3D-SIC: 3D SEMANTIC INSTANCE COMPLETION FOR RGB-D SCANS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper introduces the task of semantic instance completion: from an incomplete RGB-D scan of a scene, we aim to detect the individual object instances comprising the scene and infer their complete object geometry. This enables a semantically meaningful decomposition of a scanned scene into individual, complete 3D objects, including hidden and unobserved object parts. This will open up new possibilities for interactions with objects in a scene, for instance for virtual or robotic agents. To address this task, we propose 3D-SIC, a new data-driven approach that jointly detects object instances and predicts their completed geometry. The core idea of 3D-SIC is a novel end-to-end 3D neural network architecture that leverages joint color and geometry feature learning. The fully-convolutional nature of our 3D network enables efficient inference of semantic instance completion for 3D scans at scale of large indoor environments in a single forward pass. In a series evaluation, we evaluate on both real and synthetic scan benchmark data, where we outperform state-of-the-art approaches by over 15 in mAP@0.5 on ScanNet, and over 18 in mAP@0.5 on SUNCG.

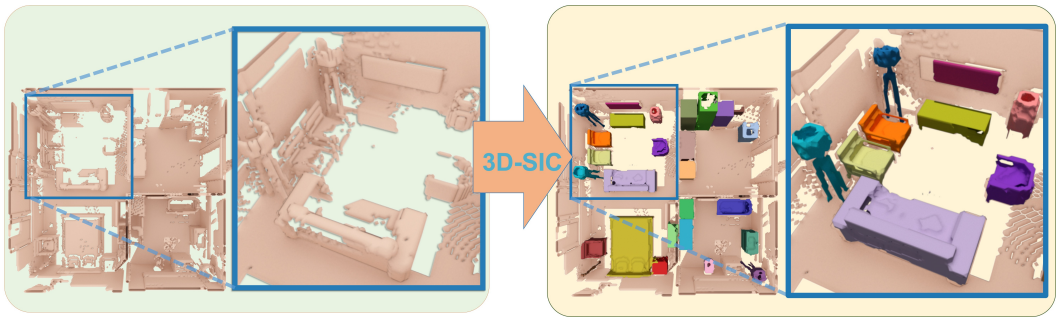

Figure 1: 3D-SIC takes an RGB-D scan as input and predicts its *semantic instance completion*: from the scan's color images and geometry (encoded as a TSDF), objects in the observed scene are detected (as 3D bounding boxes and class labels) and for each object, the complete geometry of that object is predicted as per-instance masks (in both seen and unseen regions).

1 INTRODUCTION

Understanding 3D environments is fundamental to many tasks spanning computer vision, graphics, and robotics. In particular, in order to effectively navigate, and moreover interact with an environment, an understanding of the geometry of a scene and the objects it comprises is essential. This is in contrast to the partial nature of reconstructed RGB-D scans; e.g., due to sensor occlusions. For instance, for a robot exploring an environment, it needs to infer instance-level object segmentation and complete object geometry in order to perform tasks like grasping, or estimate spatial arrangements of individual objects. Additionally, for content creation or mixed reality applications, captured scenes must be decomposable into their complete object components, in order to enable applications such as scene editing or virtual-real object interactions; i.e., it might be insufficient to predict object instance masks only for observed regions.

Thus, we aim to address this task of predicting object detection as well as instance-level completion for an input partial 3D scan of a scene; we refer to this task as **semantic instance completion**. Previous approaches have considered semantic scene segmentation jointly with scan completion (Song et al., 2017; Dai et al., 2018), but lack the notion of individual objects. In contrast, our approach focuses on the instance level, as knowledge of instances is essential towards enabling interaction with the objects in an environment.

In addition, the task of semantic instance completion is not only important towards enabling object-level understanding and interaction with 3D environments, but we also show that the prediction of complete object geometry informs the task of semantic instance segmentation. Thus, in order to address the task of semantic instance completion, we propose to consider instance detection and object completion in an end-to-end, fully differentiable fashion.

From an input RGB-D scan of a scene, our new 3D semantic instance completion network first regresses bounding boxes for objects in the scene, and then performs object classification followed by a prediction of complete object geometry. Our approach leverages a unified backbone from which instance detection and object completion are predicted, enabling information to flow from completion to detection. We incorporate features from both color image and 3D geometry of a scanned scene, as well as a fully-convolutional design in order to effectively predict the complete object decomposition of varying-sized scenes.

In summary, we present a fully-convolutional, end-to-end 3D CNN formulation to predict 3D instance completion that outperforms state-of-the-art, decoupled approaches to semantic instance completion by 15.8 in mAP@0.5 on real-world scan data, and 18.5 in mAP@0.5 on synthetic data:

- We introduce the task of *semantic instance completion* for 3D scans;
- we propose a novel, end-to-end 3D convolutional network which predicts 3D semantic instance completion as object bounding boxes, class labels, and complete object geometry,
- and we show that semantic instance completion task can benefit semantic instance segmentation performance.

2 Related Work

Object Detection and Instance Segmentation Recent advances in convolutional neural networks have now begun to drive impressive progress in object detection and instance segmentation for 2D images (Girshick, 2015; Ren et al., 2015; Liu et al., 2016; Redmon et al., 2016; Lin et al., 2017; He et al., 2017; Lin et al., 2018). Combined with the increasing availability of synthetic and real-world 3D data (Dai et al., 2017a; Song et al., 2017; Chang et al., 2017), we are now seeing more advances in object detection (Song & Xiao, 2014; 2015; Qi et al., 2017) and segmentation for 3D.

Recently, several approaches have been introduced to perform object detection and instance segmentation, applicable to single or multi-frame RGB-D input. Wang et al. (2018) introduced SGPN to operate on point clouds by clustering semantic segmentation predictions. Yi et al. (2018) leverages an object proposal-based approach to predict instance segmentation for a point cloud. Simultaneously, Hou et al. (2019) presented an approach leveraging joint color-geometry feature learning for instance segmentation on volumetric 3D data. Our approach also leverages an anchor-based object proposal mechanism for detection, but we leverage object completion to predict *instance completion*, as well as improve instance segmentation performance.

3D Scan Completion Scan completion of 3D shapes has been a long-studied problem in geometry processing, particularly for cleaning up broken mesh models. In this context, traditional methods have largely focused on filling small holes by locally fitting geometric primitives, or through continuous energy minimization (Sorkine & Cohen-Or, 2004; Nealen et al., 2006; Zhao et al., 2007). Surface reconstruction approaches on point cloud inputs (Kazhdan et al., 2006; Kazhdan & Hoppe, 2013) can also be applied in this fashion to locally optimize for missing surfaces. Other shape completion approaches leverage priors such as symmetry and structural priors (Thrun & Wegbreit, 2005; Mitra et al., 2006; Pauly et al., 2008; Sipiran et al., 2014; Speciale et al., 2016), or CAD model retrieval (Nan et al., 2012; Shao et al., 2012; Kim et al., 2012; Li et al., 2015; Shi et al., 2016) to predict the scan completion.

Recently, methods leveraging generative deep learning have been developed to predict the complete geometry of 3D shapes (Wu et al., 2015; Dai et al., 2017b; Han et al., 2017; Häne et al., 2017). Song et al. (2017) extended beyond shapes to predicting the voxel occupancy for a single depth frame. Recently, Dai et al. (2018) presented a first approach for data-driven scan completion of full 3D scenes, leveraging a fully-convolutional, autoregressive approach. Both Song et al. (2017) and Dai et al. (2018) show that inferring the complete scan geometry can improve 3D semantic segmentation. With our approach for 3D semantic instance completion, this task not only enables new applications requiring instance-based knowledge of a scene (e.g., virtual or robotic interactions with objects in a scene), but we also show that instance segmentation can benefit from instance completion.

3 METHOD OVERVIEW

Our network takes as input an RGB-D scan, and learns to join together features from both the color images as well as the 3D geometry to inform the semantic instance completion. The architecture is shown in Fig. 2. The input 3D scan is encoded as a truncated signed distance field (TSDF) in a volumetric grid. To combine this with color information from the RGB images, we first extract 2D features using 2D convolutional layers on the RGB images, which are then back-projected into a 3D volumetric grid, and subsequently merged with geometric features extracted from the geometry. The joint features are then fed into an encoder-decoder backbone, which leverages a series of 3D residual blocks to learn the representation for the task of semantic instance completion. Objects are detected through anchor proposal and bounding box regression; these predicted object boxes are then used to crop and extract features from the backbone encoder to predict the object class label as well as the complete object geometry for each detected object as per-voxel occupancies.

We adopt in total five losses to supervise the learning process illustrated in Fig. 2. Detection contains three losses: (1) objectness using binary cross entropy to indicate that there is an object, (2) box location using a Huber loss to regress the 3D bounding box locations, and (3) classification of the class label loss using cross entropy. Following detection, the completion head contains two losses: per-instance completion loss using binary cross entropy to predict per-voxel occupancies, and a proxy completion loss using binary cross entropy to classify the surface voxels belonging to all objects in the scene.

Our method operates on a unified backbone for detection followed by instance completion, enabling object completion to inform the object detection process; this results in effective 3D detection as well as instance completion. Its fully-convolutional nature enables us to train on cropped chunks of 3D scans but test on a whole scene in a single forward pass, resulting in an efficient decomposition of a scan into a set of complete objects.

4 NETWORK ARCHITECTURE

From an RGB-D scan input, our network operates on the scan's reconstructed geometry, encoded as a TSDF in a volumetric grid, as well as the color images. To jointly learn from both color and geometry, color features are first extracted in 2D with a 2D semantic segmentation network Paszke et al. (2016), and then back-projected into 3D to be combined with the TSDF features, similar to Dai & Nießner (2018); Hou et al. (2019). This enables complementary semantic features to be learned from both data modalities. These features are then input to the backbone of our network, which is structured in an encoder-decoder style.

The encoder-decoder backbone is composed of a series of five 3D residual blocks, which generates five volumetric feature maps $\mathbb{F} = \{f_i | i = 1 \dots 5\}$. The encoder results in a reduction of spatial dimension by a factor of 4, and symmetric decoder results in an expansion of spatial dimension by a factor of 4. Skip connections link spatially-corresponding encoder and decoder features. For a more detailed description of the network architecture, we refer to the appendix.

4.1 COLOR BACK-PROJECTION

As raw color data is often of much higher resolution than 3D geometry, to effectively learn from both color and geometry features, we leverage color information by back-projecting 2D CNN features learned from RGB images to 3D, similar to Dai & Nießner (2018); Hou et al. (2019). For each

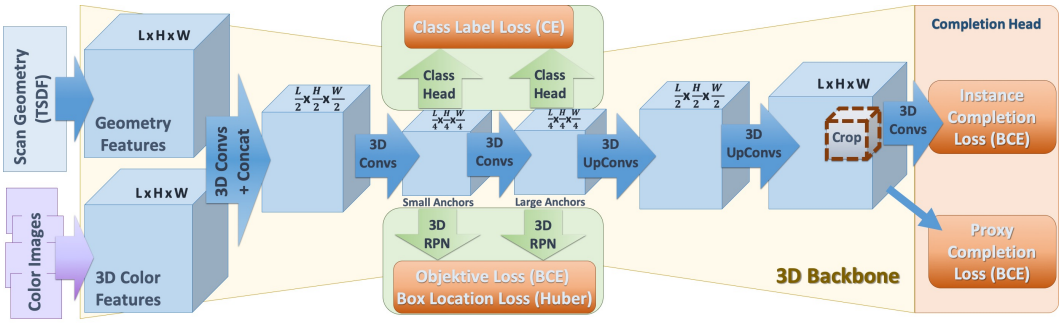

Figure 2: Our 3D-SIC network architecture takes an RGB-D scan as input. Color images are processed with 2D convolutions to spatially compress the information before back-projecting into 3D, to be merged with the 3D geometry features of the scan (following Dai & Nießner (2018); Hou et al. (2019)). These joint features are used for object detection (as 3D bounding boxes and class labels) followed by per-instance geometric completion, for the task of semantic instance completion.

voxel location $v_i = (x, y, z)$ in the 3D volumetric grid, we find its pixel location $p_i = (x, y)$ in 2D views by camera intrinsic and extrinsic matrices. We assign the voxel feature at location v_i with the learned 2D CNN feature vector at p_i. To handle multiple image observations of the same voxel v_i, we apply element-wise view pooling; this also allows our approach to handle a varying number of input images. Note that this back-projection is differentiable, allowing our model to be trained end-to-end and benefit from both RGB and geometric signal.

4.2 Object Detection

For object detection, we predict the bounding box of each detected object as well as the class label. To inform the detection, features are extracted from feature maps F_2 and F_3 of the backbone encoder. We define two set of anchors on these two features maps, $A_s = \{a_i | i = 1 \dots N_s\}$ and $A_b = \{a_i | i = 1 \dots N_b\}$ representing 'small' and 'large' anchors for the earlier F_2 and later F_3, respectively, so that the larger anchors are associated with the feature map of larger receptive field. These anchors $A_s \cup A_b$ are selected through a k-means clustering of the ground truth 3D bounding boxes. For our experiments, we use $N_s + N_b = 9$. From these $N_s + N_b$ clusters, A_b are those with any axis > 1.125m, and the rest are in A_s.

The two features maps F_2 and F_3 are then processed by a 3D region proposal to regress the 3D object bounding boxes. The 3D region proposal first employs a $1 \times 1 \times 1$ convolution layer to output objectness scores for each potential anchor, producing an objectness feature map with $2(N_s + N_b)$ channels for the positive and negative objectness probabilities. Another $1 \times 1 \times 1$ convolution layer is used to predict the 3D bounding box locations as 6-dimensional offsets from the anchors; we then apply a non-maximum suppression based on the objectness scores. We use a Huber loss on the log ratios of the offsets to the anchor sizes to regress the final bounding box predictions:

$$\Delta_x = \frac{\mu - \mu_{anchor}}{\phi_{anchor}} \qquad \Delta_w = \ln(\frac{\phi}{\phi_{anchor}})$$

where μ is the box center point and ϕ is the box width. The final bounding box loss is then:

$$L_\Delta = \begin{cases} \frac{1}{2}\Delta^2, & \text{if } |\Delta| \leq 2 \\ |\Delta|, & \text{otherwise.} \end{cases}$$

Using these predicted object bounding boxes, we then predict the object class labels using features cropped from the bounding box locations from F_2 and F_3. We use a 3D region of interest pooling layer to unify the sizes of the cropped feature maps to a spatial dimension of $4 \times 4 \times 4$ to be input to an object classification MLP.

4.3 Instance Completion

For each object, we infer its complete geometry by predicting per-voxel occupancies. Here, we crop features from feature map F_5 of the backbone, which has a feature map resolution matching

the input spatial resolution, using the predicted object bounding box. These features are processed through a series of five 3D convolutions which maintain the spatial resolution of their input. The complete geometry is then predicted as voxel occupancy using a binary cross entropy loss.

We predict N_{classes} potential object completions for each class category, and select the final prediction based on the predicted object class. We define ground truth bounding boxes b_i and masks m_i as $\gamma = \{(b_i, m_i) | i = 1 \dots N_b\}$. Further, we define predicted bounding boxes $\hat{b_i}$ along with predicted masks $\hat{m}_i$ as $\hat{\gamma} = \{(\hat{b_i}, \hat{m}_i) | i = 1 \dots \hat{N}_b\}$. During training, we only train on predicted bounding boxes that overlap with the ground truth bounding boxes:

$$\Omega = \{(\hat{b_i}, \hat{m}_i, b_i, m_i) \mid \text{IoU}(\hat{b_i}, b_i) \geq 0.5, \quad \forall (\hat{b_i}, \hat{m}_i) \in \hat{\gamma}, \forall (b_i, m_i) \in \gamma\}$$

We can then define the instance completion loss for each associated pair in Ω:

$$L_{\text{compl}} = \frac{1}{|\Omega|} \sum_{\Omega} \text{BCE}(\text{sigmoid}(\hat{m}_i), m_i'), \qquad m_i'(v) = \begin{cases} m_i(v) & \text{if } v \in \hat{b_i} \cap b_i \\ 0 & \text{otherwise.} \end{cases}$$

We further introduce a global geometric completion loss on entire scene level that serves as an intermediate proxy. To this end, we use feature map F_5 as input to a binary cross entropy loss whose target is the composition of all complete object instances of the scene:

$$L_{\text{geometry}} = \text{BCE}(\text{sigmoid}(F_5), \cup_{(b_i, m_i) \in \gamma}).$$

Our intuition is to obtain a strong gradient during training by adding this additional constraint to each voxel in the last feature map F_5. We find that this global geometric completion loss further helps the final instance completion performance; see Sec 6.

5 Network Training

5.1 Data

The input 3D scans are represented as truncated signed distance fields (TSDFs) encoded in volumetric grids. The TSDFs are generated through volumetric fusion (Curless & Levoy, 1996) during the 3D reconstruction process. For all our experiments, we used a voxel size of ≈ 4.7cm and truncation of 3 voxels. We also input the color images of the RGB-D scan, which we project to the 3D grid using their camera poses. We train our model on both synthetic and real scans, computing 9 anchors through k-means clustering; for real-world ScanNet (Dai et al., 2017a) data, this results in 4 small anchors and 5 large anchors, and for synthetic SUNCG (Song et al., 2017) data, this results in 3 small anchors and 6 large anchors.

At test time, we leverage the fully-convolutional design to input the full scan of a scene along with its color images. During training, we use random $96 \times 48 \times 96$ crops ($4.5 \times 2.25 \times 4.5$ meters) of the scanned scenes, along with a greedy selection of ≤ 5 images covering the most object geometry in the crop. Only objects with 50% of their complete geometry inside the crop are considered.

5.2 Optimization

We train our model jointly, end-to-end from scratch. We use an SGD optimizer with batch size 64 for object proposals and 16 for object classification, and all positive bounding box predictions (> 0.5 IoU with ground truth box) for object completion. We use a learning rate of 0.005, which is decayed by a factor of 0.1 every 100k steps. We train our model for 200k steps ($\approx$ 60 hours) to convergence, on a single Nvidia GTX 1080Ti. Additionally, we augment the data for training the object completion using ground truth bounding boxes and classification in addition to predicted object detection.

6 Results

We evaluate our approach on semantic instance completion performance on synthetic scans of SUNCG (Song et al., 2017) scenes as well as on real-world ScanNet (Dai et al., 2017a) scans,

	display	table	bathtub	trashbin	sofa	chair	cabinet	bookshelf	**avg**
Scene Completion + Instance Segmentation	1.65	0.64	4.55	11.25	9.09	9.09	0.18	5.45	5.24
Instance Segmentation + Shape Completion	2.27	3.90	1.14	1.68	14.86	9.93	7.11	3.03	5.49
Ours – 3D-SIC (no color)	13.16	11.28	13.64	18.19	24.79	15.87	8.60	10.60	14.52
Ours – 3D-SIC (no proxy)	21.94	7.63	12.55	28.24	20.38	22.58	13.42	9.51	17.03
Ours – 3D-SIC	**26.86**	**13.21**	**22.31**	**28.93**	**29.41**	**23.64**	**15.35**	**14.48**	**21.77**

Table 1: 3D Semantic Instance Completion on ScanNet (Dai et al., 2017a) scans with Scan2CAD (Avetisyan et al., 2019) targets at mAP@0.5. Our end-to-end formulation achieves significantly better performance than alternative, decoupled approaches that first use state-of-the-art scan completion (Dai et al., 2018) and then instance segmentation (Hou et al., 2019) method or first instance segmentation (Hou et al., 2019) and then shape completion (Dai et al., 2017b).

where we obtain ground truth object locations and geometry from CAD models aligned to ScanNet provided by (Avetisyan et al., 2019). To evaluate semantic instance completion, we use a mean average precision metric on the complete masks (at IoU 0.5). Qualitative results are shown in Figs. 3 and 4.

Comparison to state-of-the-art approaches for semantic instance completion. Tables 1 and 3 evaluate our method against alternatives for the task of semantic instance completion on our real and synthetic scans, respectively, with qualitative comparisons on ScanNet (Dai et al., 2017a) shown in Fig. 3. We compare to state-of-the-art 3D instance segmentation and scan completion approaches used sequentially; that is, first applying a 3D instance segmentation approach followed by a shape completion method on the predicted instance segmentation, as well as first applying a scene completion approach to the input partial scan, followed by a 3D instance segmentation method. For 3D instance segmentation, we evaluate 3D-SIS (Hou et al., 2019), which achieves state-of-the-art performance on a dense volumetric grid representation (the representation we use), and for scan completion we evaluate the 3D-EPN (Dai et al., 2017b) shape completion approach and ScanComplete (Dai et al., 2018) scene completion approach. Our end-to-end approach for semantic instance completion results in significantly improved performance due to information flow from instance completion to object detection. Note that the ScanComplete model applied on ScanNet data is trained on synthetic data, due to the lack of complete ground truth scene data for real-world scans.

Does instance completion help instance segmentation? We can also evaluate our semantic instance completion predictions on the task of semantic instance segmentation by taking the intersection between the predicted complete mask and the input partial scan geometry to be the predicted instance segmentation mask. In Tables 2 and 4, we evaluate our method on 3D semantic instance segmentation with and without predicting instance completion, on ScanNet (Dai et al., 2017a) and SUNCG (Song et al., 2017) scans, as well as against a state-of-the-art 3D volumetric instance segmentation approach 3D-SIS (Hou et al., 2019). Here, we find that predicting instance completion significantly benefits instance segmentation performance.

What is the effect of a global completion proxy? In Tables 1 and 3, we demonstrate the impact of the geometric completion proxy loss; here, we see that this loss improves the semantic instance completion performance on both real and synthetic data. In Tables 2 and 4, we can see that it also improves semantic instance segmentation performance.

Can color input help? We evaluate our approach with and without the color input stream; on both real and synthetic scans, the color input notably improves semantic instance completion performance, as shown in Tables 1 and 3.

	display	table	bathtub	trashbin	sofa	chair	cabinet	bookshelf	**avg**
3D-SIS (Hou et al., 2019)	19.88	16.10	23.69	13.36	20.05	38.59	23.78	10.82	20.78
Ours – 3D-SIC (no completion)	26.97	17.06	23.92	27.38	22.66	35.08	28.14	14.74	24.49
Ours – 3D-SIC (no color)	29.00	17.04	14.46	23.63	**26.09**	41.30	24.88	12.01	23.55
Ours – 3D-SIC (no proxy)	35.59	14.41	17.60	31.53	22.59	**45.32**	24.83	15.52	25.92
Ours – 3D-SIC	**40.84**	**19.68**	**32.02**	**37.98**	24.49	43.49	**27.13**	**18.52**	**30.52**

Table 2: 3D Semantic Instance Segmentation on ScanNet (Dai et al., 2017a) scans at mAP@0.5. We evaluate our instance completion predictions on the task of semantic instance segmentation. Here, predicting instance completion notably increases performance from only predicting instance segmentation (no completion).

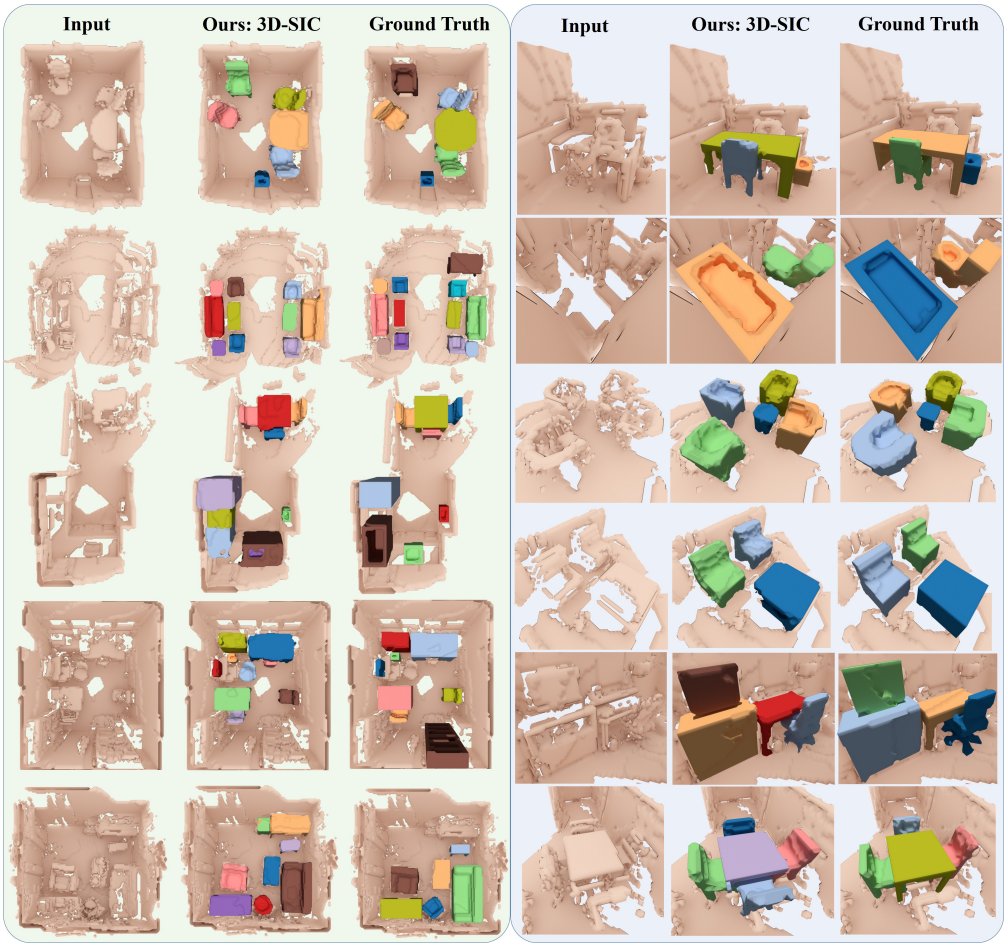

Figure 3: Qualitative results on real-world ScanNet (Dai et al., 2017a) scenes, with close-ups shown on the right. Note that different colors denote distinct object instances in the visualization. Our approach effectively predicts complete individual object geometry, including missing structural components (e.g., missing chair legs), across varying degrees of partialness in input scan observations.

	cab	bed	chair	sofa	tabl	door	wind	bkshf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	oprop	**avg**
SC + IS	3.0	0.6	19.5	0.8	18.1	**15.9**	0.00	0.0	1.0	2.3	3.0	0.0	0.5	0.0	9.2	10.4	23.9	3.4	9.1	0.0	0.0	0.0	9.1	5.5
IS + SC	0.3	0.0	7.4	0.4	3.0	9.1	0.0	0.0	0.2	0.0	0.0	0.0	2.3	0.0	3.0	0.0	2.6	0.0	1.8	0.0	0.0	0.0	4.6	1.5
no color	**19.05**	41.8	38.2	11.9	23.9	9.1	0.0	0.0	2.5	**21.6**	9.1	0.0	**12.6**	**4.6**	49.4	33.8	63.4	**36.9**	**38.8**	14.7	15.9	0.0	**23.8**	20.5
no proxy	12.9	46.1	**39.4**	26.8	**30.3**	1.0	15.9	0.0	9.1	18.2	3.4	0.0	1.1	0.0	43.6	**34.0**	**69.1**	32.4	29.6	**31.1**	14.6	0.0	23.3	20.9
Ours	14.7	**58.3**	38.2	**28.8**	29.5	0.0	**15.9**	**54.6**	**9.1**	12.1	**9.1**	**0.0**	6.2	0.0	**49.4**	33.5	61.2	34.5	29.5	27.1	**16.4**	**0.0**	23.5	**24.0**

Table 3: 3D Semantic Instance Completion on synthetic SUNCG (Song et al., 2017) scans at mAP@0.5. Our semantic instance completion approach achieves significantly better performance than alternative approaches with decoupled state-of-the-art scan completion (SC) (Dai et al., 2018) followed by instance segmentation (IS) (Hou et al., 2019), as well as instance segmentation followed by shape completion (Dai et al., 2017b). We additionally evaluate our approach without color input (no color) and without a completion proxy loss on the network backbone (no proxy).

7 Limitations

Our approach shows significant potential in the task of semantic instance completion, but several important limitations still remain. First, we output a binary mask for the complete object geometry, which can limit the amount of detail represented by the completion; other 3D representations such as distance fields or sparse 3D representations (Graham & van der Maaten, 2017) could potentially resolve greater geometric detail. Our approach also uses axis-aligned bounding boxes for object detection; it would be helpful to additionally predict the object orientation. We also do not

	cab	bed	chair	sofa	tabl	door	wind	bkshf	cntr	desk	shlf	curt	drsr	mirr	tv	nigh	toil	sink	lamp	bath	ostr	ofurn	oprop	**avg**
3D-SIS	15.5	43.6	43.9	48.1	20.4	10.0	0.0	30.0	10.0	17.4	10.0	0.0	14.50	0.0	10.0	10.0	53.5	35.1	17.2	39.7	10.0	18.9	16.2	20.6
no cmpl	19.4	66.0	**54.9**	57.6	34.5	**25.4**	0.0	0.0	5.3	24.1	**23.2**	0.0	14.8	0.0	9.1	23.0	52.1	32.5	18.2	34.6	18.2	**19.0**	16.9	23.9
no color	22.6	72.7	46.8	**79.9**	25.6	9.1	0.0	0.0	7.6	34.3	12.8	0.0	28.1	**4.6**	51.3	43.7	72.3	57.0	**51.1**	63.6	18.2	0.0	25.2	31.6
no proxy	**28.0**	**81.1**	42.7	70.3	**40.6**	12.6	15.9	0.0	**19.2**	**34.6**	8.7	0.0	25.3	0.0	44.8	41.6	71.9	55.7	42.1	62.7	**26.0**	0.0	25.9	32.6
Ours	24.9	72.7	39.6	72.3	34.8	0.0	**15.9**	**54.6**	18.6	24.6	9.1	**0.0**	**50.9**	0.0	**53.0**	**49.5**	**72.7**	**57.5**	41.9	**81.8**	25.0	9.1	**26.0**	**36.3**

Table 4: 3D Semantic Instance Segmentation on synthetic SUNCG (Song et al., 2017) scans at mAP@0.5. We compare to 3D-SIS (Hou et al., 2019), a state-of-the-art approach for 3D semantic instance segmentation on volumetric input, and additionally evaluate our approach without completion (no cmpl), without color input (no color), and without a completion proxy loss on the network backbone (no proxy). Predicting instance completion notably benefits instance segmentation.

Figure 4: Qualitative results on SUNCG dataset (Song et al., 2017) (left: full scans, right: close-ups). We sample RGB-D images to reconstruct incomplete 3D scans from random camera trajectories inside SUNCG scenes. Note that different colors denote distinct object instances in the visualization.

consider object movement over time, which contains significant opportunities for semantic instance completion in the context of dynamic environments.

8 Conclusion

In this paper, we introduced the new task of semantic instance completion along with 3D-SIC, a new 3D CNN-based approach for this task, which jointly detects objects and predicts their complete geometry. Our proposed 3D CNN learns from both color and geometry features to detect and classify objects, then predict the voxel occupancy for the complete geometry of the object in end-to-end fashion, which can be run on a full 3D scan in a single forward pass. On both real and synthetic scan data, we significantly outperform alternative approaches for semantic instance completion. We believe that our approach makes an important step towards higher-level scene understanding and helps to enable object-based interactions and understanding of scenes, which we hope will open up new research avenue.

REFERENCES

Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312. ACM, 1996.

Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017a.

Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017b.

Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. High Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *arXiv preprint arXiv:1704.00710*, 2017.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 2980–2988. IEEE, 2017.

Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

Young Min Kim, Niloy J Mitra, Dong-Ming Yan, and Leonidas Guibas. Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)*, 31(6):138, 2012.

Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34, pp. 435–446. Wiley Online Library, 2015.

Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, pp. 4, 2017.

Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. In *ACM Transactions on Graphics (TOG)*, volume 25, pp. 560–568. ACM, 2006.

Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6):137, 2012.

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pp. 381–389. ACM, 2006.

Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. Discovering structural regularity in 3d geometry. In *ACM transactions on graphics (TOG)*, volume 27, pp. 43. ACM, 2008.

Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, 2012.

Yifei Shi, Pinxin Long, Kai Xu, Hui Huang, and Yueshan Xiong. Data-driven contextual modeling for 3d scene understanding. *Computers & Graphics*, 55:55–67, 2016.

Ivan Sipiran, Robert Gregor, and Tobias Schreck. Approximate symmetry detection in partial 3d meshes. In *Computer Graphics Forum*, volume 33, pp. 131–140. Wiley Online Library, 2014.

Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pp. 634–651. Springer, 2014.

Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. *arXiv preprint arXiv:1511.02300*, 2015.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Shape Modeling Applications, 2004. Proceedings*, pp. 191–199. IEEE, 2004.

Pablo Speciale, Martin R Oswald, Andrea Cohen, and Marc Pollefeys. A symmetry prior for convex variational 3d reconstruction. In *European Conference on Computer Vision*, pp. 313–328. Springer, 2016.

Sebastian Thrun and Ben Wegbreit. Shape from symmetry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pp. 1824–1831. IEEE, 2005.

Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2569–2578, 2018.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1912–1920, 2015.

Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. *arXiv preprint arXiv:1812.03320*, 2018.

Wei Zhao, Shuming Gao, and Hongwei Lin. A robust hole-filling algorithm for triangular mesh. *The Visual Computer*, 23(12):987–997, 2007.

A Appendix

In this appendix, we detail our 3D-SIC network architecture in Section B; in Section C, we provide run-time results of our approach.

B Network Architecture

small anchors	big anchors
(9, 10, 9)	(47, 20, 23)
(17, 21, 17)	(23, 20, 47)
(12, 19, 13)	(16, 18, 30)
(16, 12, 15)	(17, 38, 17)
	(30, 18, 16)

Table 5: Anchor sizes used for region proposal on the ScanNet dataset (Dai et al., 2017a). Sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

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 (Song et al., 2017) region proposal. Sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

Table 10 details the layers used in our backbone. 3D-RPN, classification head, and mask completion head are described in Table 11. Additionally, we leverage the residual blocks in our backbone, which is listed in Table 9. Note that both the backbone and mask completion head are fully-convolutional. For the classification head, we use several fully-connected layers; however, we leverage 3D RoI-pooling on its input, we can run our method on large 3D scans of varying sizes in a single forward pass.

We additionally list the anchors used for the region proposal for our model trained on our ScanNet-based semantic instance completion benchmark (Avetisyan et al., 2019; Dai et al., 2017a) and SUNCG (Song et al., 2017) datasets in Tables 5 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 ≈ 4.69cm.

C INFERENCE TIMING

In this section, we present the inference timing with and without color projection in Table 7 and 8. Note that our color projection layer currently projects the color signal into 3D space sequentially, and can be further optimized using CUDA, so that it can project the color features back to 3D space in parallel. A scan typically contains several hundreds of images; hence, this optimization could significantly further improve inference time.

physical size (m)	4.7 x 7.7	7.9 x 9.6	10.7 x 16.5
voxel resolution	100 x 164	168 x 204	228 x 352
image count	49	107	121
color projection (s)	1.43	5.16	11.78
forward pass (s)	0.19	0.34	0.64
total (s)	1.62	5.50	12.42

Table 7: Inference timing on entire large scans with RGB input. Timings are given in seconds, physical sizes are given in meters and spatial sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

physical size (m)	5.8 x 6.4	8.3 x 13.9	10.9 x 20.1
voxel resolution	124 x 136	176 x 296	232 x 428
forward pass (s)	0.15	0.37	0.72

Table 8: Inference time on entire scenes without color signal. Timings are given in seconds, physical sizes are given in meters and spatial sizes are given in voxel units, with voxel resolution of ≈ 4.69cm

ResBlock	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
convres0	CNN feature	Conv3d	(N,X,Y,Z)	(N/2,X,Y,Z)	(1,1,1)	(1,1,1)	(0,0,0)
normres0	convres0	InstanceNorm3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
relures0	normres0	ReLU	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
convres1	relures0	Conv3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	(3,3,3)	(1,1,1)	(1,1,1)
normres1	convres1	InstanceNorm3d	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
relures1	normres1	ReLU	(N/2,X,Y,Z)	(N/2,X,Y,Z)	None	None	None
convres2	relures1	Conv3d	(N/2,X,Y,Z)	(N,X,Y,Z)	(1,1,1)	(1,1,1)	(0,0,0)
normres2	convres2	InstanceNorm3d	(N,X,Y,Z)	(N,X,Y,Z)	None	None	None
relures2	normres2	ReLU	(N,X,Y,Z)	(N,X,Y,Z)	None	None	None

Table 9: Residual block specification in 3D-SIC.

BackBone	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
geometry0	TSDF	Conv3d	(2,96,48,96)	(32,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm0	geometry0	InstanceNorm3d	(32,48,24,48)	(32,48,24,48)	None	None	None
relu0	norm0	ReLU	(32,48,24,48)	(32,48,24,48)	None	None	None
block0	relu0	ResBlock	(32,48,24,48)	(32,48,24,48)	None	None	None
color1	CNN feature	Conv3d	(128,96,48,96)	(32,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm1	color1	InstanceNorm3d	(32,48,24,48)	(32,48,24,48)	None	None	None
relu1	norm1	ReLU	(32,48,24,48)	(32,48,24,48)	None	None	None
block1	relu1	ResBlock	(32,48,24,48)	(32,48,24,48)	None	None	None
concat2	(block0,block1)	Concatenate	(32,48,24,48)	(64,48,24,48)	None	None	None
combine2	concat2	Conv3d	(64,48,24,48)	(128,24,12,24)	(2,2,2)	(2,2,2)	(0,0,0)
norm2	combine2	InstanceNorm3d	(128,24,12,24)	(128,24,12,24)	None	None	None
relu2	norm2	ReLU	(128,24,12,24)	(128,24,12,24)	None	None	None
block2	relu2	ResBlock	(128,24,12,24)	(128,24,12,24)	None	None	None
encoder3	block2	Conv3d	(128,24,12,24)	(128,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm3	combine3	InstanceNorm3d	(128,24,12,24)	(128,24,12,24)	None	None	None
relu3	norm3	ReLU	(128,24,12,24)	(128,24,12,24)	None	None	None
block3	relu3	ResBlock	(128,24,12,24)	(128,24,12,24)	None	None	None
skip4	(block, block3)	Conv3d	(128,24,12,24)	(64,48,24,48)	(2,2,2)	(2,2,2)	(0,0,0)
norm4	combine4	InstanceNorm3d	(64,48,24,48)	(64,48,24,48)	None	None	None
relu4	norm4	ReLU	(64,48,24,48)	(64,48,24,48)	None	None	None
block4	relu4	ResBlock	(64,48,24,48)	(64,48,24,48)	None	None	None
concat5	(block2,block4)	Concatenate	(64,48,24,48)	(128,48,24,48)	None	None	None
decoder5	block5	ConvTranspose3d	(128,48,24,48)	(32,96,48,96)	(2,2,2)	(2,2,2)	(0,0,0)
norm5	combine5	InstanceNorm3d	(32,96,48,96)	(32,96,48,96)	None	None	None
relu5	norm5	ReLU	(32,96,48,96)	(32,96,48,96)	None	None	None
block5	relu5	ResBlock	(32,96,48,96)	(32,96,48,96)	None	None	None
proxy5	block5	ConvTranspose3d	(32,96,48,96)	(1,96,48,96)	(1, 1, 1)	(1,1,1)	(0,0,0)

Table 10: Backbone layer specifications in 3D-SIC.

RPN	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
rpn6	block2	Conv3d	(128,24,12,24)	(256,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm6	rpn6	InstanceNorm3d	(256,24,12,24)	(256,24,12,24)	None	None	None
relu6	norm6	ReLU	(256,24,12,24)	(256,24,12,24)	None	None	None
rpncls7a	relu6	Conv3d	(256,24,12,24)	(8,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm7a	rpncls7a	InstanceNorm3d	(8,24,12,24)	(8,24,12,24)	None	None	None
rpnbbox7b	relu6	Conv3d	(24,24,12,24)	(24,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm7b	rpnbbox7b	InstanceNorm3d	(24,24,12,24)	(24,24,12,24)	None	None	None
rpn8	block3	Conv3d	(128,24,12,24)	(256,24,12,24)	(3,3,3)	(1,1,1)	(1,1,1)
norm8	rpn8	InstanceNorm3d	(256,24,12,24)	(256,24,12,24)	None	None	None
relu8	norm8	ReLU	(256,24,12,24)	(256,24,12,24)	None	None	None
rpncls9a	relu8	Conv3d	(256,24,12,24)	(8,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm9a	rpncls9a	InstanceNorm3d	(10,24,12,24)	(10,24,12,24)	None	None	None
rpnbbox9b	relu8	Conv3d	(30,24,12,24)	(30,24,12,24)	(1,1,1)	(1,1,1)	(0,0,0)
norm9b	rpnbbox9b	InstanceNorm3d	(30,24,12,24)	(30,24,12,24)	None	None	None
Class Head	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
roipool10	block2/block3	RoI Pooling	(64,arbitrary)	(64, 4, 4, 4)	None	None	None
flat10	roipool10	Flat	(64,4,4,4)	(4096)	None	None	None
cls10a	flat10	Linear	(4096)	(256)	None	None	None
relu10a	cls10a	ReLU	(256)	(256)	None	None	None
cls10b	relu10a	Linear	(256)	(128)	None	None	None
relu10b	cls10b	ReLU	(128)	(128)	None	None	None
cls10c	relu10b	Linear	(128)	(128)	None	None	None
relu10c	cls10c	ReLU	(128)	(128)	None	None	None
clscls10	relu10c	Linear	(128)	(8)	None	None	None
clsbbox10	relu10c	Linear	(128)	(48)	None	None	None
Mask Head	Input Layer	Type	Input Size	Output Size	Kernel Size	Stride	Padding
mask11	block2/block3	Conv3d	(N,arbitrary)	(N,arbitrary)	(9,9,9)	(1,1,1)	(4,4,4)
norm11	mask11	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu11	norm11	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask12	relu11	Conv3d	(N,arbitrary)	(N,arbitrary)	(7,7,7)	(1,1,1)	(3,3,3)
norm12	mask12	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu12	norm12	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask13	relu12	Conv3d	(N,arbitrary)	(N,arbitrary)	(5,5,5)	(1,1,1)	(2,2,2)
norm13	mask13	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu13	norm13	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask14	relu13	Conv3d	(N,arbitrary)	(N,arbitrary)	(3,3,3)	(1,1,1)	(1,1,1)
norm14	mask14	InstanceNorm3d	(N,arbitrary)	(N,arbitrary)	None	None	None
relu14	norm14	ReLU	(N,arbitrary)	(64,arbitrary)	None	None	None
mask15	relu14	Conv3d	(N,arbitrary)	(N,arbitrary)	(1,1,1)	(1,1,1)	(0,0,0)

Table 11: Head layer specifications of RPN, Classification and Mask Completion in 3D-SIC.