INS-Conv: Incremental Sparse Convolution for Online 3D Segmentation

Leyao Liu*[1], Tian Zheng*[1], Yun-Jou Lin[2], Kai Ni[3], and Lu Fang[1]✉

[1]Department of Electronic Engineering, Tsinghua University [2]OPPO US Research Center
[3]HoloMatic Technology (Beijing) Co., Ltd.

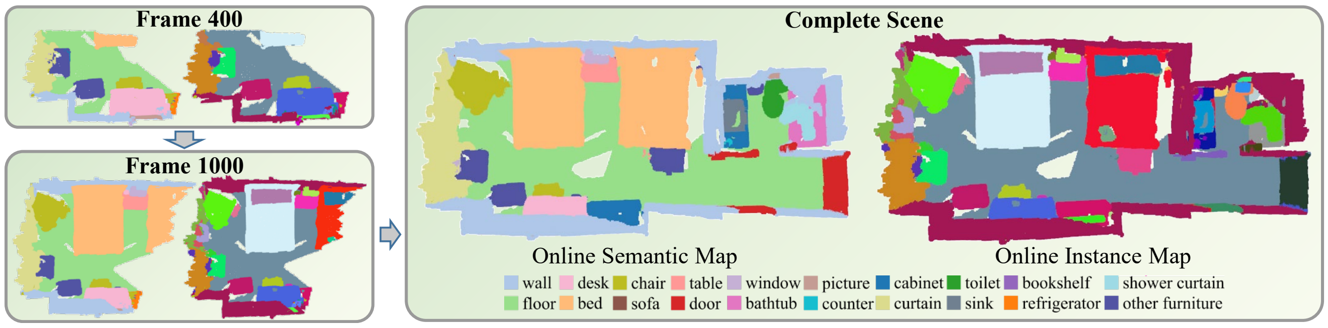


Figure 1. We propose INS-Conv, an incremental sparse convolutional network, which enables online accurate 3D semantic and instance segmentation. We generate semantic and instance labels along with 3D reconstruction, which is valuable for interactive AR/VR and robotics applications.

Abstract

*We propose **INS-Conv**, an **IN**cremental **S**parse **Conv**olutional network which enables online accurate 3D semantic and instance segmentation. Benefiting from the incremental nature of RGB-D reconstruction, we only need to update the residuals between the reconstructed scenes of consecutive frames, which are usually sparse. For layer design, we define novel residual propagation rules for sparse convolution operations, achieving close approximation to standard sparse convolution. For network architecture, an uncertainty term is proposed to adaptively select which residual to update, further improving the inference accuracy and efficiency. Based on INS-Conv, an online joint 3D semantic and instance segmentation pipeline is proposed, reaching an inference speed of 15 FPS on GPU and 10 FPS on CPU. Experiments on ScanNetv2 and SceneNN datasets show that the accuracy of our method surpasses previous online methods by a large margin, and is on par with state-of-the-art offline methods. A live demo on portable devices further shows the superior performance of INS-Conv.*

* Equal contribution.

✉ Corresponding author. Mail: fanglu@tsinghua.edu.cn.

Acknowledgement: This work is supported in part by Natural Science Foundation of China (NSFC) under contract No. 62125106, 61860206003 and 62088102, in part by National Key R&D Program of China (2021ZD0109901), in part by Beijing National Research Center for Information Science and Technology (BNRist) under Grant No. BNR2020RC01002.

1. Introduction

3D semantic and instance segmentation aims to detect objects of a 3D scene and provide per point semantic prediction simultaneously, which is fundamental to robotics or AR/VR applications. Recent methods [4, 8, 12, 13, 16, 26] that focus on offline 3D segmentation have shown great improvements in terms of segmentation accuracy, where sparse convolutional networks are widely used as backbones to extract 3D features [8, 12, 16]. While these offline methods achieve leading accuracy, they may take up to seconds to make a single update, because their backbone networks usually require global geometry as input, which cannot meet the online segmentation purpose, e.g., real-time interaction of AR agents with the surrounding environment.

For the task of online 3D segmentation, a common solution is the *2D-to-3D* approach, which means to perform 2D convolutions on RGBD frames, followed by projecting the 2D predictions to 3D space and fusing with the previous results via a probabilistic model [18, 20, 22]. These methods utilize 2D information merely, leading to low segmentation accuracy. Although recent methods achieve improvements by using 3D point convolution to process 2D features [15, 29], the problem remains unsolved, because neither 2D features nor local 3D convolutions are aware of the global information of the 3D scene. As a result, they still suffer from the low accuracy. Moreover, most online 3D segmentation methods only provide semantic predictions, without instance-level understanding. How to achieve highly accurate 3D semantic instance segmentation while enabling online inference along with 3D reconstruction is still an open question.

We propose INS-Conv, an incremental sparse convolutional network, which enables online accurate 3D semantic and instance segmentation. We observe that in online RGB-D reconstruction, the reconstructed scenes at each time step form an incrementally growing 3D geometry sequence, where the residuals between two continuous 3D frames are usually sparse. Therefore, a lot of redundant computation could be saved by performing incremental inference on the residuals of continuous frames. More specifically, for layer design, we define novel residual propagation rules for sparse convolution operations. By replacing the layers of a standard sparse convolutional network with our INS-Conv layers, we can achieve efficient incremental inference with minimal loss of accuracy. For network architecture, an uncertainty term is proposed to adaptively select which residual to update, by ignoring the unnecessary updating of the points that already have very confident predictions, while incorporating points that may have changed states in the future, further improving the inference accuracy and efficiency. Based on INS-Conv, an online joint 3D semantic and instance segmentation pipeline is proposed. At each time step, after the 3D features are extracted through the INS-Conv backbone network, we use clustering to generate instance predictions on the updated points, which are later fused into the previous results to get the final instance segmentation results using an instance fusion stage. To summarize, our contributions include:

- An incremental sparse convolutional network, INS-Conv. With the novel residual propagation strategy, along with adaptive residual selection through uncertainty prediction, it achieves fast and accurate inference of 3D convolutional networks.
- An online 3D joint semantic and instance segmentation pipeline implemented based on INS-Conv. It achieves state-of-the-art segmentation accuracy among online methods, on par with offline methods.
- A live demo of INS-Conv running locally on a portable device. The superior performance of INS-Conv in terms of both accuracy and efficiency makes it particularly adapted to AR/VR or robotics applications.
- The code is available at: `https://github.com/THU-luvision/INS-Conv`

2. Related Work

Offline Scene Segmentation 3D scene semantic and instance segmentation are widely studied topics in computer vision. For semantic segmentation tasks, most recent deep learning based methods fall into two types according to convolution type: point based [11, 24–27] and voxel based [2, 4, 12]. We focus on voxel based methods in our work. They take voxelized point clouds as input and then apply 3D convolution on the voxel grid. Early works adopt dense 3D convolutions [10, 31]. However, due to the high computation cost for high dimensional data, they could not handle large-scale voxel grids. The critical limitation is later solved by the emergence of sparse convolution [2, 4], which exploits the inherent sparsity of the 3D point cloud, demonstrating state-of-the-art segmentation accuracy. Hu *et al.* [12] later propose to jointly train 2D and 3D networks, achieving top performance. For instance segmentation, sparse convolutional networks are also widely used [8, 16, 17]. Lahoud *et al.* [17] propose a learning-then-clustering approach, performing meanshift clustering based on the per-point features extracted using a sparse convolutional network. Jiang *et al.* [16] propose to perform clustering on both the shifted coordinates and the original coordinates, and use an additional 3D network to predict the scores for the generated proposals. Han *et al.* [8] introduces an occupancy signal to guide the clustering stage. We follow the similar clustering-based approach, adding a fusion stage to fuse predictions of multiple frames.

Online Scene Segmentation Online scene segmentation has wide applications in AR/VR and robotics. The task is to predict semantic or instance labels along with 3D reconstruction system in real-time. Early works tackle this problem using the 2D-3D approach, which means to predict 2D semantic probabilities for each RGBD frame using 2D CNN and then project back to 3D space, followed by a probabilistic fusion step [18]. Narita G *et al.* [20] first perform instance segmentation in 2D, and then fuse the results to 3D to achieve online panoptic segmentation. Zhang *et al.* [29] propose to fuse 2D features by performing 3D point convolution on local neighborhoods, achieving accuracy improvements. However, it can only process very few points in order to maintain online speed. Following a similar paradigm, Huang *et al.* [15] perform 3D point convolution on supervoxels to fuse 2D features, which improves the speed, and achieves leading online semantic segmentation accuracy. However, these methods highly rely on 2D features and fail to capture global 3D information, resulting in a gap between offline and online methods. Instead, we follow the voxel based approach widely used in offline methods and perform incremental inference to achieve online performance.

Incremental CNN Several works have studied incremental inference on 2D convolutional networks, mostly targeting efficient video sequence processing. Cavigelli *et al.* [1] propose a change-based convolution layer, which performs conditional updates based on the input feature change. O'Connora and Welling [21] achieve inference speedup by performing convolutions on the quantized temporal residual of input features. Habibian *et al.* [5] further

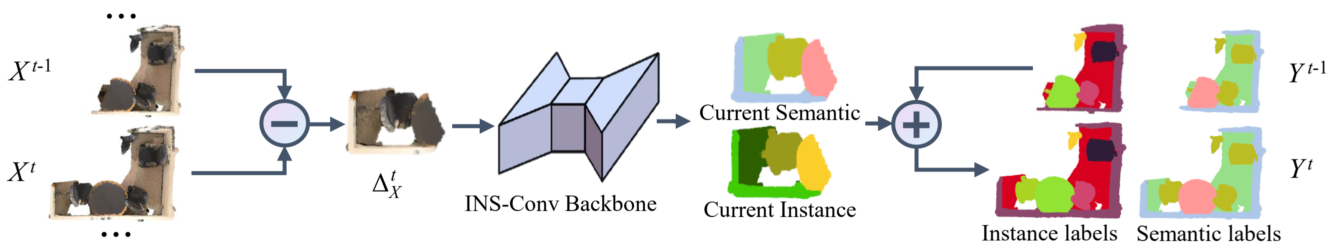


Figure 2. Overview of our incremental 3D semantic and instance segmentation pipeline. Given a sequence of gradually growing input geometry, an INS-Conv backbone network is used to extract per-point feature on the residual, followed by a clustering and fusion stage to generate the final semantic and instance segmentation results. Please refer to Sec. 3 for details.

use a learned gate function to decide which input residual is important. Xu *et al*. [28] propose to reuse feature maps of similar frames by reusable image region lookup. However, these methods only deal with dense 2D convolutions and are not directly suitable for 3D sparse convolutions.

3. Methods

An overview of our incremental semantic and instance segmentation pipeline is shown in Fig. 2. At the core is the INS-Conv backbone network, which is used for incremental feature extraction on the residuals of a series of gradually changing input geometry. Afterwards, the clustering stage and fusion stage work to generate temporally consistent semantic and instance segmentation results.

This section is organized as follows. After introducing the insight of INS-Conv in Sec. 3.1, we describe the layer design and the network architecture of INS-Conv in Sec. 3.2 and Sec. 3.3, respectively. Finally, the online 3D semantic instance segmentation pipeline is described in Sec. 3.4.

3.1. Insight of INS-Conv

Recall that a linear map is a function f that satisfies:

$$f(x+y)=f(x)+f(y),\ f(cx)=cf(x), \tag{1}$$

and the combination of linear maps is also a linear map:

$$f(g(x+y))=f(g(x)+g(y))=f(g(x))+f(g(y)). \tag{2}$$

In neural networks, many modules are linear maps, e.g., convolution layer and linear layer. Some advanced modules like batch normalization and residual blocks also satisfy the above equations by ignoring the potential bias term for simplicity. Thus, based on Eqn. 2, neural networks composed of these linear modules are linear maps as well (non-linear layers are described later in Sec. 3.2).

In our case, the neural network inference is performed on an incrementally reconstructed scene. We define x^t as color features of all voxels that have been built up at time t, and Δ_x^t as residuals (differences) between x^t and x^{t-1}. Our neural network f inputs the color features of voxels, and outputs the labels of each voxel. For current time t, the network forward can be divided into two parts:

$$f(x^t)=f(x^{t-1}+\Delta_x^t)=f(x^{t-1})+f(\Delta_x^t), \tag{3}$$

where $f(x^{t-1})$ has been computed previously. Thus, we can simply use the cached result and calculate $f(\Delta_x^t)$ merely. The calculation of $f(\Delta_x^t)$ indicates that the network is propagating residuals of features, because for every linear map layer l, $l(\Delta_x)=l(x+\Delta_x)-l(x)=\Delta_y$, where x and y denote the input and output feature of this layer.

In a short summary, we show that neural network inference on a sequence of input can be reformulated as propagating the residuals of input features, making incremental prediction possible. Based on such insight, we propose INS-Conv, an incremental sparse convolutional network that is fast and accurate for online 3D segmentation, as elaborated in the following subsections.

3.2. Layer Design of INS-Conv

Review of Sparse Convolution The key idea of a standard sparse convolution is to ignore the empty locations, only storing and calculating convolutions on the non-empty sites of the input data. To avoid dilation of non-empty sites, submanifold sparse convolution (SSC) [4] merely calculates the output features for active sites of the input. Formally, submanifold sparse convolution operations are performed on a N^D spatial neighborhood of each site u:

$$\mathbf{x}_u^{\text{out}}=\sum_{i\in N^D} W_i\mathbf{x}_{u+i}^{\text{in}} \text{ if } u\in A, \tag{4}$$

where N is a pre-defined kernel size, D indicates the dimension of spatial space (equals to 3 for 3D convolution), and W_i is the weight matrix at location i for input features $\mathbf{x}_{u+i}^{in}$. A denotes the set of non-empty sites of the input tensor. More details can be found in [4].

INS-SSC Layer We define an incremental submanifold sparse convolution (denoted as *INS-SSC*) layer that performs submanifold sparse convolution on residuals.

Recall that the sparse convolution is computed for input sites that have non-empty features. We denote this site set

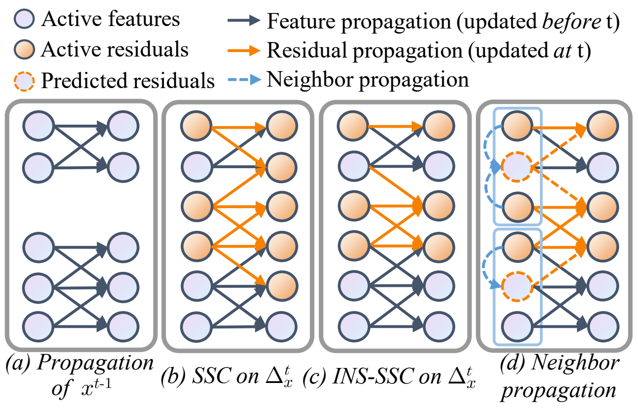

Active features
Active residuals
Predicted residuals
Feature propagation (updated *before* t)
Residual propagation (updated *at* t)
Neighbor propagation
(a) Propagation of x^{t-1}
(b) SSC on Δ_x^t
(c) INS-SSC on Δ_x^t
(d) Neighbor propagation

Figure 3. Illustration of the INS-SSC layer using a 1-D sparse convolution example with a kernel size of 3. After the propagation is performed on the previous frame as in (a), the INS-SSC layer is performed on the residuals of the current frame as in (c). (b) shows the standard SSC rule may cause residual dilation, thus not suitable for residual propagation. (d) shows INS-SSC with neighbor propagation, where residuals of unchanged sites are estimated from their neighbors. Details are in Sec. 3.2.

as A, and in addition maintain an active residual site set B, which includes input sites that have non-empty residuals. Let the input and output features of current layer at frame t be x^t and y^t, respectively. Then, the residuals of input at frame t become $\Delta_x^t = x^t - x^{t-1}$, and our goal is to compute Δ_y^t. The propagation rule for INS-SSC layer is defined as:

$$\Delta_{y_u}^t = \begin{cases} \sum_i W_i \Delta_{x_{u+i}}^t & \text{if } u \in B^t \cap A^{t-1}, \\ \sum_i W_i (\Delta_{x_{u+i}}^t + x_{u+i}^{t-1}) & \text{if } u \in B^t \setminus A^{t-1}. \end{cases} \quad (5)$$

Fig. 3 gives an intuitive illustration of INS-SSC using 1-D sparse convolution example with a kernel size of 3. Compared with the conventional SSC [4], INS-SSC is different in that: 1) INS-SSC takes residual as input; 2) INS-SSC operates on the active residual site set B, rather than the set of all active features A. Since B is much more sparse than A, INS-SSC would be more efficient; 3) INS-SSC constrains the output active residual set to be identical to the input, while SSC would 'dilate' the active residual set after each layer, as shown in Fig. 3(b)(c). 4) INS-SSC follows different convolution rules. The rules used in SSC could yield incorrect results in the case where u is a new active site that was previously inactive. Specifically, the previous feature y_u^{t-1} is set to zero under the rule of sparse convolution that ignores inactive sites, but it should exist when u becomes active at current frame, which we denote as $\hat{y}_u^{t-1}$. The compensation can be made by adding $\hat{y}_u^{t-1}$ to the propagated residual, as stated in Eqn. 5.

Neighbor Propagation The sparse residual propagation rule of INS-SSC layer assures that the active residual sites would not dilate, thus bringing computation benefits. Unfortunately, discarding the supposed residuals outside of B^t

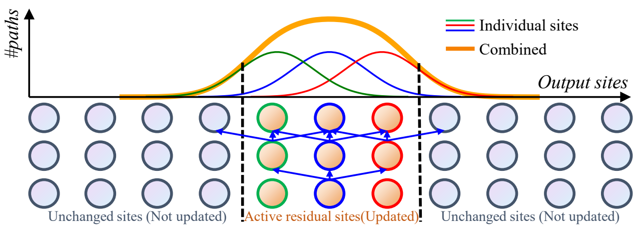
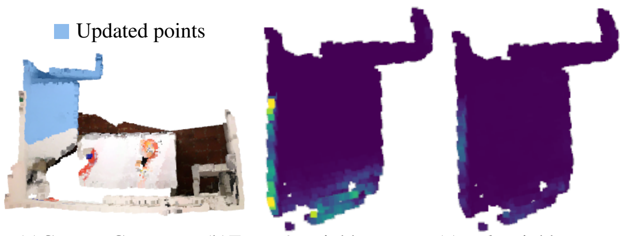

#paths
Individual sites
Combined
Output sites
Unchanged sites (Not updated)
Active residual sites(Updated)
Unchanged sites (Not updated)

Figure 4. Intuition of why the residual propagation rule of INS-SSC achieves low approximation error.



(a)Current Geometry (b)Err. *w/o* neighbor prop. (c)*with* neighbor prop.

Figure 5. Visualization of the approximate error of INS-Conv. The error is calculated by the KL divergence of the output semantic probabilities between INS-Conv and 'full' propagation.

makes INS-SSC no longer identical to 'full' propagation as SSC. To analyze the approximation error, we take 1-D convolutional network as an example in Fig. 4. According to [19], in general, the distribution of the impact of a changed input feature to deeper layers resembles a Gaussian distribution centered at the changed input site, which is measured roughly by the number of unique propagation paths. Since the active residual sites are spatially nearby in our case, the sum of impact of all active residual sites resembles a Gaussian distribution as well (plotted in orange in Fig. 4). Therefore, the effect of truncating the residual propagation outside of B^t is relatively small. Note we only care about errors in active residual sites since we only compute for these sites. As visually illustrated in Fig. 5(b), the error map shows that the overall error is quite small and mostly distributed at the border of active residual sites due to the truncation of neighboring residuals.

To further reduce the approximation error at the border, a neighbor propagation approach is proposed. Our key observation is that for spatially nearby sites, their features and residuals should be similar as well. This enlightens us that given the unchanged input sites $m \in A^t \setminus B^t$ that would not be updated in INS-SSC while have direct connections to the output active residual sites, the residuals of these sites can be approximated by their neighboring active residual sites using weighted average:

$$\Delta_{x_m}^t = \sum_{n \in \mathcal{N}_m} w_{mn} \Delta_{x_n}^t, \quad (6)$$

where the weights w_{mn} are computed by similarities of features,

$$w_{mn} = \frac{\exp(s(x_m^{t-1}, x_n^{t-1}))}{\sum_{k \in \mathcal{N}_m} \exp(s(x_m^{t-1}, x_k^{t-1}))}. \quad (7)$$

Here, $s(x_m, x_n) = l(x_m - x_n)$ denotes the similarity of feature x_m and x_n, and l is a linear layer. $\mathcal{N}_m$ denotes the neighbor region of site m and set to be a kernel size around m in the implementation.

Fig. 3(d) illustrates the INS-SSC layer with neighbor propagation. Benefiting from the predicted residuals at unchanged sites, the approximation error can be effectively reduced as shown in Fig. 5(c).

INS Convolution and Deconvolution Layer The strided sparse convolution and deconvolution layers are used to down-sample/up-sample the feature maps. The propagation rules of INS Convolution layer are the same as INS-SSC except we allow dilation of the active residual sites. The INS Deconvolution layer is simply the inverted operation of INS Convolution layer.

INS Non-linear Layer Non-linear layers are not linear maps in general, thus could not directly propagate residuals. Formally, for a non-linear function g, Δ_y^t is not equal to $g(\Delta_x^t)$. However, we can calculate the output residuals Δ_y^t using the definition of residual:

$$\Delta_{y_u}^t = g(\Delta_{x_u}^t + x_u^{t-1}) - y_u^{t-1}, \quad (8)$$

x_u^{t-1} and y_u^{t-1} are cached at the previous time step.

For sites not in active residual site set, their residuals are defined to be zero. Thus they are ignored in all layers. Based on the aforementioned layer designs, INS-Conv only needs to input voxels that have non-zero residuals. These input sites form the first layer's active residual site set.

3.3. Network Architecture of INS-Conv

We use a typical UNet-like sparse convolutional network as the backbone. At training time, it works the same way as standard sparse convolutional networks. At inference time, we replace the layers with corresponding INS-Conv layers to achieve incremental inference. Aiming for the similar task of 3D segmentation as [8], several representations are learned for each voxel i, including (1) semantic probability s_i for semantic segmentation, (2) instance embedding e_i for instance segmentation. Details are explained in supplementary material. Additionally, the uncertainty term and temporal consistency constraint are elaborated below.

Uncertainty Term Recall in INS-Conv, we choose voxels that have updated color features in the current time as input voxels. Though straightforward, we can further make this process more intelligent. As we mentioned earlier, the INS-Conv only computes feature changes for active residual sites, which are determined by the input voxels due to the no-dilate rule of INS-SSC. If we know which voxel will have large changed features in the future, we may additionally incorporate it in the input, though it may not have color changes currently due to view scope restriction. On the other hand, if we know a voxel has already been well predicted, there is no need to put it in input again.

This selection mechanism can be achieved by predicting an uncertain probability for each voxel. Here we define a voxel is uncertain if it is unable to make a correct prediction on this voxel due to the incomplete scene at current time. The more uncertain a voxel is, the more likely it will have a changed state in the future. We propose to train the network to detect uncertain voxels. We formulate it as a binary classification problem for each voxel. The uncertainty of a voxel in an incomplete scene is defined to be positive if 1) its semantic prediction is different from that in a complete scene, or 2) the distance between its instance embedding and that in the complete scene is larger than δ_d. Here δ_d is set to 0.8.

To supervise the training, we generate different completeness for each scene and put each scene along with its partial scenes in the same batch. We then use the complete scene prediction and partial scene prediction to generate the ground truth labels of the uncertainty term.

Temporal Consistency Constraint In [8], the discriminative loss function is utilized to enforce instance embeddings of voxels belonging to the same instance to be near in feature space. In our online setting, we further add a temporal consistency loss that constrains the embeddings of an instance to be near across time too. This will be useful in our instance fusion stage to match instances across time, which we will describe later in Sec. 3.4. Because of our training strategy that put complete scene along with its partial scenes in the same batch, the temporal consistency loss can be formulated as:

$$\mathcal{L}_{con} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c^k}\sum_{i=1}^{N_c^k}[||\mathbf{u_c} - e_i^k|| - \delta_v]_+^2. \quad (9)$$

Here, K means the number of partial scenes of a scene, C is the number of instances in the complete scene, N_c^k is the number of voxels of c-th instance in partial scene k, $\mathbf{u}_c$ is the mean embedding of instance c in complete scene, e_i^k is the predicted embedding of i-th voxel of instance c in partial scene k. δ_v is set to be 0.1. In short, this term enforces the embeddings of voxels in partial scenes to be near to the mean embeddings of their belonging instances in the complete scene.

3.4. Online Semantic and Instance Segmentation

For each time step, we first calculate the difference of color feature of each voxel stored in TSDF compared with the previous frame. The SLAM system will update voxels in the current view frustum, so the non-zero sites in residual volume will be a frustum shape. We utilize uncertainty

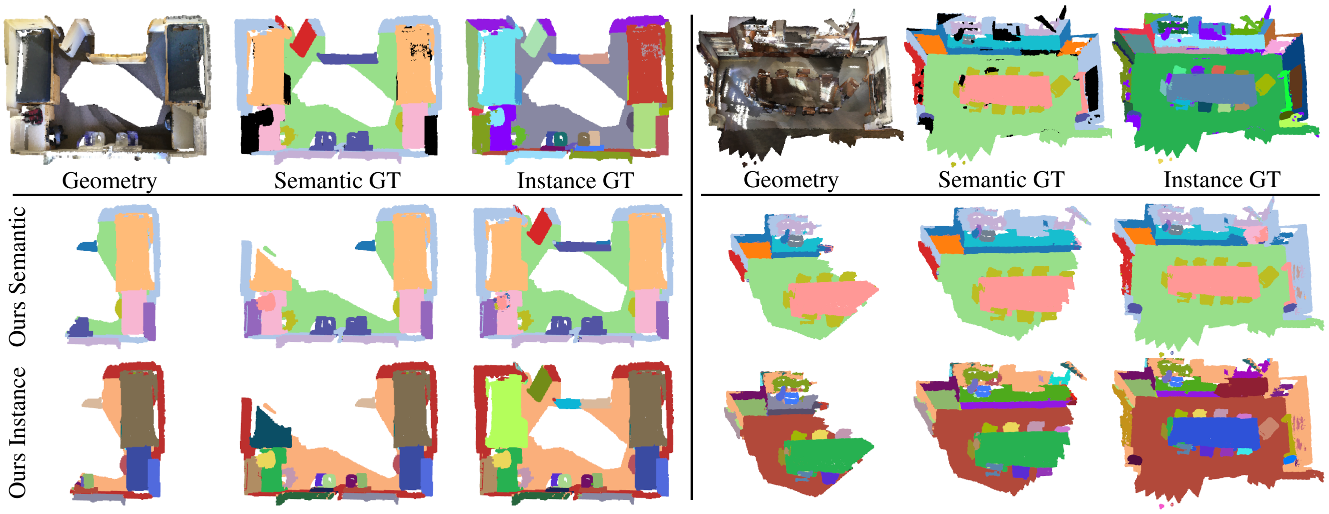


Figure 6. Visualization of online semantic and instance results on the validation set of ScanNetv2. We show the online segmentation results at 3 different time steps for each scene.

to further select voxels as input. Specifically, we filter out voxels in the frustum whose saved previous uncertainty is smaller than θ. In addition, we add voxels in the surrounding of the frustum whose uncertainty is larger than θ and treat their residuals as zeros. θ is set to be 0.4 in our case. Note that voxels in input will be set to be active residual sites, regardless of whether their residual values are zeros or not. Due to the no-dilate rule of INS-Conv, we only get the updated prediction results of voxels in input, which are then followed by a clustering and fusion stage to generate the final semantic and instance segmentation results.

Instance Segmentation We adopt the clustering-based instance segmentation scheme. Unlike offline methods [8,17], we only perform instance clustering on the updated voxels for high efficiency, using predicted embeddings. The current instance set I_c is then fused into the global instance set I_g. For every instance $i \in I_c$, we compute the similarity $S(i,j)$ between i and every instance $j \in I_g$. Previous method [20] uses only position overlap to match instances. However, it is hard to determine the overlap ratio to match, and it doesn't have error correction capability. For example, if two instances were falsely considered as one, they will never get separated. Benefiting from our temporal consistent instance embedding, we can compute the matching relation by comparing mean embeddings of instances, thus increasing the robustness. In detail, we store the mean predicted embedding $\mathbf{u}_j$ for each instance $j \in I_g$. The distance d_{ij} between i and j is computed as: $d_{ij} = \exp(-||\mathbf{u}_i - \mathbf{u}_j||^2)$. The position overlap can also help to measure similarity. The overall $S(i,j)$ is formulated as: $S(i,j) = (1 + \frac{O(i,j)}{2N_i})d_{ij}$, where $O(i,j)$ means the number of voxels of instance i that overlap with global instance j, and N_i is the number of voxels of instance i. The maximum similarity $S_{max}(i)$ of instance i and the corresponding global instance $\hat{j}$ is:

$$S_{max}(i) = \max_{j \in I_g} S(i,j), \hat{j} = \underset{j \in I_g}{argmax}\, S(i,j) \qquad (10)$$

If $S_{max}(i) > \alpha$, the instance i will match to global instance $\hat{j}$, else it will be assigned a new instance label. Here α is a hyper-parameter and set to 0.65 in our experiments.

Semantic Segmentation Instead of directly using the raw predicted semantic probabilities, we force all points in the same instance $i \in I_c$ to have the same semantic label, i.e, the majority label of i, to get a more spatially consistent semantic map. Besides, we adopt the fusion method in [20] to fuse current semantic results to global to make it temporally consistent.

4. Experiments

Both CPU-Only and GPU versions of INS-Conv are implemented based on [4] and [9], respectively. The INS-Conv is performed every frame. For every 100 frames, a network forward is performed on the current full scene using conventional sparse convolution [4] to update the internal features of the network to avoid drifting error. To show the effect of INS-Conv, the network outputs of such step are not counted for the following evaluation. Two different sized models are tested to show the accuracy and efficiency trade-off, denoted as *m32* (smaller) and *m64* (larger). Please refer to supplementary material for the details of network architecture.

4.1. 3D Semantic and Instance Segmentation

ScanNetv2 [3] Dataset ScanNetv2 includes 1513 indoor scenes with 3D semantic and instance labels for training

Table 1. Semantic and instance segmentation results on ScanNetv2. Due to the submission policy, we only report test set results of the m64 model. The FPS of online methods are fetched from their papers.

(a) Semantic Segmentation on ScanNetv2

Method	Type	mIoU		FPS	
		Val	Test	GPU	CPU
Fs-A [29]	Online	67.2	63.0	10	-
SVCNN [15]	Online	68.3	63.5	20	-
SCN [4]	Offline	69.3	72.5	-	-
MkNet [2]	Offline	72.2	73.6	-	-
Ours-m32	Online	71.5	-	15	10
Ours-m64	Online	72.4	71.7	10	8

(b) Instance Segmentation on ScanNetv2

Method	Type	mAP@50		FPS	
		Val	Test	GPU	CPU
PF [20]	Online	-	47.8	4.3	-
PointGroup [16]	Offline	56.9	63.6	-	-
OccuSeg [8]	Offline	60.7	67.2	-	-
Ours-m32	Online	57.4	-	15	10
Ours-m64	Online	61.4	65.7	10	8

and evaluation, and a hidden test set of 100 scenes used for benchmark evaluation. For the validation set results, we follow the same test/validation split as ScanNetv2 [3]. In Table. 1, we report the mean Intersection-over-Union (mIoU) for semantic segmentation, and the average precision at 0.5 IoU (mAP@50) for instance segmentation on both validation set and test set. The runtime is also reported as frame-per-second (FPS) for online methods. For those methods whose code is not accessible, we report the results from their papers, thus the FPS may not be precisely comparable due to different hardware and experiment setups.

For semantic segmentation, both two of our models achieve the highest mIoU among online methods, higher than SVCNN [15] by a large margin, while being slightly slower. Compared with offline methods, we are on par with the state-of-the-art offline methods, while being significantly faster for online segmentation purposes.

For instance segmentation, PanopticFusion (PF) [20] is the only recent online segmentation method that provides instance prediction. We achieve much higher mAP@50 (17.9%) while being faster (Note that PF requires two GPUs, while we only need one). Compared with offline methods, we achieve similar mAP@50. It is worth noting that our method does not take any post-processing technique, which however, has been widely used in offline methods to increase their accuracy numbers. Fig. 6 gives the qualitative semantic and instance results of our method.

SceneNN [14] Dataset SceneNN provides 76 indoor scenes with semantic and instance annotations. For semantic segmentation, we train our model on ScanNetv2 and evaluate on SceneNN to test the ability of generalization, following the same setting as SVCNN [15]. The evaluation metric is average mean accuracy (mAcc). As shown in Table. 2(a), our method is superior to all the previous online methods.

Table 2. Results on SceneNN dataset using the m64 model; (**a**) Semantic average mAcc(%), compared with other online methods; (**b**) Instance mAP@50(%), compared with offline methods.

(a) Semantic on SceneNN

Method(Online)	mAcc
Fs-A [29]	71.5
SVCNN [15]	76.5
Ours(Online)	79.5

(b) Instance on SceneNN

Method(Offline)	mAP@50
MLS-CRF [23]	12.1
Occuseg [8]	47.1
Ours(Online)	57.6

Table 3. Runtime (ms) of each stage of our pipeline, including network, instance clustering, instance fusion and semantic stage. The network runs in parallel with other stages.

Model	Network	Clust.	Fusion	Sem.	FPS
m32	61	32	33	2	15 Hz
m64	99	30	35	2	10 Hz

For instance segmentation, We trained from scratch on SceneNN, 50 scenes for training and 20 for testing, following the same setting and split as [8]. The comparisons with existing offline methods are shown in Table. 2(b). Surprisingly, we achieve much better mAP@50, possibly because our training strategy prevents overfitting. For per class results, please refer to supplementary material.

Runtime Analysis Following PF [20], the computation time is evaluated on *scene0645_01*, a representative large-scale scene in ScanNetv2. Experiments are performed on a computer equipped with an Intel Core i7-6800K(3.4GHz) CPU, and a Titan Xp GPU. As our 3D segmentation pipeline contains four stages: 1) network, 2) instance clustering, 3) instance fusion, 4) semantic segmentation, we report the averaged time of each stage in Table. 3. The network runs on GPU, and is parallel with other stages. The FPS of joint semantic and instance segmentation are 15Hz and 10Hz for m32 and m64 models, respectively. For the CPU version, the FPS are 10Hz and 8Hz. We also test our baseline method, predicting the current full scene every frame, using the m64 model. On average, the network forward takes $649ms$, and the instance clustering takes $1100ms$. Our INS-Conv achieves $6.5x$ faster for semantic segmentation, and $11x$ faster for joint semantic and instance segmentation. By integrating with a SLAM system [6, 7, 30], we provide an online demo of INS-Conv on a portable device. Please refer to supplementary material.

4.2. Ablation Study

We explore the effectiveness of different components in our method. The ablation experiments are conducted on the validation set of ScanNetv2. All model sizes are the same as the m64 model.

Table 4. Ablation study on INS-Conv. Semantic and instance results show that it significantly boosts the accuracy. The comparison with 'full' propagation demonstrates approximation ability.

INS-Conv	Neighbor propagation	mIoU	mAP@50
×	×	58.0	30.5
✓	×	72.1	60.9
✓	✓	72.2	61.2
Full propagation		72.2	61.3

Table 5. Approximation error to full propagation($\times 10^{-3}$), calculated by averaged per frame MSE of last layer features.

	MSE
w/o neighbor prop.	9.3
Ours	5.0

Table 6. Ablation study on temporal consistency constraint. The instance mAP@50 results are reported.

	mAP@50
w/o consist.	59.6
Ours	61.2

Effect of INS-Conv To show the effectiveness of INS-Conv, we replace INS-Conv with standard sparse convolution [4] and perform on the same input points, i.e., the points in the current view frustum. The comparison of semantic and instance results are shown in the first and second row of Table. 4. Without INS-Conv, the accuracy drops significantly. This is because INS-Conv is an approximation to the full scene inference, while the naive way can only "see" the current frustum. This shows the importance of global information in 3D semantic and instance segmentation.

The neighbor propagation module is also evaluated. It can reduce the approximation error of INS-Conv. Fig. 5 give the qualitative result of approximation error reduction, we also provide quantitative results in Table. 4 and Table. 5. By using neighbor propagation, the approximation error of last layer features is reduced by about 50%, achieving nearly the same semantic and instance results compared with 'full' propagation.

Effect of Temporal Consistency Constraint Temporal consistent embedding helps to match current instances to global instances in the instance fusion stage. To verify the importance of temporal consistency constraint, we train the network without the consistency loss term. The comparison results are shown in Table. 6. We can see a clear drop in instance mAP. This is because, without temporal consistent embedding, the current instances often fail to match to global instances. These instances will be assigned to a new label, resulting in over-segmentation.

Effect of Uncertainty Term Using uncertainty probability to select points as input makes our INS-Conv more intelligent. Fig. 7 shows the predicted uncertainty map of an incomplete scene. We can see that the uncertainty is high in points that are hard to predict due to incomplete nearby scene, while low in points that have already been predicted well. To give quantitative results, we compare our uncertainty-guided input selection method with the naive way, i.e., choosing points in the current view frustum as input. We test the accuracy, together with the time of our network running on CPU and on GPU, with the m64 model. As shown in Table. 7, by using uncertainty, we not only reduce the computation time but also increase the accuracy. The average number of points processed per frame is reduced by about 66%. We notice that the time reduction is most significant while running on CPU, this is because CPU processes data in sequential. By using uncertainty to select input, we largely reduce the data to process, thus enabling real-time CPU-only 3D segmentation.

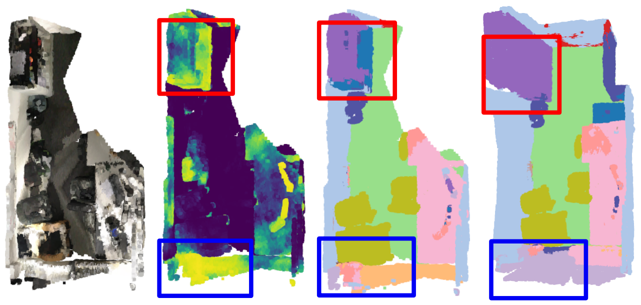

(a) Current Geometry (b) Uncertainty (c) Current Semantic (d) Complete Semantic

Figure 7. Visualization of the predicted uncertainty map. We show that high uncertainty mostly happens on points that are falsely predicted because of incomplete scene.

Table 7. Ablation study on uncertainty-guided input selection method. We report the semantic and instance accuracy results on the ScanNetv2 validation set. The average time of network running on CPU-Only and on GPU, together with the average number of input points per frame are tested on *scene0645_01*.

	mIoU	mAP@50	avg. time		#pts
			CPU	GPU	
w/o uncert.	72.2	61.2	328	122	19990
Ours	72.4	61.4	125	99	6820

5. Discussion and Conclusion

In this work, we propose INS-Conv, a 3D sparse convolutional network that enables accurate and efficient incremental inference. Based on that, we achieve online 3D semantic instance segmentation. Extensive experiments demonstrate the superior online segmentation accuracy. Our method also has a few limitations. First, the partial scene segmentation problem has not been seriously studied in our method. Incorporating 2D information might be useful. Second, we still need to perform a global network update every 100 frames to avoid drifting error. Although it performs sparsely, it is interesting to study how to avoid this step. In the future, we will explore more online tasks using INS-Conv.

References

[1] Lukas Cavigelli, Philippe Degen, and Luca Benini. Cbinfer: Change-based inference for convolutional neural networks on video data. *Proceedings of the 11th International Conference on Distributed Smart Cameras*, 2017. 2

[2] Christopher Bongsoo Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3070–3079, 2019. 2, 7

[3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443, 2017. 6, 7

[4] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018. 1, 2, 3, 4, 6, 7, 8

[5] AmirHossein Habibian, Davide Abati, Taco Cohen, and Babak Ehteshami Bejnordi. Skip-convolutions for efficient video processing. In *CVPR*, 2021. 2

[6] Lei Han and Lu Fang. Flashfusion: Real-time globally consistent dense 3d reconstruction using cpu computing. In *Robotics: Science and Systems*, volume 1, page 7, 2018. 7

[7] Lei Han, Lan Xu, Dmytro Bobkov, Eckehard Steinbach, and Lu Fang. Real-time global registration for globally consistent rgb-d slam. *IEEE Transactions on Robotics*, 35(2):498–508, 2019. 7

[8] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2937–2946, 2020. 1, 2, 5, 6, 7

[9] Lei Han, Tian Zheng, Yinheng Zhu, Lan Xu, and Lu Fang. Live semantic 3d perception for immersive augmented reality. *IEEE transactions on visualization and computer graphics*, 26(5):2012–2022, 2020. 6

[10] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4416–4425, 2019. 2

[11] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Agathoniki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11105–11114, 2020. 2

[12] Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia, and Tien-Tsin Wong. Bidirectional projection network for cross dimension scene understanding. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14368–14377, 2021. 1, 2

[13] Zeyu Hu, Mingmin Zhen, Xuyang Bai, Hongbo Fu, and Chiew-Lan Tai. Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In *ECCV*, 2020. 1

[14] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. *2016 Fourth International Conference on 3D Vision (3DV)*, pages 92–101, 2016. 7

[15] Shi-Sheng Huang, Ze-Yu Ma, Tai-Jiang Mu, Hongbo Fu, and Shi-Min Hu. Supervoxel convolution for online 3d semantic segmentation. *ACM Trans. Graph.*, 40:34:1–34:15, 2021. 1, 2, 7

[16] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4866–4875, 2020. 1, 2, 7

[17] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R. Oswald. 3d instance segmentation via multi-task metric learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9255–9265, 2019. 2, 6

[18] John McCormac, Ankur Handa, Andrew J. Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2017. 1, 2

[19] Supun Nakandala, Kabir Nagrecha, Arun Kumar, and Yannis Papakonstantinou. Incremental and approximate computations for accelerating deep cnn inference. *ACM Transactions on Database Systems (TODS)*, 45(4):1–42, 2020. 4

[20] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4205–4212, 2019. 1, 2, 6, 7

[21] Peter O'Connor and Max Welling. Sigma delta quantized networks. *ArXiv*, abs/1611.02024, 2017. 2

[22] Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Real-time progressive 3d semantic segmentation for indoor scenes. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1089–1098, 2019. 1

[23] Quang-Hieu Pham, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2019. 7

[24] C. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 2

[25] C. Qi, L. Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 2

[26] Hugues Thomas, C. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds.

2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 6410–6419, 2019. 1, 2

[27] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9613–9622, 2019. 2

[28] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. Deepcache: Principled cache for mobile deep vision. *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018. 3

[29] Jiazhao Zhang, Chenyang Zhu, Lin tao Zheng, and Kai Xu. Fusion-aware point convolution for online semantic 3d scene segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4533–4542, 2020. 1, 2, 7

[30] Tian Zheng, Guoqing Zhang, Lei Han, Lan Xu, and Lu Fang. Building fusion: Semantic-aware structural building-scale 3d reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. 7

[31] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. *ArXiv*, abs/1606.06650, 2016. 2