Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Compressão de nuvens de pontos

Miguel de Carvalho Pachá

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Engenharia Elétrica

Orientador
Prof. Dr. Eduardo Peixoto Fernandes da Silva

Brasília
2019

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Compressão de nuvens de pontos

Miguel de Carvalho Pachá

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Engenharia Elétrica

Prof. Dr. Eduardo Peixoto Fernandes da Silva (Orientador)
niversidade de Brasília

Prof. Dr. Prof. Dr.
University Institute

Prof.a Dr.a
Coordenadora do Bacharelado em Engenharia Elétrica

Brasília, 16 de março de 2019

Dedicatória

Dedicatória

Agradecimentos

Obrigado!

Resumo

O *resumo* é um texto inaugural para quem quer conhecer o trabalho, deve conter uma breve descrição de todo o trabalho (apenas um parágrafo). Portanto, só deve ser escrito após o texto estar pronto. Não é uma coletânea de frases recortadas do trabalho, mas uma apresentação concisa dos pontos relevantes, de modo que o leitor tenha uma ideia completa do que lhe espera. Uma sugestão é que seja composto por quatro pontos: 1) o que está sendo proposto, 2) qual o mérito da proposta, 3) como a proposta foi avaliada/validada, 4) quais as possibilidades para trabalhos futuros. É seguido de (geralmente) três palavras-chave que devem indicar claramente a que se refere o seu trabalho. Por exemplo: *Este trabalho apresenta informações úteis a produção de trabalhos científicos para descrever e exemplificar como utililzar a classe LaTeX do Departamento de Ciência da Computação da Universidade de Brasília para gerar documentos. A classe* `UnB-EnE` *define um padrão de formato para textos do EnE/FT-UnB, facilitando a geração de textos e permitindo que os autores foquem apenas no conteúdo. O formato foi aprovado pelos professores do Departamento e utilizado para gerar este documento. Melhorias futuras incluem manutenção contínua da classe e aprimoramento do texto explicativo.*

Palavras-chave: LaTeX, metodologia científica, trabalho de conclusão de curso

Abstract

O *abstract* é o resumo feito na língua Inglesa. Embora o conteúdo apresentado deva ser o mesmo, este texto não deve ser a tradução literal de cada palavra ou frase do resumo, muito menos feito em um tradutor automático. É uma língua diferente e o texto deveria ser escrito de acordo com suas nuances (aproveite para ler `http://dx.doi.org/10.6061%2Fclinics%2F2014(03)01`).

Por exemplo: *This work presents useful information on how to create a scientific text to describe and provide examples of how to use the Computer Science Department's LaTeX class. The* ***UnB-EnE*** *class defines a standard format for texts, simplifying the process of generating CIC documents and enabling authors to focus only on content. The standard was approved by the Department's professors and used to create this document. Future work includes continued support for the class and improvements on the explanatory text.*

Keywords: LaTeX, scientific method, thesis

Sumário

Lista de Tabelas

Lista de Abreviaturas e Siglas

BCE Biblioteca Central.

CAPES Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

CEP Comissão de Ética Pública.

CTAN Comprehensive TEX Archive Network.

EnE Departamento de Engenharia Elétrica.

UnB Universidade de Brasília.

1 Teoria e revisão bibliográfica

Este capítulo explica alguns conceitos de nuvens de pontos e de compressão de dados.

1.1 Nuvens de pontos

1.1.1 Descrição teórica

Nuvens de pontos (em inglês *point clouds*) são uma maneira de representar objetos como um conjunto de pontos em um espaço tridimensional, chamados *voxels*, do inglês *volume pixel*. É uma técnica de representação mais simples do que representações como redes poligonais (*polygon meshes*), já que não são transmitidas informações sobre as relações entre os pontos (*e.g.* arestas ou faces). Isso pode ser vantajoso para algumas aplicações já que a renderização é simplificada e a quantidade de informação transmitida é menor. Além da informação de posição, cada ponto da nuvem pode ter outros atributos, dependendo da aplicação, como cor, refletância, ou coordenadas do vetor normal à superfície. É possível criar vídeos tridimensionais com sequências onde cada um dos quadros é uma *pointcloud*

A representação em nuvens de pontos é uma representação esparsa, diferente da representação de uma imagem bidimensional como uma foto, por exemplo. Enquanto uma imagem bidimensional geralmente é entendida como uma função que associa a todo e qualquer pixel um dado de cor, as representação de uma *point cloud* não é assim, porque nem todos os *voxels* são ocupados. Para quantificar o espaço ocupado por representação de uma nuvem de pontos, mede-se quantos bits são gastos, em média, para representar um voxel ocupado. A unidade é abreviada *bpov* (*bits per occupied voxel*).

1.1.2 Aplicações

Representações em nuvens de pontos podem ser usadas para criar ambientes imersivos, seja para entreteninmento (como filmes e jogos), para comunicação pessoal (vídeo-chamadas tridimensionais), ou para ensino (por exemplo modelos anatômicos no ensino da medicina). Aliadas a sistemas de informação geográfica, nuvens de pontos podem ser

utilizadas também para representar relevo e cobertura do terreno, tendo assim uma possível aplicação na agrimensura e no manejo de recursos naturais. Também podem ser usadas para fins acadêmicos, por exemplo para a preservação e estudo de ambientes ou objetos de valor histórico.

1.1.3 Obtenção e transmissão de nuvens de pontos

Nuvens de pontos podem ser obtidas seja a partir de dados brutos gerados por tecnologias especializadas, como *LiDAR* e *scanners* tridimensionais, ou então a partir do cruzamento de imagens convencionais em duas dimensões. Este processo de obtenção de *point clouds* é denominado registro de pontos (em ingês *point set registration*). Um dos formatos de arquivo utilizado para salvar nuvens de pontos é o formato PLY (Polygon File Format - uma descrição está disponível em [1]). Apesar de ter menos informação do que representações poligonais, este tipo de modelagem gera arquivos muito grandes, o que dificulta sua transmissão. De fato, o tamanho de cada quadro dos vídeos tridimensionais estudados neste trabalho é da ordem de alguns *megabytes*. Portanto, é interessante comprimir este tipo de informação para transmissão ou arquivamento.

1.2 Compressão de dados

1.2.1 Informação e entropia

Como notam Thomas e Cover [2], "o conceito de informação é muito amplo para ser capturado em uma única definição". No entanto, informação pode ser entendida como tudo aquilo que se deseja transmitir, como textos, sinais de voz, imagens. Em teoria da informação, os dados a serem transmitidos são interpretados como uma sequência de símbolos emitidos por uma fonte. O comportamento da fonte é geralmente modelado como uma variável aleatória, cuja distribuição de probabilidade relaciona cada símbolo com a probabilidade de sua ocorrência. O conjunto de símbolos da fonte, que corresponde ao suporte da variável aleatória, é denominado alfabeto (que pode ser finito ou infinito). A taxa média de informação associada a uma variável aleatória pode ser medida com o conceito de entropia, que foi definido pela primeira vez no artigo seminal de Shannon [3].

Definição 1 *A entropia $H(X)$ de uma variável aleatória X é dada por*

$$H(X) = -\sum_{x \in X} p(x) \log_b p(x)$$

Esta medida é definida para uma base b. É usual tomar a base como 2, fazendo com que a entropia seja medida em *bits*. A entropia é o limite de compressão de um sinal.

Isso quer dizer que não é possível comprimir um sinal abaixo de sua entropia sem perder informação ou acrescentar informação oriunda de outra fonte.

1.2.2 Compressão

Sayood [4] descreve a compressão de dados como "a arte ou a ciência de representar dados de maneira compacta". As técnicas de compressão modelam estruturas ou identificam padrões na informação, com objetivo de diminuir a redundância e permitir uma transmissão mais eficiente. A compressão têm dois componentes complementares: a codificação, que diminui o tamanho da informação, e a decodificação, que reconstitui os dados. Se os dados reconstituídos são idênticos aos dados originais, a compressão é dita sem perdas (*lossless compression*). Caso a reconstituição tenha perdas (*lossy compression*), é interessante ter uma medida da qualidade da reconstrução, ou seja, da distroção do sinal reconstituído. Uma métrica que pode ser usada é o erro médio quadrático (MSE - *means squared error*).

Definição 2 *O erro quadrático médio para uma reconstituição $\hat{X}$ de N valores de um sinal X é definido por*

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(X_i - \hat{X}_i\right)^2$$

Outra medida é a razão sinal-ruído de pico (PSNR - *peak signal-to-noise ratio*, medida em decibéis.

Definição 3 *A razão sinal ruído de uma reconstituição de um sinal cujos valores máximo e mínimo são X_{max} e X_{min} é dada por*

$$PSNR = 10 \cdot \log_{10}\left(\frac{(X_{max} - X_{min})^2}{MSE}\right)$$

1.2.3 Codficação de Entropia

As técnicas de codificação de entropia são técnicas de compressão sem perdas que utilizam apenas as propriedades estatísticas da fonte, ou seja, as informações sobre a probabilidade de emissão dos diferentes símbolos. A codificação de entropia difere de outros esquemas de compressão que se adaptam à natureza semântica do conteúdo, explorando propriedades específicas de certos tipo de informação,por exemplo propriedades inerentes a sinais de vídeo, áudio, etc. Neste trabalho foi utilizado um tipo de codificação de entropia chamada a codificação aritmética, que será descrita mais a frente.

1.2.4 Modelagem de fonte e codificação de entropia

As técnicas de compresssão são tipicamente compostas por uma sequência de vários estágios. [5] A relação entre esses componentes pode ser vista na figura.
emph(incluir figura)
A primeira parte do processo de compressão consiste em extrair padrões existentes nos dados a serem transmitidos. Esta parte do processo assume certas propriedades sobre o sinal a ser comprimido. Por isso, ela deve ser adaptada à natureza do sinal, e caso ela induza perdas, ela deve ser adaptada às necessidades do usuário do produto final do processo de compressão. Tome-se por exemplo de um sinal de vídeo. Pixels de uma dada região de um quadro tendem a ser parecidos entre si. Além disso, eles também tendem a ser parecidos com os pixels da mesma região de quadros anteriores. Essa semelhança implica uma redundância que pode ser explorada na compressão. Ainda no exemplo do sinal de vídeo, é importante levar em consideração as características do espectador: por exemplo, a visão humana é mais sensível a variações na luminância do que variações na cor. Sendo assim, caso perdas sejam induzidas neste exemplo, é melhor que o erro de reconstituição seja no sinal de cor e não no sinal de luminância.
A informação gerada por esse processo é modelada estatisticamente, gerando um fluxo de símbolos que alimenta um codificador de entropia. Neste trabalho, a redundância do sinal de cor foi explorada usando o processamento de geometria, que gera um sinal que é codificado utilizando codificação diferencial, cujo resultado é passado por um codificador de entropia. O codificador de entropia escolhido foi o codificador aritmético. Essas três técnicas são explicadas nas seções que seguem.

1.3 Processamento de geometria

No contexto da codificação de nuvens de pontos, o processamento de geometria consiste em utilizar a informação das coordenadas dos *voxels* ocupados para agrupar pontos próximos entre si. Esta técnica visa explorar o fato de que *voxels* próximos têm cores parecidas. Assim, é possível transmitir a cor de *voxels* parecidos de maneira sequencial. Isso cria uma redundância no sinal que pode ser aproveitada pela codificação diferencial, que está descrita na próxima seção.

1.3.1 Encontrando os vizinhos mais próximos

Será visto no capítulo de metodologia que para gerar os filamentos, um dos passos possíveis é encontrar os n vizinho mais próximos (*kNN*, ou k *nearest neighbours*). Um algoritmo de força bruta para encontrar os k vizinhos mais próximos tem complexidade espacial

$\mathcal{O}(n)$. Aplicando-o para todos os pontos da nuvem, obtemos uma complexidade)(n^2). No entanto, é possível realizar um pré-processamento dos pontos, gerando uma estrutura denominda árvore K-D, que particiona o plano em regiões contendo quantidades iguais de pontos. Esse processo inicial tem a complexidade $\mathcal{O}(??)$. Usando essa árvore a procura pelos KNN tem complexidade $\mathcal{O}(\log n)$. Esta estrutura foi usada para encontrar rapidamente os pontos vizinhos na geração de filamentos, como será descrito na metodologia. O algoritmo FLANN é uma heurística que permite fazer isso de forma mais rápida, mas não garante uma seleção exata.

1.3.2 Curvas preenchedoras de espaço

As curvas preenchedoras de espaço (*Space-filling curves*, ou *SPF*s) são curvas fractais que preenchem o quadrado unitário. Elas preservam a localidade. Isso significa que se dois pontos estão próximos na curva, eles estão próximos no espaço $\mathbf{R}^2$. (continuar)
(Um exemplo de SFC é a curva de Hilbert.)
(Incluir figuras de SFC.) (Citar livro do Bader.) (Citar TCC Hözmuller.)

1.4 Codificação diferencial

A codificação diferencial aproveita a correlação entre o dados consecutivos para comprimir a informação a ser transmitida. Ela consiste em usar os dados anteriores para estimar o próximo valor a ser transmitido. Este tipo de codificação é útil para codificar sinais que mudam devagar (sinais "passa-baixa"). Ao invés de transmitir o sinal, é transmitido o erro de estimação. Tipicamente este erro tem propriedades estatísticas que facilitam sua compressão por um codificador de entropia: ele tende a ter uma extensão menor e também uma variância menor.

1.4.1 Gerando uma estimativa

O método mais simples é usar o último símbolo enviado como estimativa. No entanto, é possível criar estimativa usando funções mais complexas, que dependem de uma quantidade maior de símbolos anteriores. Um tipo de estatística bem estudado é uma função linear (média ponderada) dos pontos anteriores. Os coeficientes ótimos podem ser calculados, como descrito no capítulo ?? do livro [4]. Neste trabalho, será usada a média simples de m símbolos anteriores. Este parâmetro m será otimizado.

1.4.2 Quantização

Além disso, é possível obter melhores taxas de compressão usando um quantizador para os valores das diferenças, ou seja, induzindo perdas. Isso não foi feito neste trabalho.

1.5 Codificador aritmético

A codificação aritmética é uma técnica de codificação de entropia. A sua taxa de compressão aproxima-se assintoticamente da entropia para fontes independente e identicamente distribuídas (iid). Ela consiste em um algoritmo que associa de maneira única uma mensagem a um subintervalo do intervalo $[0, 1)$. Este intervalo é denominado *tag*. Para efeitos de codificação, é equivalente transmitir a *tag* ou transmitir qualquer número nela contido, acompanhado da quantidade de símbolos codificados. Nas subseções seguintes são descritos os princípios gerais do código aritmético e de sua implementação com precisão finita, denominada implementação inteira. Esta seção se baseia na explicação de [4], capítulo ??.

1.5.1 Descrição dos princípios do código aritmético

Seja $(a_1, a_2, ..., a_n)$ um arranjo do alfabeto n-ário de uma fonte de símbolos. A função de probabilidade acumulada associada a esse arranjo é dada por

$$F(k) = \sum_{i=0}^{k} p(a_i)$$

De maneira geral, pode-se associar a um intervalo $I_i = [l_i, u_i)$ uma partição

$$P_{I_i}(k) = \frac{l_i + F(k-1)}{u_i - l_i}, \frac{l_i + F(k)}{l_i - u_i}$$

Para transmitir uma mensagem $(m_1, m_2, ..., m_N)$ parte-se do intervalo $I_0 = [0, 1)$ e obtém-se intervalos sucessivamente menores usando a fórmula

$$I_n = P_{I_{n-1}}(m_n)$$

. O último intervalo dessa sequência, I_n, é a *tag* associada a mensagem. Basta transmitir qualquer número t_0 pertencente a este intervalo, a quantidade N de símbolos transmitidos, e a probabilidade função de probabilidade acumulada. Para decodificar, basta partir do número recebido t_0 e proceder da seguinte maneira. Seja $D_I(t)$ a função que associa t ao intervalo da partição P_I que o contém. Basta então calcular sucessivamente, para

$< n \leq N$:

$$\hat{m}_n = D_{I_n}(t_n)$$
$$t_n = \frac{l + F(\hat{m}_n)}{u - l}$$
$$I_n = P_{I_{n-1}}(m_n)$$

.

1.5.2 Implementação com precisão finita

((Aqui devo explicar a necessidade de implementar o algoritmo com precisão finita, e como isso é feito.)) Muitas plataformas não podem trabalhar com precisão arbitrária (depende de cálculo simbólico e é computacionalmente custosa, ou simplesmente não está implementada). Por isso é interessante estudar a implementação com precisão finita, que foi empregada no presente trabalho. Ao invés de trabalhar com o subintervalos do intervalo unitário $[0, 1)$, são usados subintervalos do intervalo $[0, M)$ onde M é o máximo inteiro que pode ser representado com a dada precisão. Para uma precisão de n *bits*, $M = 2^n - 1$. Os cálculos para codificação e decodificação são adaptados à aritmética inteira, usando a função $floor$.

1.5.3 Considerações práticas

(Aqui devo explicar porque mensagens muito curtas não são bem compactadas pelo codificador aritmético.)

1.6 Estado da arte em compressão de nuvens de pontos

É usual na compressão de nuvens de pontos separar a informação correspondente à geometria do objeto (coordenadas dos pontos) daquela que representa o resto dos atributos (por exemplo o valor de cor de cada ponto). Vários trabalhos já apresentaram esquemas de compressão do canal de cor de *point clouds* com base na geometria. De forma geral, estes esquemas partem do princípio que pontos espacialmente próximos tendem a ter cores parecidas. Assim, os algoritmos apresentados organizam os pontos de forma que pontos parecidos sejam transmitidos sequencialmente, gerando por assim dizer um sinal passa-baixa. É importante salientar que nestes esquemas a transmissão da geometria do objeto não pode ter sem perdas. Já existem algoritmos para comprimir a geometria de vídeos de *point clouds* [6].

Já o sinal de cor pode ser transmitido com ou sem perdas, dependendo do algoritmo. O estado da arte é o algoritmo da *Region-Adaptive Hierarchical Transform* (RAHT - Transformada Hierárquica adaptada à região), descrito em [7]. No artigo, os pontos são agrupados por proximidade em uma estrutura de árvore. Os trabalhos [8] e [9] dividem a imagem em sequências de pontos próximos denominadas filamentos. [9] utiliza a transformada wavelet para cada codificar cada filamento. [8] utiliza um esquema simples de codificação diferencial, e este esquema foi o ponto de partida para o presente trabalho. No capítulo de Resultados o algoritmo proposto no presente trabalho será comparado com os trabalhos anteriores.

2 Metodologia

Este capítulo apresenta a metodologia aplicada.

2.1 Visão geral

2.2 Processamento da geometria

Consiste em obter filamentos a partir da imagem. Inicialmente a imagem é dividida em camadas de mesma coordenada. Escolhe-se um ponto arbitrariamente, e a partir deste ponto procura-se o ponto mais próximo que ainda não foi visitado. Repete-se esse processo até não sobrarem pontos na camada, sempre guardando a distância em um vetor auxiliar. Em seguida, este filamento é cortado quando as ditâncias estão acima de um certo limiar que é um parâmetro de entrada a ser otimizado. Assim é obtida uma lista de filamentos. (incluir uma figura ilustrando camada, outra ilustrando filamento).

2.3 Escolha da codificação

Como visto anteriormente, é difícil codificar de maneira eficiente sequências muito pequenas pois a informação lateral necessária é proporcionalmente grande. Então, para filamentos muito curtos, não há codificação, tendos os seus valores transmitidos diretamente pelo canal lateral. O limiar de comprimento a partir do qual o filamento é codificado é um parâmetro de entrada a ser otimizado.

2.4 Codificação diferencial

Para obter a estimativa da codificação diferencial, foi utilizada a média aritmética dos últimos n voxels do filamento. Foi variado o parâmetro n para otimizar a codificação (diminuindo a variância do sinal de erro).

2.5 Modelagem da fonte e codificação de entropia

Como visto anteriormente, quanto melhor o modelo que se tem da fonte, melhor são os resultados do codificador aritmético. Vários esquemas de transmissão do contexto são possíveis. É possível enviar uma descrição completa da distribuição de probabilidade, mas isso ocupa bastante espaço. É possível modelar a distribuição de de probabilidade usando a distribuição laplaciana. Nese casso é necessário enviar apenas os parâmetros da laplaciana. É possível usar um codificador adaptativo, em que as distribuições são atualizadas a cada símbolo processado. Mesmo com uma distribuição inicial errada, o codificador adaptativo aprende a seguir a variável com o tempo, aproximando-se da entropia. Partindo do princípio que as distribuições dos símbolos oriundos da codificação diferencial são parecidas para filamentos diferentes, é possível usar um esquema que reaproveita o contexto para todos os filamentos.

2.6 Implementação

Os algoritmos foram implementados usando a linguagem de programação Python (versão 3.7) [10]. Foram usadas as seguintes bibliotecas:

NumPy Para funções numéricas, especialmente para vetores e matrizes.[11]

Open3D Para processamento de nuvens de pontos (leitura de arquivos PLY, algoritmos de busca e renderização). [12]

bitstring Para a escrita e leitura *bit*-a-*bit* dos arquivos.[13]

Matplotlib para gerar os gráficos presentes deste relatório. [14]

Foram desenvolvidos módulos para processar a geometria, codificar diferencialmente e implementar o código aritmético.

2.7 Procedimentos experimentais

Foram realizados vários tipos de experimentos diferentes para otimizar os parâmetros de compressão. Os testes foram feitas sobre alguns quadros da base de dados *Microsoft Voxelized Upper Bodies* [15]. A base de dados é constituída por vídeos tridimensionais representando humanos da cintura para cima. Cada quadro está salvo em um arquivo PLY separado. Em todas as nuvens estudadas, cada *voxel* tem 9 bits de para cada coordenada geométrica, ou seja, cada uma delas está contida em um cubo de dimensão 512. Para cada voxel, cada canal de cor comporta 8 bits de informação. Assim os arquivos possuem 27

Tabela 2.1: *Point clouds* escolhidas para o trabalho.

Nome	**Vídeo**	**Número de quadro**	**Voxels ocupados**
Pointcloud 1	Ricardo9	0001	214656
Pointcloud 2	Ricardo9	0091	337803
Pointcloud 3	Phil9	0116	385467
Pointcloud 4	Andrew9	0262	277583
Pointcloud 5	Sarah9	0146	304597

bpov de informação de geometria e 24 bpov de informação de cor. Os quadros escolhidos estão descritos na tabela 2.1 (incluir uma figura com renderizações das pointclouds)

2.7.1 Parâmetros variados

Foi estudada a variação dos comprimentos dos filamentos em função do limiar de distância, para diferentes métodos de geração de filamentos.
Foi variada a quantidade de pontos utilizada na estimativa.
Para determinar o limiar de codificação ideal, foi medida a taxa de compressão em função do comprimento dos filamentos, comparando com a transmissão sem codificação.

Referências

[1] Bourke, Paul: *Ply - polygon file format.* `http://paulbourke.net/dataformats/ply/`, acesso em 2019-10-31. 2

[2] Thomas M. Cover, Joy A. Thomas: *Elements of Information Theory, Second Edition.* Wiley-Interscience, 2006. 2

[3] Shannon, C. E.: *A mathematical theory of communication.* The Bell System Technical Journal, 27(3):379–423, July 1948. 2

[4] Sayood, Khalid: *Introduction to Data Compression, Fifth Edition.* Morgan Kaufmann, 2018. 3, 5, 6

[5] Sayood, Khalid: *Lossless Data Compression Handbook.* Academic Press, 2003. 4

[6] De Queiroz, Ricardo, Diogo Garcia, Philip Chou e Dinei Florencio: *Distance-based probability model for octree coding.* IEEE Signal Processing Letters, 25:1–1, abril 2018. 7

[7] Queiroz, Philip A. Chou Ricardo L. de: *Compression of 3d point clouds using a region-adaptive hierarchical transform.* IEEE Transactions on Image Processing, 25(8):3947–3956, 2016. 8

[8] Reis, Bruno José Bergamaschi Kumer: *Codificação das cores de uma point cloud através da sua divisão em filamentos*, 2018. 8

[9] Macedo, Lucas Rezende de: *Compressão de cor de point clouds utilizando transformada wavelets*, 2018. 8

[10] Foundation, Python Software: *Python language reference, version 3.7.0.* `http://www.python.org`, acesso em 2019-10-31. 10

[11] van der Walt, S., S. C. Colbert e G. Varoquaux: *The numpy array: A structure for efficient numerical computation.* Computing in Science Engineering, 13(2):22–30, March 2011. 10

[12] Zhou, Qian Yi, Jaesik Park e Vladlen Koltun: *Open3D: A modern library for 3D data processing.* arXiv:1801.09847, 2018. 10

[13] Griffiths, Scott: *Bitstring, a python module to help you manage your bits.* `https://scott-griffiths.github.io/bitstring/`, acesso em 2019-10-31. 10

[14] Hunter, J. D.: *Matplotlib: A 2d graphics environment*. Computing in Science Engineering, 9(3):90–95, May 2007. 10

[15] Microsoft (Charles Loop, Qin Cai, Sergio Orts Escolano e Philip A. Chou): *Jpeg pleno database: Microsoft voxelized upper bodies - a voxelized point cloud dataset*. `https://jpeg.org/plenodb/pc/microsoft/`, acesso em 2019-10-31. 10