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

1. Introduction

Semantic scene understanding is critical to many real- world computer vision applications. It is fundamental to- wards 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 scenar- ios using mobile and AR/VR devices. In all these applica- tions, we would not only want semantic inference of single images, but importantly, also require understanding of spa- tial relationships and layouts of objects in 3D environments.

With recent breakthroughs in deep learning and the in- creasing 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 [19, 13, 21], object detection [11, 26], and semantic instance segmentation [12]. The primary focus of these impressive works lies in the analysis of visual input from a single im- age; however, in many real-world computer vision scenar- ios, 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 seg- mentation, it is quite disadvantageous to run methods in- dependently on single images given that instance associa- tions 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 odom- etry 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 in- stances 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 fea- tures 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 inter- est pooling layer (3D-RoI) to infer object bounding box lo- cations, 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 result- ing 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 ef- ficiently infer predictions on large 3D environments in a sin- gle shot. In comparison to state-of-the-art approaches that operate on individual RGB images, such as Mask R-CNN [12], 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 archi- tectures, we have now seen impressive progress on object detection and semantic instance segmentation in 2D im- ages [11, 27, 18, 26, 16, 12, 17]. Notably, Ren et al. [27] 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 [12] ex- panded this work to semantic instance segmentation by pre- dicting a per-pixel object instance masks. An alternative direction for detection is the popular Yolo work [26], which also defines anchors on grid cells of an image.

This progress in 2D object detection and instance seg- mentation has inspired work on object detection and seg- mentation in the 3D domain, as we see more and more video and RGB-D data become available. Song et al. pro- posed Sliding Shapes to predict 3D object bounding boxes from single RGB-D frame input with handcrafted feature design [30], and then expanded the approach to operate on learned features [31]. The latter direction leverages the RGB frame input to improve classification accuracy of de- tected objects; in contrast to our approach, there is no ex- plicit spatial mapping between RGB and geometry for joint feature learning. An alternative approach is taken by Frus- tum PointNet [22], where detection is performed a 2D frame and then back-projected into 3D from which final bound- ing box predictions are refined. Wang et al. [35] base their SGPN approach on semantic segmentation from a Point- Net++ 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 [15]. 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 vol- umetric grids, which for instance embed a surface repre- sentation as an implicit signed distance field [4]. With the availability of 3D shape databases [36, 3, 32] and anno- tated RGB-D datasets [29, 1, 5, 2], these network archi- tectures are now being used for 3D object classification [36, 20, 24, 28], semantic segmentation [5, 34, 6], and ob- ject or scene completion [8, 32, 9]. An alternative represen- tation to volumetric grids are the popular point-based archi- tectures, such as PointNet [23] or PointNet++ [25], which leverage a more efficient, although less structured, repre- sentation of 3D surfaces. Multi-view approaches have also been proposed to leverage RGB or RGB-D video informa- tion. Su et al. proposed one of the first multi-view archi- tectures for object classification by view-pooling over 2D predictions [33], and Kalogerakis et al. recently proposed an approach for shape segmentation by projecting predicted 2D confidence maps onto the 3D shape, which are then ag- gregated through a CRF [14]. 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 geom- etry 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 con- sider synthetic ground truth data from SUNCG [32], 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 convolu- tions 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, associ- ated 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 fash- ion. In Sec. 6, we describe the training and implementation details of our approach, and in Sec. 7, we evaluate our ap- proach.

4. Training Data

Data Representation We use a truncated sign distance field (TSDF) representation to encode the reconstructed ge- ometry of the 3D scan inputs. The TSDF is stored in a reg- ular volumetric grid with truncation of 3 voxels. In addi- tion 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 re- spective 3D reconstruction algorithm.

For the training data, we subdivide each 3D scan into chunks of 4.5m $\times$ 4.5m $\times$ 2.25m, 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 $\approx$ 4.69cm3 . In our experiments, for training, we associate 5 RGB images at a resolution of 328x256 pixels in every chunk, with training images selected based on the average voxel-to-pixel cover- age 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 (approxi- mately the height of a room), with the voxel size also fixed at 4.69cm3 . Additionally, at test time, we use all RGB im- ages available for inference. In order to evaluate our algo- rithm, 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 [32] 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 < 600m3). 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 vir- tual 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 associa- tions 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 fioor cate- gories which do not have well-defined instances.

Real-world Data For training and evaluating our algo- rithm 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 con- straints for our 3D region proposal.

We follow the public train/val/test split originally pro- posed by ScanNet of 1045 (train), 156 (val), 312 (test)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 Scan- Net 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 com- posed 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 back- bones are then fed into the classification and the voxel- instance mask prediction heads, respectively.

The object detection component of the network com- prises the detection backbone, a 3D region proposal net- work (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 fea- tures which are input to the 3D-RPN and 3D-RoI to pre- dict bounding box locations and object class labels, respec- tively. 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 an- chor. If it does, a second loss regresses the 3D object bound- ing box; if not, no additional loss is considered. In addi- tion, we classify the 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 pre- diction 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 predic- tions end-to-end; see Sec. 6 for more detail. In the follow- ing, 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

To this end, we first pre-train a 2D semantic segmen- tation network based on the ENet architecture [21]. The 2D architecture takes single 256 × 328 RGB images as in- put, 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 vox- els. In order to combine features from multiple views, we perform view pooling through an element-wise max pool- ing over all RGB images available.

For training, the voxel volume is fixed to 96 × 96 × 48 voxels, resulting in a 128 × 96 × 96 × 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 resolu- tion is dynamic, given by the spatial extent of the environ- ment; 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 pro- vided; we use them directly for our back-projection layer. For SUNCG [32], 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 pro- cessed symmetrically with a 3D ResNet block before join- ing 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 in- put train chunk of 96 × 96 × 48, we obtain a features of size 24 ×24 × 12). We then apply another 3D convolutional layer, maintaining the same spatial dimensions, to provide fea- tures maps with larger receptive fields. We define anchors on these two feature maps, splitting the anchors into ‘small’ and ‘large’ anchors (small anchors < 1m3), with small an- chors associated with the first feature map of smaller re- ceptive 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 de- tection: 3D bounding box regression and classification.

The instance segmentation backbone also takes the 3D geometry and the back-projected 2D CNN features as in- put. The geometry and color features are first processed independently with two 3D convolutions, and then concate- nated channel-wise and processed with another two 3D con- volutions 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 in- put to predict the final instance mask segmentation.

In contrast to single backbone, we found that this two- backbone structure both converged more easily and pro- duced significantly better instance segmentation perfor- mance (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 $2 \times$ Nanchors , where Nanchors = (3, 11) for small and large anchors, respectively. These represent the positive and negative scores of object- ness 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 $6 \times$ Nanchors , which represent the 3D bounding box locations as ($\triangle$x, $\triangle$y, $\triangle$z, $\triangle$w, $\triangle$h, $\triangle$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 IoU $>$ 0.35, we consider an anchor to be positive (and it will be regressed to the associated box), and if the 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 objec- tiveness, and for the bounding box regression we use a Huber loss on the prediction ($\triangle$x, $\triangle$y, $\triangle$z, $\triangle$w, $\triangle$h, $\triangle$l) against the log ratios of the ground truth box and anchors ($\triangle$t , $\triangle$t , $\triangle$t , $\triangle$t , $\triangle$t , $\triangle$t), where

$\triangle x =$ $\triangle w = \ln( )$ (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 dimen- sionality 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.

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 main- tain the same spatial resolutions, in order to maintain spa- tial correspondence with the raw inputs, which we found to significantly improve performance. We then use the pre- dicted 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 cth channel for pre- dicted 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 overlap- ping 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 (us- ing 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, mo- mentum 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 suppres- sion 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 back- bone, 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 3D detection and in- stance segmentation predictions, comparing to several state- of-the-art approaches, on synthetic scans of SUNCG [32] data and real-world scans from the ScanNetV2 dataset [5]. To compare to previous approaches that operate on single RGB or RGB-D frames (Mask R-CNN [12], Deep Sliding Shapes [31], Frustum PointNet [22]), we first obtain predic- tions on each individual frame, and then merge all predic- tions together in the 3D space of the scene, merging predic- tions if the predicted class labels match and the IoU > 0.5. We further compare to SGPN [35] which performs instance segmentation on 3D point clouds. For both detection and in- stance segmentation tasks, we project all results into a voxel space of 4.69cm voxels and evaluate them with a mean aver- age precision metric. We additionally show several variants of our approach for learning from both color and geome- try 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 segmenta- tion 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 [32], using 23 class categories. Table 4 shows 3D detection per- formance compared to state-of-the-art approaches which operate on single frames. Table 1 shows a quantitative eval- uation of our approach, the SGPN for point cloud instance segmentation [35], their proposed Seg-Cluster baseline, and Mask R-CNN [12] 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.

7.2. 3D Instance Analysis on Real-World Scans

We further evaluate our approach on ScanNet dataset [5], which contains 1513 real-world scans. For training and evaluation, we use ScanNetV2 annotated ground truth as well as the proposed 18-class instance benchmark. We show qualitative results in Figure 3. In Table 5, we quantita- tively evaluate our object detection against Deep Sliding Shapes and Frustum PointNet, which operate on RGB-D frame, as well as Mask R-CNN [12] projected to 3D. 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 SGPN instance segmentation [35], their proposed Seg- Cluster baseline, and Mask R-CNN [12] projected into 3D. Our formulation for learning from both color and geometry features brings notable improvement over state of the art.

Finally, we evaluate our model on the ScanNetV2 3D in- stance segmentation benchmark on the hidden test set; see Table 2. Our final model (geo+5views) significantly outper- forms previous (Mask R-CNN [12], SGPN [35]) and con- current (MTML, 3D-BEVIS [10], R-PointNet [37]) state- of-the-art methods in mAP@0.5. ScanNetV2 benchmark data was accessed on 12/17/2018.

8. Conclusion

In this work, we introduce 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 to jointly learn features from RGB and geometry data using multi-view RGB-D input recorded with commodity RGB-D sensors. The network is fully-convolutional, and thus can run efficiently in a sin- gle shot on large 3D environments. In comparison to exist- ing state-of-the-art methods that typically operate on single RGB frame, we achieve significantly better 3D detection and instance 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..