

3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans

Ji Hou

Angela Dai

Matthias Nießner

Technical University of Munich

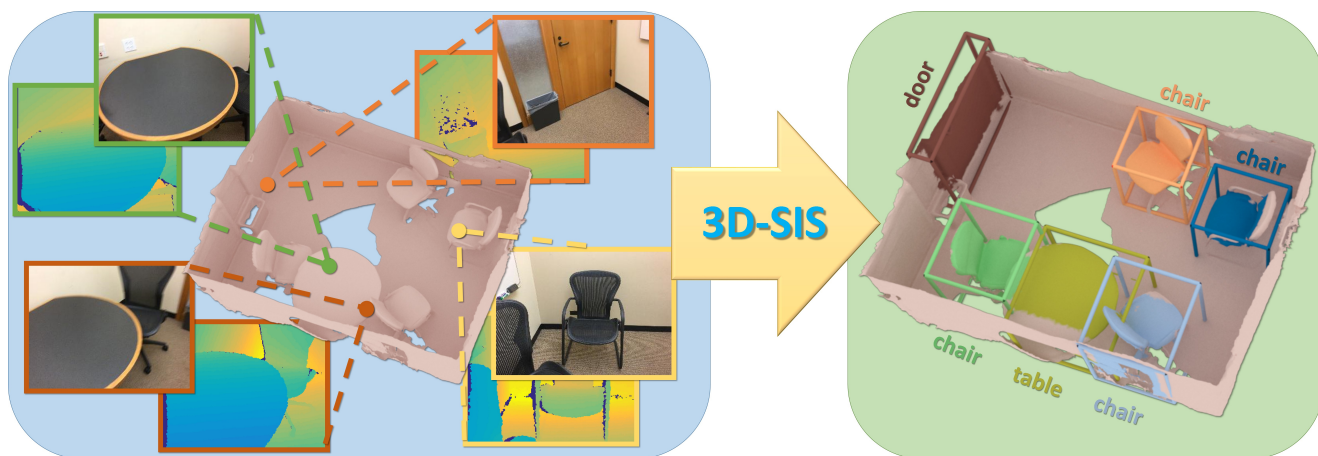


Figure 1: 3D-SIS performs 3D instance segmentation on RGB-D scan data, learning to jointly fuse both 2D RGB input features with 3D scan geometry features. In combination with a fully-convolutional approach enabling inference on full 3D scans at test time, we achieve accurate inference for object bounding boxes, class labels, and instance masks.

Abstract

We introduce 3D-SIS, a novel neural network architecture for 3D semantic instance segmentation in commodity RGB-D scans. The core idea of our method is to jointly learn from both geometric and color signal, thus enabling accurate instance predictions. Rather than operate solely on 2D frames, we observe that most computer vision applications have multi-view RGB-D input available, which we leverage to construct an approach for 3D instance segmentation that effectively fuses together these multi-modal inputs. Our network leverages high-resolution RGB input by associating 2D images with the volumetric grid based on the pose alignment of the 3D reconstruction. For each image, we first extract 2D features for each pixel with a series of 2D convolutions; we then backproject the resulting feature vector to the associated voxel in the 3D grid. This combination of 2D and 3D feature learning allows significantly higher accuracy object detection and instance segmentation than state-of-the-art alternatives. We show results on both synthetic and real-world public benchmarks, achieving an improvement in mAP of over 13 on real-world data.

1. Introduction

Semantic scene understanding is critical to many real-world computer vision applications. It is fundamental towards enabling interactivity, which is core to robotics in both indoor and outdoor settings, such as autonomous cars, drones, and assistive robotics, as well as upcoming scenarios using mobile and AR/VR devices. In all these applications, we would not only want semantic inference of single images, but importantly, also require understanding of spatial relationships and layouts of objects in 3D environments.

With recent breakthroughs in deep learning and the increasing prominence of convolutional neural networks, the computer vision community has made tremendous progress on analyzing images in the recent years. Specifically, we are seeing rapid progress in the tasks of semantic segmentation [18, 12, 20], object detection [10, 25], and semantic instance segmentation [11]. The primary focus of these impressive works lies in the analysis of visual input from a single image; however, in many real-world computer vision scenarios, we rarely find ourselves in such a single-image setting. Instead, we typically record video streams of RGB input sequences, or as in many robotics and AR/VR applications, we have 3D sensors such as LIDAR or RGB-D cameras.

In particular, in the context of semantic instance segmentation, it is quite disadvantageous to run methods independently on single images given that instance associations must be found across a sequence of RGB input frames. Instead, we aim to infer spatial relationships of objects as part of a semantic 3D map, learning prediction of spatially-consistent semantic labels and the underlying 3D layouts *jointly* from all input views and sensor data. This goal can also be seen as similar to traditional sensor fusion but for deep learning from multiple inputs.

We believe that robustly-aligned and tracked RGB frames, and even depth data, from SLAM and visual odometry provide a unique opportunity in this regard. Here, we can leverage the given mapping between input frames, and thus learn features jointly from all input modalities. In this work, we specifically focus on predicting 3D semantic instances in RGB-D scans, where we capture a series of RGB-D input frames (e.g., from a Kinect Sensor), compute 6DoF rigid poses, and reconstruct 3D models. The core of our method learns semantic features in the 3D domain from both color features, projected into 3D, and geometry features from the signed distance field of the 3D scan. This is realized by a series of 3D convolutions and ResNet blocks. From these semantic features, we obtain anchor bounding box proposals. We process these proposals with a new 3D region proposal network (3D-RPN) and 3D region of interest pooling layer (3D-RoI) to infer object bounding box locations, class labels, and per-voxel instance masks. In order to jointly learn from RGB frames, we leverage their pose alignments with respect to the volumetric grid. We first run a series of 2D convolutions, and then backproject the resulting features into the 3D grid. In 3D, we then join the 2D and 3D features in end-to-end training constrained by bounding box regression, object classification, and semantic instance mask losses.

Our architecture is fully-convolutional, enabling us to efficiently infer predictions on large 3D environments in a single shot. In comparison to state-of-the-art approaches that operate on individual RGB images, such as Mask R-CNN [11], our approach achieves significantly higher accuracy due to the joint feature learning.

To sum up, our contributions are the following:

- We present the first approach leveraging joint 2D-3D end-to-end feature learning on both geometry and RGB input for 3D object bounding box detection and semantic instance segmentation on 3D scans.
- We leverage a fully-convolutional 3D architecture for instance segmentation trained on scene parts, but with single-shot inference on large 3D environments.
- We outperform state-of-the-art by a significant margin, increasing the mAP by 13.5 on real-world data.

2. Related Work

2.1. Object Detection and Instance Segmentation

With the success of convolutional neural network architectures, we have now seen impressive progress on object detection and semantic instance segmentation in 2D images [10, 26, 17, 25, 15, 11, 16]. Notably, Girshick et al. [10] introduced an anchor mechanism to predict objectness in a region and regress associated 2D bounding boxes while jointly classifying the object type. Mask R-CNN [11] expanded this work to semantic instance segmentation by predicting a per-pixel object instance masks. An alternative direction for detection is the popular Yolo work [25], which also defines anchors on grid cells of an image.

This progress in 2D object detection and instance segmentation has inspired work on object detection and segmentation in the 3D domain, as we see more and more video and RGB-D data become available. Song et al. proposed Sliding Shapes to predict 3D object bounding boxes from single RGB-D frame input with handcrafted feature design [29], and then expanded the approach to operate on learned features [30]. The latter direction leverages the RGB frame input to improve classification accuracy of detected objects; in contrast to our approach, there is no explicit spatial mapping between RGB and geometry for joint feature learning. An alternative approach is taken by Frustum PointNet [21], where detection is performed a 2D frame and then back-projected into 3D from which final bounding box predictions are refined. Wang et al. [34] base their SGPN approach on semantic segmentation from a PointNet++ variation. They formulate instance segmentation as a clustering problem upon a semantically segmented point cloud by introducing a similarity matrix prediction similar to the idea behind panoptic segmentation [14]. In contrast to these approaches, we explicitly map both multi-view RGB input with 3D geometry in order to jointly infer 3D instance segmentation in an end-to-end fashion.

2.2. 3D Deep Learning

In the recent years, we have seen impressive progress in developments on 3D deep learning. Analogous to the 2D domain, one can define convolution operators on volumetric grids, which for instance embed a surface representation as an implicit signed distance field [4]. With the availability of 3D shape databases [35, 3, 31] and annotated RGB-D datasets [28, 1, 5, 2], these network architectures are now being used for 3D object classification [35, 19, 23, 27], semantic segmentation [5, 33, 6], and object or scene completion [8, 31, 9]. An alternative representation to volumetric grids are the popular point-based architectures, such as PointNet [22] or PointNet++ [24], which leverage a more efficient, although less structured, representation of 3D surfaces. Multi-view approaches have also

been proposed to leverage RGB or RGB-D video information. Su et al. proposed one of the first multi-view architectures for object classification by view-pooling over 2D predictions [32], and Kalogerakis et al. recently proposed an approach for shape segmentation by projecting predicted 2D confidence maps onto the 3D shape, which are then aggregated through a CRF [13]. Our approach joins together many of these ideas, leveraging the power of a holistic 3D representation along with features from 2D information by combining them through their explicit spatial mapping.

3. Method Overview

Our approach infers 3D object bounding box locations, class labels, and semantic instance masks on a per-voxel basis in an end-to-end fashion. To this end, we propose a neural network that jointly learns features from both geometry and RGB input. In the following, we refer to bounding box regression and object classification as object detection, and semantic instance mask segmentation for each object as mask prediction.

In Sec. 4, we first introduce the data representation and training data that is used by our approach. Here, we consider synthetic ground truth data from SUNCG [31], as well as manually-annotated real-world data from ScanNetV2 [5]. In Sec. 5, we present the neural network architecture of our 3D-SIS approach. Our architecture is composed of several parts; on the one hand, we have a series of 3D convolutions that operate in voxel grid space of the scanned 3D data. On the other hand, we learn 2D features that we back-project into the voxel grid where we join the features and thus jointly learn from both geometry and RGB data. These features are used to detect object instances; that is, associated bounding boxes are regressed through a 3D-RPN and class labels are predicted for each object following a 3D-ROI pooling layer. For each detected object, features from both the 2D color and 3D geometry are forwarded into a per-voxel instance mask network. Detection and per-voxel instance mask prediction are trained in an end-to-end fashion. In Sec. 6, we describe the training and implementation details of our approach, and in Sec. 7, we evaluate our approach.

4. Training Data

Data Representation We use a truncated sign distance field (TSDF) representation to encode the reconstructed geometry of the 3D scan inputs. The TSDF is stored in a regular volumetric grid with truncation of 3 voxels. In addition to this 3D geometry, we also input spatially associated RGB images. This is feasible since we know the mapping between each image pixel with voxels in the 3D scene grid based on the 6 degree-of-freedom (DoF) poses from the respective 3D reconstruction algorithm.

For the training data, we subdivide each 3D scan into chunks of $4.5\text{m} \times 4.5\text{m} \times 2.25\text{m}$, and use a resolution of $96 \times 96 \times 48$ voxels per chunk (each voxel stores a TSDF value); i.e., our effective voxel size is $\approx 4.69\text{cm}^3$. In our experiments, for training, we associate 5 RGB images at a resolution of 328×256 pixels in every chunk, with training images selected based on the average voxel-to-pixel coverage of the instances within the region.

Our architecture is fully-convolutional (see Sec. 5), which allows us to run our method over entire scenes in a single shot for inference. Here, the xy-voxel resolution is derived from a given test scene’s spatial extent. The z (height) of the voxel grid is fixed to 48 voxels (approximately the height of a room), with the voxel size also fixed at 4.69cm^3 . Additionally, at test time, we use all RGB images available for inference. In order to evaluate our algorithm, we use training, validation, test data from synthetic and real-world RGB-D scanning datasets.

Synthetic Data For synthetic training and evaluation, we use the SUNCG [31] dataset. We follow the public train/val/test split, using 5519 train, 40 validation, and 86 test scenes (test scenes are selected to have total volume $< 600\text{m}^3$). From the train and validation scenes, we extract 97,918 train chunks and 625 validation chunk. Each chunk contains an average of ≈ 4.3 object instances. At test time, we take the full scan data of the 86 test scenes.

In order to generate partial scan data from these synthetic scenes, we virtually render them, storing both RGB and depth frames. Trajectories are generated following the virtual scanning approach of [9], but adapted to provide denser camera trajectories to better simulate real-world scanning scenarios. Based on these trajectories, we then generate partial scans as TSDFs through volumetric fusion [4], and define the training data RGB-to-voxel grid image associations based on the camera poses. We use 23 class categories for instance segmentation, defined by their NYU40 class labels; these categories are selected for the most frequently-appearing object types, ignoring the wall and floor categories which do not have well-defined instances.

Real-world Data For training and evaluating our algorithm on real-world scenes, we use the ScanNetV2 [5] dataset. This dataset contains RGB-D scans of 1513 scenes, comprising ≈ 2.5 million RGB-D frames. The scans have been reconstructed using BundleFusion [7]; both 6 DoF pose alignments and reconstructed models are available. Additionally, each scan contains manually-annotated object instance segmentation masks on the 3D mesh. From this data, we derive 3D bounding boxes which we use as constraints for our 3D region proposal.

We follow the public train/val/test split originally proposed by ScanNet of 1045 (train), 156 (val), 312 (test)

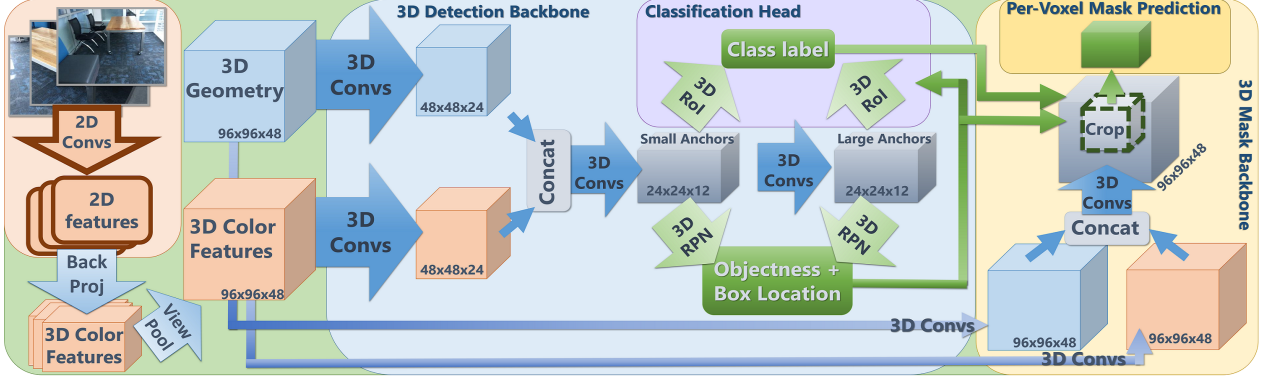


Figure 2: 3D-SIS network architecture. Our architecture is composed of a 3D detection and a 3D mask pipeline. Both 3D geometry and 2D color images are taken as input and used to jointly learn semantic features for object detection and instance segmentation. From the 3D detection backbone, color and geometry features are used to propose the object bounding boxes and their class labels through a 3D-RPN and a 3D-RoI layer. The mask backbone also uses color and geometry features, in addition to the 3D detection results, to predict per-voxel instance masks inside the 3D bounding box.

scenes, respectively. From the train scenes, we extract 108241 chunks, and from the validation scenes, we extract 995 chunks. Note that due to the smaller number of train scans available in the ScanNet dataset, we augment the train scans to have 4 rotations each. We adopt the same 18-class label set for instance segmentation as proposed by the ScanNet benchmark.

Note that our method is agnostic to the respective dataset as long as semantic RGB-D instance labels are available.

5. Network Architecture

Our network architecture is shown in Fig. 2. It is composed of two main components, one for detection, and one for per-voxel instance mask prediction; each of these pipelines has its own feature extraction backbone. Both backbones are composed of a series of 3D convolutions, taking the 3D scan geometry along with the back-projected RGB color features as input. We detail the RGB feature learning in Sec. 5.1 and the feature backbones in Sec. 5.2. The learned 3D features of the detection and mask backbones are then fed into the classification and the voxel-instance mask prediction heads, respectively.

The object detection component of the network comprises the detection backbone, a 3D region proposal network (3D-RPN) to predict bounding box locations, and a 3D-region of interest (3D-RoI) pooling layer followed by classification head. The detection backbone outputs features which are input to the 3D-RPN and 3D-RoI to predict bounding box locations and object class labels, respectively. The 3D-RPN is trained by associating predefined anchors with ground-truth object annotations; here, a per-anchor loss defines whether an object exists for a given anchor. If it does, a second loss regresses the 3D object bounding box; if not, no additional loss is considered. In addition, we classify the object class of each 3D bounding

box. For the per-voxel instance mask prediction network (see Sec. 5.4), we use both the input color and geometry as well as the predicted bounding box location and class label. The cropped feature channels are used to create a mask prediction which has n channels for the n semantic class labels, and the final mask prediction is selected from these channels using the previously predicted class label. We optimize for the instance mask prediction using a binary cross entropy loss. Note that we jointly train the backbones, bounding box regression, classification, and per-voxel mask predictions end-to-end; see Sec. 6 for more detail. In the following, we describe the main components of our architecture design, for more detail regarding exact filter sizes, etc., we refer to the supplemental material.

5.1. Back-projection Layer for RGB Features

In order to jointly learn from RGB and geometric features, one could simply assign a single RGB value to each voxel. However, in practice, RGB image resolutions are significantly higher than the available 3D voxel resolution due to memory constraints. This 2D-3D resolution mismatch would make learning from a per-voxel color rather inefficient. Inspired by the semantic segmentation work of Dai et al. [6], we instead leverage a series of 2D convolutions to summarize RGB signal in image space. We then define a back-projection layer and map these features on top of the associated voxel grid, which are then used for both object detection and instance segmentation.

To this end, we first pre-train a 2D semantic segmentation network based on the ENet architecture [20]. The 2D architecture takes single 256×328 RGB images as input, and is trained on a semantic classification loss using the NYUv2 40 label set. From this pre-trained network, we extract a feature encoding of dimension 32×41 with 128 channels from the encoder. Using the corresponding depth

image, camera intrinsics, and 6DoF poses, we then back-project each of these features back to the voxel grid (still 128 channels); the projection is from 2D pixels to 3D voxels. In order to combine features from multiple views, we perform view pooling through an element-wise max pooling over all RGB images available.

For training, the voxel volume is fixed to $96 \times 96 \times 48$ voxels, resulting in a $128 \times 96 \times 96 \times 48$ back-projected RGB feature grid in 3D; here, we use 5 RGB images for each training chunk (with image selection based on average 3D instance coverage). At test time, the voxel grid resolution is dynamic, given by the spatial extent of the environment; here, we use all available RGB images. The grid of projected features is processed by a set of 3D convolutions and is subsequently merged with the geometric features.

In ScanNet [5], the camera poses and intrinsics are provided; we use them directly for our back-projection layer. For SUNCG [31], extrinsics and intrinsics are given by the virtual scanning path. Note that our method is agnostic to the used 2D network architecture.

5.2. 3D Feature Backbones

For jointly learning geometric and RGB features for both instance detection and segmentation, we propose two 3D feature learning backbones. The first backbone generates features for detection, and takes as input the 3D geometry and back-projected 2D features (see Sec. 5.1).

Both the geometric input and RGB features are processed symmetrically with a 3D ResNet block before joining them together through concatenation. We then apply a 3D convolutional layer to reduce the spatial dimension by a factor of 4, followed by a 3D ResNet block (e.g., for an input train chunk of $96 \times 96 \times 48$, we obtain a features of size $24 \times 24 \times 12$). We then apply another 3D convolutional layer, maintaining the same spatial dimensions, to provide features maps with larger receptive fields. We define anchors on these two feature maps, splitting the anchors into ‘small’ and ‘large’ anchors (small anchors $< 1\text{m}^3$), with small anchors associated with the first feature map of smaller receptive field and large anchors associated with the second feature map of larger receptive field. For selecting anchors, we apply k-means algorithm ($k=14$) on the ground-truth 3D bounding boxes in first 10k chunks. These two levels of features maps are then used for the final steps of object detection: 3D bounding box regression and classification.

The instance segmentation backbone also takes the 3D geometry and the back-projected 2D CNN features as input. The geometry and color features are first processed independently with two 3D convolutions, and then concatenated channel-wise and processed with another two 3D convolutions to produce a mask feature map prediction. Note that for the mask backbone, we maintain the same spatial resolution through all convolutions, which we found to be

critical for obtaining high accuracy for the voxel instance predictions. The mask feature map prediction is used as input to predict the final instance mask segmentation.

In contrast to single backbone, we found that this two-backbone structure both converged more easily and produced significantly better instance segmentation performance (see Sec. 6 for more details about the training scheme for the backbones).

5.3. 3D Region Proposals and 3D-RoI Pooling for Detection

Our 3D region proposal network (3D-RPN) takes input features from the detection backbone to predict and regress 3D object bounding boxes. From the detection backbone we obtain two feature maps for small and large anchors, which are separately processed by the 3D-RPN. For each feature map, the 3D-RPN uses a $1 \times 1 \times 1$ convolutional layer to reduce the channel dimension to $2 \times N_{\text{anchors}}$, where $N_{\text{anchors}} = (3, 11)$ for small and large anchors, respectively. These represent the positive and negative scores of objectness of each anchor. We apply a non-maximum suppression on these region proposals based on their objectness scores. The 3D-RPN then uses another $1 \times 1 \times 1$ convolutional layer to predict feature maps of $6 \times N_{\text{anchors}}$, which represent the 3D bounding box locations as $(\Delta_x, \Delta_y, \Delta_z, \Delta_w, \Delta_h, \Delta_l)$, defined in Eq. 1.

In order to determine the ground truth objectiveness and associated 3D bounding box locations of each anchor during training, we perform anchor association. Anchors are associated with ground truth bounding boxes by their IoU: if the $\text{IoU} > 0.35$, we consider an anchor to be positive (and it will be regressed to the associated box), and if the $\text{IoU} < 0.15$, we consider an anchor to be negative (and it will not be regressed to any box). We use a two-class cross entropy loss to measure the objectiveness, and for the bounding box regression we use a Huber loss on the prediction $(\Delta_x, \Delta_y, \Delta_z, \Delta_w, \Delta_h, \Delta_l)$ against the log ratios of the ground truth box and anchors $(\Delta_x^{gt}, \Delta_y^{gt}, \Delta_z^{gt}, \Delta_w^{gt}, \Delta_h^{gt}, \Delta_l^{gt})$, where

$$\Delta_x = \frac{\mu - \mu_{\text{anchor}}}{\phi_{\text{anchor}}} \quad \Delta_w = \ln\left(\frac{\phi}{\phi_{\text{anchor}}}\right) \quad (1)$$

where μ is the box center point and ϕ is the box width.

Using the predicted bounding box locations, we can then crop out the respective features from the global feature map. We then unify these cropped features to the same dimensionality using our 3D Region of Interest (3D-RoI) pooling layer. This 3D-RoI pooling layer pools the cropped feature maps into $4 \times 4 \times 4$ blocks through max pooling operations. These feature blocks are then linearized for input to object classification, which is performed with an MLP.

	cab	bed	chair	sofa	tabl	door	wind	bkskf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	opropr	avg
Seg-Cluster	16.8	16.2	15.6	11.8	14.5	10.0	11.7	27.2	20.0	25.7	10.0	0.0	15.0	0.0	20.0	27.8	39.5	22.9	10.7	38.9	10.4	0.0	12.3	16.4
Mask R-CNN [11]	14.9	19.0	19.5	13.5	12.2	11.7	14.2	35.0	15.7	18.3	13.7	0.0	24.4	23.1	26.0	28.8	51.2	28.1	14.7	32.2	11.4	10.7	19.5	19.9
SGPN [34]	18.6	39.2	28.5	46.5	26.7	21.8	15.9	0.0	24.9	23.9	16.3	20.8	15.1	10.7	0.0	17.7	35.1	37.0	22.9	34.2	17.7	31.5	13.9	22.5
Ours(geo only)	23.2	78.6	47.7	63.3	37.0	19.6	0.0	0.0	21.3	34.4	16.8	0.0	16.7	0.0	10.0	22.8	59.7	49.2	10.0	77.2	10.0	0.0	19.3	26.8
Ours(geo+1view)	22.2	70.8	48.5	66.6	44.4	10.0	0.0	63.9	25.8	32.2	17.8	0.0	25.3	0.0	0.0	14.7	37.0	55.5	20.5	58.2	18.0	20.0	17.9	29.1
Ours(geo+3views)	26.5	78.4	48.2	59.5	42.8	26.1	0.0	30.0	22.7	39.4	17.3	0.0	36.2	0.0	10.0	10.0	37.0	50.8	16.8	59.3	10.0	36.4	17.8	29.4
Ours(geo+5views)	20.5	69.4	56.2	64.5	43.8	17.8	0.0	30.0	32.3	33.5	21.0	0.0	34.2	0.0	10.0	20.0	56.7	56.2	17.6	56.2	10.0	35.5	17.8	30.6

Table 1: 3D instance segmentation on synthetic scans from SUNCG [31]. We evaluate the mean average precision with IoU threshold of 0.25 over 23 classes. Our joint color-geometry feature learning enables us to achieve more accurate instance segmentation performance.

5.4. Per-Voxel 3D Instance Segmentation

We perform instance mask segmentation using a separate mask backbone, which similarly as the detection backbone, takes as input the 3D geometry and projected RGB features. However, for mask prediction, the 3D convolutions maintain the same spatial resolutions, in order to maintain spatial correspondence with the raw inputs, which we found to significantly improve performance. We then use the predicted bounding box location from the 3D-RPN to crop out the associated mask features from the mask backbone, and compute a final mask prediction with a 3D convolution to reduce the feature dimensionality to n for n semantic class labels; the final mask prediction is the c^{th} channel for predicted object class c . During training, since predictions from the detection pipeline can be wrong, we only train on predictions whose predicted bounding box overlaps with the ground truth bounding box with at least 0.5 IoU. The mask targets are defined as the ground-truth mask in the overlapping region of the ground truth box and proposed box.

6. Training

To train our model, we first train the detection backbone and 3D-RPN. After pre-training these parts, we add the 3D-RoI pooling layer and object classification head, and train these end-to-end. Then, we add the per-voxel instance mask segmentation network along with the associated backbone. In all training steps, we always keep the previous losses (using 1:1 ratio between all losses), and train everything end-to-end. We found that a sequential training process resulted in more stable convergence and higher accuracy.

We use an SGD optimizer with learning rate 0.001, momentum 0.9 and batch size 64 for 3D-RPN, 16 for classifi-

cation, 16 for mask prediction. The learning rate is divided by 10 every 100k steps. We use a non-maximum suppression for proposed boxes with threshold of 0.7 for training and 0.3 for test. Our network is implemented with PyTorch and runs on a single Nvidia GTX1080Ti GPU. The object detection components of the network are trained end-to-end for 10 epochs (≈ 24 hours). After adding in the mask backbone, we train for an additional 5 epochs (≈ 16 hours). For mask training, we also use ground truth bounding boxes to augment the learning procedure.

7. Results

We evaluate our approach on both its 3D detection and instance segmentation predictions, comparing to several state-of-the-art approaches, first on synthetic scans of SUNCG [31] data and then on real-world scans from the ScanNetV2 dataset [5]. In order to compare to previous approaches that operate on single RGB or RGB-D frames (Mask R-CNN [11], Deep Sliding Shapes [30], Frustum PointNet [21]), we first obtain predictions on each individual frame, and then merge all predictions together in the 3D space of the scene, merging predictions if the predicted class labels match and the IoU > 0.5 . We further compare to SGPN [34] which performs instance segmentation on 3D point clouds. For both detection and instance segmentation tasks, we project all results into a voxel space of 4.69cm voxels and evaluate them with a mean average precision metric. We additionally show several variants of our approach for learning from both color and geometry features, varying the number of color views used during training. We consistently find that training on more color views improves both the detection and instance segmentation performance.

	cab	bed	chair	sofa	tabl	door	wind	bkskf	pic	cntr	desk	curt	fridge	showr	toil	sink	bath	ofurn	avg
Mask R-CNN [11]	5.3	0.2	0.2	10.7	2.0	4.5	0.6	0.0	23.8	0.2	0.0	2.1	6.5	0.0	2.0	1.4	33.3	2.4	5.8
SGPN [34]	6.5	39.0	27.5	35.1	16.8	8.7	13.8	16.9	1.4	2.9	0.0	6.9	2.7	0.0	43.8	11.2	20.8	4.3	14.3
MTML	2.7	61.4	39.0	50.0	10.5	10.0	0.3	33.7	0.0	0.0	0.1	11.8	16.7	14.3	57.0	4.6	66.7	2.8	21.2
3D-BEVIS	3.5	56.6	39.4	60.4	18.1	9.9	17.1	7.6	2.5	2.7	9.8	3.5	9.8	37.5	85.4	12.6	66.7	3.0	24.8
R-PointNet [36]	34.8	40.5	58.9	39.6	27.5	28.3	24.5	31.1	2.8	5.4	12.6	6.8	21.9	21.4	82.1	33.1	50.0	29.0	30.6
3D-SIS (Ours)	13.4	55.4	58.7	72.8	22.4	30.7	18.1	31.9	0.6	0.0	12.1	0.0	54.1	1.0	88.9	4.5	66.7	21.0	36.2

Table 2: Instance segmentation results on the official ScanNetV2 3D semantic instance benchmark (hidden test set). Our final model (geo+5views) significantly outperforms previous (Mask R-CNN, SGPN) and concurrent (MTML, 3D-BEVIS, R-PointNet) state-of-the-art methods in mAP@0.5. ScanNetV2 benchmark data accessed on 12/17/2018.



	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	avg
Seg-Cluster	11.8	13.5	18.9	14.6	13.8	11.1	11.5	11.7	0.0	13.7	12.2	12.4	11.2	18.0	19.5	18.9	16.4	12.2	13.4
Mask R-CNN [11]	15.7	15.4	16.4	16.2	14.9	12.5	11.6	11.8	19.5	13.7	14.4	14.7	21.6	18.5	25.0	24.5	24.5	16.9	17.1
SGPN [34]	20.7	31.5	31.6	40.6	31.9	16.6	15.3	13.6	0.0	17.4	14.1	22.2	0.0	0.0	72.9	52.4	0.0	18.6	22.2
Ours(geo only)	22.1	48.2	64.4	52.2	16.0	13.4	0.0	17.2	0.0	20.7	17.4	13.9	23.6	33.0	45.2	47.7	61.3	14.6	28.3
Ours(geo+1view)	25.4	60.3	66.2	52.1	31.7	27.6	10.1	16.9	0.0	21.4	30.9	18.4	22.6	16.0	70.5	44.5	37.5	20.0	31.8
Ours(geo+3views)	28.3	52.3	65.0	66.5	31.4	27.9	10.1	17.9	0.0	20.3	36.3	20.1	28.1	31.0	68.6	41	66.8	24.0	35.3
Ours(geo+5views)	32.0	66.3	65.3	56.4	29.4	26.7	10.1	16.9	0.0	22.1	35.1	22.6	28.6	37.2	74.9	39.6	57.6	21.1	35.7

Table 3: 3D instance segmentation on real-world scans from ScanNetV2 [5]. We evaluate the mean average precision with IoU threshold of 0.25 over 18 classes. Our explicit leveraging of the spatial mapping between the 3D geometry and color features extracted through 2D convolutions enables significantly improved instance segmentation performance.

7.1. 3D Instance Analysis on Synthetic Scans

We evaluate 3D detection and instance segmentation on virtual scans taken from the synthetic SUNCG dataset [31], using 23 semantic class categories. Table 4 shows 3D detection performance compared to state-of-the-art approaches which operate on single frames. In Table 1, we show a quantitative evaluation of our instance segmentation, the SGPN approach for point cloud instance segmentation [34], their proposed Seg-Cluster baseline, and Mask R-CNN [11] projected into 3D. For both tasks, our joint color-geometry approach along with a global view of the 3D scenes at test time enables us to achieve significantly improved detection and segmentation results.

	mAP@0.25	mAP@0.5
Deep Sliding Shapes [29]	12.8	6.2
Mask R-CNN 2D-3D [11]	20.4	10.5
Frustum PointNet [21]	24.9	10.8
Ours – 3D-SIS (geo only)	27.8	21.9
Ours – 3D-SIS (geo+1view)	30.9	23.8
Ours – 3D-SIS (geo+3views)	31.3	24.2
Ours – 3D-SIS (geo+5views)	32.2	24.7

Table 4: 3D detection in synthetic scans from SUNCG [31]. We evaluate the mean average precision over 23 classes. Our holistic approach to interpreting 3D scans as well as the combination of color and geometric features results in significantly improved detection results over previous approaches which operate on individual input frames.

7.2. 3D Instance Analysis on Real-World Scans

We further evaluate our approach on the ScanNet dataset [5], which contains 3D semantic instance annotations on 1513 real-world scans. For training and evaluation, we use the ScanNetV2 annotated ground truth instance and class labels as well as the proposed 18-class instance benchmark. We show qualitative results in Figure 3, and in Table 5, we quantitatively evaluate our object detection performance against Deep Sliding Shapes and Frustum Pointnet, which operate on RGB-D frame data, as well as Mask R-CNN [11] projected to 3D. Here, our fully-convolutional approach enabling inference on full test scenes achieves

significantly better detection performance. Table 3 shows our 3D instance segmentation in comparison to the SGPN point cloud instance segmentation [34], their proposed Seg-Cluster baseline, and Mask R-CNN [11] projected into 3D. Our formulation for learning from both color and geometry features brings notable improvement over state of the art.

	mAP@0.25	mAP@0.5
Deep Sliding Shapes [29]	15.2	6.8
Mask R-CNN 2D-3D [11]	17.3	10.5
Frustum PointNet [21]	19.8	10.8
Ours – 3D-SIS (geo only)	29.7	16.3
Ours – 3D-SIS (geo+1view)	32.6	16.6
Ours – 3D-SIS (geo+3views)	36.8	18.5
Ours – 3D-SIS (geo+5views)	37.8	20.0

Table 5: 3D detection on real-world scans from ScanNetV2 [5], evaluated using mean average precision over 18 classes. In contrast to previous approaches operating on individual frames, our fully-convolutional approach for inference on full test scenes enables significantly improved object detection.

Finally, we evaluate our model on the ScanNetV2 3D instance segmentation benchmark using the automated evaluation script on the hidden test set; see Table 2. Our final model (geo+5views) significantly outperforms previous (Mask R-CNN [11], SGPN [34]) and concurrent (MTML, 3D-BEVIS, R-PointNet [36]) state-of-the-art methods in mAP@0.5. ScanNetV2 benchmark data was accessed on 12/17/2018.

8. Limitations

While our 3D instance segmentation approach leveraging joint color-geometry feature learning achieves marked performance gain over state of the art, there are still several important limitations. For instance, our current 3D bounding box predictions are axis-aligned to the grid space of the 3D environment. Generally, it would be beneficial to additionally regress the orientation for object instances; e.g., in the form of a rotation angle. Note that this would need to account for symmetric objects where poses might be ambiguous. At the moment, our focus is also largely on indoor

environments as we use commodity RGB-D data such as a Kinect or Structure Sensor. However, we believe that the idea of taking multi-view RGB-D input is agnostic to this specific setting; for instance, we could very well see applications in automotive settings with LIDAR and panorama data. Another limitation of our approach is the focus on static scenes. Ultimately, the goal is to handle dynamic or at least semi-dynamic scenes where objects are moving, which we would want to track over time. Here, we see a significant research opportunities and a strong correlation to tracking and localization methods that would benefit from semantic 3D segmentation priors.

9. Conclusion

In this work, we have introduced 3D-SIS, a new approach for 3D semantic instance segmentation of RGB-D scans, which is trained in an end-to-end fashion to detect object instances and infer a per-voxel 3D semantic instance segmentation. The core of our method is a neural network that jointly learns features from RGB and geometry data using multi-view RGB-D input recorded with commodity RGB-D sensors. The learned network is fully-convolutional, and thus can be efficiently run in a single shot on large 3D environments. In comparison to existing state-of-the-art methods that typically operate on single RGB frames, we achieve significantly higher 3D detection and segmentation results, improving on mAP by over 13. We believe that this is an important insight to a wide range of computer vision applications given that many of them now capture multi-view RGB and depth streams; e.g., autonomous cars, AR/VR applications, etc..

Acknowledgments

This work was supported by a Google Research Grant, a TUM Foundation Fellowship, a TUM-IAS Rudolf Mößbauer Fellowship, and the ERC Starting Grant *Scan2CAD* (804724).

References

- [1] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. [2](#)
- [2] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. [2](#)
- [3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. [2](#)
- [4] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. [2, 3](#)
- [5] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. [2, 3, 5, 6, 7, 8, 11, 13](#)
- [6] A. Dai and M. Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. *arXiv preprint arXiv:1803.10409*, 2018. [2, 4](#)
- [7] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017. [3](#)
- [8] A. Dai, C. R. Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. [2](#)
- [9] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. *arXiv preprint arXiv:1712.10215*, 2018. [2, 3](#)
- [10] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. [1, 2](#)
- [11] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. [1, 2, 6, 8, 13](#)
- [12] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014. [1](#)
- [13] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3d shape segmentation with projective convolutional networks. *Proc. CVPR, IEEE*, 2, 2017. [3](#)
- [14] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. *arXiv preprint arXiv:1801.00868*, 2018. [2](#)
- [15] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. [2](#)
- [16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. [2](#)
- [17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. [2](#)
- [18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. [1](#)
- [19] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. [2](#)
- [20] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. [1, 4](#)

- [21] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017. 2, 6, 8
- [22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 2
- [23] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016. 2
- [24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017. 2
- [25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2
- [26] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2
- [27] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2
- [28] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015. 2
- [29] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pages 634–651. Springer, 2014. 2, 8
- [30] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. *arXiv preprint arXiv:1511.02300*, 2015. 2, 6
- [31] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 3, 5, 6, 8, 11, 13
- [32] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953, 2015. 3
- [33] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2
- [34] W. Wang, R. Yu, Q. Huang, and U. Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018. 2, 6, 8, 13
- [35] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 2
- [36] L. Yi, W. Zhao, H. Wang, M. Sung, and L. Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. *arXiv preprint arXiv:1812.03320*, 2018. 6, 8

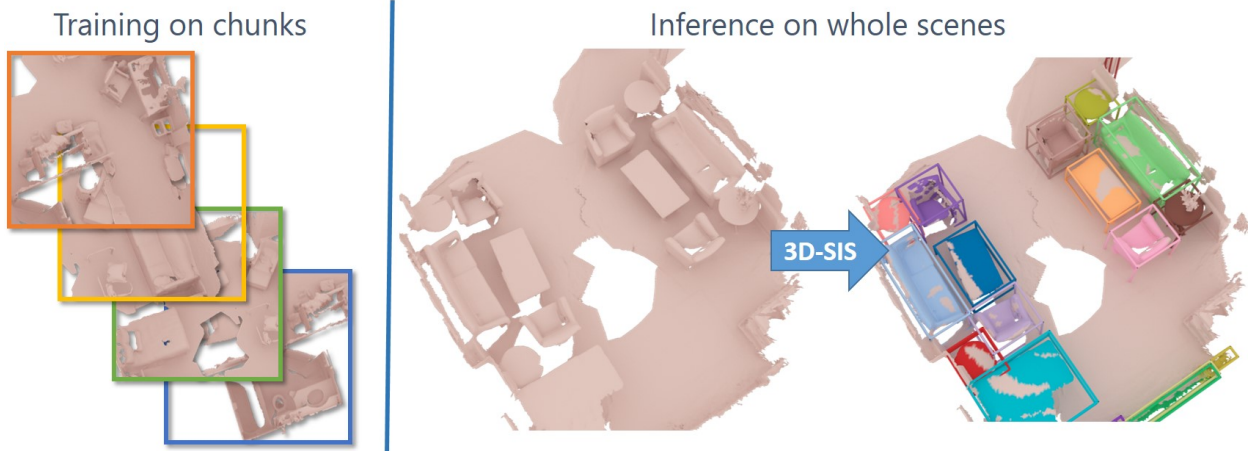


Figure 4: 3D-SIS trains on chunks of a scene, and leverages fully-convolutional backbone architectures to enable inference on a full scene in a single forward pass, producing more consistent instance segmentation results.

A. Appendix

In this supplemental document, we describe the details of our 3D-SIS network architecture in Section A.1. In Section A.2, we describe our training scheme on scene chunks to enable inference on entire test scenes, and finally, in Section A.3, we show additional evaluation on the ScanNet [5] and SUNCG [31] datasets.

A.1. Network Architecture

Table 8 details the layers used in our detection backbone, 3D-RPN, classification head, mask backbone, and mask prediction. Note that both the detection backbone and mask backbone are fully-convolutional. For the classification head, we use several fully-connected layers; however, due to our 3D RoI-pooling on its input, we can run our entire instance segmentation approach on full scans of varying sizes.

We additionally list the anchors used for the region proposal for our model trained on the ScanNet [5] and SUNCG [31] datasets in Tables 7 and 6, respectively. Anchors for each dataset are determined through k -means clustering of ground truth bounding boxes. The anchor sizes are given in voxels, where our voxel size is $\approx 4.69\text{cm}$.

small anchors	big anchors
(8, 6, 8)	(12, 12, 40)
(22, 22, 16)	(8, 60, 40)
(12, 12, 20)	(38, 12, 16)
	(62, 8, 40)
	(46, 8, 20)
	(46, 44, 20)
	(14, 38, 16)

Table 6: Anchor sizes (in voxels) used for SUNCG [31] region proposal. Sizes are given in voxel units, with voxel resolution of $\approx 4.69\text{cm}$

A.2. Training and Inference

In order to leverage as much context as possible from an input RGB-D scan, we leverage fully-convolutional detection and mask backbones to infer instance segmentation on varying-sized scans. To accommodate memory and efficiency constraints during training, we train on chunks of scans, i.e. cropped volumes out of the scans, which we use to generalize to the full scene at test time (see Figure 4). This also enables us to avoid inconsistencies which can arise with individual frame input, with differing views of the same object; with the full view of a test scene, we can more easily predict consistent object boundaries.

A.3. Additional Experiment Details

We additionally evaluate mean average precision on SUNCG [31] and ScanNetV2 [5] using an IoU threshold of 0.5 in Tables 9 and 10. Consistent with evaluation at an IoU threshold of 0.25, our approach leveraging joint color-geometry feature learning and inference on full scans

small anchors	big anchors
(8, 8, 9)	(21, 7, 38)
(14, 14, 11)	(7, 21, 39)
(14, 14, 20)	(32, 15, 18)
	(15, 31, 17)
	(53, 24, 22)
	(24, 53, 22)
	(28, 4, 22)
	(4, 28, 22)
	(18, 46, 8)
	(46, 18, 8)
	(9, 9, 35)

Table 7: Anchor sizes used for region proposal on the ScanNet dataset [5]. Sizes are given in voxel units, with voxel resolution of $\approx 4.69\text{cm}$

layer name	input layer	type	output size	kernel size	stride	padding
geo_1	TSDf	conv3d	(32, 48, 24, 48)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
geo_2	geo_1	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_3	geo_2	conv3d	(32, 48, 24, 48)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
geo_4	geo_3	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_5	geo_4	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_6	geo_5	conv3d	(32, 48, 24, 48)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
geo_7	geo_6	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_8	geo_7	conv3d	(64, 24, 12, 24)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
geo_9	geo_1	conv3d	(32, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_10	geo_2	conv3d	(32, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
geo_11	geo_3	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_12	geo_4	conv3d	(32, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
geo_13	geo_5	conv3d	(32, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
geo_14	geo_6	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_1	projected 2D features	conv3d	(64, 48, 24, 48)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
color_2	color_1	conv3d	(32, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_3	color_2	conv3d	(32, 48, 24, 48)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
color_4	color_3	conv3d	(64, 48, 24, 48)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_5	color_4	maxpool3d	(64, 48, 24, 48)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
color_6	color_5	conv3d	(64, 24, 12, 24)	(2, 2, 2)	(2, 2, 2)	(0, 0, 0)
color_7	color_6	conv3d	(32, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_8	color_7	conv3d	(32, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
color_9	color_8	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
color_10	color_9	maxpool3d	(64, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
concat_1	(geo_14, color_10)	concat	(128, 24, 12, 24)	None	None	None
combine_1	concat_1	conv3d	(128, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
combine_2	combine_1	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
combine_3	combine_2	conv3d	(64, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
combine_4	combine_3	conv3d	(128, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
combine_5	combine_4	conv3d	(64, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
combine_6	combine_5	conv3d	(64, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
combine_7	combine_6	conv3d	(128, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
rpn_1	combine_7	conv3d	(256, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
rpn_cls_1	rpn_1	conv3d	(6, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
rpn_bbox_1	rpn_1	conv3d	(18, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
rpn_2	combine_5	conv3d	(256, 24, 12, 24)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
rpn_cls_2	rpn_2	conv3d	(22, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
rpn_bbox_2	rpn_2	conv3d	(66, 24, 12, 24)	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)
cls_1	combine_7	FC	$128 \times 4 \times 4 \rightarrow 256$	None	None	None
cls_2	cls_1	FC	$256 \rightarrow 256$	None	None	None
cls_3	cls_2	FC	$256 \rightarrow 128$	None	None	None
cls_cls	cls_3	FC	$128 \rightarrow N_{cls}$	None	None	None
cls_bbox	cls_3	FC	$128 \rightarrow N_{cls} \times 6$	None	None	None
mask_geo_1	TSDf	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
mask_geo_2	mask_geo_1	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
mask_color_1	cnn feature	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
mask_color_2	mask_color_1	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
mask_combine_1	(mask_geo_2, mask_color_2)	conv3d	(64, 96, 48, 96)	(3, 3, 3)	(1, 1, 1)	(1, 1, 1)
mask_combine_2	mask_combine_1	conv3d	$(N_{cls}, 96, 48, 96)$	(1, 1, 1)	(1, 1, 1)	(0, 0, 0)

Table 8: 3D-SIS network architecture layer specifications.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	oprop	avg
Seg-Cluster	10.1	10.9	10.4	10.1	10.3	0.0	0.0	12.9	10.7	15.2	0.0	0.0	10.0	0	0.0	11.2	26.1	12.1	0	16.5	0	0	10	7.7
Mask R-CNN [11]	0.0	10.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.8	11.4	10.8	18.8	13.5	0.0	11.5	0.0	0.0	10.7	4.3
SGPN [34]	15.3	28.7	23.7	29.7	17.6	15.1	15.4	0.0	10.8	16.0	0.0	10.9	0.0	0.0	0.0	12.3	33.7	25.9	19.2	31.7	0.0	10.4	10.5	14.2
Ours(geo only)	12.6	60.5	38.6	45.8	21.8	16.8	0.0	0.0	10.0	18.5	10.0	0.0	14.0	0.0	0.0	14.9	64.2	30.8	17.6	35.2	10.0	0.0	16.9	19.1
Ours(geo+1view)	13.9	42.4	35.3	52.9	22	10	0.0	35.0	13.4	21.4	10.0	0.0	13.5	0.0	0.0	10.0	33.8	29.2	17.7	48.3	10.0	16.9	10.0	19.4
Ours(geo+3views)	15.4	58.5	35.5	34.5	24.4	16.6	0.0	20.0	10.0	17.6	10.0	0.0	24.3	0.0	10.0	10.0	34.6	28.5	15.6	40.7	10.0	24.9	15.5	19.8
Ours(geo+5views)	15.5	43.6	43.9	48.1	20.4	10.0	0.0	30.0	10.0	17.4	10.0	0.0	14.5	0.0	10.0	10.0	53.5	35.1	17.2	39.7	10.0	18.9	16.2	20.6

Table 9: 3D instance segmentation on synthetic scans from SUNCG [31]. We evaluate the mean average precision with IoU threshold of 0.5 over 23 classes. Our joint color-geometry feature learning enables us to achieve more accurate instance segmentation performance.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn	avg
Seg-Cluster	10.4	11.9	15.5	12.8	12.4	10.1	10.1	10.3	0.0	11.7	10.4	11.4	0.0	13.9	17.2	11.5	14.2	10.5	10.8
Mask R-CNN [11]	11.2	10.6	10.6	11.4	10.8	10.3	0.0	0.0	11.1	10.1	0.0	10.0	12.8	0.0	18.9	13.1	11.8	11.6	9.1
SGPN [34]	10.1	16.4	20.2	20.7	14.7	11.1	11.1	0.0	0.0	10.0	10.3	12.8	0.0	0.0	48.7	16.5	0.0	0.0	11.3
Ours(geo only)	11.5	17.5	18.0	26.3	0.0	10.1	0.0	10.3	0.0	0.0	0.0	0.0	24.4	21.5	25.0	17.2	34.9	10.1	12.6
Ours(geo+1view)	12.5	15.0	17.8	23.7	0.0	19.0	0.0	11.0	0.0	0.0	10.5	11.1	13.0	19.4	22.5	14.0	40.5	10.1	13.3
Ours(geo+3views)	14.4	19.9	48.4	37.3	16.9	18.3	0.0	11.0	0.0	0.0	10.5	13.1	16.3	15.3	51.3	13.0	12.9	13.4	17.3
Ours(geo+5views)	19.7	37.7	40.5	31.9	15.9	18.1	0.0	11.0	0.0	0.0	10.5	11.1	18.5	24.0	45.8	15.8	23.5	12.9	18.7

Table 10: 3D instance segmentation on real-world scans from ScanNetV2 [5]. We evaluate the mean average precision with IoU threshold of 0.5 over 18 classes. Our explicit leveraging of the spatial mapping between the 3D geometry and color features extracted through 2D convolutions enables significantly improved instance segmentation performance.

enables significantly better instance segmentation performance. We also submit our model the ScanNet Benchmark, and we achieve the state-of-the-art in all three metrics.