NeuralBF: Neural Bilateral Filtering for Top-down Instance Segmentation on Point Clouds

Weiwei Sun[1,3*] Daniel Rebain[1] Renjie Liao[1] Vladimir Tankovich[3]
Soroosh Yazdani[3] Kwang Moo Yi[1] Andrea Tagliasacchi[2,3]

[1]University of British Columbia [2]Simon Fraser University [3]Google Research

https://neuralbf.github.io

Abstract

We introduce a method for instance proposal generation for 3D point clouds. Existing techniques typically directly regress proposals in a single feed-forward step, leading to inaccurate estimation. We show that this serves as a critical bottleneck, and propose a method based on iterative bilateral filtering with learned kernels. Following the spirit of bilateral filtering, we consider both the deep feature embeddings of each point, as well as their locations in the 3D space. We show via synthetic experiments that our method brings drastic improvements when generating instance proposals for a given point of interest. We further validate our method on the challenging ScanNet benchmark, achieving the best instance segmentation performance amongst the sub-category of top-down methods.

1. Introduction

Instance segmentation is a critical component of semantic 3D understanding, with applications including robotic manipulation [16, 45, 31, 27, 30] and autonomous driving [36, 50, 3, 44, 29]. An essential step of instance segmentation [13, 46, 43, 42, 19] is to generate a set of reliable instance proposals. For natural images, state-of-the-art methods generally follow the *top-down* paradigm [43, 42], where one first *detects* candidate instance proposals and then *prunes* them via non-maximum suppression (NMS). Conversely, *bottom-up* methods [21, 41] learn per-point *embeddings* that are then used to *cluster* points into a disjoint set of proposals.

It is quite surprising that the dominance of top-down methods in natural images (2D) is not reaffirmed when we change our domain to point clouds (3D), where bottom-up methods dominate public leaderboards [40, 1]. While they perform well, bottom-up methods rely heavily on handcrafted heuristics in the clustering step, such as the specification of spatial distance thresholds [19] and average instance sizes [1]. Still, because of the performance gap recent 3D computer vision literature naturally focused on incremental contributions attempting to improve the performance of bottom-up techniques, leaving top-down methods relatively under-investigated. Therefore, one is left to wonder why such a striking difference in approaching 2D vs 3D instance segmentation exists, and whether it is possible to devise a competitive top-down method.

In this work, we argue that a critical bottleneck exists in the proposal generation process for point clouds. Early works follow a similar process as for natural images, where

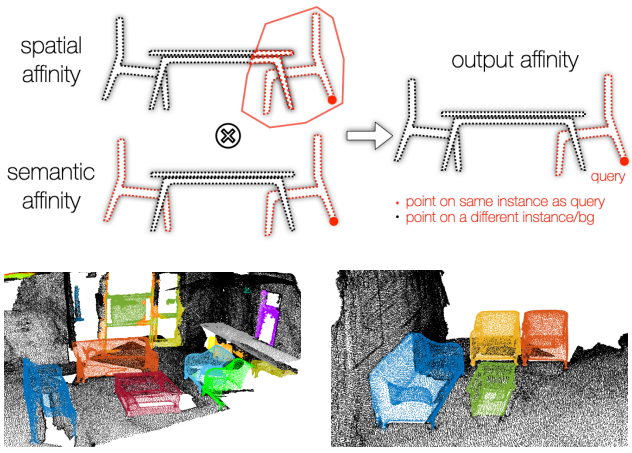

Figure 1. **Teaser –** (top) Given a query point, we define the corresponding instance as the element-wise *product* of feature (points with similar class) and spatial (points with similar localization) affinities; this leads to formulating instance segmentation as a neural bilateral filter. (bottom) Our technique can be applied to large-scale instance segmentation of real-world ScanNet scenes, leading to the best-performance-in-class amongst top-down methods.

*Work partially done during an internship at Google.

bounding boxes are *regressed* [24, 17, 49, 48], but this regression does not generally lead to sufficiently accurate proposals. We ablate these techniques on a simple synthetic dataset, demonstrating how they lead to weak performance (i.e. mAP$<50\%$), while we achieve near-perfect results.

Our top-down technique generates the proposal associated to a given *query* on the input point cloud; see Figure 1. We encode a proposal as an *affinity score*: a [0,1] point-wise labeling of the point cloud that is conditional on a query point (i.e. as the query point changes, the affinity scores changes). We draw inspiration from bottom-up methods [40], and determine two points belong to the same instance if they are "close" to each other in *both* space and semantic class; see Figure 1. Unlike bottom-up methods which would utilize the dual-space affinity to group all points into distinct clusters [19, 40], our proposal generation step identifies the points that are affiliated with a given query. The *semantic* affinity compares the similarity of semantic features in order to separate points from distinct object types. The *spatial* affinity is responsible for bounding the spatial extent of the instance in order to separate semantically-similar objects from each other. Hence, query-conditional affinity can be factorized in two terms, leading us naturally to a *neural bilateral filter* formulation.

In representing *spatial* affinity, we note the predominant representation employed in the 2D image domain axis-aligned *bounding boxes*. And while parameterizing spatial affinity with 3D bounding boxes is possible, this either requires careful handling of SE(3) equivariance [7, 26, 32], or careful prediction of rotations [22]. We avoid this issue by introducing the use of differentiable convex hulls [5] for instance proposal. Note that convex hulls are universal approximators of bounding boxes, capable of implicitly modelling rotated bounding boxes. From a different standpoint, our technique, which models convexes as the level set of a field, can be viewed as the first method that attempts to apply the rapidly growing area of Neural Fields [47] to 3D instance segmentation.

Contributions. We validate the effectiveness of our method on both synthetic and real datasets leading to the following contributions:

- we introduce a simple synthetic dataset that reveals a major bottleneck in instance proposal generation for point clouds;
- we pose the problem of instance segmentation as generating the affinity of points in the cloud to a query point;
- we formulate the computation of affinity as a neural bilateral filter, and demonstrate how an iterative formulation improves its performance;
- we introduce the use of coordinate networks representing convex domains to model the spatial affinity in our neural bilateral filter;
- collectively, these contributions results in a method that tops the leaderboard in point cloud instance segmentation on ScanNet amongst top-down methods.

2. Related works

We briefly describe the recent works on 2D and 3D instance segmentation and review methods on mean shift and bilateral kernel. For a survey on 3D instance segmentation, please refer to [16], and to [12] for 2D instance segmentation.

2D instance segmentation. Top-down methods [13, 46] predict redundant instance proposals for sampled locations in images, which typically requires NMS to remove the overlap. Mask-RCNN [13] detects a set of bounding boxes as the initial instance proposals, and then applies a segmentation module and NMS to output the final mask. PolarMask [46] enhances the performance by using "center priors" – locations close the center of object tends to predict better bounding boxes. SOLO [42, 43] predict instance masks for every location, obviating the need of segmentation module. This is similar to our method where we also output instance masks without segmentation module. Other mainstream instance segmentation pipelines [21, 4] follow the bottom-up paradigm clustering pixels into segments as instance proposals, resulting in performance typically inferior to that of top-down methods.

3D instance segmentation. In contrast to the 2D image domain, bottom-up methods dominate 3D instance segmentation benchmarks. PointGroup [19] first labels points with semantic prediction and center votes, and then cluster points into segments as the instance proposals. Follow-up works [1, 40] further enhance the clustering method in different aspects. HAIS [1] develops hierarchical clustering to have better instance proposals. SoftGroup [40] proposes to group points using soft semantic scores and introduces a hybrid top-down/bottom-up technique via a proposal refinement module. While bottom-up methods rely on the heuristics such as object sizes and distance threshold, top-down methods largely lag in performance. Top-down methods [48, 49] rely on precise bounding box prediction as the initial instance proposal. In more detail, 3DBoNet [48] directly predicts a fixed set of 3D bounding boxes, while GSPN [49] proposes a synthesis-and-analysis strategy to predict better bounding boxes.

Neural Bilateral Filtering. The idea of combining bilateral filtering with neural networks has been mostly in the context of filtering and enhancing natural images [18, 10, 28]. However, to the best of our knowledge, learned bilateral filtering has not been applied to the context of 3D point clouds, although classical point cloud processing layers for point clouds do exist [8].

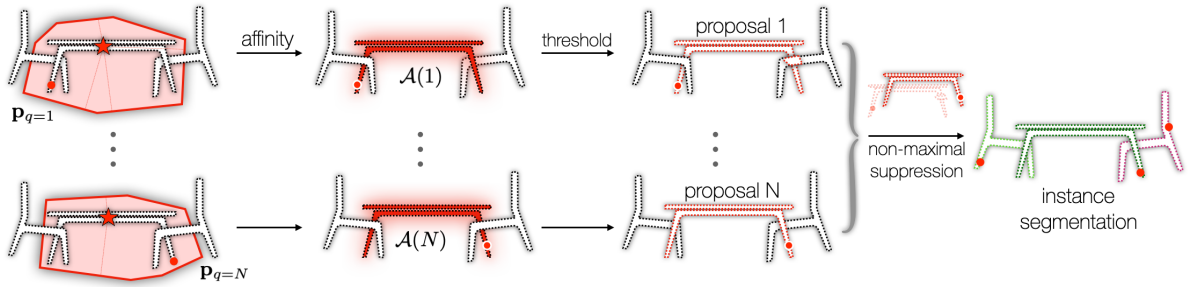

Figure 2. **Overview –** (a) Given a query (⊙), we regress the bounding hull of the corresponding instance; (b) Together with semantic segmentation, this defines an affinity function on the entire point cloud; (c) This affinity can be threshold to generate a candidate instance proposal; (d) Instance proposals are then grouped by non-maximal suppression to generate the scene's instance segmentation.

3. Method – Figure 2

Given a point cloud of N points in D-dimensional space $\mathbf{P}=\{\mathbf{p}_n\}$ and corresponding C-dimensional features $\mathbf{F}=\{\mathbf{f}_n\}=\mathcal{F}(\mathbf{P};\boldsymbol{\theta}_\mathcal{F})$, computed by a deep learning backbone with learnable parameters $\boldsymbol{\theta}_\mathcal{F}$, we generate instance proposals by regressing the bounding volume (i.e. a convex hull in $\mathbb{R}^D$) corresponding to the instance of a query point $(\mathbf{p}_q, \mathbf{f}_q)$, where $q\sim[1, N]$ is the index of query. Together with segmentation features, bounding volumes imply an affinity $\mathbf{A}\in\mathbb{R}^N$ between the query $(\mathbf{p}_q, \mathbf{f}_q)$ and the whole point cloud, which can be thresholded to generate an instance proposal (Section 3.1). These instance proposals are then aggregated by classical non-maximum suppression (NMS) to generate the desired instance segmentation.

3.1. Affinity definition

As illustrated in Figure 1, the affinity of points in the point cloud to a query $(\mathbf{p}_q, \mathbf{f}_q)$ can be intuitively defined as the *element-wise product* of two affinities:

- **Affinity in feature space**: whether a point in the point cloud belongs to the same class as the query;
- **Affinity in geometric space**: whether a point in the point cloud belongs to the same spatial region as the query.

More formally, we define our affinity function $\mathcal{A}(q)$ as:

$$\mathcal{A}(q) = \mathcal{A}_\mathbf{p}(q) \odot \mathcal{A}_\mathbf{f}(q), \tag{1}$$

$$\mathcal{A}_\mathbf{f}(q)[n] = \exp(-\tau_\mathbf{f} \cdot \mathcal{K}_\mathbf{f}(q, n)), \tag{2}$$

$$\mathcal{A}_\mathbf{p}(q)[n] = \exp(-\tau_\mathbf{p} \cdot \mathcal{K}_\mathbf{p}(q, n)), \tag{3}$$

where $\odot$ is the element-wise product, $[n]$ indexes the n-th element of the array, and τ are hyperparameters controlling the bandwidth of the kernels. We can then learn the parameters of kernels $\mathcal{K}_\mathbf{f}$ and $\mathcal{K}_\mathbf{p}$, whose internals are provided in what follows, by directly attempting to reproduce the target affinity given a randomly drawn query point:

$$\mathbb{E}_{q\sim[1,N]} \left\|\mathcal{A}(q) - \mathcal{A}^{gt}(q)\right\|_1^2. \tag{4}$$

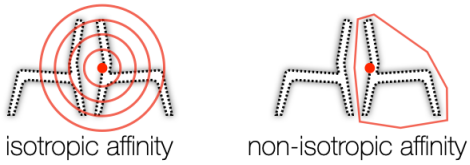

Figure 3. **Spatial similarity –** The semantic feature is uninformative in separating the two instances: (left) an isotropic affinity kernel w.r.t. the query point would mistakenly assign points on the left instance to the right one, regardless of bandwidth choice; (right) a non-isotropic kernel does not suffer this shortcoming.

$\mathcal{K}_\mathbf{f}$ **– semantic similarity**. We measure whether two points have similar semantic classes via:

$$\mathcal{K}_\mathbf{f}(q, n) = \|\phi(\mathbf{f}_q; \boldsymbol{\theta}_\phi) - \phi(\mathbf{f}_n; \boldsymbol{\theta}_\phi)\|_2^2. \tag{5}$$

where $\phi(\cdot; \boldsymbol{\theta}_\phi)$ is a small projection layer with parameters $\boldsymbol{\theta}_\phi$ that extracts *semantic similarity features* from the (task agnostic) backbone features $\mathbf{f}$.

$\mathcal{K}_\mathbf{p}$ **– spatial similarity**. While classical bilateral filtering employs *isotropic* kernels to account for spatial similarity (i.e. gaussian with tunable bandwidth), this is not optimal for instance segmentation. We illustrate our intuition in Figure 3, where the proximity of two objects of the *same* semantic class implies that no isotropic kernel centered at the query point could be used to isolate the desired instance. We achieve this while retaining *commutative symmetry*[1]:

$$\mathcal{K}_\mathbf{p}(q, n) = \mathcal{C}(\mathbf{p}_n - \mathbf{p}_q; \psi(\mathbf{f}_q; \boldsymbol{\theta}_\psi)) + \mathcal{C}(\mathbf{p}_q - \mathbf{p}_n; \psi(\mathbf{f}_n; \boldsymbol{\theta}_\psi)), \tag{6}$$

where $\psi(\cdot; \boldsymbol{\theta}_\psi)$ is a small projection layer with parameters $\boldsymbol{\theta}_\psi$ that extracts *spatial similarity features* from the generic backbone features $\mathbf{f}$. This leads us to the question of how to design the function $\mathcal{C}(\mathbf{x}; \mathbf{f})$. One potential solution is to define $\mathcal{C}$ as a *coordinate neural network* [47] whose shape is described by the feature $\mathbf{f}$, and that is evaluated at location

[1] Affinity ought to be symmetric, because if point $\mathbf{p}_n$ belongs to the same instance as $\mathbf{p}_q$ then we should ideally have $\mathcal{K}(q, n) \equiv \mathcal{K}(n, q)$.

$\mathbf{x}$. We opt to model $\mathcal{C}$ with CvxNet [5] – a coordinate neural network that is a universal approximator of convex domains. This choice is particularly well-suited, because:

- convex hulls are a topologically equivalent, yet more flexible and detailed replacement for 2D/3D bounding boxes, the core representation employed in 2D/3D object detection/instance segmentation, making them a particularly well-suited choice for our problem;
- compared to coordinate neural networks implemented as multi-layer perceptrons, CvxNet-like hyper-networks generate very small output networks and are more memory efficient, allowing us to use larger mini-batch sizes leading to faster training.

We further detail the design of $\mathcal{C}$ in Section 3.2, which will fulfill the following base property with respect to the convex domain specified by the feature $\mathbf{f}$:

$$\mathcal{C}(\mathbf{x};\mathbf{f}) \begin{cases} = 0 & \text{if } \mathbf{x} \text{ inside convex defined by } \mathbf{f}, \\ > 0 & \text{otherwise } (\approx \text{boundary distance}). \end{cases} \tag{7}$$

3.2. Convex parameterization $\mathcal{C}(\mathbf{x};\mathbf{f})$

From $\mathbf{f}$, via a fully connected decoder (with *shared* parameters $\boldsymbol{\theta}_{\mathcal{D}}$), we derive the normals $\{\mathbf{n}_h{\in}\mathbb{R}^D \mid \|\mathbf{n}_h\|_2{=}1, h{\sim}[1,H]\}$ specifying the H half-space orientations, and their distances $\{d_h{\in}\mathbb{R}^+\}$ from the origin $\mathbf{o}{\in}\mathbb{R}^D$:

$$\mathbf{o}, \{\mathbf{n}_h\}, \{d_h\} = \mathcal{D}(\mathbf{f};\boldsymbol{\theta}_{\mathcal{D}}), \tag{8}$$

and define the distance of $\mathbf{x}$ from the $h{-}$th hyperplane as:

$$\mathcal{H}_h(\mathbf{x}) = \mathbf{n}_h \cdot (\mathbf{x} + \mathbf{o}) + d_h, \tag{9}$$

which can be assembled into an (approximate, see [5]) distance function from the convex polytope as:

$$\Phi(\mathbf{x};\mathbf{f}) = \max_h\{\mathcal{H}_h(\mathbf{x})\}, \tag{10}$$

finally leading to our convex spatial proximity:

$$\mathcal{C}(\mathbf{x};\mathbf{f}) = \max(\Phi(\mathbf{x};\mathbf{f}), 0), \tag{11}$$

which can then, if necessary, be converted as an indicator function (i.e. occupancy) for a convex [5]:

$$\mathcal{O}(\mathbf{x};\mathbf{f}) = \text{sigmoid}(-\Phi(\mathbf{x};\mathbf{f})). \tag{12}$$

3.3. Neural Bilateral Filter – Figure 4

The resemblance of (1) to the product of kernels in *bilateral filtering* [14, 39] inspired us to investigate the use of *iterative* inference. Specifically, given a query, we advect *both* query position and features, where the advection weights are given by the affinity definition from (1). Note that the point cloud $\mathbf{P}$ and corresponding features $\mathbf{F}$ remain *unchanged*, only the query is affected. With a slight abuse of notation, we denote $\mathbf{A}^{(t)}(q)$ as the affinity at t-th iteration for the query q. The outcome is simply that, rather than attention $\mathbf{A}^{(0)}(q){=}\mathcal{A}(q)$ in downstream processing, $\mathbf{A}^{(T)}(q)$ will instead be used.

Input:
  $q \in [1, N]$ ▷ query index
  $\mathbf{P} \in \mathbb{R}^{N\times D}$ ▷ (const) cloud positions
  $\mathbf{F} \in \mathbb{R}^{N\times C}$ ▷ (const) cloud features
function NEURALBILATERALFILTER
  $\mathbf{p}^{(0)} = \mathbf{p}_q = \mathbf{P}[q]$
  $\mathbf{f}^{(0)} = \mathbf{f}_q = \mathbf{F}[q]$
  for $t = 1, \ldots, T$ **do**
    $\mathbf{A}^{(t-1)}(q) = \mathcal{A}(\mathbf{p}^{(t-1)}, \mathbf{f}^{(t-1)})$
    $\mathbf{A}^{(t-1)}(q) = \mathbf{A}^{(t-1)}(q) / \|\mathbf{A}^{(t-1)}(q)\|_1$
    $\mathbf{p}^{(t)} = \mathbf{A}^{(t-1)}(q) \cdot \mathbf{P}$
    $\mathbf{f}^{(t)} = \mathbf{A}^{(t-1)}(q) \cdot \mathbf{F}$
  end for
  Return $\mathbf{A}^{(T)}(q)$
end function

Figure 4. **Neural Bilateral Filter (Section 3.3)–** Given a query (position and feature) we iteratively apply the learned filters to advect the query point position and features. Ultimately, the final attention $\mathbf{A}^{(T)}(q)$ is used for downstream tasks.

3.4. Training

To train our network, we optimize:

$$\underset{\boldsymbol{\theta}_\phi, \boldsymbol{\theta}_\psi, \boldsymbol{\theta}_{\mathcal{D}}, \boldsymbol{\theta}_{\mathcal{F}}}{\arg\min} \quad \mathcal{L}_{\text{affinity}} + \mathcal{L}_{\text{sem}} + \mathcal{L}_{\text{poly}} + \mathcal{L}_{\text{shift}}. \tag{13}$$

Of these losses $\mathcal{L}_{\text{affinity}}$ is our core loss, while the rest provide "skip-connection" supervision to the network to facilitate learning. Since our method performs iterative inference, we *discount* ($\alpha{=}0.8$) contributions of later iterations – we found empirically that focusing on later iterations cause training instability:

$$\mathcal{L}_{\text{affinity}} = \mathbb{E}_{q\sim[1,N]} \sum_{t=1}^{T} \alpha^t \left\| \mathbf{A}^{(t)}(q) - \mathbf{A}^{gt}(q) \right\|_1^2 \tag{14}$$

Semantic supervision. To encourage the semantic features in (5) to represent *only* the semantic similarity, we inject semantic information by mapping intermediate point-wise backbone feature to semantic logits (see Figure 5), and supervising with ground truth labels $\mathbf{s}_q^{gt}$:

$$\mathcal{L}_{\text{sem}} = \mathbb{E}_{q\sim[1,N]} \left[\text{CrossEntropy}(\mathbf{s}_q, \mathbf{s}_q^{gt})\right] \tag{15}$$

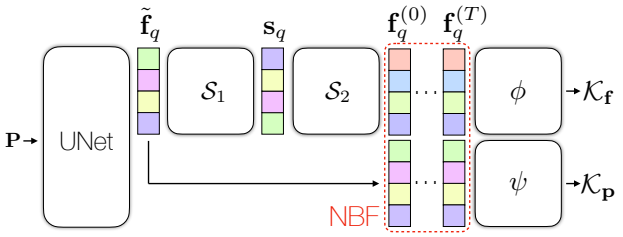

Figure 5. **Architecture –** The point cloud is processed by a backbone to produce $\mathbf{f}_q{=}\mathbf{f}_q^{(0)}$, which is then processed by a neural bilateral filter by our kernels $\mathcal{K}_*$. We supervise $\mathbf{s}_q$ via ground truth semantic classification labels, where $\mathcal{S}_1$ is a 2-layer MLP, and $\mathcal{S}_2$ is a linear layer.

Instance centroid supervision. To minimize the learning complexity of $\mathcal{D}$, we incentivize predicted convex hulls to be expressed with respect to a *stable* coordinate frame[2]. We employ the ground truth instance origin $\mathbf{c}_q^{gt}$ and supervise the predicted origin relative offset:

$$\mathcal{L}_{\text{shift}} = \mathbb{E}_{q\sim[1,N]} \sum_{t=1}^{T} \alpha^t \left\| (\mathbf{p}_q + \mathbf{o}_q^{(t)}) - \mathbf{c}_q^{gt} \right\|_1 \qquad (16)$$

where the offset $\mathbf{o}_q^{(t)}$ is computed from $\mathcal{D}(\mathbf{f}_q^{(t)})$.

Convex occupancy supervision. Note the affinity supervision in (14) only penalizes points that are incorrectly marked as *outside* the convex hull. To correct this, let $\mathcal{O}_q^{\text{gt}}(p)$ be the ground truth occupancy of point p belonging to the convex hull of query q, we then penalize:

$$\mathcal{L}_{\text{poly}} = \mathbb{E}_q \mathbb{E}_n \sum_{t=1}^{T} \omega_{q,n} \alpha^t \| \mathcal{O}(\mathbf{p}_n - \mathbf{p}_q; \psi(\mathbf{f}_q^{(t)})) - \mathcal{O}_q^{\text{gt}}(p) \|_2^2 \qquad (17)$$

where $\omega_{q,n}$ is a term to control class imbalance: if the instance corresponding to q has Q points and the scene has N points, then $\omega_{q,n}{=}1/Q$ if point n belongs to the instance, and $\omega_{q,n}{=}1/(N{-}Q)$ otherwise.

3.5. Implementation details

We briefly discuss the core implementation details.

Network architecture. For the backbone we utilize the U-Net-like backbone in [19, 1] which is implemented with sparse convolution [11]. We set the dimension C of the backbone feature $\mathbf{f}$ to 32 as in [1].

The projection layers $\phi(\cdot;\boldsymbol{\theta}_\phi)$ in (5) and $\psi(\cdot;\boldsymbol{\theta}_\psi)$ in (6). The layer ϕ is composed of semantic layers ($\mathcal{S}_1$) and an embedding layer ($\mathcal{S}_2$). The semantic layers convert the backbone features into semantic scores with a two-layer MLP with 32 neurons and then outputs the semantic feature with a linear layer of C neurons. Note that during the iterative process, we directly update the query's semantic feature without re-using the semantic branch. The embedding layer is a linear layer of C neurons. For ψ we use a small projection layer and rely on the $\mathcal{D}$ for reasoning about the 3D convex polytopes. Specifically, we use a simple identity mapping layer as the ψ, which we found to be good enough.

The polytope network $\mathcal{D}(\cdot;\boldsymbol{\theta}_\mathcal{D})$ in (6). The network $\mathcal{D}$ consists of two MLP blocks. The first block – a two-layer ReLU-activated MLP with 128 neurons – predicts $\mathbf{o}$ from the query feature $\mathbf{f}$ and a residual to $\mathbf{f}$. We then add the residual to $\mathbf{f}$ and utilize the second block – a three-layer ReLU-activated MLP with 128 neurons – to predict normals and offsets. For predicting the plane offset d_h, we use the strategy from [9] and discretize the offset values into 32 equal bins in the range $[0, 8]$ meters, and obtain the predicted value via the weighted sum of classification scores. We represent each 3D convex polytope with twelve planes, striking a good balance between precision and computational load, which linearly increases with the number of planes. Finally, $\tau_\mathbf{F}{=}1$ and $\tau_\mathbf{P}{=}50$ in (2) and (3).

Forming the training batch. While possible, training with all points in the point cloud is impractical and inefficient, as it would create a quadratic increase in both memory and computation. We use a batch of four scenes, and randomly sample 32 random points/scene during training to form a single training sample. Our algorithm has the computation and space complexity linear to the number of sampled queries. We further set the number of mean shift iterations $T{=}2$, which we ablate in Sec. 4.3.

Training. As in RAFT [37], we detach the gradient flow between different iterations to stabilize training. We use the Adam optimizer [20] with cosine annealing for the learning rate [25], with 0.001 as the initial learning rate. We further follow standard data augmentation/voxelization schemes of existing instance segmentation methods [1]. Coefficients for all loss terms are set as one.

Non-maximum suppression. To obtain the final instance segmentation results for the ScanNet dataset we use standard non-maximum suppression [43, 35] to remove redundant proposals. In more details, we visit a queue of input candidate proposals in confidence score order; see Sec. 4.2. For each candidate proposal, we compute the IoU with all other candidates and merge/prune those that have IoU higher than 0.25.

4. Results

In our results section, we:

- Sec. 4.1 – validate our method in a controlled *synthetic* setup, where we show that current proposal generation methods have limited effectiveness;

[2]If two points a and b belong to the same instance, then the predicted convex origin $\mathbf{o}_a \equiv \mathbf{o}_b$, and the same half-space configuration can be used for all queries within an instance; Note this is similar to the coordinate frame normalization in NASA [6].

	Line segment			*Circle*			*Average*		
	mAP	AP_{50}	AP_{25}	mAP	AP_{50}	AP_{25}	mAP	AP_{50}	AP_{25}
BBox	$46.4_{\pm 1.1}$	$67.7_{\pm 1.9}$	$69.8_{\pm 1.1}$	$21.2_{\pm 1.4}$	$54.7_{\pm 2.3}$	$90.6_{\pm 0.3}$	$33.8_{\pm 0.9}$	$61.2_{\pm 1.7}$	$80.2_{\pm 0.7}$
BBox w/ center	$54.1_{\pm 1.6}$	$77.9_{\pm 1.5}$	$80.4_{\pm 1.2}$	$28.0_{\pm 0.8}$	$64.0_{\pm 0.7}$	$89.2_{\pm 0.7}$	$41.0_{\pm 1.0}$	$71.0_{\pm 0.7}$	$84.8_{\pm 0.8}$
BBox + GT filtering	$53.9_{\pm 1.4}$	$68.2_{\pm 1.6}$	$69.0_{\pm 1.1}$	$31.9_{\pm 1.9}$	$71.1_{\pm 1.7}$	$91.7_{\pm 0.5}$	$42.9_{\pm 1.2}$	$69.7_{\pm 1.2}$	$80.3_{\pm 0.5}$
BBox w/ center + GT filtering	$65.3_{\pm 1.7}$	$79.3_{\pm 1.5}$	$80.1_{\pm 1.5}$	$41.4_{\pm 1.1}$	$75.4_{\pm 1.4}$	$90.3_{\pm 0.5}$	$53.3_{\pm 1.1}$	$77.3_{\pm 0.6}$	$85.2_{\pm 0.9}$
Ours	$\mathbf{95.9}_{\pm 0.3}$	$\mathbf{97.6}_{\pm 0.4}$	$\mathbf{97.9}_{\pm 0.3}$	$\mathbf{98.2}_{\pm 0.5}$	$\mathbf{98.9}_{\pm 0.3}$	$\mathbf{99.3}_{\pm 0.3}$	$\mathbf{97.1}_{\pm 0.2}$	$\mathbf{98.3}_{\pm 0.2}$	$\mathbf{98.6}_{\pm 0.1}$

Table 1. **Query-conditioned instance proposal generation** – we randomly sample a single query for each instance and generate the non-overlapped proposals. We report the mean and standard deviation of average precision by running the evaluation pipeline five times.

- Sec. 4.2 – demonstrate the potential of our method in a more complex instance segmentation pipeline on the *real-world* ScanNet dataset [2];
- Sec. 4.3 – perform an *ablation* study.

4.1. Synthetic dataset

We create a 2D synthetic dataset composed of lines, circles, and random noise; see Fig. 6. For each scene, we randomly place 16 primitives sampled from a large pool (10k in total) of randomly generated line segments and circles in a 2D space. We sample 4096 points for foreground instances and 512 points for the background noise. To keep a similar point density for instances of different sizes, we make the number of points for each instance proportional to the length of the primitive instance. We generate these scenes on-the-fly while training and keep 100 scenes for testing. We limit the 2D coordinates to be within $[-4, 4]$ to match the typical size of ScanNet scenes, allowing us to reuse the same backbone across both synthetic and real scenes.

Metrics. With the dataset, to show that instance proposals are a bottleneck, we are interested in their *direct* evaluation without any downstream Non-Maximum Suppression (NMS) heuristic. We randomly select a *single* point for each instance in the point cloud and measure the quality of the generated proposal for the selected point. Once the proposals are provided, we use the standard metrics used in the ScanNet benchmark [2]—AP_{50} and AP_{25}, which are the accuracy computed with the intersection-over-union (IoU) threshold of 50% and 25%, respectively, and mAP, which is the average AP over different thresholds ranging from 50% to 95% with the step size of 5%.

Baselines. A commonly-used baseline is to directly predict the bounding box for each instance, within which post-processing is applied [24, 17, 49, 48]. To do so, similarly to GICN [24], we train a 2-layer MLP that predicts the bounding box, parameterized by its two corners relative to the query. We further compare against VoteNet [33], where one first regresses a spatial offset given a query point and then regresses the bounding box corners relative to the offset. For these baselines, the bounding box often contains points from noise or other classes (lines vs circle), so we utilize the semantic predictions from the backbone to filter those points out of each instance proposal. Clearly, this would not perfectly filter out cases where the same class instances overlap; hence, we further propose an oracle baseline which uses *ground-truth* semantic and instance labels as an oracle for filtering, thus emulating an ideal post-processing step for the methods based on bounding boxes. We train with 10k iterations, which is enough for all methods to converge on this simple dataset.

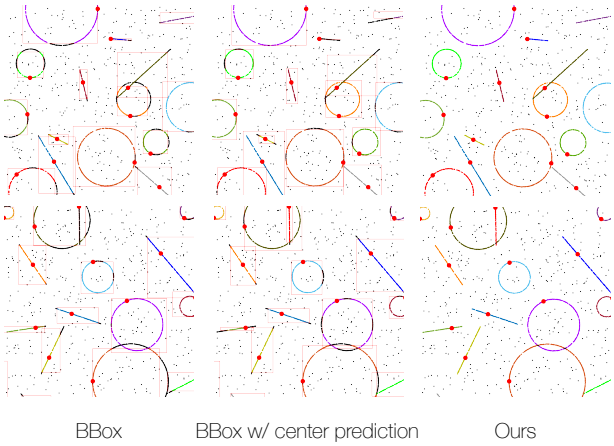

Figure 6. **Qualitative/Synthetic –** Our method generates nearly perfect query-conditioned instance proposals while the baseline is limited by the noisy bounding box. Note the red large points are the sampled queries. We color the points detected by different instance proposals. And the black points are the background points or the points detected by more than two instance proposals.

Results – Tab. 1 and Fig. 6. Our method outperforms the baselines by a significant margin. Despite the success of bounding box proposals in 2D images, these method achieves a surprisingly low performance on this simple synthetic dataset, even when ground-truth filtering is employed. On the other hand, our method delivers near-perfect results, as one would expect for such a simple dataset. For the baselines, as shown in the examples in Fig. 6, we find that many of the proposals are slightly off, with some being *completely* off. While small errors in bounding box position/size are not critical for detection in 2D images, they can be catastrophic for point clouds sampled from the object surface near the bounding box exterior, where a small mis-

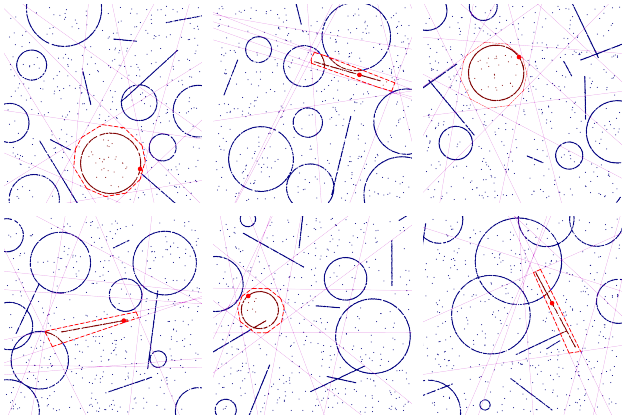

Figure 7. **Visualizing the spatial kernels –** Our method learns the effective convex hulls that act as the tight bounding box of the target instance. For each convex hull, the magenta lines are the learned half-planes. The red polygon is the intersection between half-planes. Points are colored with spatial similarity where red means larger similarity while blue means smaller similarity.

	Methods	Validation			Test		
		mAP	AP_{50}	AP_{25}	mAP	AP_{50}	AP_{25}
Bottom-up	PointGroup [19]	34.8	56.7	71.3	40.7	63.6	77.8
	SSTNet [23]	49.4	64.3	74.0	50.6	69.8	78.9
	HAIS [1]	43.5	64.4	75.6	45.7	69.9	80.3
Mix	Dyco3D [15]	35.4	57.6	72.9	39.5	64.1	76.1
	SoftGroup [40]	–	67.6	78.9	50.4	76.1	86.5
Top-down	3D-SIS [17]	–	18.7	35.7	16.1	38.2	55.8
	GSPN [49]	19.3	37.8	53.4	–	30.6	–
	3D-Bonet [48]	–	–	–	25.3	48.8	68.7
	Ours	**36.0**	**55.5**	**71.1**	**35.3**	**55.5**	**71.8**

Table 2. **Quantitative/ScanNetV2 –** instance segmentation benchmark; our method provides the top performance for the top-down category. For looser thresholds our method performs slightly worse, which may be improved with advanced post-processing.

alignment could remove entire sections of geometry. For example, in Fig. 6 top-left, the bottom right circle is detected with a bounding box that would be considered accurate should one consider only the bounding box, but the majority of the point cloud points for this circle lie outside, as the box is slightly smaller than the actual circle.

Visualizing the spatial kernels – Fig. 7. We visualize the learned spatial kernel. As shown, the learned spatial kernel forms a polytope that tightly bounds the instance in question as desired. These learned kernels enable our method to easily separate different instances spatially, even without considering semantics. Such easy separation would not be possible, for example with standard Euclidean distance as points far away from each other on the line or on the circle would be confused with other nearby points.

4.2. Instance segmentation on ScanNetV2

The ScanNetV2 [2] dataset consists of 1613 scenes in total with 1201, 312, and 100 scenes dedicated for training, validation, and testing, respectively. We use the standard evaluation pipeline and report the standard metrics, the same ones as the ones used for the 2D synthetic data. To evaluate our method for *top-down* instance segmentation pipeline for point clouds, we introduce basic post-processing steps that are commonly used in the literature [38, 43, 42, 24], as well as a scoring function to provide confidence scores for each instance proposal, as required by the benchmark protocol. Notably, our post-processing steps are relatively simple compared to tricks like "matrix NMS" [42] and query sampling using "center priors" [24, 38]. Specifically, we first segment out all background points using $\phi(\mathbf{f}_i)$ in (5). We then sample 256 query points from the predicted foreground points and generate 256 instance proposals. When sampling queries, we apply farthest point sampling [34] to ensure maximum coverage. We then remove redundant instance proposals by applying Non-Maximum Suppression (NMS) to instance proposals with an IoU threshold of 30%. We train the entire pipeline end-to-end for 500 epochs as in [1].

Confidence scores. As the benchmark protocol requires instance proposals to have associated confidence scores, we provide a confidence score for each proposal based on both the semantic segmentation score (provided by $\mathbf{s}_q$) and an MLP that is trained to regress the IoU of each proposal with respect to ground-truth. Specifically, we train a two-layer MLP with an ℓ_1 loss for the IoU. The final confidence score for each proposal is computed by multiplying the regressed IoU value and the average semantic segmentation confidence. We also use these confidence scores and NMS to filter our background points, which typically have low foreground semantic confidence.

Dropping low-confidence proposals. In addition to the above, we drop proposals that have low confidence values (i.e. proposals with semantic confidence lower than 0.1, or with estimated IoU less than 0.2). Furthermore, we drop proposals that have different predicted labels for the proposal and the query point. These proposals are from points that are often located where two different instances of different classes meet, and hence are unreliable.

State-of-the-art comparisons – Table 2 and Fig. 8. Our method shows promising results compared to the state-of-the-art. Among purely top-down methods, our method achieves top performance validating the effectiveness of our instance proposals generation. We leave further improvement via better post-processing steps for future work. While our method performs worse than the most recent bottom-up methods or hybrid methods [40], we note that these are methods heavily fine-tuned to achieve SOTA benchmark results, whereas ours is not. Note that our top-down method beats the leading bottom-up method that was SOTA around CVPR 2020 [19]. Considering that top-down methods have intriguing properties (*e.g.*, their dominant performance in image benchmarks [43, 42] and better gen-

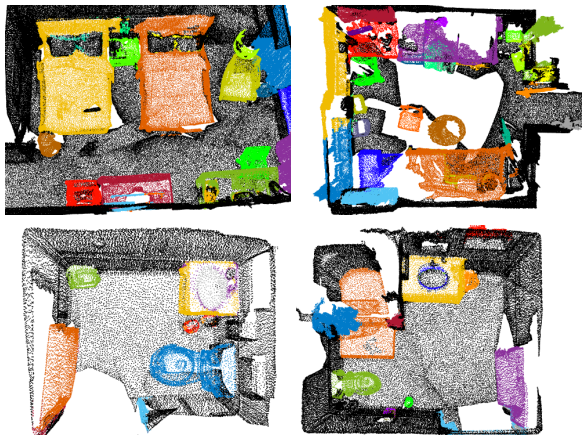

Figure 8. **Qualitative/ScanNet –** Instance segmentation results on test set.

num_iter	1	2	3	4
mAP	$94.46_{\pm 0.55}$	$\mathbf{96.80}_{\pm 0.35}$	$96.54_{\pm 0.40}$	$96.18_{\pm 0.68}$

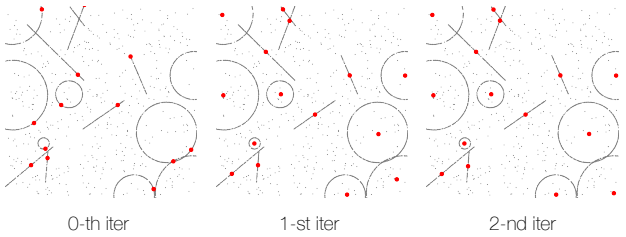

Figure 9. **Ablation: number of iterations** T **–** (top) We report the average and standard deviation of mAP by repeating the experiment 5 times. (bottom) Our algorithm shifts the queries (red points) to the centroid after a small number of iterations.

eralization ability), and are worthwhile to explore further, we believe our work provides progress for instance segmentation on point clouds.

4.3. Ablations

Number of iterations – Fig. 9. Our algorithm is capable to shift query points to the center of each instance in just two iterations. This leads to queries from the same instance to share a similar coordinate frame, leading to a reduction of representation complexity as noted in NASA [6]. This is beneficial, as a smaller number of iterations reduces the GPU memory load of training. Training with more than two iterations seems to simply cause training to become less stable and introduces slight performance degradation.

Losses – Fig. 10. With the proposed regularizers, our algorithm learns tighter instance polytopes (w/ $\mathcal{L}_{\text{poly}}$ and $\mathcal{L}_{\text{offset}}$) and semantic similarity (w/ $\mathcal{L}_{\text{sem}}$), leading to significantly improved performance. Note that, since we evaluate AP for each the semantic category, we provide ground-truth semantic label for the models trained without semantic prediction (i.e., w/o $\mathcal{L}_{\text{sem}}$), Finally, note also that even without $\mathcal{L}_{\text{poly}}$ and $\mathcal{L}_{\text{offset}}$, our algorithm can still learn polytopes that *roughly* segment instances.

loss	w/o $\mathcal{L}_{\text{poly}}$	w/o $\mathcal{L}_{\text{shift}}$	w/o $\mathcal{L}_{\text{sem}}$	Full
mAP	$95.42_{\pm 0.79}$	$95.38_{\pm 0.34}$	$95.06_{\pm 0.62}$	$\mathbf{96.80}_{\pm 0.35}$

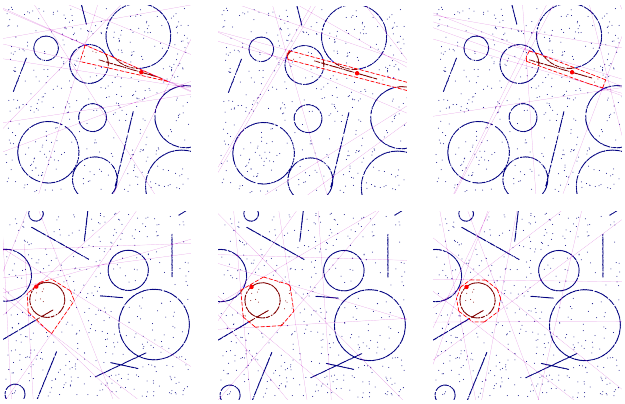

Figure 10. **Ablation: losses –** We report the average and standard deviation of mAP by repeating the experiment 5 times.

5. Conclusions

We have proposed an instance proposal method for point clouds. We formulate instance proposals as a query-conditioned attention model and employ neural bilateral filtering to provide much more accurate proposals than direct regression. We demonstrate through synthetic data that the proposal generation process is indeed a bottleneck, which our method can significantly improve. We further demonstrate the potential of our method on the ScanNet dataset, achieving competitive performance amongst top-down methods.

Limitations and Future works. While we have shown clearly that a bottleneck exists, and that it can be avoided, its benefit has not been as strikingly revealed when compared to pipelines that are carefully engineered for instance segmentation. We believe there is much room for this potential to be realized, similar to how top-down methods are the dominant strategy for natural images [43, 42].

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, NSERC Collaborative Research and Development Grant, Google, Digital Research Alliance of Canada, and Advanced Research Computing at the University of British Columbia.

References

[1] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical Aggregation for 3d Instance Segmentation. In *ICCV*, 2021.

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d Reconstructions of Indoor Scenes. In *CVPR*, 2017.

[3] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic Instance Segmentation for Autonomous Driving. In *CVPRW*, 2017.

[4] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic Instance Segmentation with a Discriminative Loss Function. *CVPRW*, 2017.

[5] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *CVPR*, 2020.

[6] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. NASA: Neural Articulated Shape Approximation. In *ECCV*, 2020.

[7] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021.

[8] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral Mesh Denoising. *ACM TOG*, 2003.

[9] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[10] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM TOG*, 2017.

[11] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *CVPR*, 2018.

[12] Wenchao Gu, Shuang Bai, and Lingxing Kong. A Review on 2D instance Segmentation Based on Deep Neural Networks. *IVC*, 2022.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[14] Kaiming He, Jian Sun, and Xiaoou Tang. Guided Image Filtering. *IEEE TPAMI*, 2012.

[15] Tong He, Chunhua Shen, and Anton van den Hengel. Dyco3d: Robust instance segmentation of 3d point clouds through dynamic convolution. In *CVPR*, 2021.

[16] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, Yaonan Wang, Qiang Fu, Yanmei Zou, and Ajmal Mian. Deep Learning based 3D Segmentation: A survey. *arXiv Preprint*, 2021.

[17] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d Semantic Instance Segmentation of Rgb-d Scans. In *CVPR*, 2019.

[18] Varun Jampani, Martin Kiefel, and Peter V. Gehler. Learning Sparse High Dimensional Filters: Image Filtering, Dense CRFs and Bilateral Neural Networks. In *CVPR*, 2016.

[19] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set Point Grouping for 3d Instance Segmentation. In *CVPR*, 2020.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.

[21] Shu Kong and Charless C Fowlkes. Recurrent Pixel Embedding for Instance Grouping. In *CVPR*, 2018.

[22] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *NeurIPS*, 33:22554–22565, 2020.

[23] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance Segmentation in 3d Scenes using Semantic Superpoint Tree Networks. In *ICCV*, 2021.

[24] Shih-Hung Liu, Shang-Yi Yu, Shao-Chi Wu, Hwann-Tzong Chen, and Tyng-Luh Liu. Learning Gaussian Instance Segmentation in Point Clouds. *arXiv Preprint*, 2020.

[25] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic Gradient Descent with Warm Restarts. *ICLR*, 2017.

[26] Shitong Luo, Jiahan Li, Jiaqi Guan, Yufeng Su, Chaoran Cheng, Jian Peng, and Jianzhu Ma. Equivariant point cloud analysis via learning orientations for message passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18932–18941, 2022.

[27] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *ISRR*, 2019.

[28] Mehdi Khoshboresh Masouleh and Reza Shah-Hosseini. Fusion of Deep Learning with Adaptive Bilateral Filter for Building Outline Extraction from Remote Sensing Imagery. *Journal of Applied Remote Sensing*, 2018.

[29] Andres Milioto, Jens Behley, Chris McCool, and Cyrill Stachniss. Lidar panoptic segmentation for autonomous driving. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8505–8512. IEEE, 2020.

[30] Douglas Morrison, Adam W Tow, Matt Mctaggart, R Smith, Norton Kelly-Boxall, Sean Wade-Mccue, Jordan Erskine, Riccardo Grinover, Alec Gurman, T Hunn, et al. Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7757–7764. IEEE, 2018.

[31] Takaya Ogawa and Tomohiro Mashita. Occlusion Handling in Outdoor Augmented Reality using a Combination of Map Data and Instance Segmentation. In *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2021.

[32] Omri Puny, Matan Atzmon, Heli Ben-Hamu, Edward J Smith, Ishan Misra, Aditya Grover, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021.

[33] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep Hough Voting for 3d Object Detection in Point Clouds. In *CVPR*, 2019.

[34] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017.

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards Real-time Object Detection with Region Proposal Networks. *NeurIPS*, 2015.

[36] Kshitij Sirohi, Rohit Mohan, Daniel Büscher, Wolfram Burgard, and Abhinav Valada. Efficientlps: Efficient lidar panoptic segmentation. *IEEE Transactions on Robotics*, 2021.

[37] Zachary Teed and Jia Deng. Raft: Recurrent All-pairs Field Transforms for Optical Flow. In *ECCV*, 2020.

[38] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully Convolutional One-stage Object Detection. In *CVPR*, 2019.

[39] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[40] Thang Vu, Kookhoi Kim, Tung M Luu, Thanh Nguyen, and Chang D Yoo. SoftGroup for 3D Instance Segmentation on Point Clouds. In *CVPR*, 2022.

[41] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively Segmenting Instances and Semantics in Point Clouds. In *CVPR*, 2019.

[42] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and Fast Instance Segmentation. *NeurIPS*, 2020.

[43] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Solo: A Simple Framework for Instance Segmentation. *IEEE TPAMI*, 2021.

[44] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying Unknown Instances for Autonomous Driving. In *Conference on Robot Learning*, 2020.

[45] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. Unseen Object Instance Segmentation for Robotic Environments. *IEEE Transactions on Robotics*, 2021.

[46] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single Shot Instance Segmentation with Polar Representation. In *CVPR*, 2020.

[47] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022.

[48] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. *NeurIPS*, 2019.

[49] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative Shape Proposal Network for 3d Instance Segmentation in Point Cloud. In *CVPR*, 2019.

[50] Dingfu Zhou, Jin Fang, Xibin Song, Liu Liu, Junbo Yin, Yuchao Dai, Hongdong Li, and Ruigang Yang. Joint 3d instance segmentation and object detection for autonomous driving. In *CVPR*, 2020.