Local spatial feature learning techniques in 3D object detection

Qi Ma
MAVT
ETH Zurich
qimaqi@student.ethz.ch

Xiaoao Song
Department of Informatics
UZH Zurich
xisong@student.ethz.ch

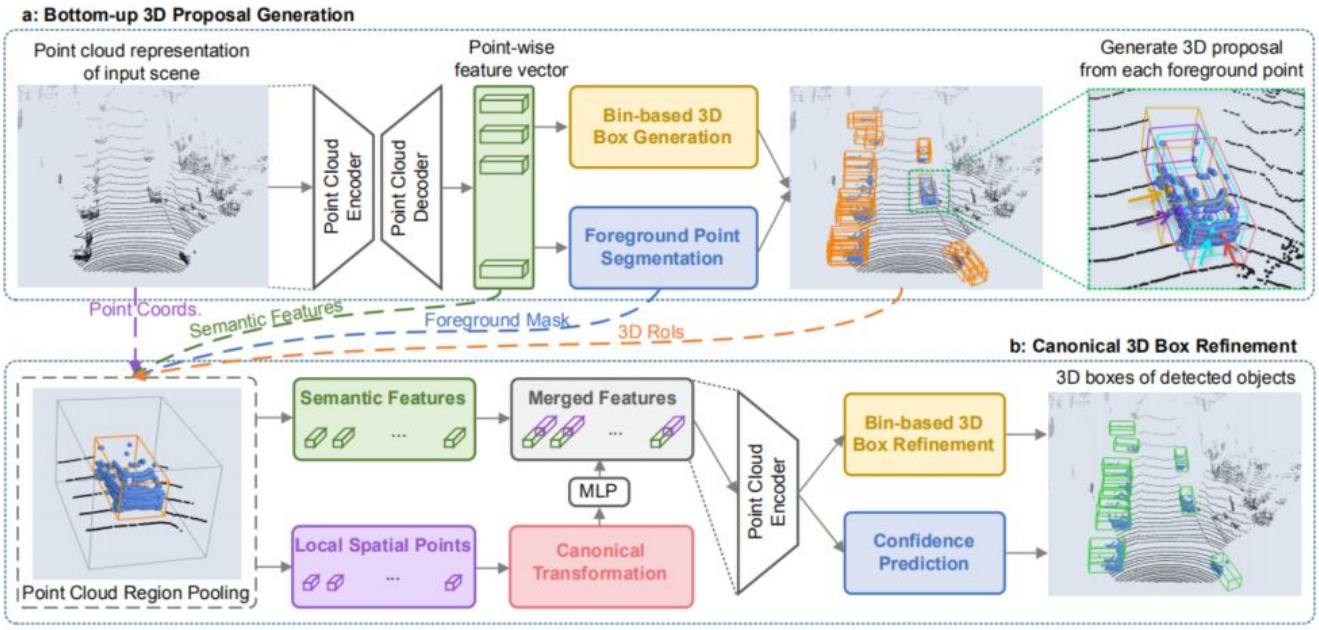

Figure 1. The representative of two-stage region proposal-based methods :PointRCNN architecture.[5]

Abstract

With the popularity of 3D acquisition technologies and deep learning techniques,

AR, VR and other mobile devices with cameras, there's a growing concern for privacy. For indicated devices which extract local feature from images, though the original scene is often discarded for privacy concerns, it has been shown these features can be potentially used to reconstruct original images. Nowadays, the network based features are more and more popular for their superior performance in many tasks. In this paper, we show they may pose a greater treat to piracy with their more accurate interest point detector and more informative feature descriptor.[1] *Besides, we also carried out exhaustive experiments towards the performance of three reconstruction network backbones as InvNet, UNet and UNet++ with different input feature sparsity levels on SuperPoint features and R2D2 features respectively, with several metrics for evaluation. After reconstructing gray scale images, we performed detailed investigation into different colorization methods to provide realistic color information of reconstructed images.*

1. Introduction

For 3D Object detection the main task is to draw an oriented 3D bounding box around corresponding object by taking and processing point cloud. The methods of 3D object detection can be divided into region proposal based method and single shot method[2]. From the KITTI benchmark [1] the two categories outperforms single shot methods by a

[1]Dataset and Codes for evaluation will be publicly available at https://github.com/AlexanderNevada/3dv_pytorch

CT	RPN features	camera depth	seg. mask	AP_E	AP_M	AP_H
×	✓	✓	✓	7.64	13.68	13.94
✓	×	✓	✓	84.75	74.96	74.29
✓	✓	×	✓	87.34	76.79	75.46
✓	✓	✓	×	86.25	76.64	75.86
✓	✓	✓	✓	**88.45**	**77.67**	**76.30**

Figure 2. Without the proposed canonical transformation the performance of refinement sub-network is poor[5].

large margin. Therefore in this work we mainly analyze how the methods of region proposal-based methods learn and use local spatial feature.

Frustum-based Methods. 2D object detection techniques are well leveraged in these methods. First a 2D candidate regions of objects are proposed by 2D detector from image view. Then a 3D frustum proposal is extracted. The performance of these methods are limited by 2D image detectors. Work[6] from Wang generated a sequence of frustums along the frustum axis and use the obtained frustums to group local points. This work outperforms all existing works. Refinement with normalized points by translation and rotation resolves well the inaccuracy of 2D object proposal. Moreover, the regression loss is calculated based on offset formulas and bin loss.

Multi-view based Methods. A significant improvement on 2D, 3D and BEV detection was achieved by the Work [3]. Multiple sensors and tasks are exploit to generate accurate 3D object detection. The precise rotated ROI feature is proposed which consider rotational periodicity and it also re-parametrized with respect to object orientation axes.

Segmentation-based Methods. This methods take advantage of existing semantic segmentation techniques and foreground points are extracted from most background points thus speeds up training. Compared to other methods the segmentation-based methods earns higher recall rates. Adopting the Region Proposal Network(RPN), the [5] generates highly-accurated 3D boxes by fusing semantic feature and local spatial features. Similar to the re-parametrization and normalization above, canonical transformation is proposed for better local spatial feature learning. Moreover, a ablation study proves that canonical transformation improves the performance of detection significantly.

Summary. After analyzing different methods in 3D object detection the following observation can be made: In order to learn local spatial feature a normalized technique is leveraged in all kinds of 3D object detection techniques. However, to the best of our knowledge, only a few analysis is made about how big is the improvement of this normalization and how it should correspond to other input attributes and loss function.

2. Methods

Since our baseline network is highly similar to PointRCNN architecture2 and inspired by the extensively-used normalization methods we first change the input xyz with respect to object coordinate and analyze how the performance is changed compared to the baseline.3

```
### canonical transformation
batch_size = xyz.shape[0]
roi_center = assinged_target[:,0:3]
xyz[:, :, 0:3] -= roi_center.unsqueeze(dim=1)

for k in range(batch_size):
    xyz[k,:,0:3] = rotate_pc_along_y_torch_dlad(xyz[k,:,0:3],assinged_target[k,6])
```

Figure 3. All points are translated and rotated to object canonical coordinates [5].

Then we upsample the canioncal position information and build the local spatial feature. By merging the local spatial feature with the semantic feature from PointNet++[4] in stage 1. This method shows the necessities for local feature convolution operations.8

```
### concat input from canonical transformation
pts_feature = xyz.transpose(1, 2).unsqueeze(dim=3).contiguous().float()    # depth or other input attributes

### Upsample
xyz_feature = self.xyz_up_layer(pts_feature)

### Merge local feature with semantic feature and downsample
merged_feature = torch.cat((xyz_feature, feat), dim=1)
merged_feature = self.merge_down_layer(merged_feature)
```

Figure 4. Convolute local point position to local spatial features and merge it with semantic feature [5].

Finally, since the loss function calculation consists of three term: location loss, size loss and orientation loss. The input of refinment Encoder should contain information about the position of each point in world-coordinate. So in contrast to PointRCNN[5] we replace xyz local position with xyz global position. Because in PointRCNN loss is calculated with bin-based box, input with normalized xyz will accelerated the training process.

In summary, the main changes we made compared to baseline network is adding local feature generated from canonical coordinates. The local spatial feature helps network learn the feature and information which is robust to rotation and translation. Different from PointRCNN, the input to PointCloud Encode use xyz in world coordinate instead of canonical coordinates which matches the loss function in our case.

3. Result

In this section we compare the training loss between the baseline and our proposal, since the aws resources is limited and very unstable, a detailed numerical analysis is unrealistic. Most results crashed more then 3 times so the curve is discrete and not continuous.

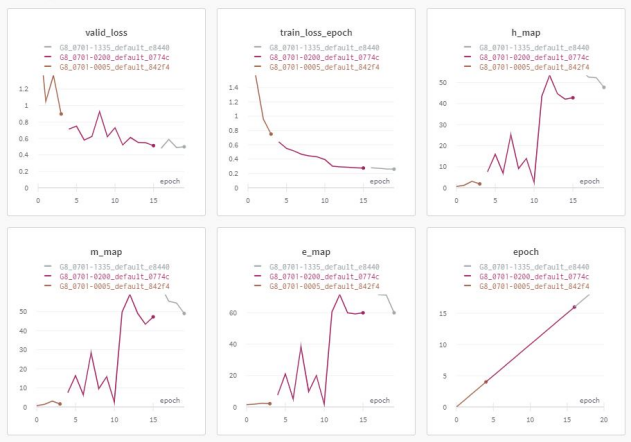

Figure 5. Baseline result.

Figure 6. Canioncal xyz input result.[5]

Figure 7. Fusion of local feature and semantic feature result.[5]

Figure 8. Fusion of local feature and semantic feature result with xyz in world coordinate.[5]

Even without numerical analysis, some observation can be made through this simple ablation study. 1. Compared to baseline, adding Canioncal transform will slightly decrease the training speed. The loss drops slowly then without Canioncal transform.
2. By fusing local feature with semantic feature, the training speed is increased and the loss is smaller, because by convolution the local spatial information is extracted. 3. Matching the loss function by using xyz in world coordinate will have the fastest convergence speed.

4. Summary

In this project, we analyze the how Canioncal transformation will improve the 3D object detection result and which input attributes should be choosed to match the loss function. From simple ablation study it shows that by merging local feature the loss will drop and the training will speed up a little. If loss function consists of location loss the input to refinement network is better without Canioncal transformation.

Method			Modality	Speed (fps)	Cars			Pedestrians			Cyclists		
					E	M	H	E	M	H	E	M	H
Region Proposal -based Methods	Multi-view Methods	MV3D [4]	L & I	2.8	86.62	78.93	69.80	-	-	-	-	-	-
		AVOD [126]	L & I	12.5	89.75	84.95	78.32	42.58	33.57	30.14	64.11	48.15	42.37
		ContFuse [127]	L & I	16.7	94.07	85.35	75.88	-	-	-	-	-	-
		MMF [128]	L & I	12.5	93.67	88.21	81.99	-	-	-	-	-	-
		SCANet [39]	L & I	11.1	90.33	82.85	76.06	-	-	-	-	-	-
		RT3D [131]	L & I	11.1	56.44	44.00	42.34	-	-	-	-	-	-
	Segmentation -based Methods	IPOD [132]	L & I	5.0	89.64	84.62	79.96	60.88	49.79	45.43	78.19	59.40	51.38
		PointRCNN [133]	L	10.0	92.13	87.39	82.72	54.77	46.13	42.84	82.56	67.24	60.28
		PointRGCN [134]	L	3.8	91.63	87.49	80.73	-	-	-	-	-	-
		PointPainting [135]	L & I	2.5	92.45	88.11	83.36	58.70	49.93	46.29	83.91	71.54	62.97
		STD [138]	L	12.5	94.74	89.19	86.42	60.02	48.72 44.55	81.36	67.23	59.35	
	Frustum -based Methods	F-PointNets [139]	L & I	5.9	91.17	84.67	74.77	57.13	49.57	45.48	77.26	61.37	53.78
		SIFRNet [140]	L & I	-	-	-	-	-	-	-	-	-	-
		PointFusion [142]	L & I	-	-	-	-	-	-	-	-	-	-
		RoarNet [143]	L & I	10.0	88.20	79.41	70.02	-	-	-	-	-	-
		F-ConvNet [144]	L & I	2.1	91.51	85.84	76.11	57.04	48.96	44.33	84.16	68.88	60.05
		Patch Refinement [145]	L	6.7	92.72	88.39	83.19	-	-	-	-	-	-
	Other Methods	3D IoU loss [146]	L	12.5	91.36	86.22	81.20	-	-	-	-	-	-
		Fast Point R-CNN [147]	L	16.7	90.76	85.61	79.99	-	-	-	-	-	-
		PV-RCNN [148]	L	12.5	94.98	90.65	86.14	-	-	-	82.49	68.89	62.41
		VoteNet [124]	L	-	-	-	-	-	-	-	-	-	-
		Feng et al. [149]	L	-	-	-	-	-	-	-	-	-	-
		ImVoteNet [150]	L & I	-	-	-	-	-	-	-	-	-	-
		Part-A^2 [151]	L	12.5	91.70	87.79	84.61	-	-	-	81.91	68.12	61.92
Single Shot Methods	BEV-based Methods	PIXOR [129]	L	28.6	83.97	80.01	74.31	-	-	-	-	-	-
		HDNET [152]	L	20.0	89.14	86.57	78.32	-	-	-	-	-	-
		BirdNet [153]	L	9.1	76.88	51.51	50.27	20.73	15.80	14.59	36.01	23.78	21.09
	Discretization -based Methods	VeloFCN [154]	L	1.0	0.02	0.14	0.21	-	-	-	-	-	-
		3D FCN [155]	L	<0.2	70.62	61.67	55.61	-	-	-	-	-	-
		Vote3Deep [156]	L	-	-	-	-	-	-	-	-	-	-
		3DBN [157]	L	7.7	89.66	83.94	76.50	-	-	-	-	-	-
		VoxelNet [136]	L	2.0	89.35	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
		SECOND [158]	L	26.3	89.39	83.77	78.59	55.99	45.02	40.93	76.50	56.05	49.45
		MVX-Net [159]	L & I	16.7	92.13	86.05	78.68	-	-	-	-	-	-
		PointPillars [137]	L	62.0	90.07	86.56	82.81	57.60	48.64	45.78	79.90	62.73	55.58
		SA-SSD [160]	L	25.0	95.03	91.03	85.96	-	-	-	-	-	-
	Point-based Methods	3DSSD [161]	L	25.0	92.66	89.02	85.86	60.54	49.94	45.73	85.04	67.62	61.14
	Other Methods	LaserNet [162]	L	83.3	79.19	74.52	68.45	-	-	-	-	-	-
		LaserNet++ [163]	L & I	26.3	-	-	-	-	-	-	-	-	-
		OHS-Dense [164]	L	33.3	93.73	88.11	84.98	50.87	44.59	42.14	82.13	66.86	60.86
		OHS-Direct [164]	L	33.3	93.59	87.95	83.21	55.90	49.48	45.79	79.66	67.20	61.04
		Point-GNN [125]	L	1.7	93.11	89.17	83.90	55.36	47.07	44.61	81.17	67.28	59.67

Figure 9. Comparative 3D object detection results on the KITTI test BEV detection benchmark.[2]

References

[1] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[2] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[3] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

[4] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.

[5] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] Z. Wang and K. Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1742–1749. IEEE, 2019.