GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud – Supplementary Material

Li Yi[1] Wang Zhao[2] He Wang[1] Minhyuk Sung[1] Leonidas Guibas[1,3]
[1]Stanford University [2]Tsinghua University [3]Facebook AI Research

This document provides a list of supplemental materials that accompany the main paper.

- **PartNet** - We provide more details about the PartNet dataset in Section 1.
- **Runtime Comparison** - We provide the runtime comparison between GSPN and a 3D bounding box regression baseline in Section 2.
- **GSPN Model Design** - We discuss the influence of different design choices in our Generative Shape Proposal Network (GSPN) in Section 3.
- **Additional Comparison with 3D Object Detection Approaches** - We compare our approach with previous state-of-the-art object detection approaches on ScanNet and show the comparison in Section 4.
- **Additional Comparison on NYUv2** - We compare our approach with additional baseline methods on NYUv2 dataset to validate its effectiveness. (see Section 5).
- **Architecture Details** - We provide architecture details about GSPN and Region-based PointNet (R-PointNet) in Section 6.
- **Additional Visualizations** - In Section 7, we provide additional visualizations for the instance segmentation results generated by our R-PointNet on different datasets.

1. PartNet

PartNet is a consistent, large-scale dataset of 3D objects annotated with fine-grained, instance-level, and hierarchical 3D part information. The dataset consists of 573,585 part instances over 26,671 3D models covering 24 object categories. This dataset is very suitable for evaluating instance segmentation algorithms since each part instance is segmented out with their semantic labels annotated. Also different from objects in a scene, parts in an object are more structured but more adjacent with each other with usually complex topology, thus providing a new angle for evaluating instance segmentation frameworks. For this reason, we evaluate our framework on PartNet to test its generalizability.

2. Runtime Comparison

We have shown a comparison of different 3D object proposal approaches in the main paper. GSPN clearly outperforms the basic 3D bounding box regression baseline by a large margin, both w.r.t. the proposal quality and w.r.t. the final segmentation mAP. To compare the computation cost of GSPN with the bounding box regression baseline, we measure the runtime of both proposal approaches on a single TitanX GPU following the same configuration. On average, it takes the bounding box regression baseline 0.6s to process a single scene in ScanNet while GSPN costs 0.88s. This shows GSPN achieves significantly better performance without introducing much additional computation cost compared with the baseline.

3. GSPN Model Design

In GSPN, we use a variation of CVAE which takes multi-scale context around each seed point as input and predicts the corresponding object center before the generation process. To validate our design choices, we experiment with three additional settings: replacing CVAE with naive encoder-decoder structure (E-D), using a single scale context as input, and removing the center prediction network from GSPN. In the naive encoder-decoder structure, we encode the multi-scale context into a latent code and directly decode the object instance from the code. For the second setting, we choose the largest context as input to guarantee the largest objects in the scene can be roughly included. We use chamfer distance (CD) and mIoU as our evaluation metrics, where CD is computed between the generated shape and the ground truth shape, and mIoU is computed between the induced axis-aligned bounding boxes from the shapes. The comparison is reported in Table 1.

When replacing CVAE with E-D, we observe difficulties for the network to produce good shape reconstructions. We conjecture the instance guidance in CVAE training makes such a difference. The recognition network encodes the ground truth instances into a proposal distribution, which provides strong guidance to the prior network to learn a semantic meaningful and compact prior shape distribution. We also find using single context fails to encode small objects well, leading to worse object proposals. The experiment also shows it is important to explicitly predict object

center to learn the generation process in a normalized space, otherwise the reconstruction quality will be influenced.

	Ours	E-D	1-Context	No Center Pred
CD	**0.0450**	0.0532	0.0524	0.0571
mIoU	**0.581**	0.408	0.486	0.409

Table 1. Evaluation of different generative model designs. Using CVAE with multi-scale context inputs, along with a center prediction network for translation factorization, gives us the best proposal generation quality.

4. Additional Comparison with 3D Object Detection Approaches

To further justify the effectiveness of GSPN, we compare our approach with previous state-of-the-art object detection approaches on ScanNet, including Deep Sliding Shapes [5] and Frustum Pointnet [4], which operate on RGB-D frame data, as well as MaskR-CNN [3] projected to 3D. We evaluate using mean average precision (mAP) over 18 semantic classes. In contrast to previous approaches operating on individual frames, GSPN proposes objects directly in 3D, exploiting generative shape priors and achieving significantly better detection performance as shown in Table 2.

	Ours	[37]	[30]	[16]
mAP@0.25	**30.6**	15.2	19.8	17.3
mAP@0.5	**17.7**	6.8	10.8	10.5

Table 2. Additional comparison with object detection baselines.

5. Additional Comparison on NYUv2

To better evaluate our approach, we provide an additional comparison on semantic instance segmentation with one of the previous state-of-the-art approaches, Mask R-CNN, on the NYUv2 dataset. AP with an IoU threshold of 0.25 is used as the evaluation metric and the IoU is computed between predicted segmentations and ground truth segmentations.

Mask R-CNN is initially designed for RGB images. To adapt it for RGBD image processing, we convert the depth channel into an HHA image following [2] and concatenate it with the original RGB image to form a 6-channel input to the network. We initialize the whole network, except for the first convolution layer, with pre-trained coco weights. We carefully train Mask R-CNN following the guideline provided by [1]. To be specific, we first freeze the feature backbone, train the conv1, and heads for 20 epochs with a learning rate 0.001. And then Resnet 4+ layers are finetuned with another 15 epochs using lower learning rate (0.0001). Finally Resnet 3+ layers are open to train for 15 epochs with a learning rate 0.00001. Due to the small size of training data set (only 795 images), data augmentations (Fliplr & Random Rotation & Gamma Adjustment), high weight decay (0.01) and simple architecture (Resnet-50) are applied to avoid severely overfitting. This approach is called MR-CNN*. We also train Mask R-CNN on the RGB images only to analyze the effectiveness of 3D learning. The qualitative comparison is shown in Table 3.

For shape categories with strong appearance signatures but weak geometric features, such as monitor, Mask R-CNN achieves the best performance. This indicates our way of using color information is not as effective as Mask R-CNN. Introducing depth information to Mask R-CNN does not improve its performance dramatically, even on categories with a strong geometric feature which could be easily segmented out by R-PointNet such like bathtub. This justifies the necessity of 3D learning while dealing with RGBD images.

6. Architecture Details

In GSPN, we have a center prediction network, a prior network, a recognition network and a generation network. We use PointNet/PointNet++ as their backbones. Following the same notations in PointNet++, SA$(K, r, [l_1, ..., l_d])$ is a set abstraction (SA) layer with K local regions of ball radius r. The SA layer uses PointNet of d 1 x 1 $conv$ layers with output channels $l_1, ..., l_d$ respectively. FP$(l_1, ..., l_d)$ is a feature propagation (FP) layer with d 1 x 1 $conv$ layers, whose output channels are $l_1, ..., l_d$. Deconv$(C, [h, w], [s_1, s_2])$ means a deconv layer with C output channels, a kernel size of $[h, w]$ and a stride of $[s_1, s_2]$. MLP$([l_1, ..., l_d])$ indicates several multi-layer perceptrons (MLP) with output channels $l_1, ..., l_d$. FC$([l_1, ..., l_d])$ is the same as MLP. We report the details of the network design in Table 4. The center prediction network is essentially a PointNet++. The prior network takes three contexts as input and uses three PointNets to encode each context. The parameters of the MLP used within each PointNet are shown in the first list. Then their features are concatenated and several MLPs are applied to transform the aggregated features to get the mean & variance of the latent variable z. In the recognition network, the first list of MLPs is used to extract shape features and the second list of MLPs is used to output the mean & variance of the latent variable z. In the generation network, we use both $deconv$ and fc layers to generate shape proposals. This two branches (deconv and fc) generate parts of the whole point set independently which are combined together in the end.

The R-PointNet consists of three heads: a classification head, a segmentation head and a bounding box refinement head. For the classification head and the bounding box refinement head, we first use an MLP with feature dimensions (128, 256, 512) to transform the input features. Then after max-pooling, we apply several fully-connected layers with output dimensions (256, 256, num_category) and (256, 256, num_category*6) to get the classification scores and the bounding box updates, respectively. For the segmentation head, we choose to use a small PointNet segmentation architecture with MLP([64, 64]) for local feature extraction,

	Mean	bath-tub	bed	book-shelf	box	chair	coun-ter	desk	door	dres-ser	gar-bin	lamp	moni-tor	night-stand	pil-low	sink	sofa	table	TV	toilet
MRCNN	29.3	26.3	54.1	23.4	3.1	39.3	34.0	6.2	17.8	23.7	**23.1**	31.1	**35.1**	25.4	26.6	36.4	47.1	21.0	**23.3**	58.8
MRCNN*	31.5	24.7	**66.3**	20.1	1.4	44.9	43.9	6.8	16.6	29.5	22.1	29.2	29.3	36.9	34.6	37.1	48.4	26.6	21.9	58.5
Ours	**39.3**	**62.8**	51.4	**35.1**	**11.4**	**54.6**	**45.8**	**38.0**	**22.9**	**43.3**	8.4	**36.8**	18.3	**58.1**	**42.0**	**45.4**	**54.8**	**29.1**	20.8	**67.5**

Table 3. Additional instance segmentation comparison on NYUv2 dataset.

MLP([64, 128, 512]) & Max-pooling for global feature extraction and 1x1 conv(256, 256, num_categroy) for segmentation label prediction. we predict one segment for each category and the weights are updated only based on the prediction for the ground truth category during training. During the inference time, the predicted RoIs are refined based on the bounding box refinement head, which then goes through Point RoIAlign to generate RoI features for the segmentation head.

In addition, we provide various configuration parameters in Table 5. "num_sample" represents the number of seed points we used for shape proposal. "spn_pre_nms_limit" represents the object proposals we keep after GSPN by filtering out proposals with low objectness scores. "spn_nms_max_size" is the maximum number of object proposals we keep after a non-maximum suppression operation following GSPN. "spn_iou_threshold" is the 3D IoU threshold we used for the non-maximum suppression operation. "num_point_ins_mask" is the number of points in each of our generated shape proposals. "train_rois_per_image" is the maximum number of RoIs we use for training within each image in each mini-batch. "detection_min_confidence" is the confidence threshold we use during the inference time, where detections with confidence scores lower than this threshold are filtered out. "detection_max_instances" is the maximum number of instances we detection from a single scene or object.

7. Additional Visualizations

In the main paper, we have evaluated our R-PointNet with GSPN on three datasets: ScanNet, PartNet and NYUv2. In this section, we provide more visualizations for the instance segmentation results generated by R-PointNet on these datasets. Specifically, we show instance segmentation results on ScanNet, PartNet and NYUv2 in Figure 1, Figure 2 and Figure 3 respectively.

Sub-Networks	Architecture
Center Prediction Net	SA(2048, 0.2, [32, 32, 64]) SA(512, 0.4, [64, 64, 128]) SA(128, 0.8, [128, 128, 256]) SA(32, 1.6, [256, 256, 512]) FP(256, 256) FP(256, 256) FP(256, 128) FP(128, 128, 128)
Prior Net	MLP([64, 128, 256]) (For context) MLP([256, 512, 512]) (After concat)
Recognition Net	MLP([64, 256, 512, 256]) MLP([256, 512, 512])
Generation Net	Deconv(512, [3,3], [1,1]) Deconv(256, [3,3], [2,2]) Deconv(128, [4,4], [2,2]) Deconv(3, [1,1], [1,1]) FC([512, 512, 256*3])

Table 4. Architecture details of GSPN.

Configurations	Train	Inference
num_sample	512	2048
spn_pre_nms_limit	192	1536
spn_nms_max_size	128	384
spn_iou_threshold	0.5	0.5
num_point_ins_mask	256	1024
train_rois_per_image	64	-
detection_min_confidence	0.5	0.5
detection_max_instances	100	100

Table 5. Main configuration parameters used during train and inference.

References

[1] Mask r-cnn wiki. https://github.com/matterport/Mask_RCNN/wiki. 2

[2] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014. 2

[3] 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. 2

[4] 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

[5] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016. 2

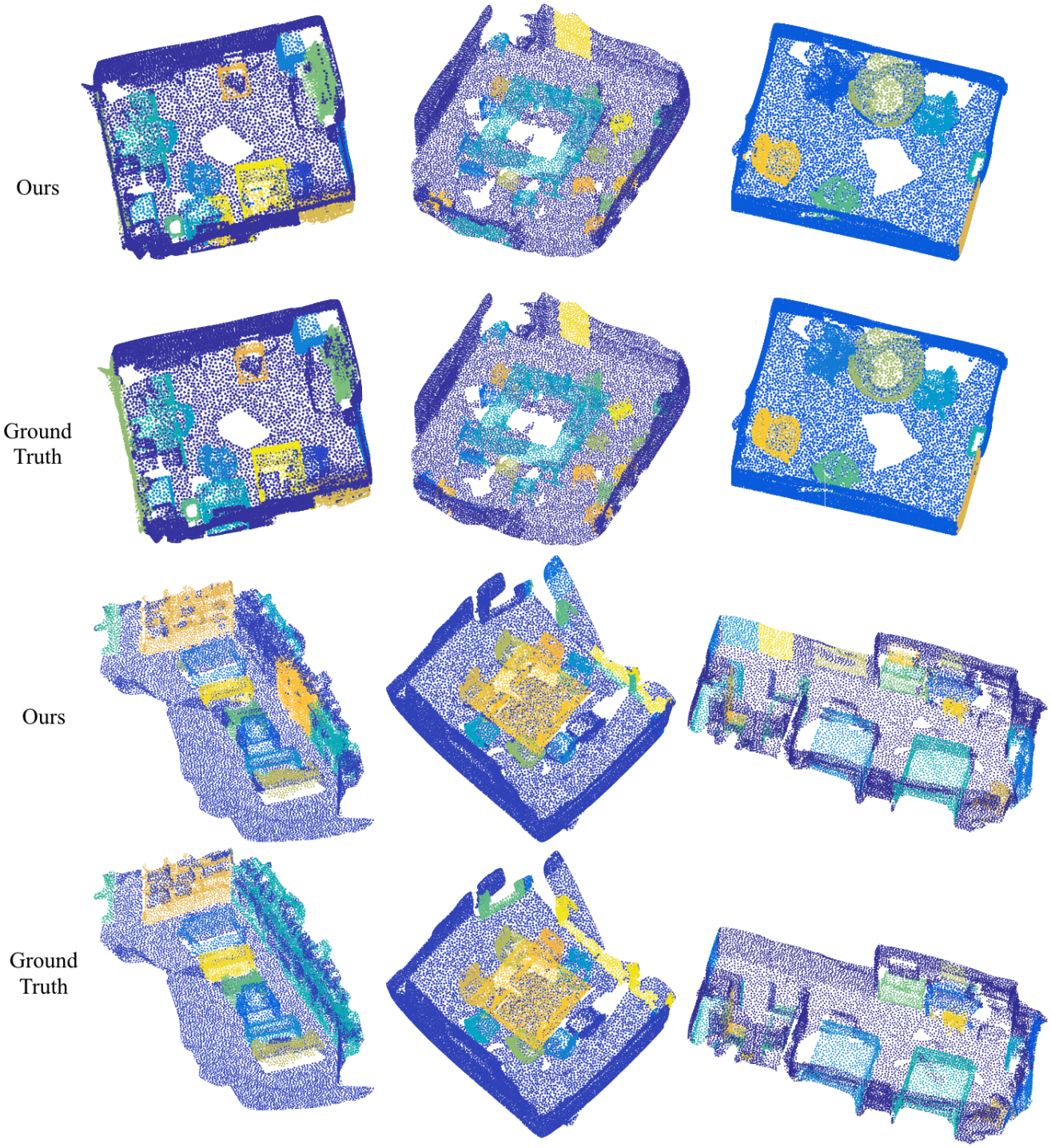

Figure 1. Visualization for ScanNet instance segmentation results. Different colors indicate different instances.

Figure 2. Visualization for PartNet instance segmentation results. Different colors indicate different instances.

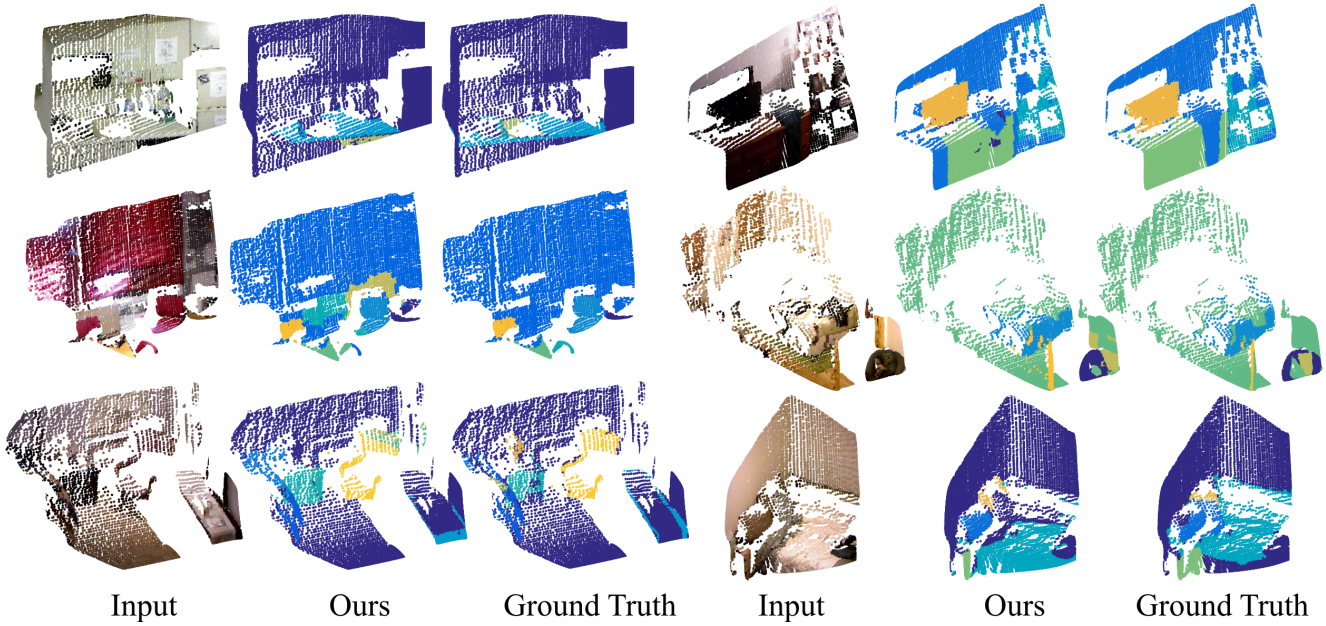

Figure 3. Visualization for NYUv2 instance segmentation results. Different colors indicate different instances.