Ajuste de Primitivas Geométricas a Nuvens de Pontos de Grandes Estruturas

Igor P. Maurell [*] **Paulo Drews-Jr** [*] **Carlos A. Eguti** [**]

[*] *NAUTEC - Grupo de Automação e Robótica Inteligente*
Universidade Federal do Rio Grande - FURG, RS
(e-mail: igormaurell@furg.br, paulodrews@furg.br)
[**] *IEM - Divisão de Engenharia Mecânica*
Instituto Tecnológico de Aeronáutica - ITA, SP
(e-mail: eguti@ita.br)

Abstract: The geometric primitive fitting is extremely important in tasks like 3D data simplification and automatic As Built. In this context, only few works address this problem in large structures data, where the point clouds are larger and more complex. Thus, the present work brings a methodology to deal with the problem, as well as a complete discussion about the state of the art in large structures data processing and in geometric primitive fitting.

Resumo: O ajuste de primitivas geométricas é extremamente importante em tarefas como simplificação de dados 3D e *As Built* automático. Nesse contexto, poucos trabalhos abordam essa problemática em dados de grandes estruturas, onde as nuvens de pontos são maiores e mais complexas. Dessa forma, o presente trabalho traz uma metodologia para lidar com o problema, bem como uma completa discussão sobre o estado da arte em processamento de dados de grandes estruturas e em ajuste de primitivas geométricas.

Keywords: 3D data; RANSAC; large scale; geometric primitive; large structures.
Palavras-chaves: dados 3D; RANSAC; larga escala; primitivas geométricas; grandes estruturas.

1. INTRODUÇÃO

Parafraseando Kaiser et al. (2019), pode-se dizer que "a quantidade de dados 3D capturados está continuamente crescendo, com a democratização das câmeras de profundidade, o desenvolvimento de sistemas modernos de estéreo de múltiplas vistas e o surgimento de sistemas de captura 3D monoculares baseados em aprendizado de máquina.". Nesse contexto, motivado pelo alto volume e variedade dos dados, novas técnicas estão constantemente surgindo com as mais diversas finalidades. Dentro dessas, a segmentação e ajuste paramétrico das formas geométricas implícitas nos dados recebe muita atenção dos pesquisadores. Porém, quando se trata de grandes estruturas, como: construções civis ou plantas industriais, os métodos de ajuste paramétrico de primitivas geométricas focam, geralmente, só em planos, como pode ser observado em Kaiser et al. (2019). Além disso, nos casos em que outros tipos de primitiva são abordados, é observado que o ajuste é apenas um passo de uma finalidade maior que se busca alcançar com os métodos. Sendo assim, as primitivas geométricas encontradas não são muito refinadas e os resultados são inferiores aos obtidos com os métodos focados na resolução do problema para pequenas estruturas.

A forma mais fundamental de representação 3D são as nuvens de pontos, as quais são geradas por grande parte dos métodos de captura. Essas nuvens, consistem em vetores ou matrizes que abrigam os pontos, os quais podem não ser representados somente por suas coordenadas cartesianas, como também pela cor, vetor normal, curvatura da superfície, etc. Com isso, o problema de ajuste de primitivas geométricas busca encontrar as diferentes formas geométricas básicas que estão implícitas nos dados, segmentando a nuvem em uma série de primitivas e ajustando os parâmetros para essas. De maneira formal, baseado no que foi definido em Schnabel et al. (2007), dado um conjunto de N pontos $P = \{p_1, p_2, ..., p_N\}$, com opcionalmente suas normais associadas $\{n_1, n_2, ..., n_N\}$, o algoritmo deve produzir um conjunto de K primitivas $S = \{s_1, s_2, ..., s_K\}$, as quais são representadas por seus parâmetros, e seus respectivos conjuntos de pontos $P_{s_1} \subset P, P_{s_2} \subset P, ..., P_{s_K} \subset P$. Dessa forma, pode-se observar que este se trata de um clássico problema *chicken-and-egg*, como demonstrado em Li and Feng (2019), onde duas tarefas interdependentes devem ser resolvidas simultaneamente. Essa constatação se dá pelo fato de que, tendo os parâmetros de todas as primitivas, a segmentação se torna um problema de distância mínima ponto-primitivas, além disso, no caso de se ter uma segmentação perfeita dos pontos, diferentes métodos robustos de ajuste podem ser aplicados gerando os melhores parâmetros possíveis para a distribuição dos dados da entrada.

Apesar de o problema de ajuste de primitivas geométricas em nuvens de pontos ser explorado na literatura, este se diferencia em contextos de pequenas e grandes estruturas, principalmente na segmentação, visto que a quantidade de pontos tende a ser muito diferente para uma mesma resolução da nuvem de entrada. Além disso, os aspectos construtivos da estrutura, que configuram a maneira como as primitivas estão distribuídas no espaço, também são

muito diferentes. Junto a isso, apesar da grande quantidade de dados 3D disponíveis de forma livre, não há conhecimento sobre a existência de conjuntos de dados públicos até o presente momento que, junto com as nuvens de pontos, tenham valores de referência para o ajuste ideal de primitivas no contexto de grandes estruturas. Assim, a falta desses valores chamados de *ground truth* impede uma comparação quantitativa confiável de diferentes soluções, bem como, o uso de técnicas de aprendizado supervisionado, o que impõe uma forte limitação ao trabalho.

O objetivo do projeto é avaliar métodos do estado da arte no ajuste de primitivas geométricas, aplicado a nuvens de pontos de grandes estruturas. Para que esse objetivo seja cumprido, alguns outros devem ser alcançados anteriormente, são eles:

- Entender o estado da arte em processamento de nuvem de pontos de grandes estruturas
- Entender o estado da arte em ajuste de primitivas
- Propor metodologias para lidar com grandes estruturas
- Realizar reparametrizações nas técnicas existentes para obter melhores resultados nessa problemática
- Investigar conjuntos de dados reais públicos para realizar validação.

O trabalho está dividido da seginte forma: na Sec. 2 serão apresentados os trabalhos relacionados com a temática proposta, na Sec. 3 será apresentada a metodologia utilizada para enfrentar o problema e, finalizando, na Sec. 4 os experimentos realizados serão descritos e os resultados serão demonstrados.

2. TRABALHOS RELACIONADOS

Com o objetivo de levantar o estado da arte relacionado com o problema a ser resolvido, alguns diferentes tipos de métodos podem ser considerados, tanto por estarem trabalhando no contexto do processamento de dados 3D de grandes estruturas, quanto pelo motivo de estarem se propondo a resolver o problema do ajuste de primitivas geométricas a nuvens de pontos.

2.1 Trabalhos no Contexto de Grandes Estruturas

Pătrăucean et al. (2015) revisa métodos de *As Built* automático, os quais buscam formar um projeto da estrutura como está construída, a partir da captura e processamento de dados 3D. Junto a isso, o trabalho também comenta sobre diferentes técnicas de pré-processamento dos dados, onde enfatiza que segmentação em partes menores, filtro de *downsampling*, filtro remoção de *outliers* e filtro de ruído, são muito utilizados na literatura.

Dentro da área de processamento de dados 3D de grandes estruturas, grande parte dos trabalhos são voltados para o meio civil, tendo como objeto de estudo residências ou prédios, de forma interna ou externa à construção. Porém, alguns trabalhos do meio industrial se destacam por processarem dados de grandes estruturas industriais e por utilizarem passos de ajuste de primitivas geométricas. Nesse contexto, Rabbani et al. (2007) investiga uma maneira de acelerar o processo de registro de nuvens de pontos utilizando detecção e correspondência de primitivas geométricas, as quais são ajustadas através da optimização de problemas de mínimos quadrados. Já Pang et al. (2015), busca realizar classificação e modelagem de nuvem de pontos utilizando alguns princípios do ajuste de primitivas planares e cilíndricas, ajustadas através de RANSAC, entre outras técnicas de aprendizado de máquina.

Analisando trabalhos mais novos, alguns já se baseiam em técnicas de aprendizado profundo para realizar algumas tarefas em nuvens de pontos de larga-escala, as quais geralmente representam grandes estruturas. Nesse contexto, Landrieu and Simonovsky (2018) busca realizar a tarefa de segmentação semântica em grandes nuvens de pontos a partir da divisão em partes menores, formação de um grafo de super pontos e de redes de convolução em grafos. Já em Hu et al. (2020), o mesmo problema é enfrentado, porém de maneira mais eficiente e sem passos de pré-processamento, como a divisão em partes menores feita anteriormente. Nesse método, denominado RandLA-Net, a nuvem é continuamente amostrada aleatoriamente para resoluções menores e, a cada amostra, características locais são calculadas a partir de um paradigma denominado "*Local Feature Agregation*". Ao final, as características são utilizadas para segmentar semanticamente a nuvem de entrada.

dowAtravés da análise dos trabalhos do contexto de grandes estruturas, é possível perceber que não é comum a existência de técnicas que tenham como objetivo o ajuste de múltiplas primitivas geométricas. Visto que, o Indoor Scan2Bim busca a modelagem da informação da construção a partir apenas do ajuste planar para paredes, chão e teto. Já o Rabbani et al. (2007), apesar de ajustar múltiplos tipos de primitivas, não foca na qualidade do ajuste, e sim na simplificação, a partir das primitivas, do processo de registro de nuvens de pontos. Já o método Pang et al. (2015), apesar de não ter um código público, parece focar especificamente em plantas industriais com grandes linhas de tubulação. Logo, mesmo tendo o ajuste de primitivas como um de seus passos, seu foco não é um ajuste genérico em qualquer grande estrutura. Já olhando para os métodos baseados em aprendizado, é possível notar uma evolução gradual das técnicas para que consigam lidar com nuvens de pontos de larga-escala, porém, a falta de conjuntos de dados com *ground truth* torna o ajuste de primitivas geométricas em grandes estruturas uma questão ainda não explorada dentro dessa fundamentação teórica. Contudo, os trabalhos trazem algumas discussões importantes sobre o pré-processamento de grandes nuvens de pontos, além de discussões sobre as dificuldades de lidar com essa quantidade imensa de dados.

2.2 Ajuste de Primitivas Geométricas

Kaiser et al. (2019) avalia os principais métodos de ajuste de primitivas geométricas que foram publicados até o ano de 2016, nessa revisão os métodos são tabelados, principalmente, com base em: tipos de primitivas que ajustam, contexto dos dados, tipo de dado de entrada e a fundamentação teórica que o método se baseia. Para o presente trabalho, se busca métodos que realizam o ajuste de múltiplos tipos de primitivas, que funcionem para grandes estruturas, que usem nuvem de pontos como dado de entrada e que tenham como finalidade, e não como meio, o

ajuste de primitivas geométricas. Logo, com base nas premissas mencionadas, o Efficient RANSAC (Schnabel et al., 2007) se demonstrou como o método de maior relevância e adaptabilidade da literatura, considerando o estado da arte clássico, ou seja, não baseado em aprendizado de máquina, servindo como base de solução para o problema proposto.

O Efficient RANSAC pode ser considerado como o estado da arte clássico, ou seja, não baseado em aprendizado de máquina, em ajuste de primitivas geométricas. Apesar de ser classificado por Kaiser et al. (2019) como um método focado apenas em pequenas estruturas, essa informação não é especificada no artigo original, no qual, inclusive, alguns resultados são demonstrados para o ajuste em nuvens com uma quantidade alta de pontos. Contudo, apesar de larga escala não representar necessariamente grandes estruturas, pelo menos o método mostra a capacidade de trabalhar com dados de alta dimensionalidade. Através do paradigma de *Random Sample Consensus*, o Efficient RANSAC busca ajustar cinco tipos de primitivas, são elas: plano, esfera, cilindro, cone e toro. Para isso, os dados de entrada são tanto as posições dos pontos, como também os vetores normais da superfície ao redor de cada ponto. Assim, a cada iteração, um conjunto de amostras é gerado a partir de uma estratégia de amostragem localizada baseada em *Octree*. As amostras são compostas por um conjunto de pontos e um tipo de primitiva, as quais podem ser avaliadas com base numa função de pontuação e, a cada iteração, somente a melhor amostra segue para as próximas. Finalizando, o algoritmo tem bases probabilísticas fortes na amostragem, na análise da qualidade das primitivas e na condição de parada do método.

Em contraponto aos métodos com fundamentação teórica clássica, a revolução ocasionada pelos métodos aprendizado profundo também se mostra muito presente no tratamento de dados 3D. Ahmed et al. (2018) faz uma revisão sobre essas diferentes técnicas, que possibilitaram o grande avanço na direção da resolução de diversos problemas do processamento de dados tridimensionais utilizando aprendizado profundo. Baseado nessas técnicas, o SPFN (Li et al., 2018) e o ParSeNet (Sharma et al., 2020) foram desenvolvidos como métodos de aprendizado profundo que realizam ajuste de primitivas geométricas a nuvens de pontos.

O SPFN foi o primeiro método, baseado em aprendizado profundo e desenvolvido com a intenção de resolver o problema, que teve resultados competitivos. Seu modelo é dividido em diversos módulos diferenciaveis que, quando juntos, podem ser treinados fim-a-fim. Nesse método, a segmentação é feita por uma PointNet++ (Qi et al., 2017) e o seu ajuste de primitivas é feito pela resolução de diversos problemas de mínimos quadrados. Os tipos de primitivas que podem ser ajustados são: plano, esfera, cilindro e cone. Além disso, o método recebe apenas a posição dos pontos na entrada e o número de pontos é tipicamente baixo, na ordem de 10^4 pontos, e há uma quantidade máxima de primitivas que podem ser ajustadas, que é tipicamente 24 primitivas. Seguindo o mesmo fundamento teórico do SPFN, o ParSeNet busca o ajuste de superfícies paramétricas a partir de um modelo de aprendizado profundo. Para isso, foi criado um modelo também modular e totalmente diferenciável para que possa ser treinado fim-a-fim. Nesse caso, o termo utilizado é superfícies paramétricas porque, além das primitivas que o SPFN já realiza ajuste, o ParSeNet também ajusta *b-splines*, que não são primitivas geométricas. O método recebe as posições e as normais na entrada e, como primeira parte do processamento, a segmentação é feita por um *Mean Shift* em um espaço de 128 dimensões gerado por uma rede de *embedding*. Logo após, as primitivas geométricas são ajustadas da mesma forma que no SPFN e as *b-splines* são ajustadas por uma SplineNet (Sagar and Sharma, 2018).

Apesar do fato de que, no contexto de objetos individuais, o ParSeNet é o método com os melhores resultados, isso não necessariamente é verdade no contexto de grandes estruturas. Assim, as diferenças na estratégia adotada pelo SPFN, mesmo que tenha a mesma fundamentação teórica, ainda podem promover a obtenção de resultados superiores aos da ParSeNet. Junto a isso, o Efficient RANSAC, apesar de mais antigo, não tem um contexto de aplicação especificado e pode ter resultados ainda superiores a essas novas técnicas baseadas em aprendizado profundo. Contudo, essa comparação seria mais justa, caso existissem dados de grandes estruturas com *ground truth* para treinar os modelos baseados em aprendizado e comparar os métodos utilizando métricas mais confiáveis. Porém, na falta desses, os modelos utilizados devem ser aqueles treinados para objetos individuais e o método de comparação deve ser mais focado na qualidade visual da segmentação de primitivas.

3. METODOLOGIA

Para cumprir os objetivos propostos, primeiramente deve-se olhar para os dados que serão processados, tanto para a maneira de se obtê-los, quanto para as heterogeneidades das diferentes fontes, o que é feito em 3.1. Logo após, é essencial realizar uma discussão sobre os métodos de pré-processamento, isso é feito em 3.2. Finalizando, os métodos de ajuste de primitivas geométricas a serem utilizados, serão esclarecidos e discutidos em 3.3.

3.1 Dados

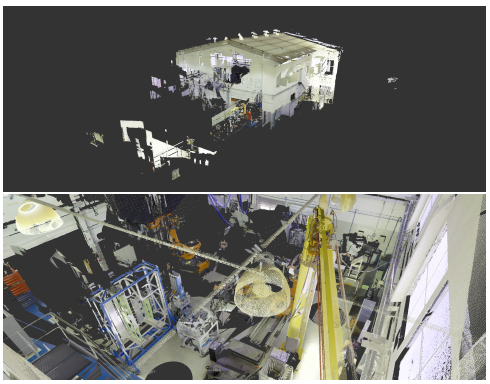

Figura 1. Nuvem de Pontos do *Dataset Scan Real*.

A obtenção de dados 3D geralmente passa pela realização de experimentos controlados de aquisição, os quais tem determinado custo e qualidade do resultado final. Dependendo da técnica de aquisição, coisas como: precisão, resolução, partes oclusas e quantidade de *outliers* variam

muito. Nesse trabalho, um conjunto de dados, capturado através de escâner 3D, de um laboratório do ITA será utilizado. Esse conjunto da dados tem 55 milhões de pontos com informação de posição e cor, é chamado de *Dataset Scan Real* e está demonstrado na Figura 1. A nuvem de pontos disponível no conjunto de dados foi adquirida a partir do registro, em algumas localizações diferentes, das leituras feitas por um escâner Leica RTC360.

3.2 Pré-processamento

O pré-processamento é um passo necessário para a maioria das técnicas que tem como objetivo processar nuvens de pontos. Nesse contexto, temos técnicas de filtragem, técnica de estimativa de normal e técnicas de normalização. Todos os métodos de pré-processamento foram implementados em um código em C++ chamado *large_point_cloud_processing* [1] . Se inspirando nos filtros mais utilizados na literatura de processamento de dados grandes estruturas, demonstrada em 2.1, alguns foram implementados, são eles:

- **Filtro de Corte:** utilizando o filtro *ConditionalRemoval* da PCL (Rusu and Cousins, 2011) é possível estabelecer limites para cada um dos eixos coordenados, assim, todos os pontos fora desse limite parametrizado são deletados da nuvem de pontos.
- **Filtro de *Downsampling*:** utilizando o filtro *VoxelGrid* da PCL é possível estabelecer as dimensões dos *voxels*, ou cubos, que vão discretizar o espaço 3D. Assim, para cada *voxel*, todos os pontos internos ao seu espaço serão substituídos pelo centroide calculado a partir da média das posições, ocasionando a diminuição significativa da quantidade de pontos e o controle da densidade ou resolução da nuvem.
- **Filtro de Remoção de *Outliers*:** utilizando o filtro *StatisticalOutlierRemoval* da PCL, é possível realizar uma análise da estatística de distâncias locais cada ponto. Para isso, a média das distâncias de um ponto a todos os seus vizinhos é computada, e caso essa esteja além de uma distribuição normal com média e desvio padrão definidos pelo usuário, o ponto é descartado.

Como foi visto em 2.2, métodos como o Efficient RANSAC (Schnabel et al., 2007) e o ParSeNet (Sharma et al., 2020) necessitam da informação da normal da superfície ao redor de cada ponto como dado de entrada. Dessa forma, integrando o módulo de pré-processamento, o *NormalEstimationOMP* da PCL foi utilizado para estimar a normal de cada ponto da nuvem.

Métodos baseados em aprendizado, como o SPFN (Li et al., 2018) e o ParSeNet (Sharma et al., 2020) exigem passos de normalização de forma anterior a execução para não direcionar o aprendizado do modelo as características particulares de cada d Toado de treino. Assim, da mesma forma como os dados são normalizados no processo de treino, também devem ser normalizados no processo de teste e validação. Portanto, um módulo de normalização foi incorporado ao pré-processamento com as seguintes funções:

- **Reescala:** todos os pontos são divididos por um valor de reescala para trocar a unidade de medida dos pontos da nuvem.
- **Centralização:** todos os pontos são subtraídos do centroide da nuvem de pontos, fazendo com que a origem do sistema de coordenadas fique no centro geométrico da nuvem.
- **Alinhamento Canônico:** a nuvem de pontos é rotacionada para que o eixo x do sistema de coordenadas aponte na direção de menor variabilidade da nuvem.
- **Adição de Ruído:** visto que os dados simulados, utilizados para treinar, não tem a presença de ruídos inerentes do processo de captura, é possível simular ruído a partir desse módulo. Neste, é aplicado, na direção da normal de cada ponto, ruído que segue distribuição uniforme aleatória com limite definido pelo usuário.
- **Reescala Baseada em Cubo:** um passo de normalização muito importante para o SPFN e o ParSeNet. Neste, todos os pontos são divididos pelo tamanho do maior eixo do sistema de coordenadas, visando fazer com que toda a nuvem caiba em um cubo unitário. Logo após, o módulo do cubo pode ser aumentado multiplicando todos os pontos por algum valor de escala.

3.3 Ajuste de Primitivas Geométricas

Primeiramente, o Efficient RANSAC utilizado foi a implementação disponível na CGAL (Oesau et al., 2020), na qual, os parâmetros descritos no artigo tem seus valores modificáveis, e foi configurada para ajustar: planos, esferas, cilindros e cones. Assim como demonstrado em Schnabel et al. (2007), diferentes nuvens de pontos de entrada necessitam de diferentes parâmetros para ter um ajuste de primitivas geométricas adequado. Dessa forma, a implementação utilizada define esses parâmetros de forma dinâmica, porém, isso não significa que a maneira escolhida para definir gera os melhores resultados possíveis. Logo, para alcançar resultados superiores, os parâmetros devem ser entendidos e estimados com base nas características individuais da nuvem de entrada e do processo de filtragem. Os parâmetros do método são:

- ***probability:*** probabilidade de perder a maior primitiva a cada iteração.
- ***min_points:*** número mínimo de pontos para ser considerado primitiva.
- ***epsilon:*** distância mínima entre uma primitiva e um ponto para que ele seja consideração pertencente a ela.
- ***cluster_epsilon:*** distância utilizada para que dois pontos vizinhos sejam considerados, ou não, adjacentes.
- ***normal_threshold:*** limiar do quanto a normal de um ponto pode diferir da normal da primitiva na projeção daquele ponto.

Depois de executado, a saída do método é: um conjunto de primitivas com tipo e parâmetros, uma nuvem de pontos dividia por cores que representa a segmentação, um conjunto de pontos que não foi atribuído a nenhuma primitiva e a métrica de *P coverage*, que é a divisão da quantidade de pontos ajustados pelo número total de pontos na nuvem de

[1] Repositório no Github: https://github.com/igormaurell/large_point_cloud_processing

entrada. Junto a isso, por se tratar de um método de fundamentação teórica estocástica, seu resultado pode variar um pouco dentro de todas as vezes que for executado, assim, o indicado na literatura é executar o mesmo processo três vezes e utilizar o resultado de maior *P coverage*.

Já os modelos baseados em aprendizado, SPFN e ParSeNet, ambos foram treinados a partir de suas implementações oficiais [2] [3]. Porém, mesmo com todos os passos de normalização realizados corretamente, ambos modelos não consegue realizar ajuste de primitivas geométricas em nuvens de pontos de grandes estruturas com qualidade. Isso se dá ao fato de que os dois modelos são treinados em nuvens com poucos pontos e sempre no contexto de objetos individuais, o que não provoca generalização suficiente para que permaneça funcionando adequadamente mesmo mudando o contexto de aplicação. Para além disso, nenhum dos dois modelos se propõe a enfrentar o problema da larga escala das nuvens, assim como é feito pela RandLA-Net (Hu et al., 2020) no problema de segmentação semântica. Logo, a escalabilidade com relação a tempo de execução e consumo de memória de ambos torna improvável que seja apenas um problema de dados. Assim, além de um conjunto de dados com *ground truth*, novos modelos baseados em aprendizado profundo serão necessários para realizar o ajuste de primitivas geométricas em nuvens de pontos de grandes estruturas de forma adequada.

4. EXPERIMENTOS

4.1 Avaliação e parametrização do Efficient RANSAC

Esse experimento tem como objetivo, avaliar o Efficient RANSAC frente ao conjunto de dados apresentado em 3.1. Contudo, é importante pontuar que os parâmetros do Efficient RANSAC são muito dependentes das características de cada nuvem de entrada e do nível de detalhamento que se planeja alcançar com as primitivas finais. Logo, nesse experimento se planeja analisar como esses parâmetros podem ser modificados, além de demonstrar a melhoria dos resultados visuais da segmentação de primitivas com base nessas alterações. Para isso, as versões com e sem reparametrização serão testadas em um conjunto de dados denso adquirido por um sensor de alta precisão.

Anterior ao ajuste de primitivas geométricas, os dados devem passar pela camada de pré-processamento. Assim, primeiramente, o filtro de corte removeu pontos fora da estrutura principal a partir de limiares obtidos manualmente. Como uma das principais características do Efficient RANSAC é ser robusto a *outliers*, o filtro de remoção não é aplicado nesse experimento. Depois, finalizando, se utilizou uma resolução de $0,125$ $pontos/cm^3$ no filtro de *downsampling*, resultando em uma nuvem com cerca de 3 milhões 207 mil pontos. Nesse contexto, é importante lembrar que a nuvem original desse conjunto de dados tem cerca de 55 milhões de pontos e, é claro, que uma redução de mais de dez vezes na densidade piora a qualidade dos resultados. Assim, é necessário encontrar uma resolução suficiente para representar todos os detalhes contidos na estrutura, sem que o tempo de execução fique muito alto.

[2] Repositório no Github: https://github.com/lingxiaoli94/SPFN
[3] Repositório no Github: https://github.com/Hippogriff/parsenet-codebase

Logo após, as normais foram estimadas utilizando uma região esférica de 5 *cm* ao redor de cada ponto da nuvem já filtrada. Como o método não é baseado em aprendizado, a normalização não é obrigatória, porém a centralização e o alinhamento foram aplicados para padronizar a visualização.

Como discutido em 3.3, cinco parâmetros podem ser alterados para que se tenha uma melhor performance do método nos dados que se planeja realizar o ajuste de primitivas. Contudo, *normal_threshold* tem um valor padrão razoável de 0.9 *rad*, considerando que a técnica de estimativa de normal tem uma alta quantidade de pontos para trabalhar, e que a nuvem de entrada foi capturada por um sensor de alta precisão. Os parâmetros padrões da CGAL são definidos assim como demonstrado na linha 1 da Tabela 1, onde N é o número de pontos da nuvem de entrada e BBD é o tamanho da diagonal das caixas delimitadoras da *Octree* utilizada na amostragem localizada, discutida em 2.2.

Tabela 1. Parâmetros utilizados para o Efficient RANSAC.

	probability	*min_points*	*epsilon (m)*	*cluster_epsilon (m)*
Padrão	0,05	0,01*N	0,01*BBD	0,01*BBD
R1	0,05	4000	0,1	0,069
R2	0,99	4000	0,1	0,069

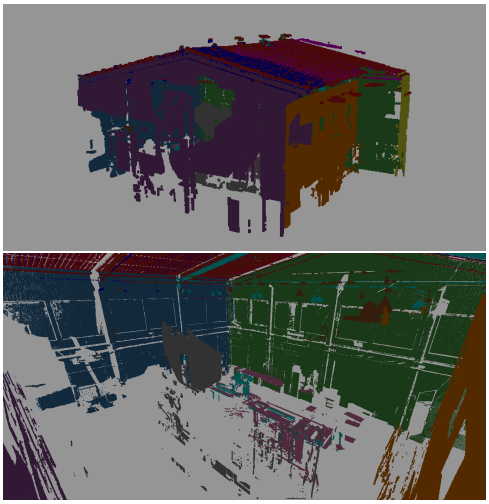

Figura 2. Resultado do Efficient RANSAC Padrão no *Dataset Scan Real*.

Sem um reajuste dos parâmetros, primitivas próximas e de mesmo tipo acabam por se unir. Esse fato é observado em alguns planos paralelos na visão interna da Figura 2, que demonstra os resultados do Efficient RANSAC padrão nos dados já preparados. Logo, temos indícios de que os parâmetros de distância, *epsilon* e *cluster_epsilon*, podem ser definidos de maneira mais ajustada aos resultados que se quer obter e a nuvem que se está trabalhando. Além disso, assim como demonstrado na Tabela 2, apenas 12 primitivas foram segmentadas, o que, claramente, não é o suficiente para descrever toda essa grande estrutura, demonstrando que o valor de 32070 pontos para o *min_points*, calculado utilizando a definição da Tabela 1, é muito alto e faz com que diversas primitivas menores perdidas. Além disso, observa-se que o custo computacional do processo, mesmo com um alto valor para o *min_points*, é muito

elevado na versão padrão, isso se da pelo fato de que o *probrability* utilizado é extremamente baixo. Dessa forma, uma estratégia interessante é aumentar esse parâmetro e diminuir o *min_points*, visando ajustar primitivas menores, sem impossibilitar que o método funcione devido ao custo computacional. Junto a isso, é importante pontuar que, mesmo com esses problemas, o *P coverage* do método padrão é próximo ao das outra versões, isso acontece pelo fato de que muitas primitivas são agregadas as outras, devido aos parâmetros de distância, não gerando danos a essa métrica e, somado a isso, as primitivas menores que são perdidas, tem menor quantidade de pontos, logo, baixa contribuição na formação desse valor.

Para alcançar melhores resultados, novos parâmetros foram escolhidos com base no entendimento do processo, acurácia desejada, testes e conhecimento sobre o pré-processamento aplicado. Logo, duas versões novas do método foram geradas a partir da reparametrização, o R1 e R2, como mostrado na Tabela 1. Para isso, algumas premissas foram utilizadas para modificar os 4 parâmetros, são elas:

- **Pequenos Detalhes das Grandes Estruturas (*min_points*):** apesar dos dados terem uma quantidade alta de pontos, há muitos detalhes a serem primitivados que são compostos por uma quantidade baixa de pontos, dessa forma diminuir o *min_points* possibilita englobar esses detalhes no ajuste final.
- **Diminuição do Custo Computacional (*probability*)**: o parâmetro *probability* está diretamente ligado ao custo computacional do método, visto que esse é a sua condição de parada. Como visto na Tabela 1, o valor padrão é de 0.05, o qual é muito inferior ao valor usado por Schnabel et al. (2007), que é 0.99. Assim, para diminuir o custo computacional e possibilitar a diminuição do *min_points* é necessário o aumento do *probability*, ao custo de encontrar primitivas de menor qualidade.
- **Restrições Impostas pelo Filtro de *Downsampling*(*cluster_epsilon*):** Schnabel et al. (2007) define que o *cluster_epsilon* deve ser: "a menor resolução de amostragem satisfeita em todas as partes do dado". Logo, a resolução do filtro de *downsampling* deve ser levada em consideração, visto que, nesse processo, ocorre uma grande mudança na distância média entre os pontos. Logo, para a resolução de $0,125$ $pontos/cm^3$, os *voxels* utilizados no filtro tem arestas de 2 *cm* e, em um esquema de 26 adjacências por *voxel*, a distância máxima entre dois adjacentes é duas vezes a diagonal do *voxel*. Portanto, temos: $cluster\_epsilon = 2 * 0,02 * \sqrt{3} \equiv 0,069$.
- **Primitivas nem Sempre Passam Sobre Pontos (*epsilon*)**:
  apesar da definição do *cluster_epsilon* ser muito clara sobre a distância máxima entre dois pontos adjacentes, esta não vale da mesma forma para uma adjacência primitiva-ponto. Em casos de pontos muito ruidosos, a primitiva pode estar definida para além da cobertura dos *voxels* vizinhos, o que exige um *epsilon* maior, para que possa tolerar esses casos. Contudo, o quanto ele deve ser superior ao valor do *cluster_epsilon* é a incógnita, por um lado, valores menores gerarão uma segmentação maior da nuvem.

Através das premissas, os parâmetros de distância foram ajustados para valores que fazem mais sentido, dado o conhecimento sobre o processo. Porém, o parâmetro de número de pontos continua sendo dependente do grau de primitivação que se pretende alcançar, dessa forma, R1 varia apenas nesse quesito. Porém, é observado na Tabela 2 um aumento significativo do tempo de processamento, o que não é desejado. Assim, no R2, o *probability* é aumentado, para demonstrar a sua influência no custo computacional e, ao mesmo tempo, a perda na qualidade do ajuste. Além disso, o aumento do *probability* ainda possibilita uma diminuição ainda maior do *min_points*, o que pode ser interessante em alguns casos. Logo, a Tabela 2 demonstra tanto o número de primitivas de cada tipo que foram segmentadas junto do número total, quanto o valor do *P coverage*, que é a métrica que diz qual a proporção de pontos que foi primitivada. Assim, fica claro, como dito anteriormente, que a redução do *min_points* gera um grande aumento no número de primitivas encontradas, bem como, um ganho no *P coverage*. Junto a isso, quanto menores forem o valores de *epsilon* e *cluster_epsilon*, maior será o número primitivas, porém, como esses valores são dependentes do BBD na versão Padrão, não é possível analisar o seu impacto nos resultados da Tabela 2.

Tabela 2. Resultados do Efficient RANSAC no *Dataset Scan Real*. A execução foi feita em uma máquina com 8GB de RAM e um CPU Intel Core i5-9300H.

	Planos	Esferas	Cilindros	Cones	Primitivas	*P Coverage*	Tempo (s)
Padrão	5	0	4	3	12	0,701559	70
R1	**33**	**2**	**37**	**17**	**89**	**0,741984**	**640,62**
R2	24	0	16	5	45	0,627597	27,7

Como comentado anteriormente, sobre-segmentação pode ocorrer quando os parâmetros de distância ou o *min_points* tem valores muito baixos. Analisando a Figura 3, que demonstra os resultados do R1, pode-se perceber novas pequenas primitivas sendo detectadas, principalmente na parte interna, e uma sobre segmentação na região do teto, que provavelmente é decorrente de um valor muito baixo para o *epsilon*. Já na Figura 4, que traz os resultados do R2, os resultados claramente pioram, tanto nas primitivas internas detectadas quanto na qualidade da segmentação das paredes. Porém, esse aumento feito no *probability* possibilita uma diminuição de mais de $10X$ no custo computacional do método.

Com base nos resultados apresentados, é possível perceber a forte influência dos parâmetros no resultado final. Analisando de forma mais especifica, o *min_points* permite um aumento fácil do número de primitivas ajustadas, porém, seu valor deve ser definido com relação a resolução da nuvem de entrada, e não com relação ao número de pontos da mesma, visto que independentemente das dimensões da estrutura que é apresentada na entrada, o nível de detalhes a serem capturados devem ser os mesmos. Já com relação ao *probability*, as figuras 3 e 4 deixam claro a importância de seu ajuste, demonstrando que valores extremamente baixos geram uma segmentação de muito melhor qualidade, ao custo de um tempo de processamento extremamente alto e não-escalável. Dessa forma, o processo de reparametrização mais importante é a relação entre o *min_points* e o *probability*, objetificando sempre, o melhor ajuste com o mínimo de tempo de processamento.

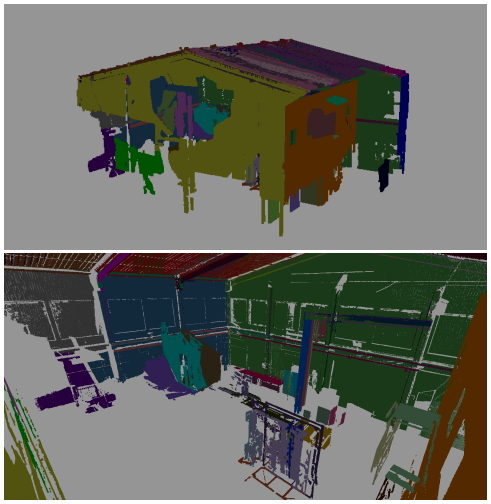

Figura 3. Resultado do Efficient RANSAC R1 no *Dataset Scan Real*.

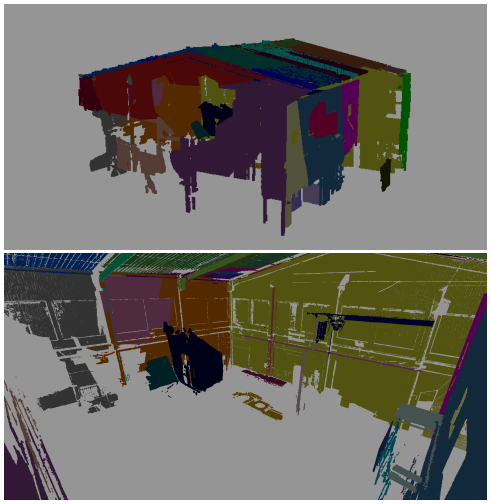

Figura 4. Resultado do Efficient RANSAC R2 no *Dataset Scan Real*.

Ao contrário, os parâmetros de distância como o *epsilon* e o *cluster_epsilon* podem ser definidos de maneira mais analítica, apesar de ainda ser possível fazer leves ajustes em ambos para melhorar os resultados. Portanto, não existe a reparametrização perfeita, em todos os casos há uma relação de perda e ganho. Porém, um conjunto de dados de grandes estruturas com *ground truth* para ajuste de primitivas geométricas possibilitaria um processo de validação cruzada para obtenção de parâmetros mais adequados.

4.2 Validação com Dataset Público

O experimento anterior foi feito em conjunto de dados próprio, porém, com o objetivo de realizar uma validação mais completa o Semantic3D (Hackel et al., 2017) será utilizado. Assim, o Efficient RANSAC com parâmetros mais adequados, será utilizado em todas as nuvens de pontos desse conjunto de dados público.

O conjunto de dados conta com 30 nuvens de pontos reais, de ambientes urbanos e rurais, divididas meio a meio para treino e teste. Assim, as 15 nuvens de pontos de teste são utilizadas para validar o que foi discutido no trabalho. Para isso, primeiramente, as nuvens devem ser filtradas, contudo, nesse caso, apenas o filtro de *downsampling*, para a resolução de $0,125$ $pontos/cm^3$, foi aplicado. Assim, a média dos resultados do processo de filtragem pode ser vista, em número de pontos, na Tabela 3. Seguindo, os processos de estimativa de normal e normalização são feitos da mesma maneira que no experimento anterior. A Figura 5 mostra duas vistas da nuvem *sg27_10* após o pré-processamento.

Tabela 3. Preparação dos dados do Semantic3D.

	Normal (milhões)	Filtrada (milhões)
Média	139,46	10,17

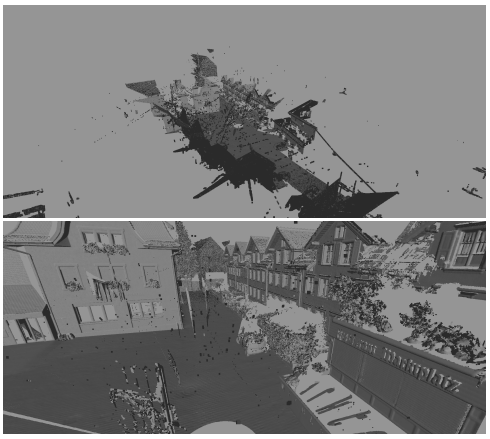

Figura 5. Preparação dos dados do conjunto de dados Semantic3D.

Com a utilização de nuvens de pontos maiores, o cuidado com os parâmetros do Efficient RANSAC tem que ser ainda maior, visto que como comentado no experimento anterior, o ajuste inadequado do *probability* e do *min_points* pode aumentar muito o tempo de processamento. Dessa forma, o *probability* foi ajustado para o valor de $0,9$, na intenção de baixar o custo computacional relativo ao R1 ($probability = 0.05$), e reduzir a sobre-segmentação com relação a R2 ($probability = 0,99$). Além disso, o *min_points* utilizado é de 1000, o que é ainda menor do que os 4000 utilizados para o R2 no experimento anterior. Já com relação aos parâmetros de distância, *epsilon* e *cluster_epsilon* foram ajustados com os mesmo valores de R1 e R2, visto que a resolução utilizada na preparação dos dados é a mesma.

A Tabela 4 mostra a média dos resultados quantitativos no conjunto de dados. Através desses dados, é possível perceber que, apesar do alto tempo de processamento, o método é capaz de ajustar um número significativo de primitivas a nuvens de pontos de grandes estruturas. Além disso, é importante pontuar que todos os dados foram adquiridos em ambientes externos com a presença de muitos detalhes naturais, onde grande parte deles é vegetação, como demonstrado em algumas estruturas de segunda linha das figuras 5 6. Assim, apesar dessas estruturas também serem primitivadas, esse processo não passa de uma aproximação, visto que primitivas geométricas perfeitas são incomuns na natureza. Caso essa aproximação seja um problema, o *min_points* deve ser aumentado consideravelmente, ou filtros de pós processamento podem ser passados.

Tabela 4. Resultados do Efficient RANSAC no Semantic3D. A execução foi feita em uma máquina com 8GB de RAM e um CPU Intel Core i5-9300H.

	Planos	Esferas	Cilindros	Cones	Primitivas	*P Coverage*	Tempo (s)
Média	155,9	1	387,4	275,7	818	0,636441	2141,31

Já para uma análise qualitativa, entender os resultados da Figura 6 é de exímia importância, nela pode-se visualizar um bom ajuste dos grandes planos, que são: chão, telhados, fachadas de casas e muros. Além disso, na visão oferecida na segunda linha, é possível observar alguns pequenos planos, como: janelas, laterais de carro e, até mesmo, roupas no varal. Porém, o baixo *min_points* faz com que, alguns ruídos de leitura mais fortes e pontos de vegetação, sejam segmentados em uma série de pequenas primitivas, das quais grande parte são cilindros e cones, o que justifica seus altos números na Tabela 4.

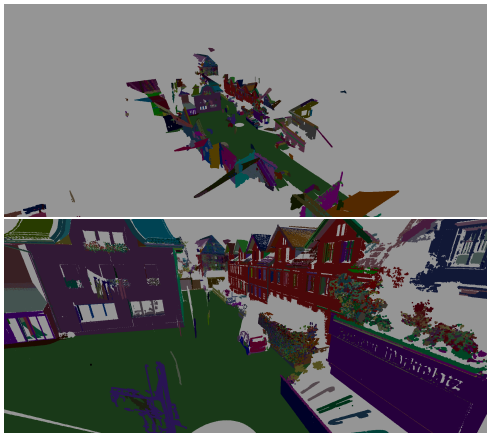

Figura 6. Resultado do Efficient RANSAC no conjunto de dados Semantic3D.

Com base nos resultados apresentados, conclui-se que o Efficient RANSAC é capaz de realizar o ajuste de primitivas geométricas a, até mesmo, conjuntos de dados com uma altíssima quantidade de pontos. Porém, dependendo da finalidade para qual a primitivação está sendo feita, o método pode não alcançar resultados bons o bastante ou pode exigir uma configuração exaustiva dos parâmetros para cada novo dado a ser utilizado. Além disso, em contextos de ambientes externos, é importante que os parâmetros tenham valores mais baixos, principalmente o *probability* e o *min_points*, para que a vegetação não seja primitivada, caso isso seja um problema.

AGRADECIMENTOS

Agradecimento ao ITA pela disponibilização dos dados e ao PRH-ANP 22 pelo financiamento da pesquisa.

REFERÊNCIAS

Ahmed, E., Saint, A., Shabayek, A., Cherenkova, K., Das, R., Gusev, G., Aouada, D., and Ottersten, B. (2018). Deep learning advances on different 3d data representations: A survey. *arXiv preprint arXiv:1808.01462*, 1.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J.D., Schindler, K., and Pollefeys, M. (2017). SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, 91–98.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., and Markham, A. (2020). Randlanet: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11108–11117.

Kaiser, A., Ybanez Zepeda, J.A., and Boubekeur, T. (2019). A survey of simple geometric primitives detection methods for captured 3d data. In *Computer Graphics Forum*, volume 38, 167–196. Wiley Online Library.

Landrieu, L. and Simonovsky, M. (2018). Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4558–4567.

Li, D. and Feng, C. (2019). Primitive fitting using deep geometric segmentation. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 36, 780–787. IAARC Publications.

Li, L., Sung, M., Dubrovina, A., Yi, L., and Guibas, L.J. (2018). Supervised fitting of geometric primitives to 3d point clouds. *CoRR*, abs/1811.08988. URL `http://arxiv.org/abs/1811.08988`.

Oesau, S., Verdie, Y., Jamin, C., Alliez, P., Lafarge, F., Giraudot, S., Hoang, T., and Anisimov, D. (2020). Shape detection. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.1.1 edition. URL `https://doc.cgal.org/5.1.1/Manual/packages.html#PkgShapeDetection`.

Pang, G., Qiu, R., Huang, J., You, S., Neumann, U., et al. (2015). Automatic 3d industrial point cloud classification and modeling. In *SPE Western Regional Meeting*. Society of Petroleum Engineers.

Pătrăucean, V., Armeni, I., Nahangi, M., Yeung, J., Brilakis, I., and Haas, C. (2015). State of research in automatic as-built modelling. *Advanced Engineering Informatics*, 29(2), 162–171.

Qi, C.R., Yi, L., Su, H., and Guibas, L.J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413. URL `http://arxiv.org/abs/1706.02413`.

Rabbani, T., Dijkman, S., van den Heuvel, F., and Vosselman, G. (2007). An integrated approach for modelling and global registration of point clouds. *ISPRS journal of Photogrammetry and Remote Sensing*, 61(6), 355–370.

Rusu, R.B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China.

Sagar, S. and Sharma, A. (2018). Splinenet: B-spline neural network for efficient classification of 3d data. 1–8. doi:10.1145/3293353.3293426.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, 214–226. Wiley Online Library.

Sharma, G., Liu, D., Maji, S., Kalogerakis, E., Chaudhuri, S., and Měch, R. (2020). Parsenet: A parametric surface fitting network for 3d point clouds. In *European Conference on Computer Vision*, 261–276. Springer.