitivamente o número de bits necessários para codificar a PC quando comparado à RAHT aplicado independentemente para cada uma das câmeras.

Mostramos também que é possível melhorar o desempenho da RAHT-RLGR através do reordenamento dos coeficientes transformados, superando até mesmo a RAHT-AC. Acreditamos que o algoritmo RAHT-RLGR com o ordenamento por profundidade como proposto neste trabalho seja o novo estado-da-arte em codificação de *point clouds*.

## REFERÊNCIAS

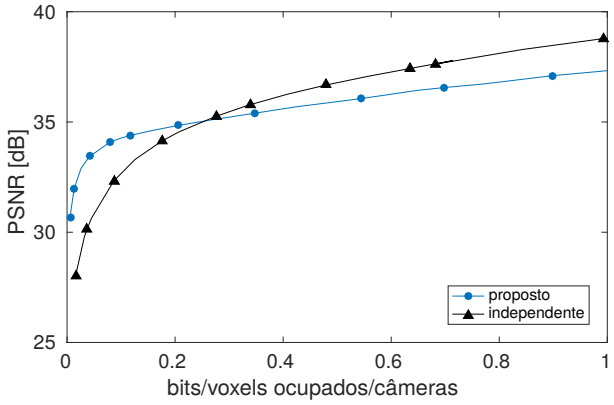
[1] R. L. de Queiroz and P. A. Chou, "Compression of 3d point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

[2] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *IEEE International Conf. Image Process. (ICIP)*, pp. 2066–2070, 2014.

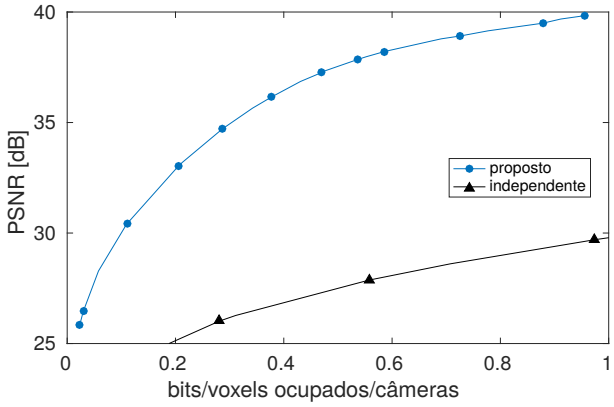
[3] R. de Queiroz and P. A. Chou, "Transform coding for point clouds using a Gaussian Process Model," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3507–3517, 2017.

[4] R. Malvar, "Adaptive run-length / golomb-rice encoding of quantized generalized gaussian sources with unknown statistics," *Proc. Data Compression Conference*, Snowbird, UT, USA, , March 2006.

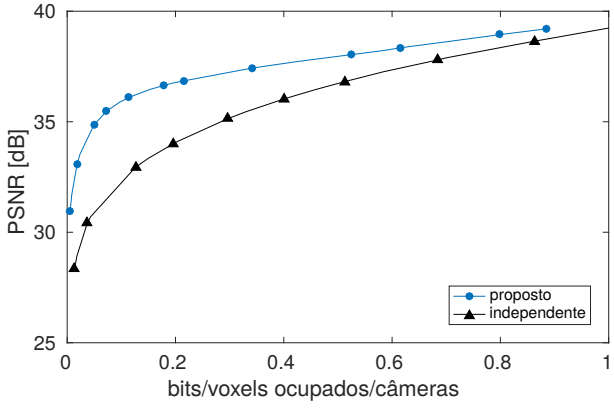
[5] E. Adelson and J. Bergen, "The plenoptic function and the elements of early vision," *Comput. Models of Visual Process.*, pp. 3–20, 1991.




(a) boxer

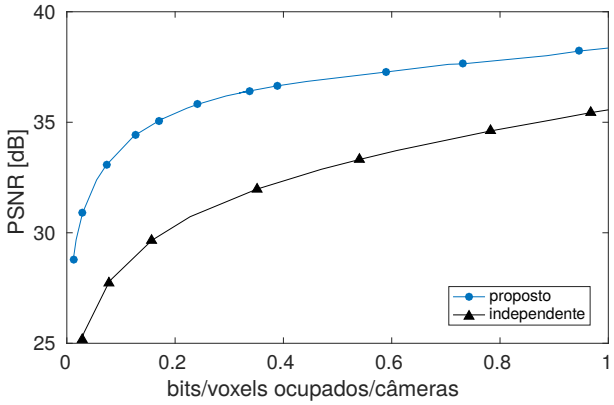



(b) longdress

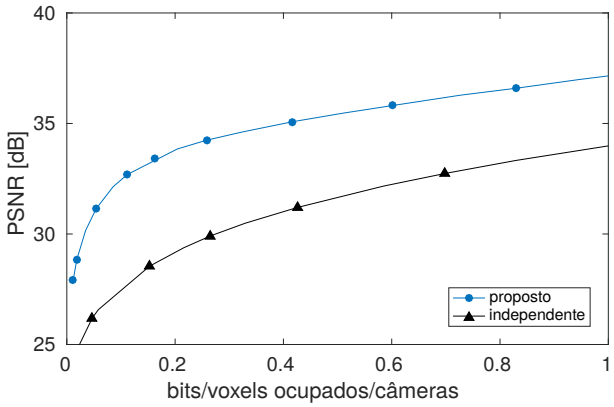



(c) loot

Fig. 8. Curvas de taxa-distorção para as *point clouds* (a) boxer, (b) longdress e (c) loot.




(d) redandblack




(e) soldier

Fig. 9. Curvas de taxa-distorção para as *point clouds* (d) redandblack e (e) soldier.


[6] S. Orts-Escolano *et al.*, "Holoportation: Virtual 3d teleportation in real-time," in *Proc. of Annual Symp. User Interf. Soft. and Tech. (UIST)*, pp. 741–754, 2016.

[7] A. P.-Miro, J. R.-Hidalgo, and J. R. Casas, "Registration of images to unorganized 3D point clouds using contour cues," in *European Signal Process. Conf. (EUSIPCO)*, pp. 81–85, 2017.

[8] C. Perra, F. Murgia, and D. Giusto, "An analysis of 3D point cloud reconstruction from light field images," in *International Conf. Image Process. Theory, Tools and Applications (IPTA)*, pp. 1–6, 2016.

[9] D. N. Wood *et al.*, "Surface light fields for 3D photography," in *Proc. Annual Conf. Computer Graphics and Interactive Techniques*, pp. 287–296, 2000.

[10] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk, "Light field mapping: Efficient representation and hardware rendering of surface light fields," *ACM Trans. on Graphics*, vol. 21, no. 3, pp. 447–456, 2002.

[11] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Proc. Eurographics / IEEE VGTC Conf. Point-Based Graphics*, pp. 111–121, 2006.

[12] Y. Huang, J. Peng, C.-C. Jay Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440–453, 2008.

[13] T. Ochotta and D. Saupe, "Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields," in *Proc. Eurographics Conf. on Point-Based Graphics*, pp. 103–112, 2004.

[14] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," in *IEEE International Conf. Image Process. (ICIP)*, pp. 3235–3239, 2015.

[15] J. Kammerl *et al.*, "Real-time compression of point cloud streams," in *IEEE International Conf. Robotics and Automation (ICRA)*, pp. 778–785, 2012.

[16] R. L. de Queiroz and P. A. Chou, " Transform coding for point clouds using a Gaussian process model," *IEEE Trans. on Image Processing,* Vol. 26, No. 7, July 2017.

[17] P. A. Chou, E. Pavez, R. L. de Queiroz, and A. Ortega, *Dynamic Polygon Clouds: Representation and Compression for VR/AR,* Microsoft Research Technical Report MSR-TR-2016-59, Jan. 2017.

[18] R. de Queiroz and P. A. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3947–3956, 2016.