

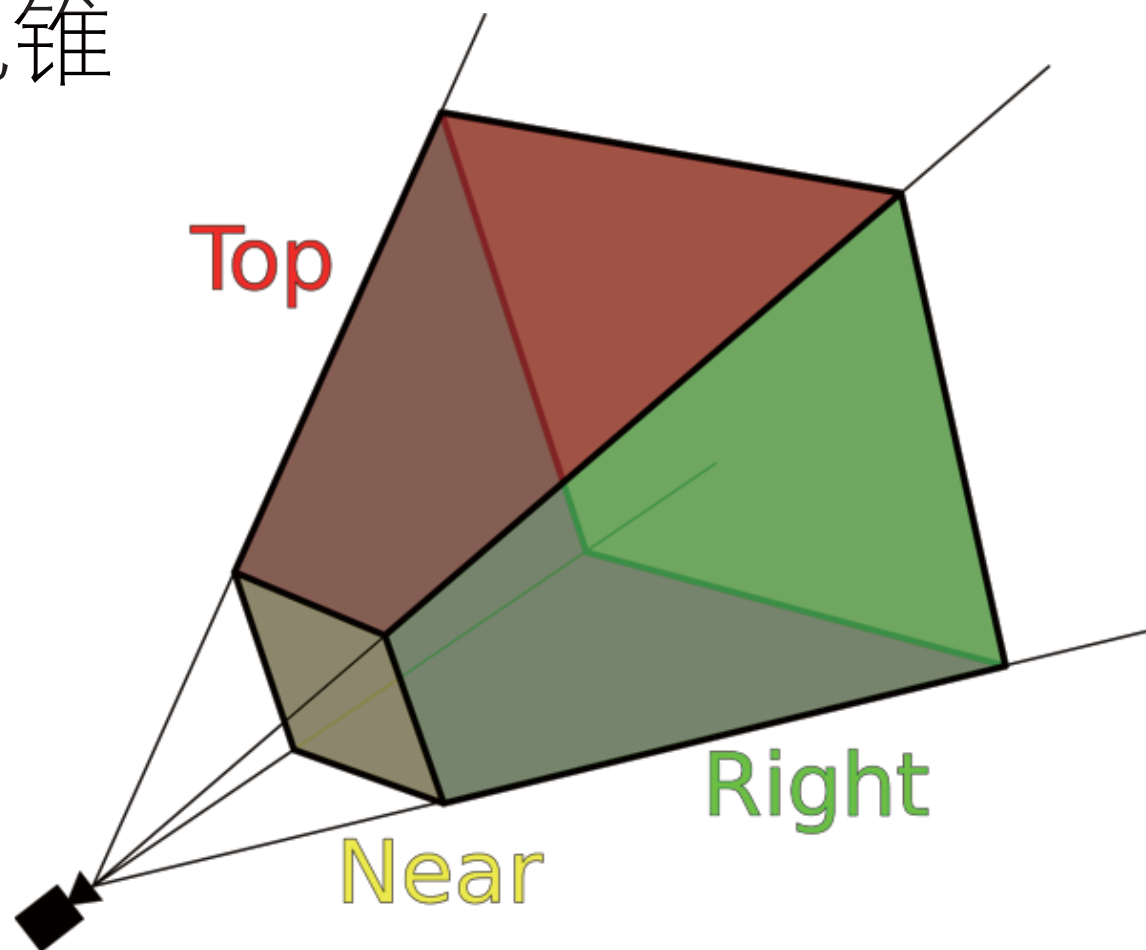
Frustum PointNets for 3D Object Detection from RGB-D Data

Charles Qi

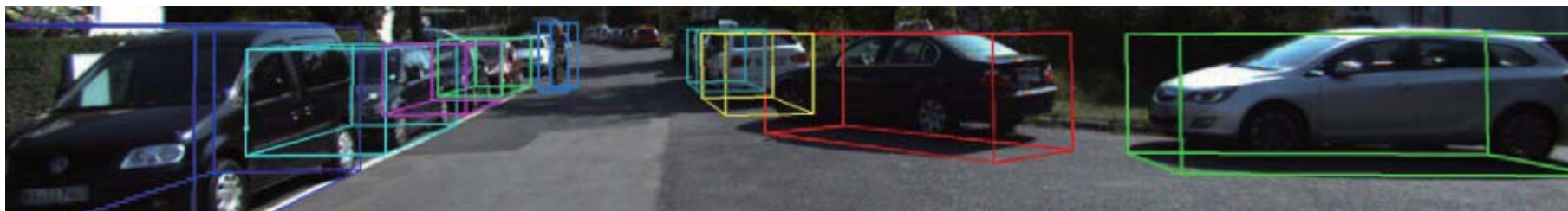
GAMES Webinar Jan 17, 2019

*joint work with Wei Liu, Chenxia Wu, Hao Su and Leonidas Guibas
CVPR 2018*

Frustum 视锥



What is 3D Object Detection?



What is 3D Object Detection?

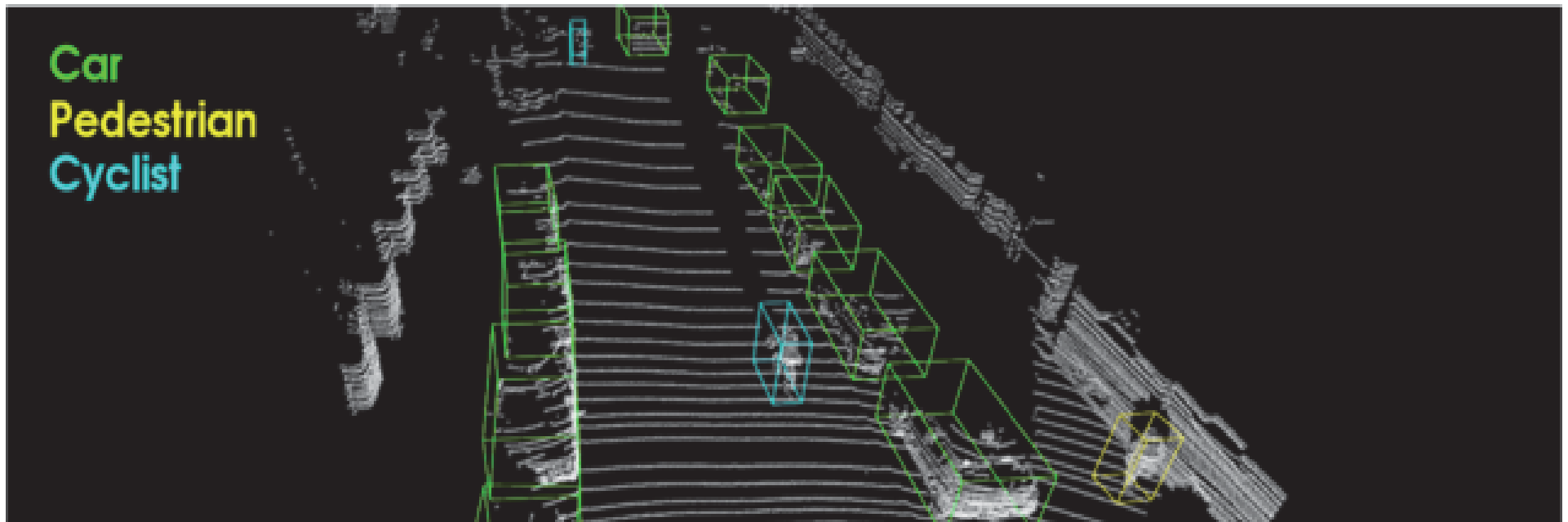


Figure from the recent VoxelNet paper from Apple.

What is 3D Object Detection?

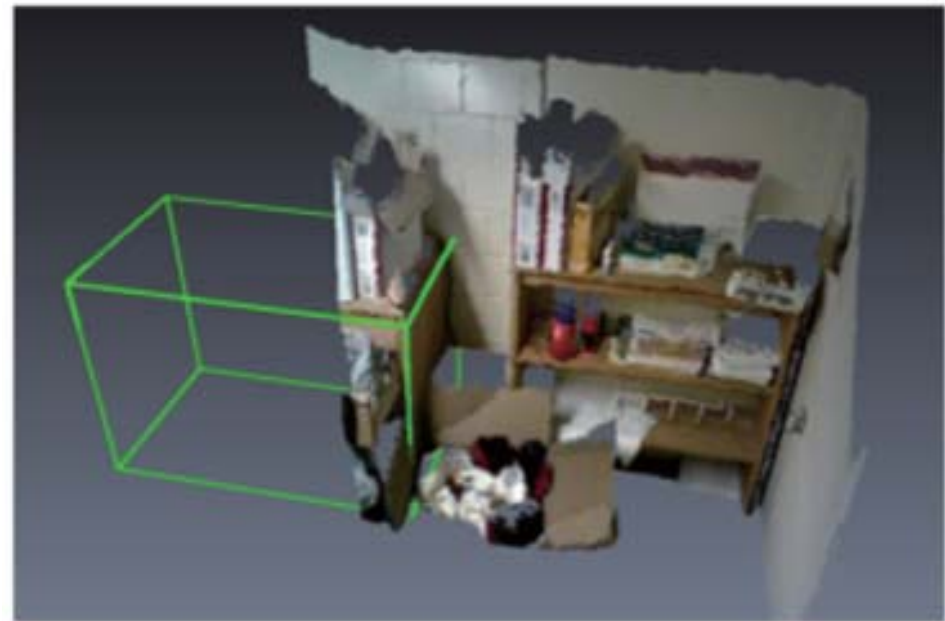
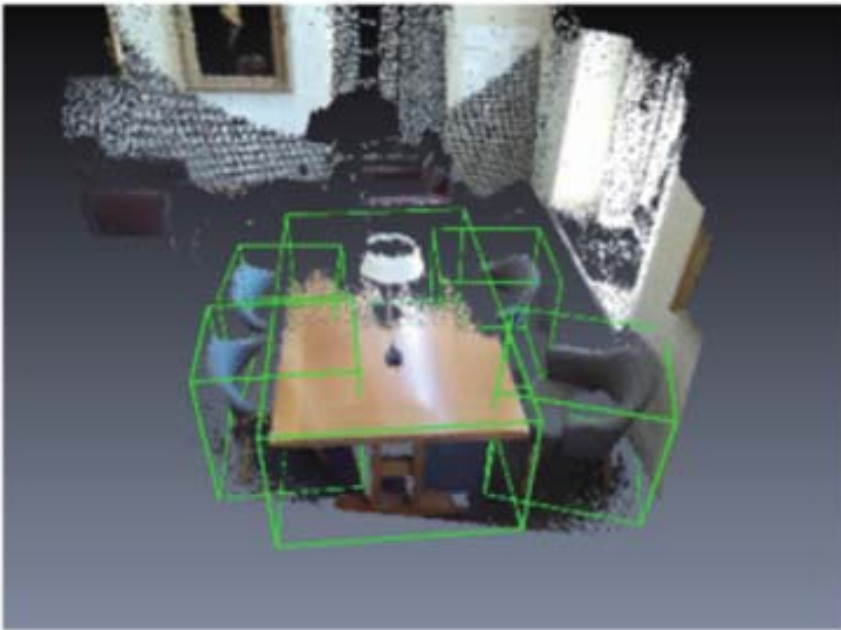


Figure from ICCV17 paper 2d-driven 3d object detection.

What is 3D Object Detection?

Input: RGB-D data

“D” can be sparse point cloud from LiDAR or dense depth map from indoor depth sensors

What is 3D Object Detection?

Input: RGB-D data

“D” can be sparse point cloud from LiDAR or dense depth map from indoor depth sensors

Output: Amodal 3D bounding boxes and semantic class labels for objects in the scene

What is 3D Object Detection?

Input: RGB-D data

“D” can be sparse point cloud from LiDAR or dense depth map from indoor depth sensors

Output: Amodal 3D bounding boxes and semantic class labels for objects in the scene

“amodal” means the 3D box is for the “complete” object even if part of it is invisible.

What is 3D Object Detection?

Input: RGB-D data

“D” can be sparse point cloud from LiDAR or dense depth map from indoor depth sensors

Output: Amodal 3D bounding boxes and semantic class labels for objects in the scene

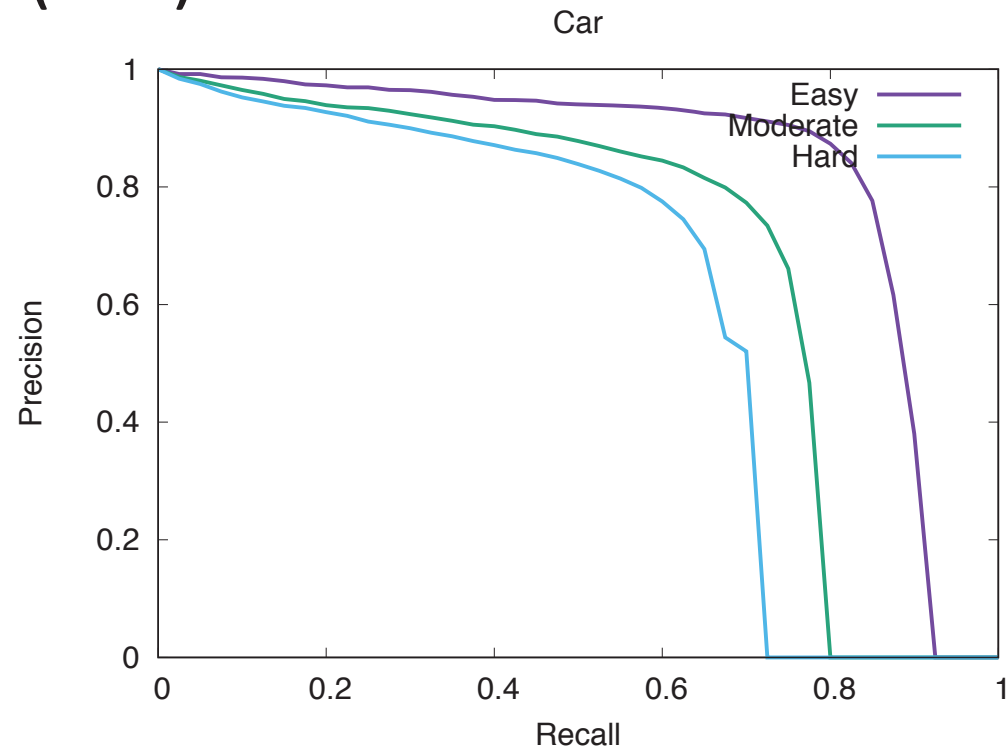
“amodal” means the 3D box is for the “complete” object even if part of it is invisible.

3D box parameterization: $c_x, c_y, c_z \quad h, w, l \quad \theta, \phi, \psi$

What is 3D Object Detection?

Evaluation Metric: Average Precision (AP) with a 3D Intersection over Union (IoU) threshold

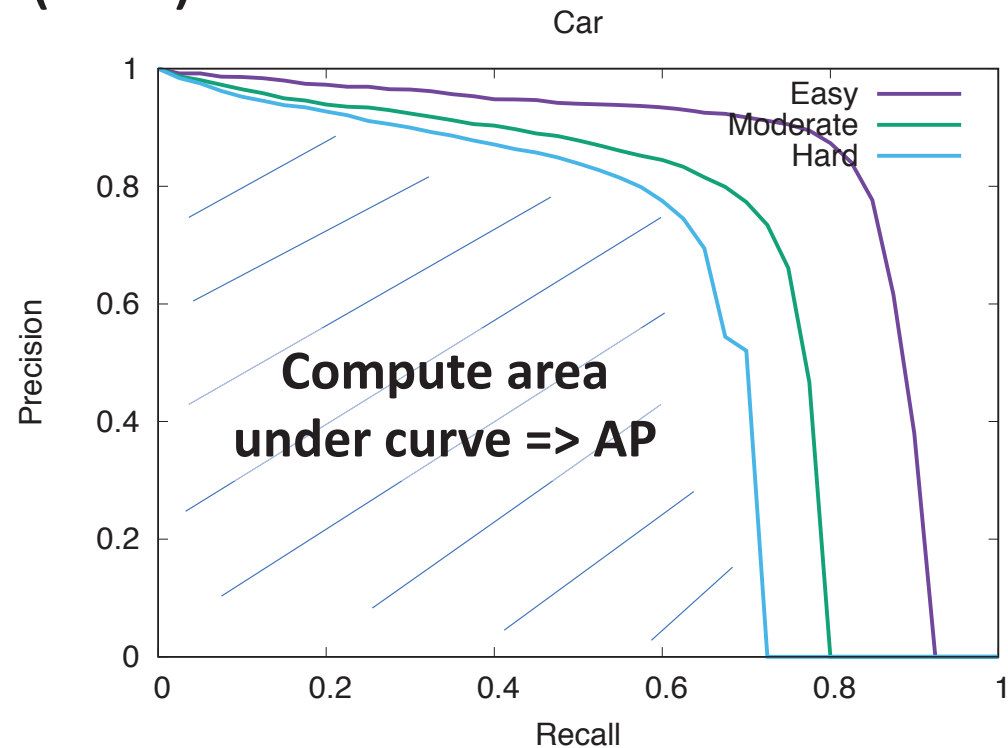
Example Precision-Recall (PR) curves:



What is 3D Object Detection?

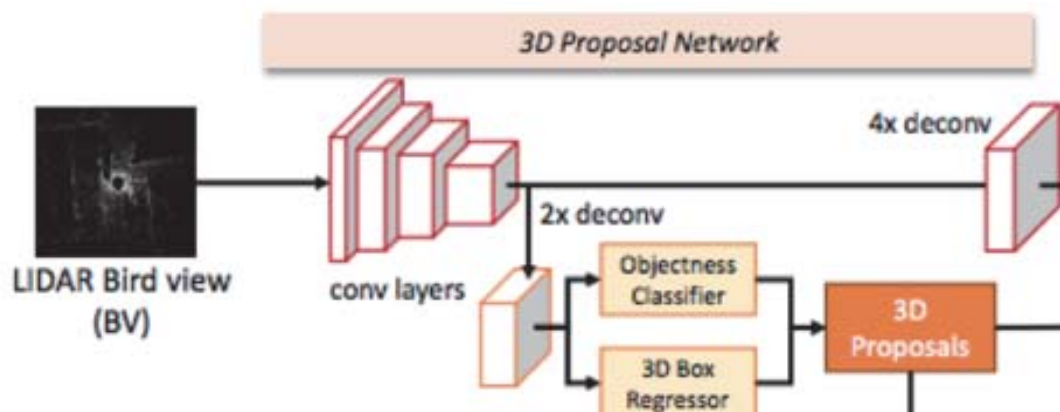
Evaluation Metric: Average Precision (AP) with a 3D Intersection over Union (IoU) threshold

Example Precision-Recall (PR) curves:



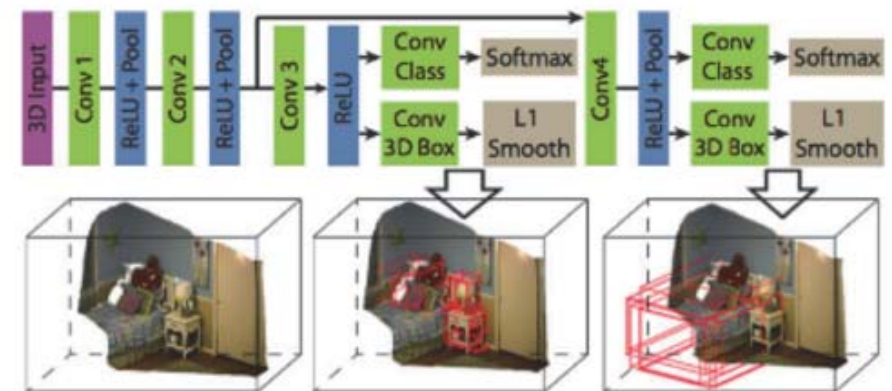
Previous Work (1/2)

- 3D based object proposal + classification
- **Con: Make no use of state-of-the-art RGB detectors**



MV3D by Chen et al.

Bird-eye view proposal (hard to detect smaller objects like pedestrian and cyclists)

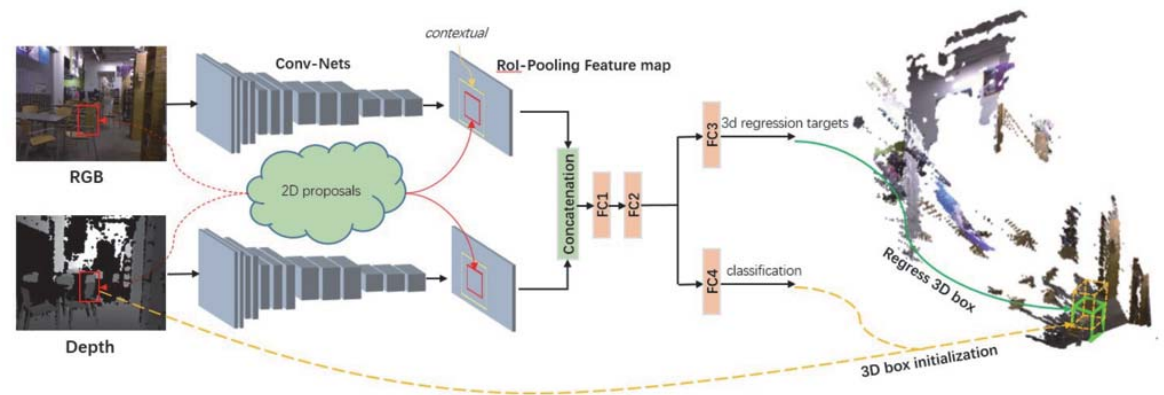
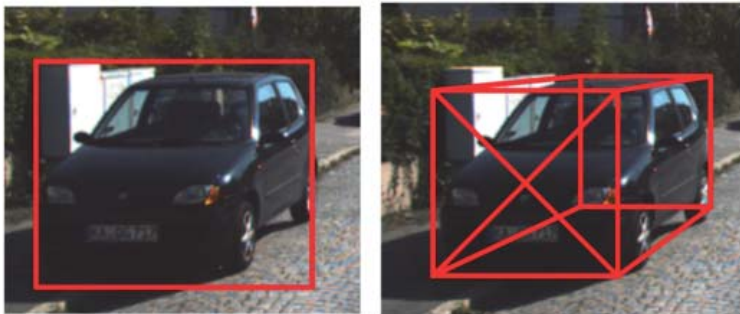


Deep Sliding Shapes by Song et al.

3D CNN region proposal (huge search space, expensive computation)

Previous Work (2/2)

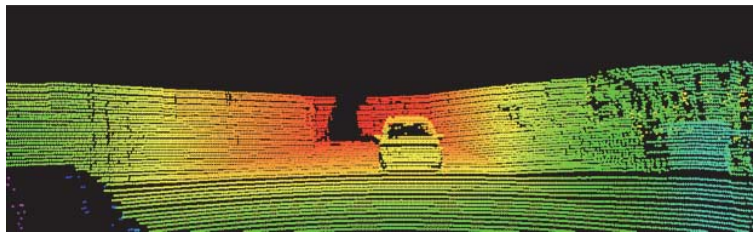
- RGB or RGB-D image based object detection
- **Con: Perspective projection makes it hard to infer precise 3D information such as object depth and size**



CVPR17 paper by Mousavian et al.
Use size prior and projection error as supervision for 3D object detection from monocular RGB image.

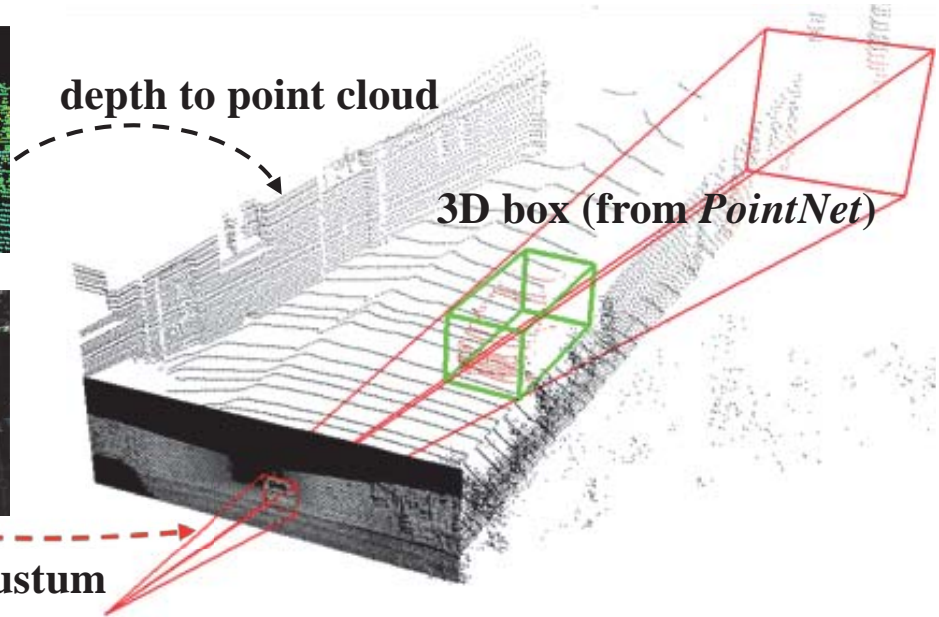
CVPR17 paper by Deng et al.
Use 2D CNNs to generate region proposals from RGB and depth images. Regress to 3D box based on an initialization from 2D depth map region.

Our work: Frustum PointNets for 3D Object Detection



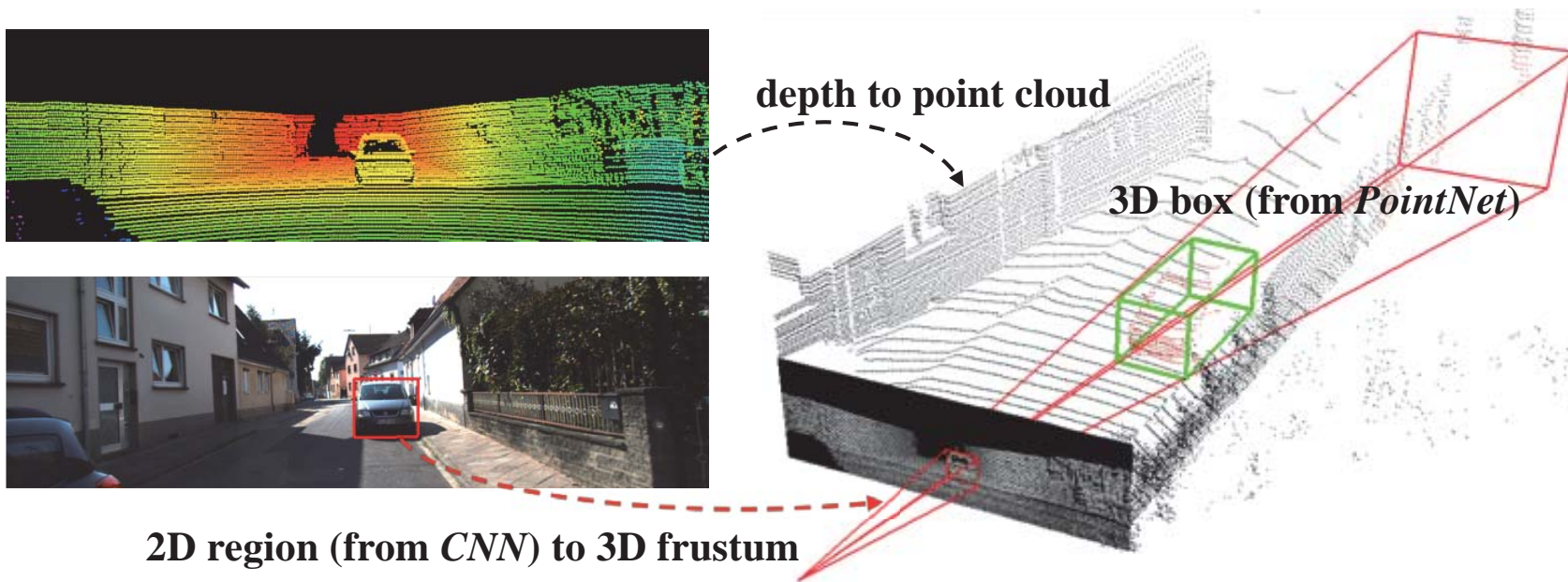
2D region (from CNN) to 3D frustum

depth to point cloud



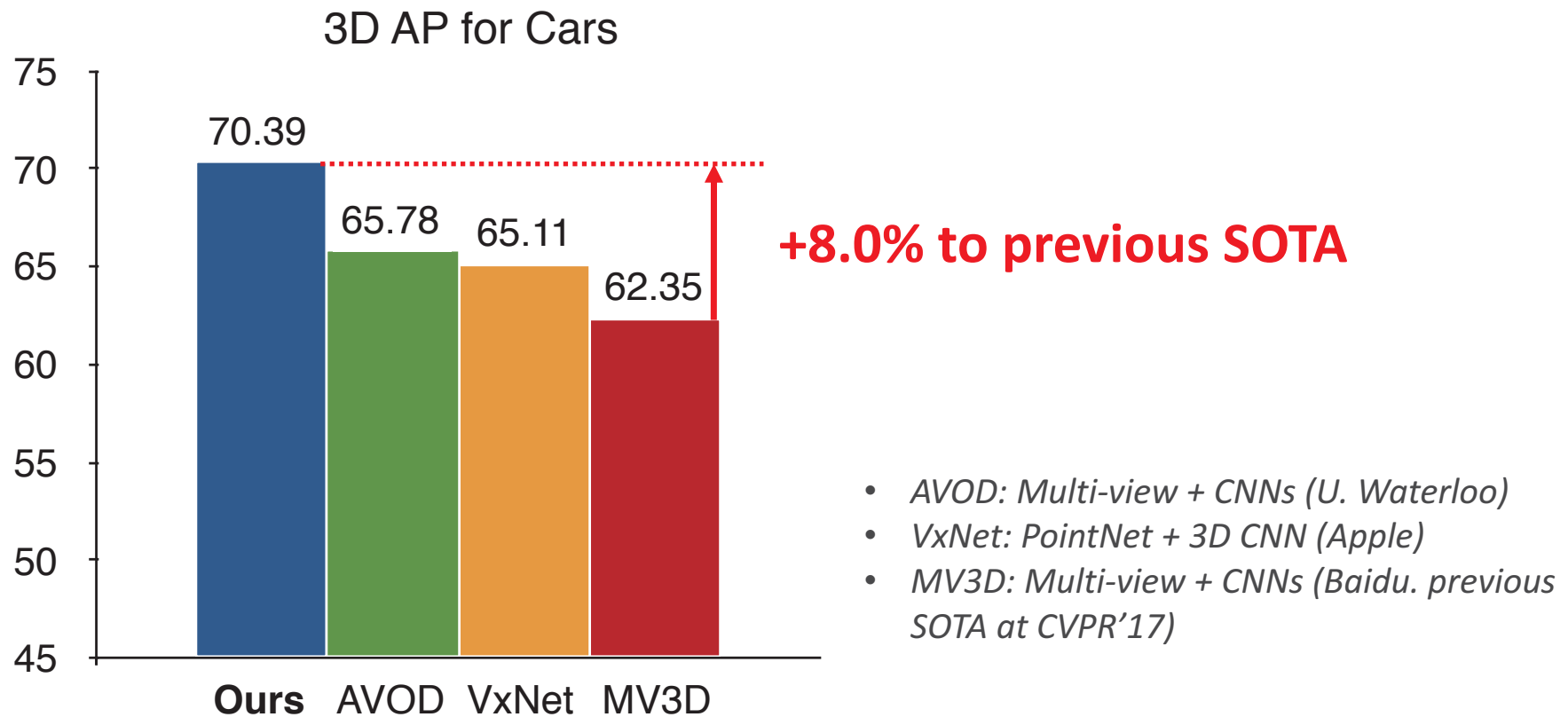
3D box (from PointNet)

Our work: Frustum PointNets for 3D Object Detection

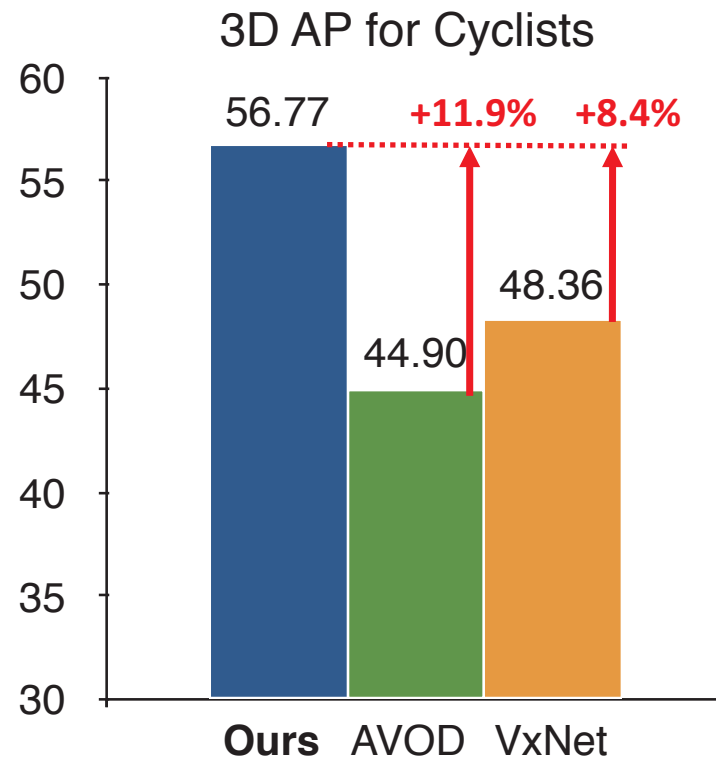
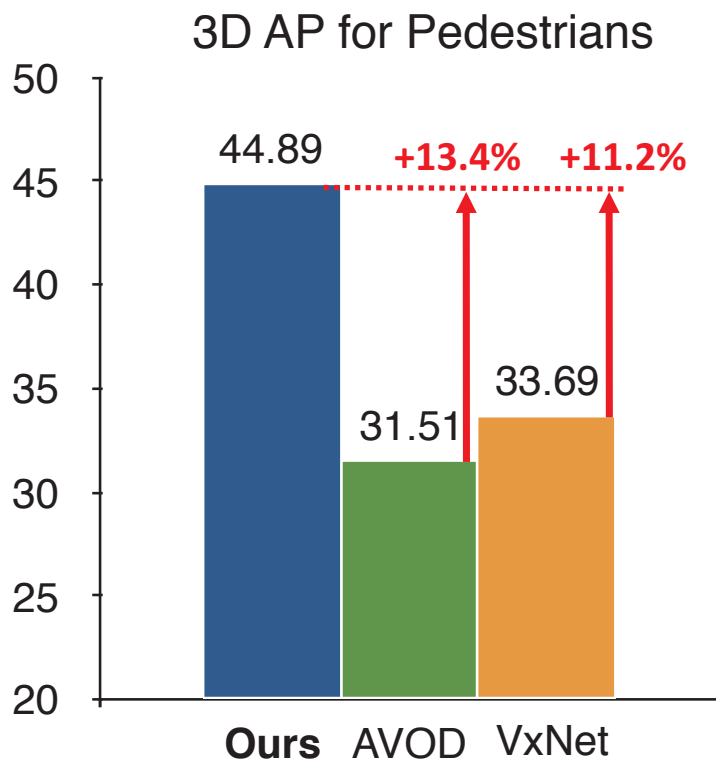


- + **Leveraging mature 2D detectors** for region proposal. 3D search space reduced.
- + Solving 3D detection problem with **3D data and 3D deep learning** architectures.

Our method ranked No. 1 on KITTI 3D Object Detection Benchmark






Our method ranked No. 1 on KITTI 3D Object Detection Benchmark



After ~1.5 years, our method still ranks No. 1 in pedestrian detection

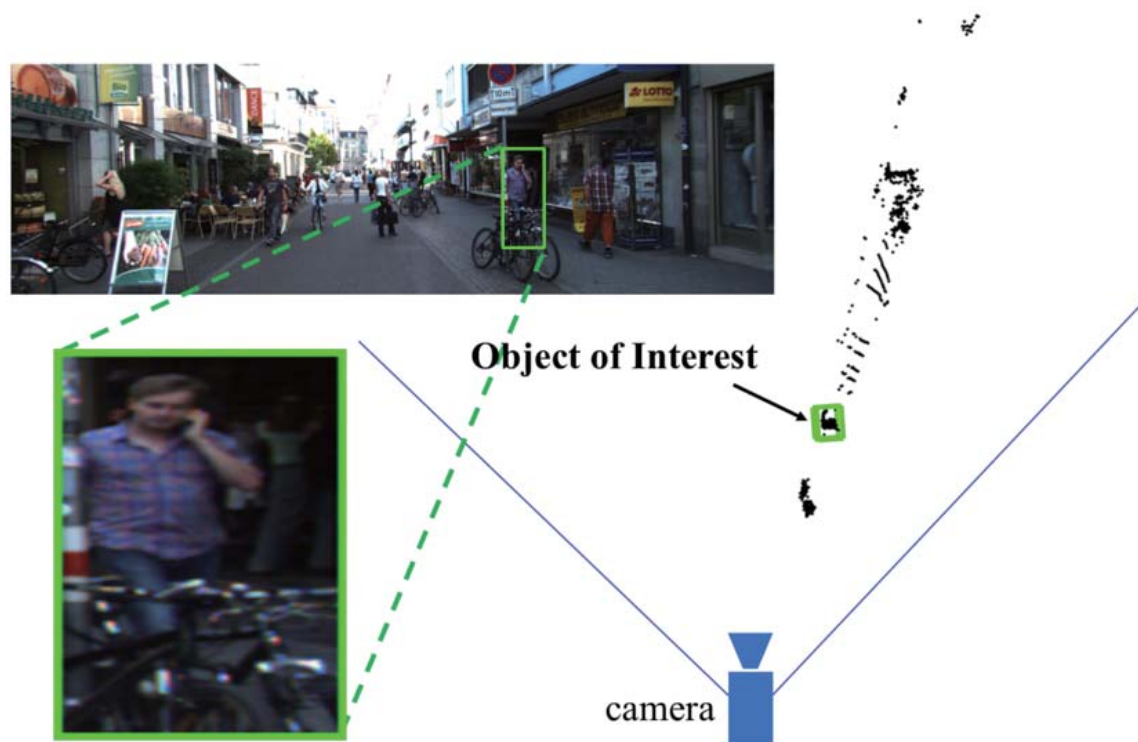
Pedestrian

our method

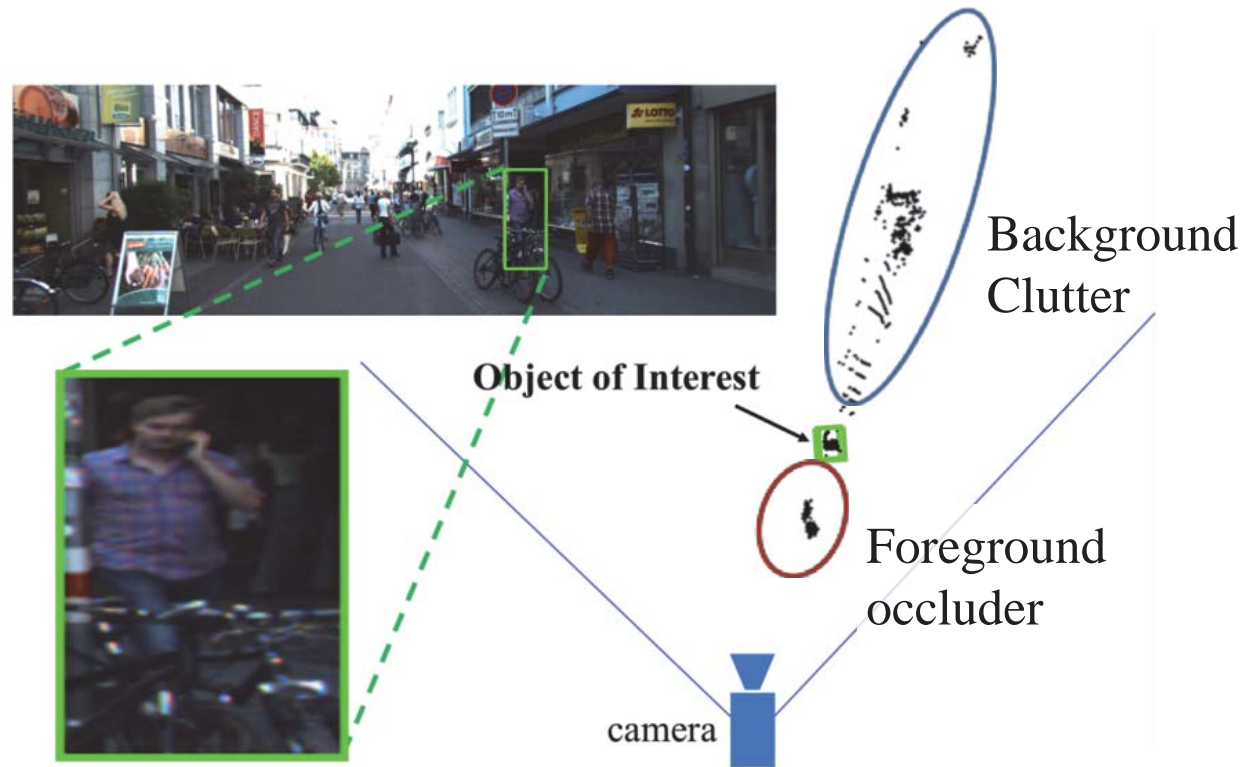
	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	F-PointNet		code	44.89 %	51.21 %	40.23 %	0.17 s	GPU @ 3.0 Ghz (Python)	<input type="checkbox"/>
<small>C. Qi, W. Liu, C. Wu, H. Su and L. Guibas: Frustum PointNets for 3D Object Detection from RGB-D Data. arXiv preprint arXiv:1711.08488 2017.</small>									
2	IPOD			44.68 %	56.92 %	42.39 %	0.2 s	GPU @ 2.5 Ghz (Python + C/C++)	<input type="checkbox"/>
<small>Z. Yang, Y. Sun, S. Liu, X. Shen and J. Jia: IPOD: Intensive Point-based Object Detector for Point Cloud. arXiv preprint arXiv:1812.05276 2018.</small>									
3	PointPillars			43.53 %	52.08 %	41.49 %	16 ms	1080ti GPU and Intel i7 CPU	<input type="checkbox"/>
4	AVOD-FPN		code	42.81 %	50.80 %	40.88 %	0.1 s	Titan X (Pascal)	<input type="checkbox"/>
<small>J. Ku, M. Mozifian, J. Lee, A. Harakeh and S. Waslander: Joint 3D Proposal Generation and Object Detection from View Aggregation. IROS 2018.</small>									
5	SECOND		code	42.56 %	51.07 %	37.29 %	38 ms	1080Ti	<input type="checkbox"/>
<small>Y. Yan, Y. Mao and B. Li: SECOND: Sparsely Embedded Convolutional Detection. Sensors 2018.</small>									

http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d

Frustum-based 3D Object Detection: Challenges



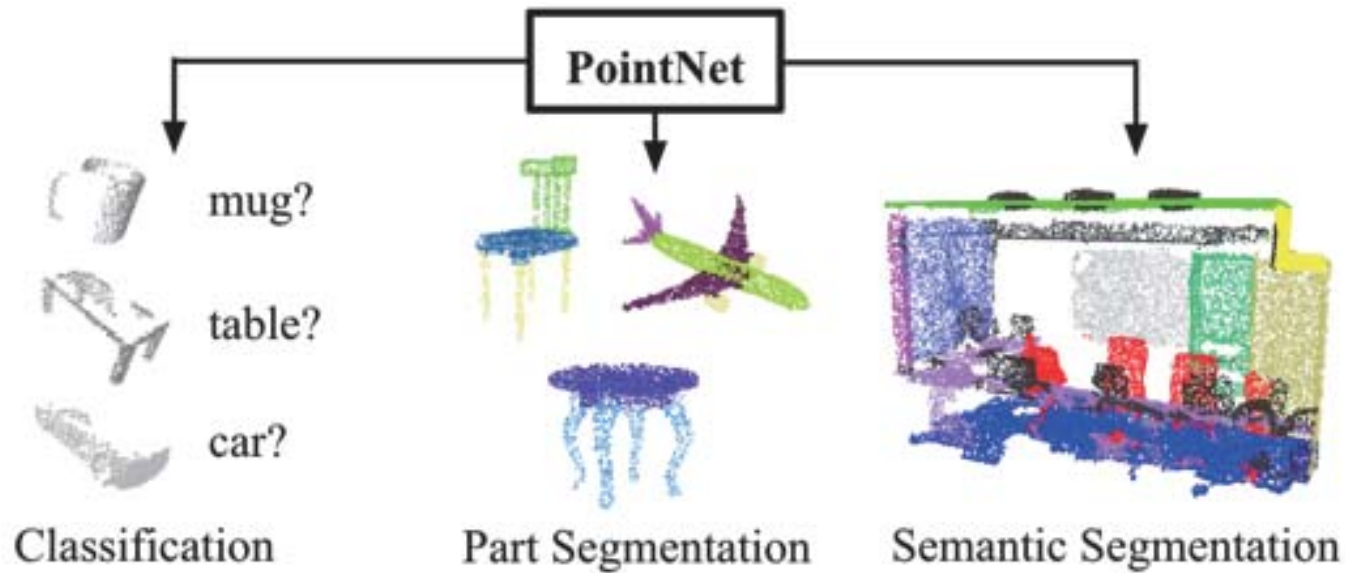
Frustum-based 3D Object Detection: Challenges



- Occlusions and clutters are common in frustum point clouds
- Large range of point depths

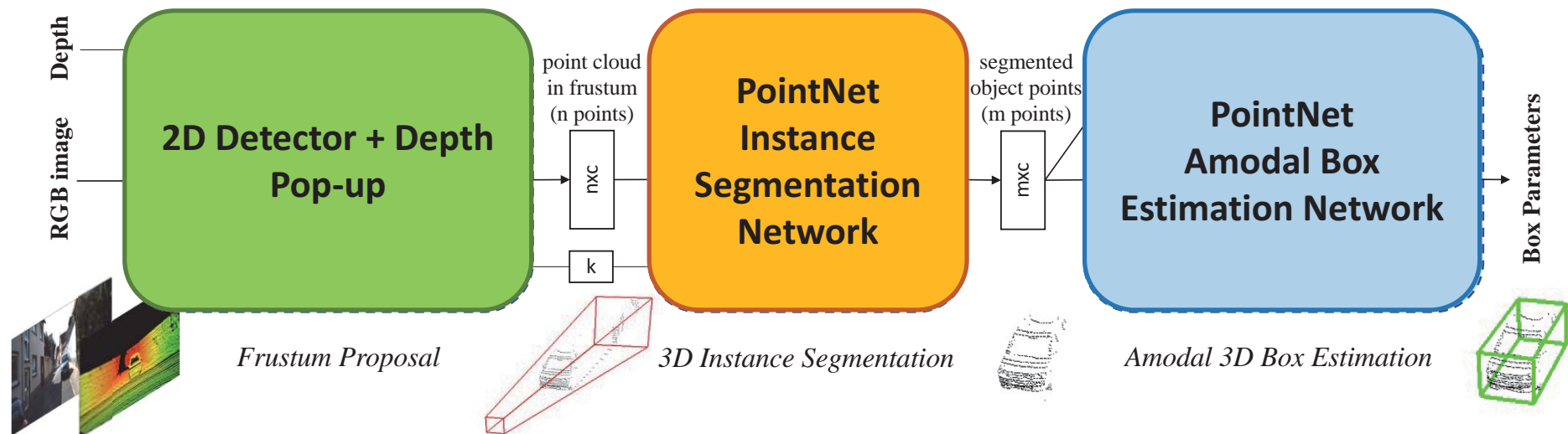
Frustum PointNets

Use **PointNets** for **data-driven** object detection in frustums.



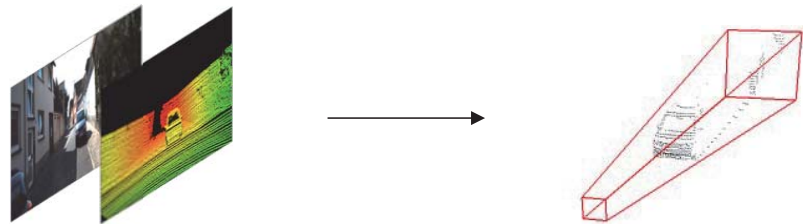
Frustum PointNets

Use **PointNets** for **data-driven** object detection in frustums.



Frustum Proposal

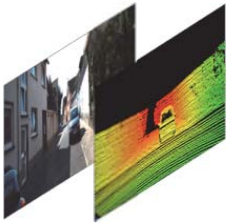
Propose 3D frustums for objects by 2D region proposals in images and depth data back projection.



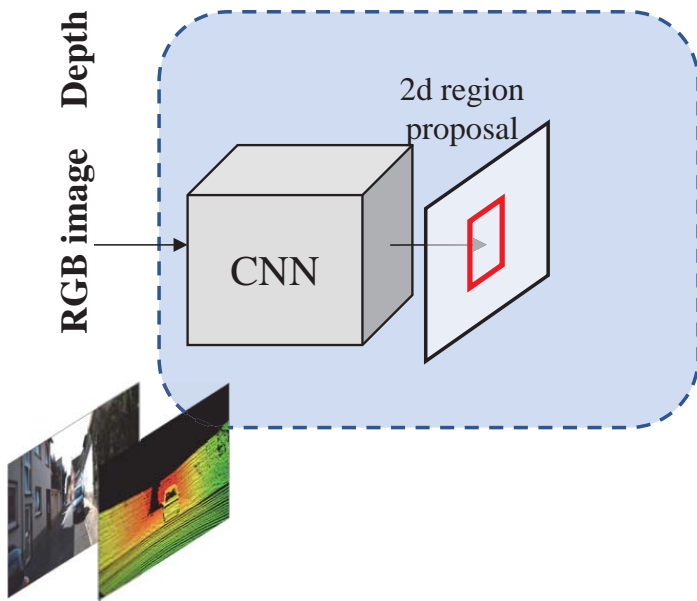
Frustum Proposal

Input: RGB-D data

RGB image Depth



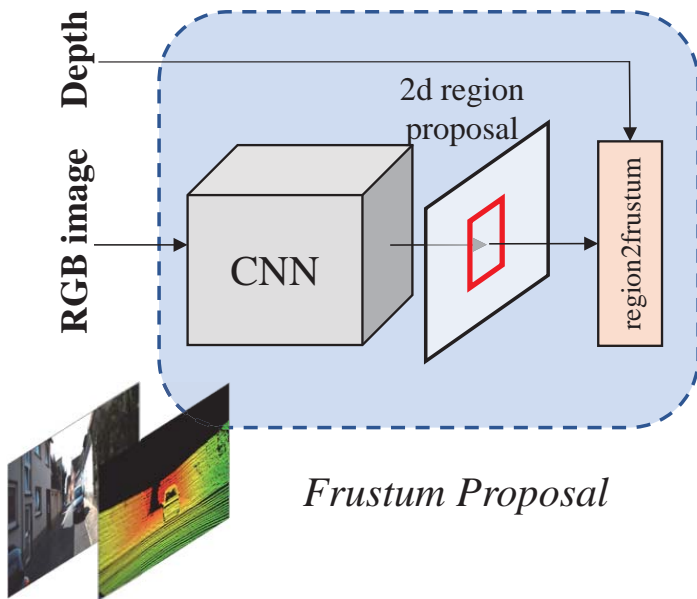
Frustum Proposal



Input: RGB-D data

Image region proposal using a 2D detector on RGB images (high resolution)

Frustum Proposal

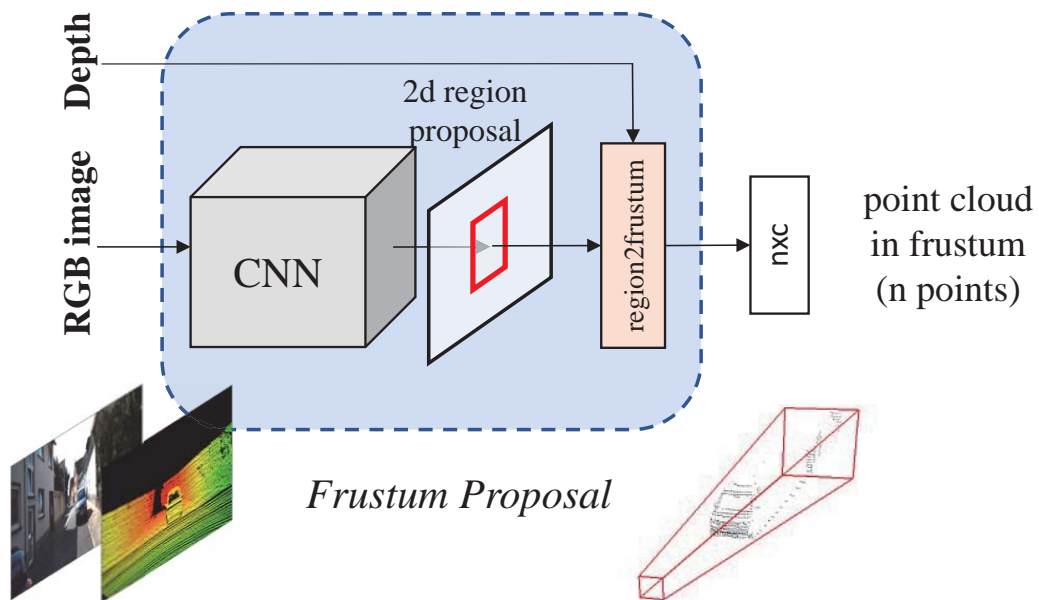


Input: RGB-D data

Image region proposal using a 2D detector on RGB images (high resolution)

Frustum proposal by lifting a 2D region into a 3D frustum.

Frustum Proposal



Input: RGB-D data

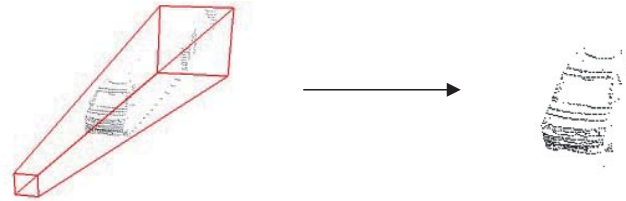
Image region proposal using a 2D detector on RGB images (high resolution)

Frustum proposal by lifting a 2D region into a 3D frustum.

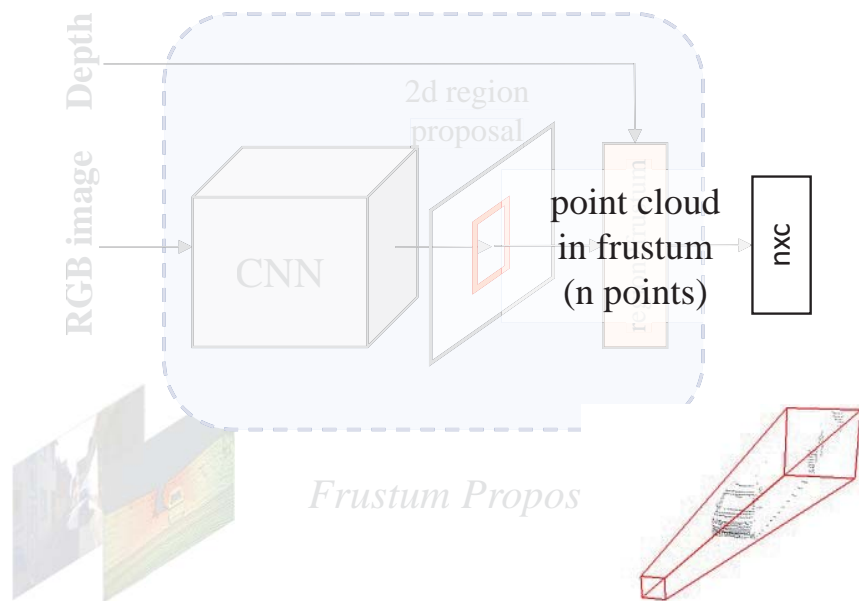
Points in the frustum are extracted and are called a *frustum point cloud*.

3D Instance Segmentation in Frustums

Localize object in frustum by point cloud segmentation.

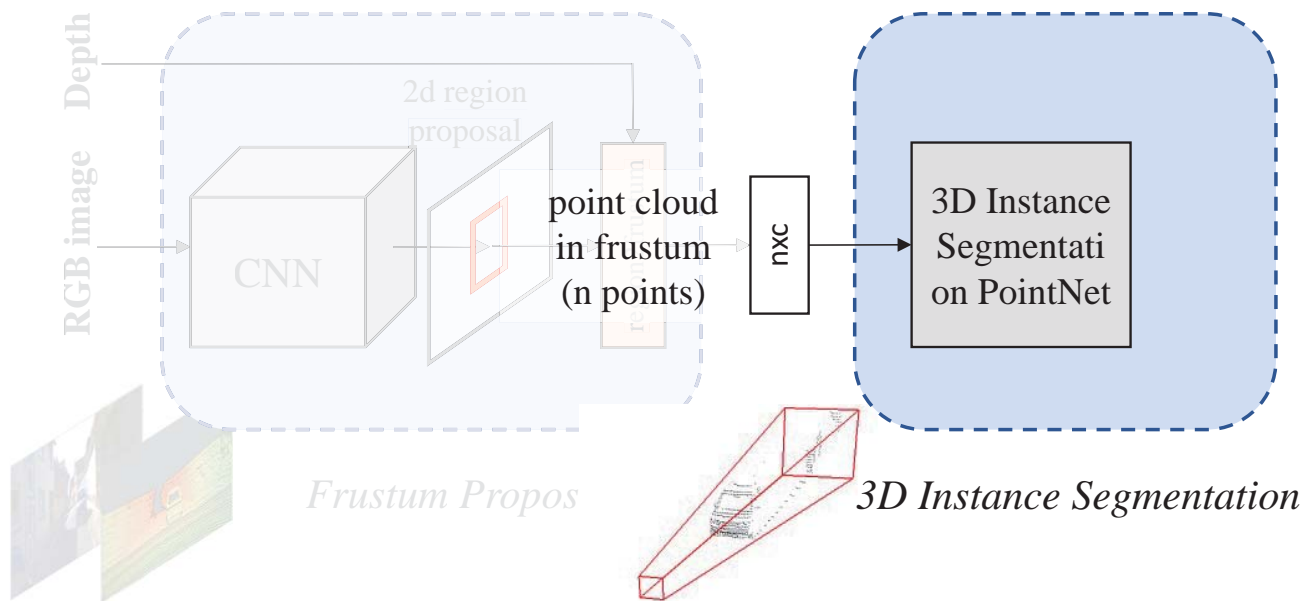


3D Instance Segmentation in Frustums



Input: frustum point cloud

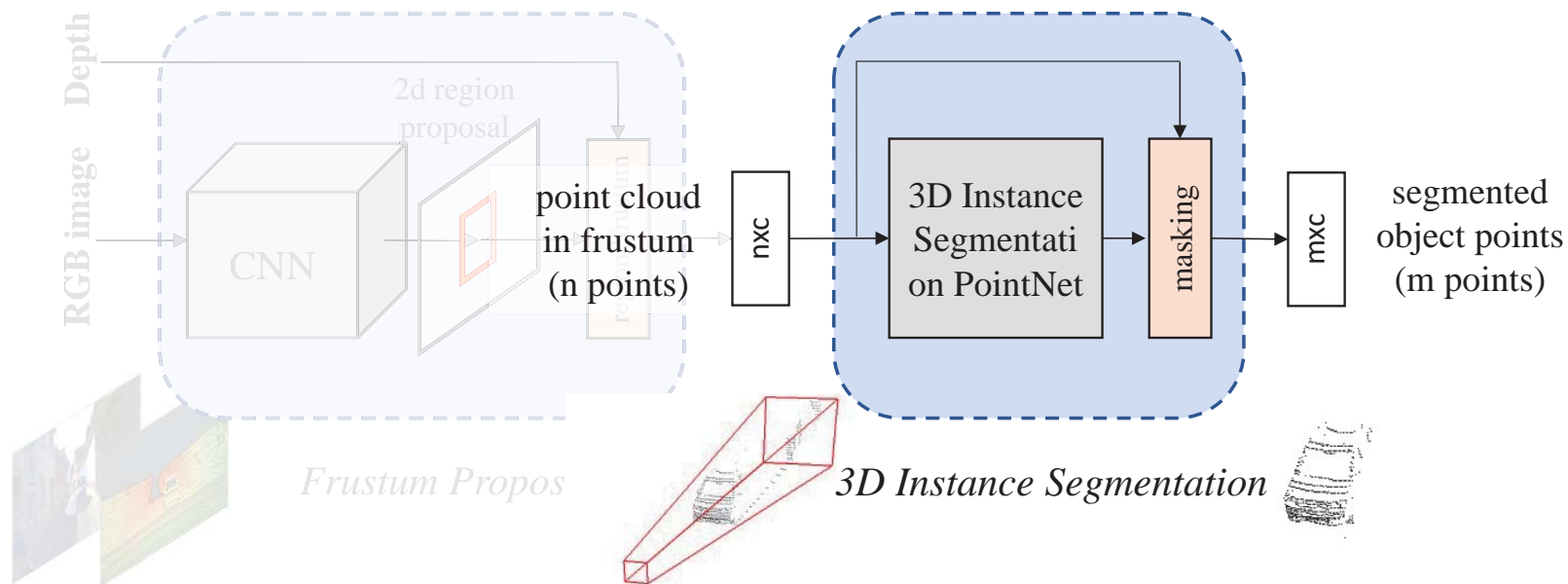
3D Instance Segmentation in Frustums



Input: frustum point cloud

Point cloud binary segmentation with PointNet: object of interest v.s. others

3D Instance Segmentation in Frustums

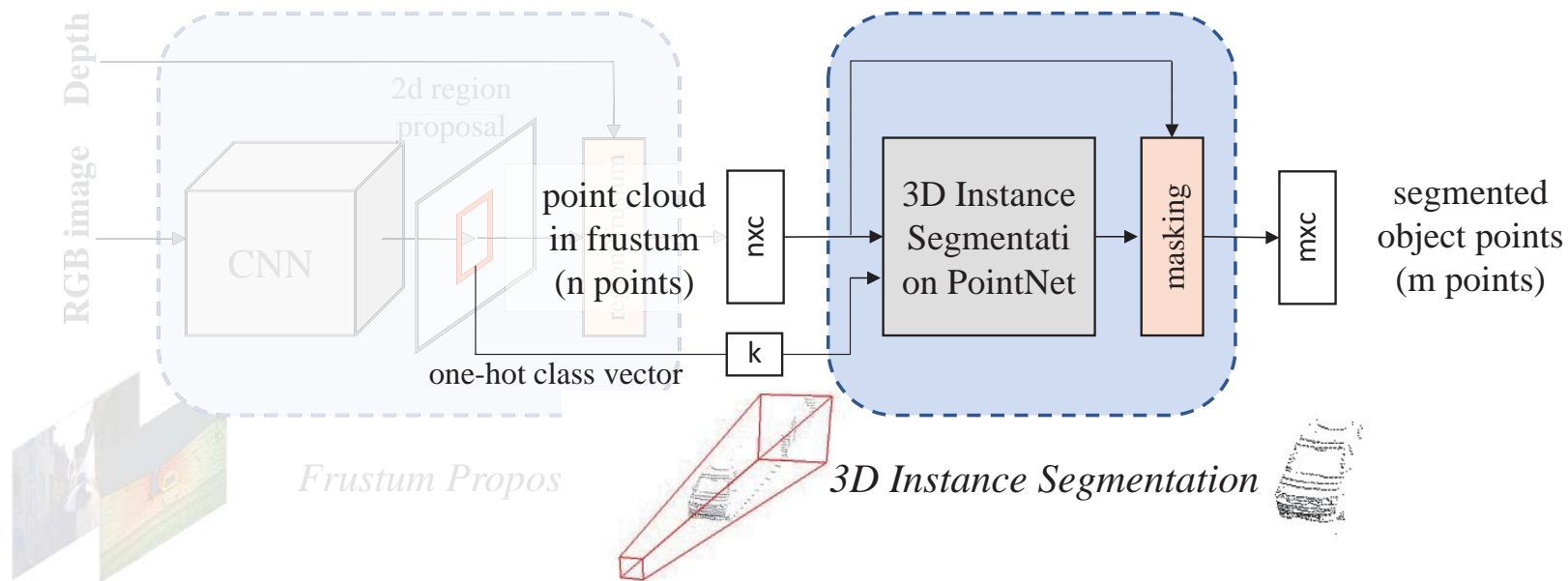


Input: frustum point cloud

Point cloud binary segmentation with PointNet: object of interest v.s. others

Points that are classified as object points are extracted for the next step.

3D Instance Segmentation in Frustums



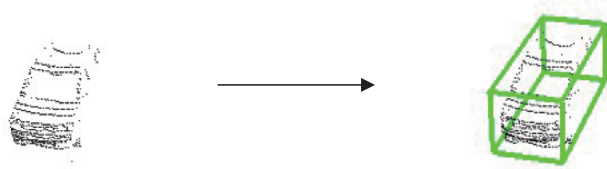
Input: frustum point cloud

Point cloud binary segmentation with PointNet: object of interest v.s. others

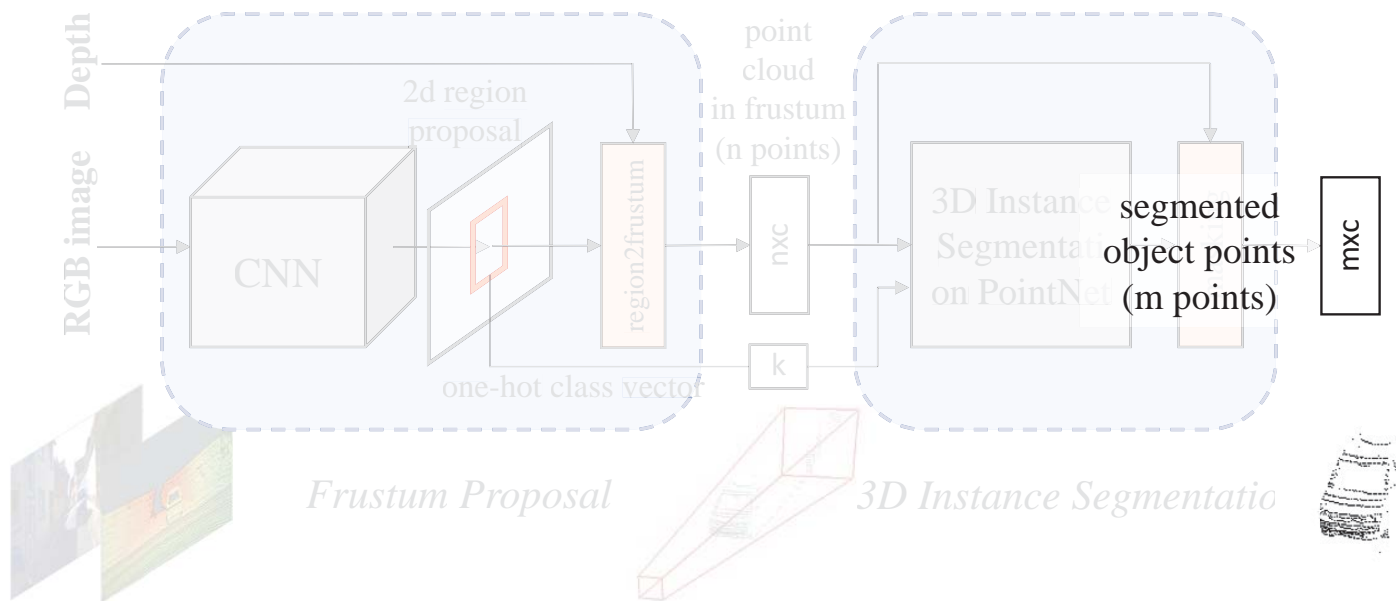
Points that are classified as object points are extracted for the next step.

Amodal 3D Box Estimation

Estimate 3D bounding boxes from segmented object point clouds.

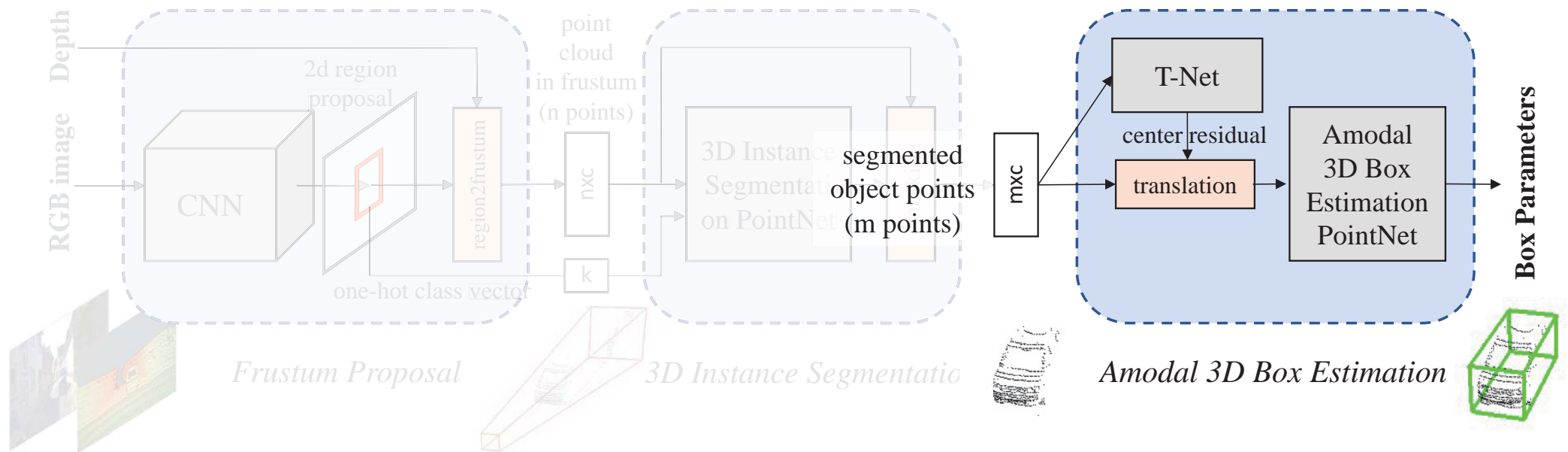


Amodal 3D Box Estimation



Input: object point cloud

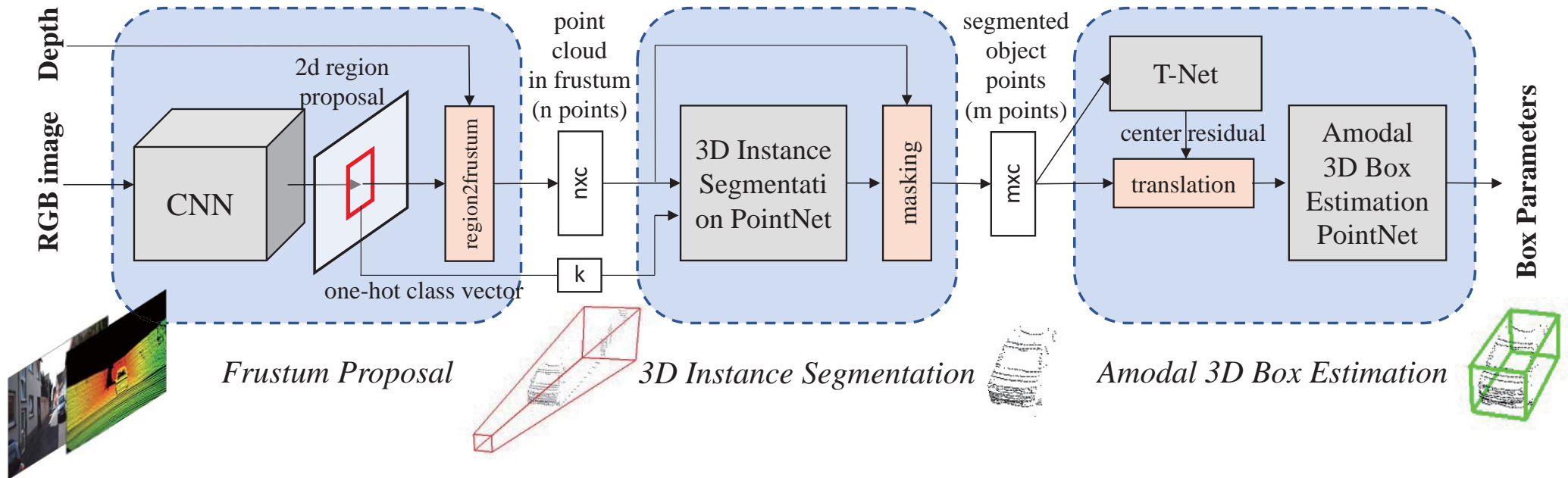
Amodal 3D Box Estimation



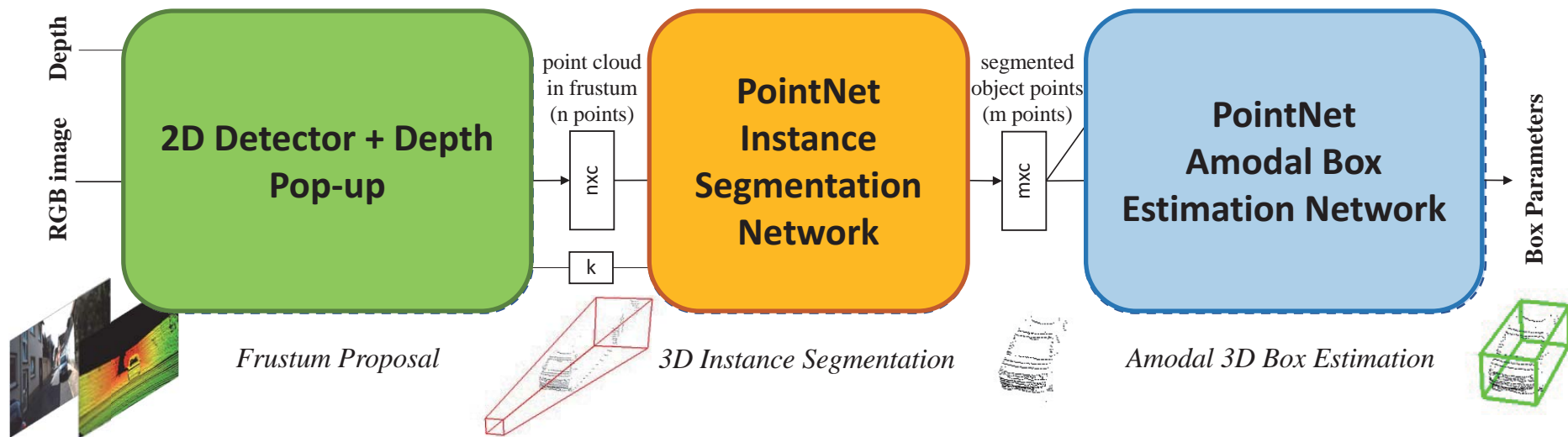
Input: object point cloud

A regression PointNet estimates amodal 3D bounding box for the object

Frustum PointNets



Frustum PointNets



In comparison with Mask R-CNN

Mask R-CNN: 2D box -> 2D segmentation

Naive 3D version of Mask-RCNN: 2D box -> 2D segmentation -> 3D amodal box

Frustum PointNets: 2D box -> 3D frustum -> 3D segmentation -> 3D amodal box

Frustum PointNets: Key to our Success

- **Representation.** We use PointNets for 3D estimation in raw point clouds.
- **Coordinates Normalization.** A series of coordinate transformations canonicalize the learning problems.
- **Loss function.** We design specialized loss functions for 3D bounding box regression.

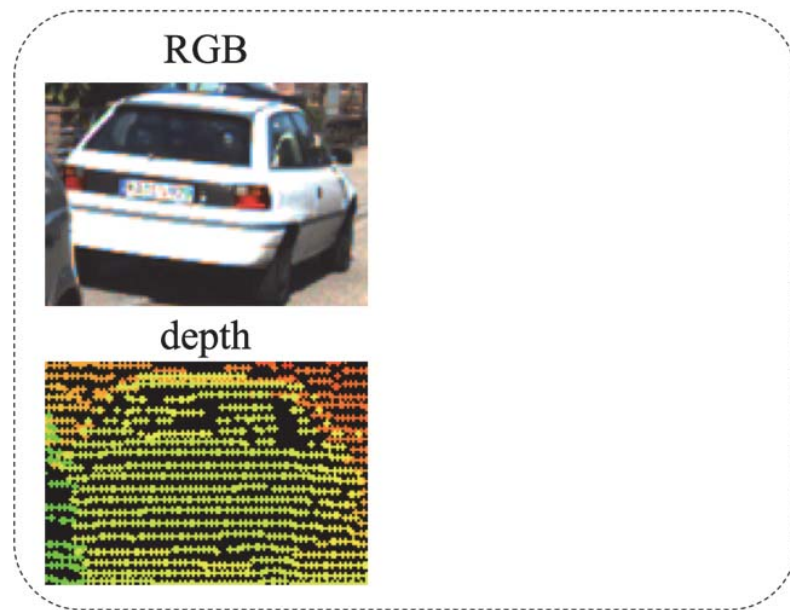
Frustum PointNets: Key to our Success

- **Representation.** We use PointNets for 3D estimation in raw point clouds.
- **Coordinates Normalization.** A series of coordinate transformations canonicalize the learning problems.
- **Loss function.** We design specialized loss functions for 3D bounding box regression.

Frustum PointNets: Key to our Success

- **Representation matters — 2D v.s. 3D**

An alternative representation for 3D object detection is RGB-D image.

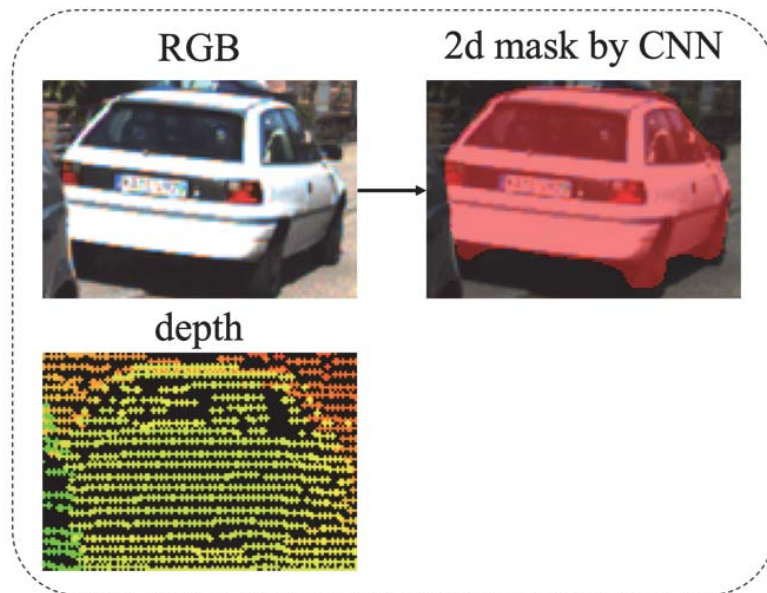


Frustum PointNets: Key to our Success

- **Representation matters — 2D v.s. 3D**

An alternative representation for 3D object detection is **RGB-D image**.

Instead of instance segmentation in 3D, we can segment objects in 2D image.

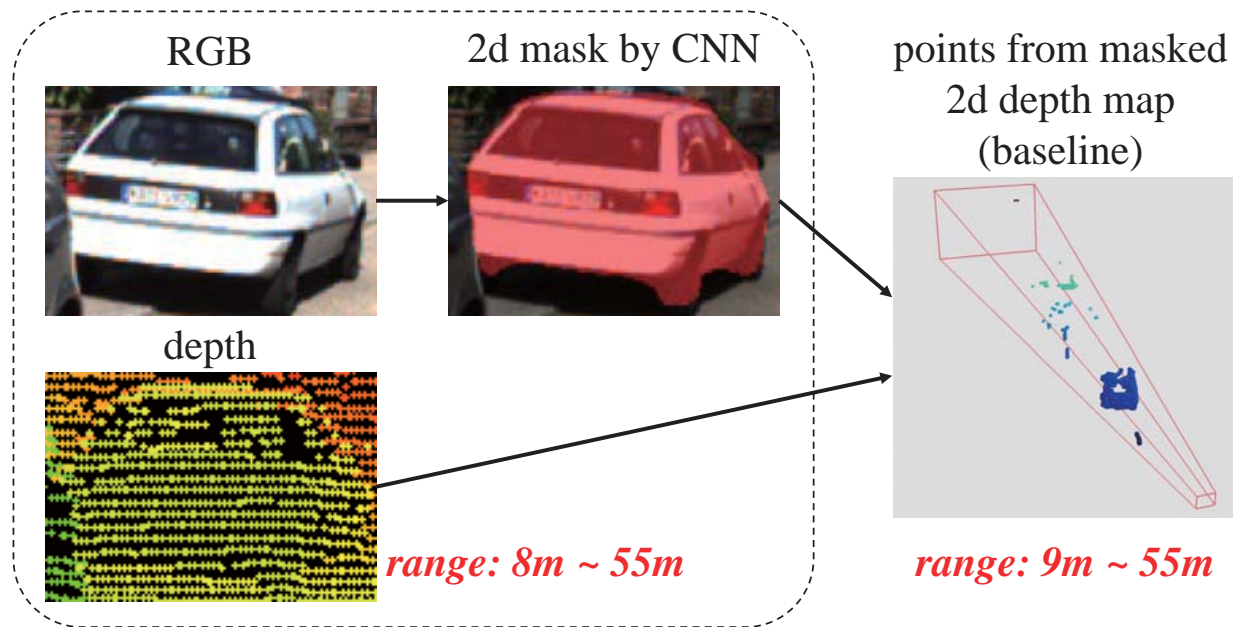


Frustum PointNets: Key to our Success

- **Representation matters — 2D v.s. 3D**

An alternative representation for 3D object detection is **RGB-D image**.

Instead of instance segmentation in 3D, we can segment objects in 2D image.

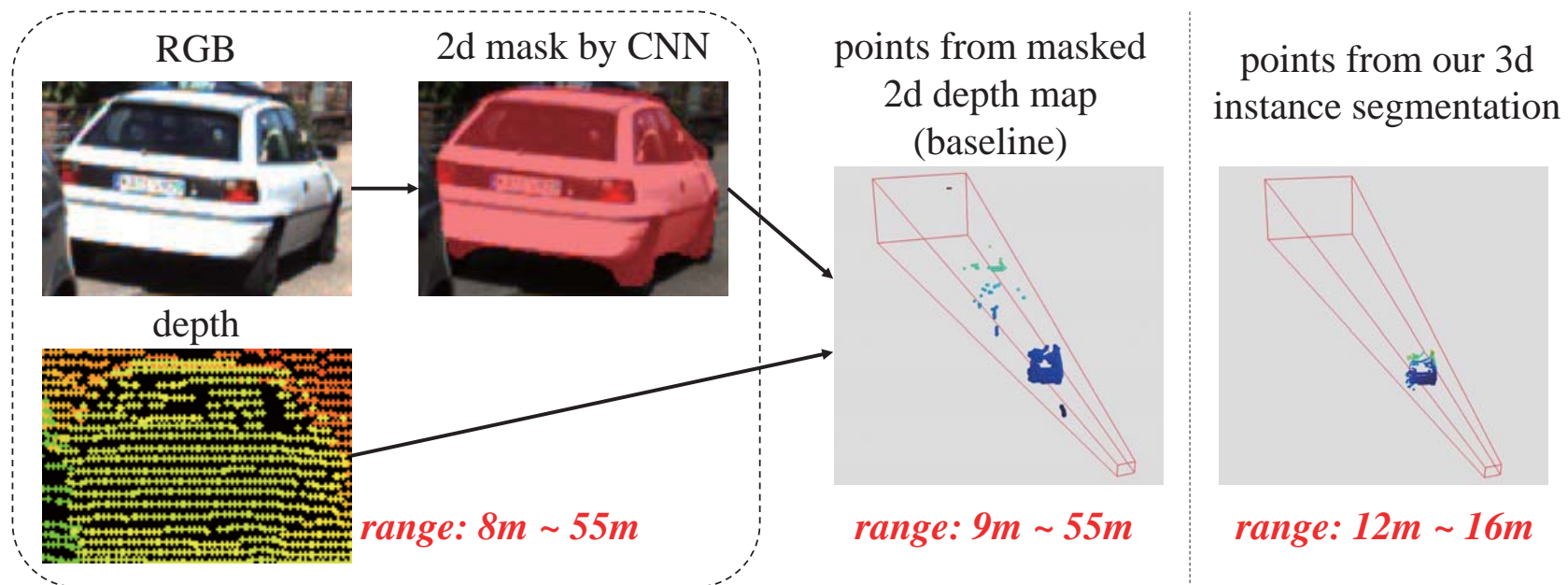


Frustum PointNets: Key to our Success

- **Representation matters — 2D v.s. 3D**

An alternative representation for 3D object detection is **RGB-D image**.

Instead of instance segmentation in 3D, we can segment objects in 2D image.



Frustum PointNets: Key to our Success

- **Representation matters — 2D v.s. 3D**

Comparing 2D and 3D methods.

network arch.	mask	depth representation	accuracy
ConvNet	-	image	18.3
ConvNet	2D	image	27.4
PointNet	-	point cloud	33.5
PointNet	2D	point cloud	61.6
PointNet	3D	point cloud	74.3
PointNet	2D+3D	point cloud	70.0

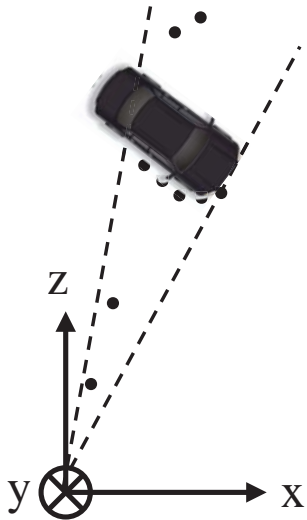
dataset: KITTI; metric: 3D bounding box estimation accuracy (%) under IoU 0.7

Frustum PointNets: Key to our Success

- **Representation.** We use PointNets for 3D estimation in raw point clouds.
- **Coordinates Normalization.** A series of coordinate transformations canonicalize the learning problems.
- **Loss function.** We design specialized loss functions for 3D bounding box regression.

Frustum PointNets: Key to our Success

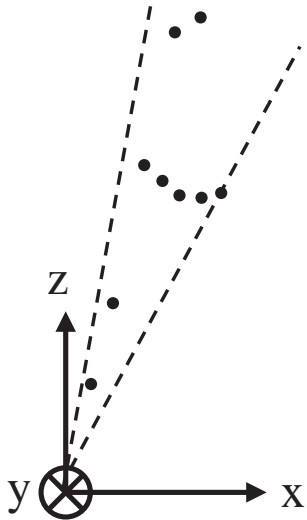
- **Canonicalize** the problem with coordinate normalizations



(a) camera
coordinate

Frustum PointNets: Key to our Success

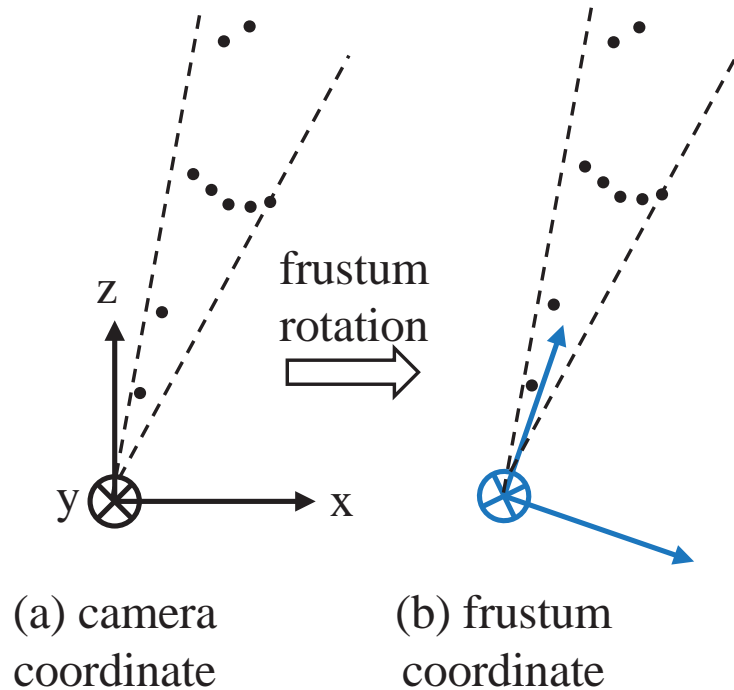
- **Canonicalize** the problem with coordinate normalizations



(a) camera
coordinate

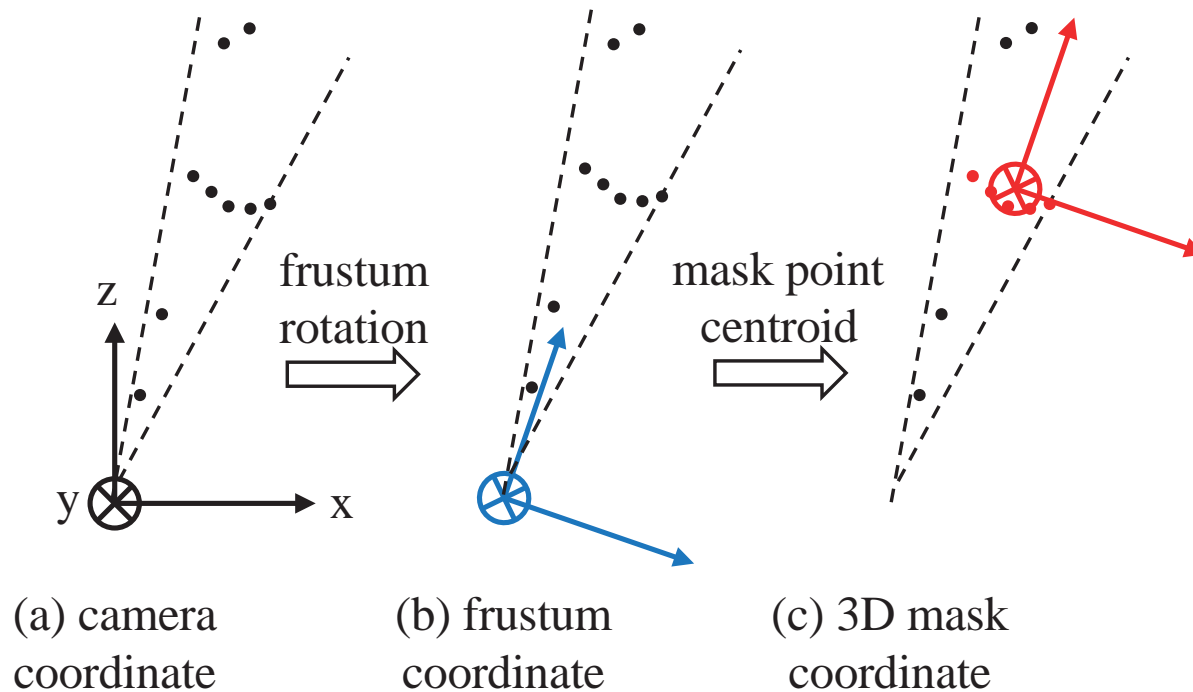
Frustum PointNets: Key to our Success

- **Canonicalize** the problem with coordinate normalizations



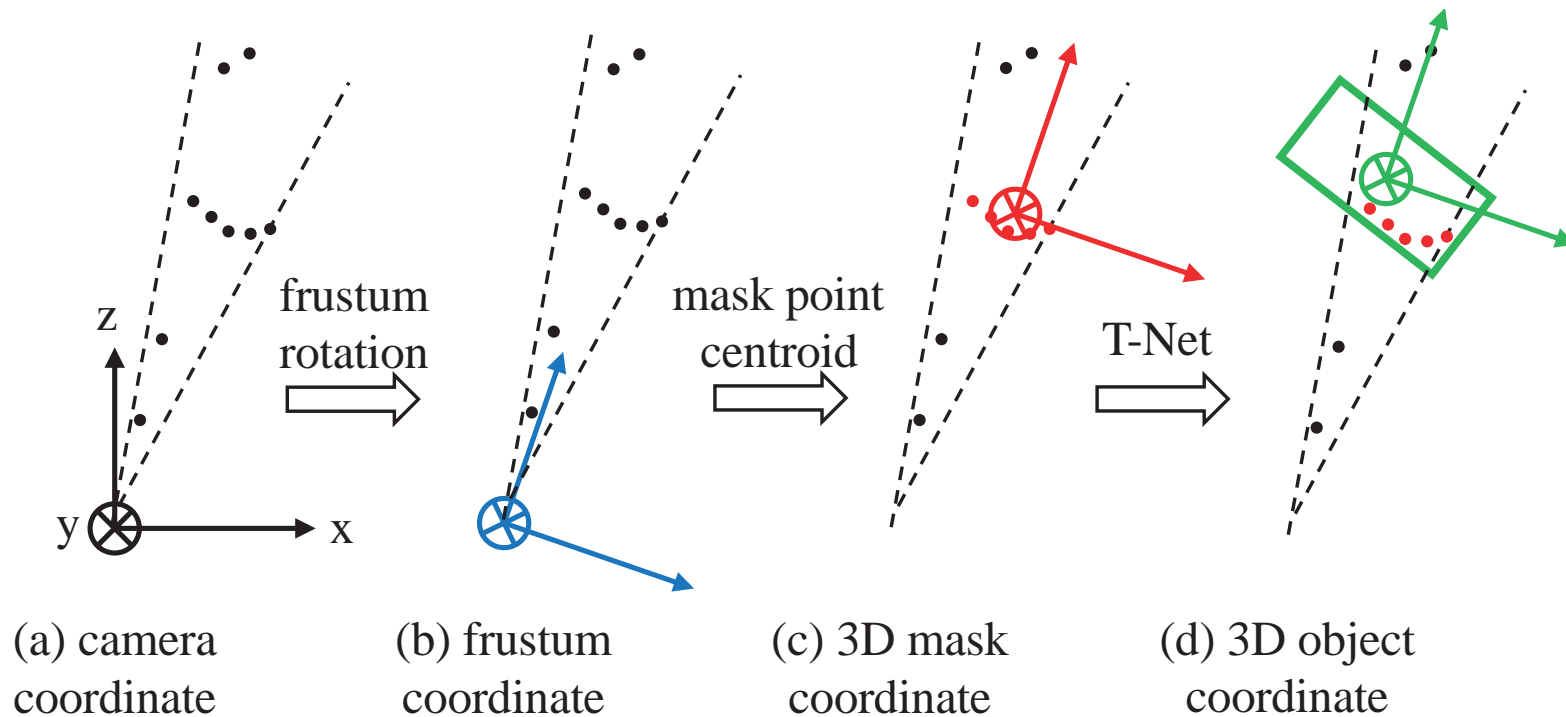
Frustum PointNets: Key to our Success

- **Canonicalize** the problem with coordinate normalizations



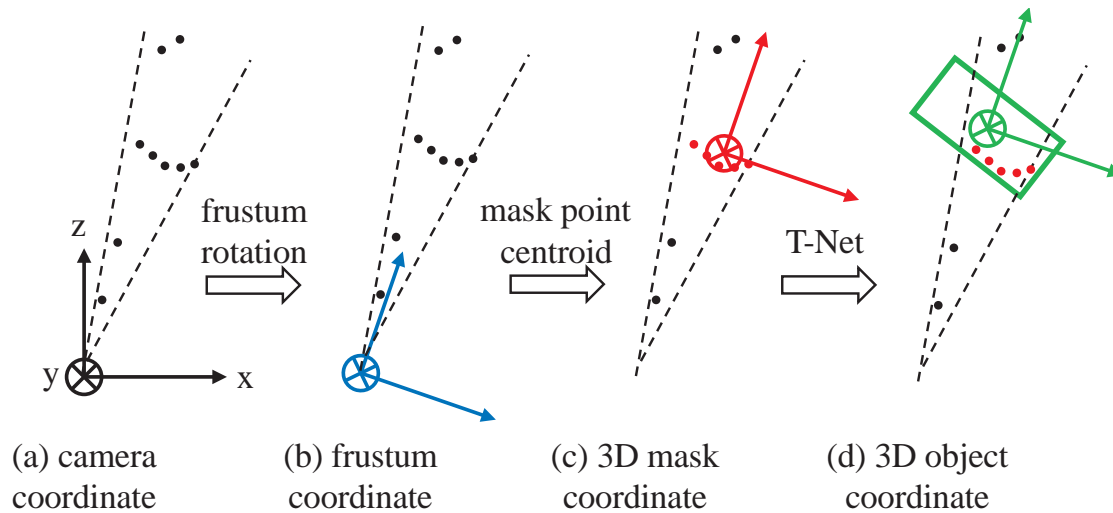
Frustum PointNets: Key to our Success

- **Canonicalize** the problem with coordinate normalizations



Frustum PointNets: Key to our Success

- **Canonicalize** the problem with coordinate normalizations



frustum rot.	mask centralize	t-net	accuracy
-	-	-	12.5
✓	-	-	48.1
-	✓	-	64.6
✓	✓	-	71.5
✓	✓	✓	74.3

dataset: KITTI; metric: 3D bounding box estimation accuracy (%) under IoU 0.7

Frustum PointNets: Key to our Success

- **Representation.** We use PointNets for 3D estimation in raw point clouds.
- **Coordinates Normalization.** A series of coordinate transformations canonicalize the learning problems.
- **Loss function.** We design specialized loss functions for 3D bounding box regression.

Loss Functions for 3D Box Regression

A 3D bounding box is parameterized by its center c_x, c_y, c_z , its size h, w, l and its orientation θ, ϕ, ψ relative to a canonical pose for each category.

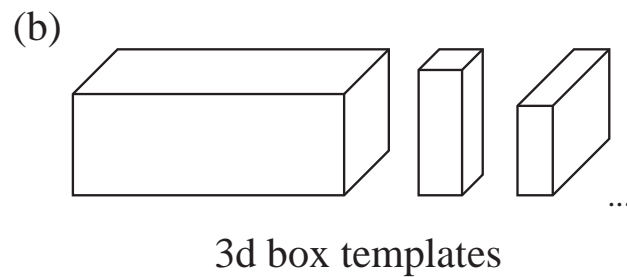
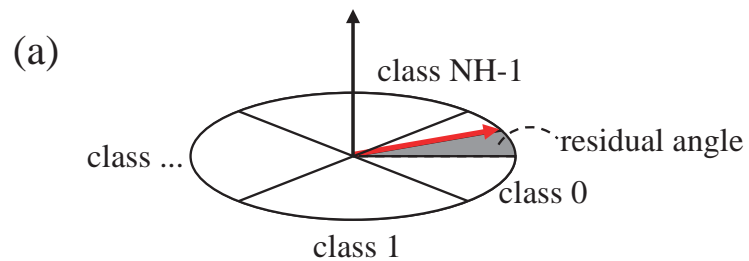
Loss Functions for 3D Box Regression

A 3D bounding box is parameterized by its center c_x, c_y, c_z , its size h, w, l and its orientation θ, ϕ, ψ relative to a canonical pose for each category.

In our case we assume an object sits on a flat plane and only estimate the orientation θ around the up-axis.

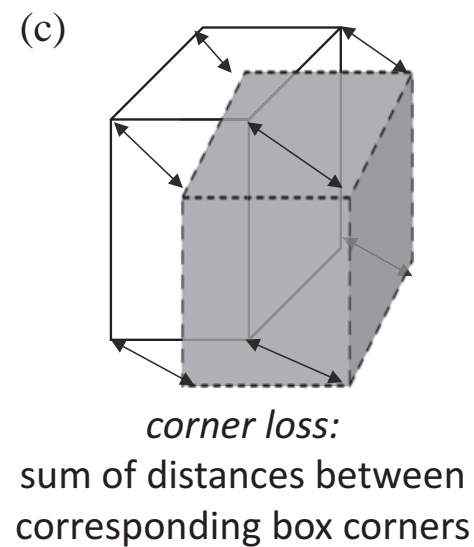
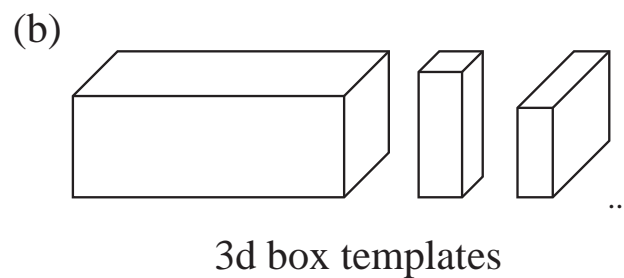
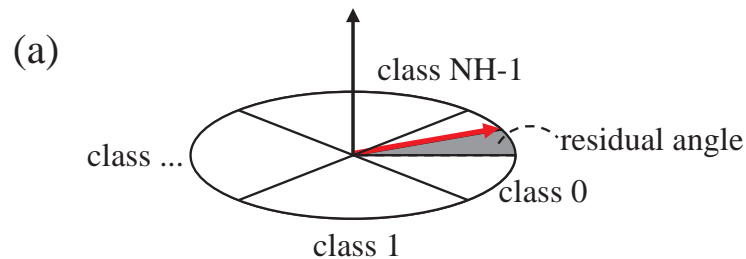
Loss Functions for 3D Box Regression

- Instead of regressing to absolute values, we use a hybrid of classification and regression (cls-reg) formulation.



Loss Functions for 3D Box Regression

- Instead of regressing to absolute values, we use a hybrid of classification and regression formulation.
- A regularizing loss (corner loss) for joint optimization of center, size and heading angle



Loss Functions for 3D Box Regression

- Instead of regressing to absolute values, we use a hybrid of classification and regression formulation.
- A regularizing loss (corner loss) for joint optimization of center, size and heading angle
- Multi-task learning

$$L_{multi-task} = L_{seg} + \lambda(L_{c1-reg} + L_{c2-reg} + L_{h-cls} + L_{h-reg} + L_{s-cls} + L_{s-reg} + \gamma L_{corner})$$

Loss Functions for 3D Box Regression

loss type	regularization	accuracy
regression only	-	62.9
cls-reg	-	71.8
cls-reg (normalized)	-	72.2
cls-reg (normalized)	corner loss	74.3

Table 8. **Effects of 3D box loss formulations.** Metric is 3D box estimation accuracy with IoU=0.7.

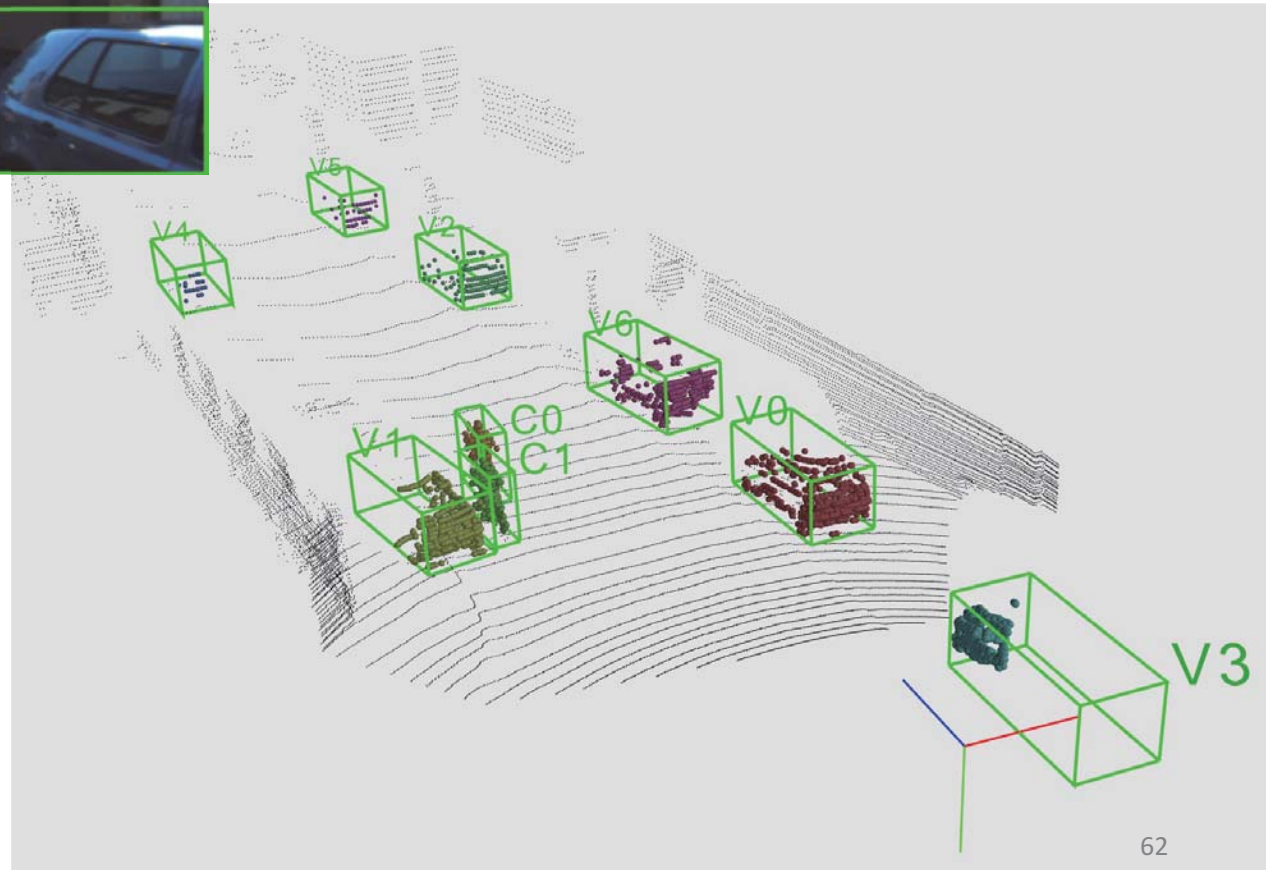
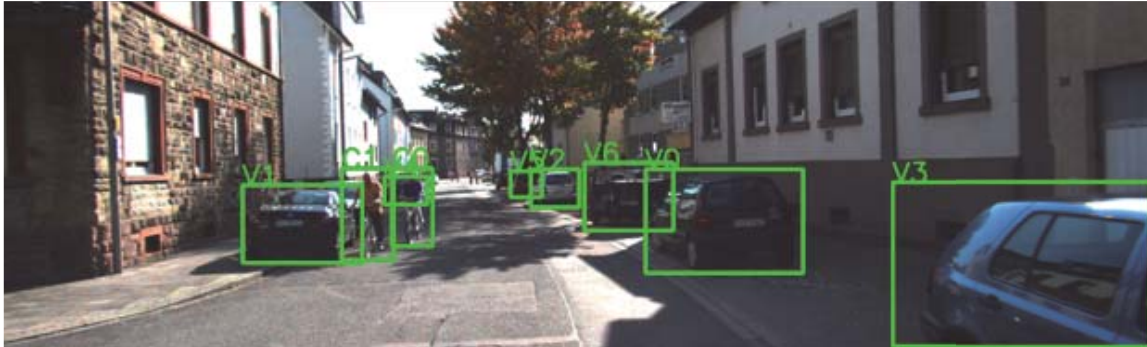
Frustum PointNets: Key to our Success

Respect and exploit 3D

- **Representation matters** — using 3D representation and 3D deep learning for the 3D problem.
- **Canonicalize the problem** — exploiting geometric transformations in point clouds.
- **Special 3D loss functions.**

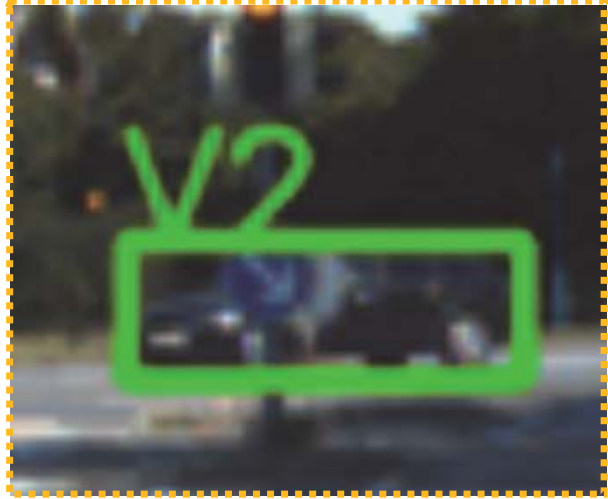
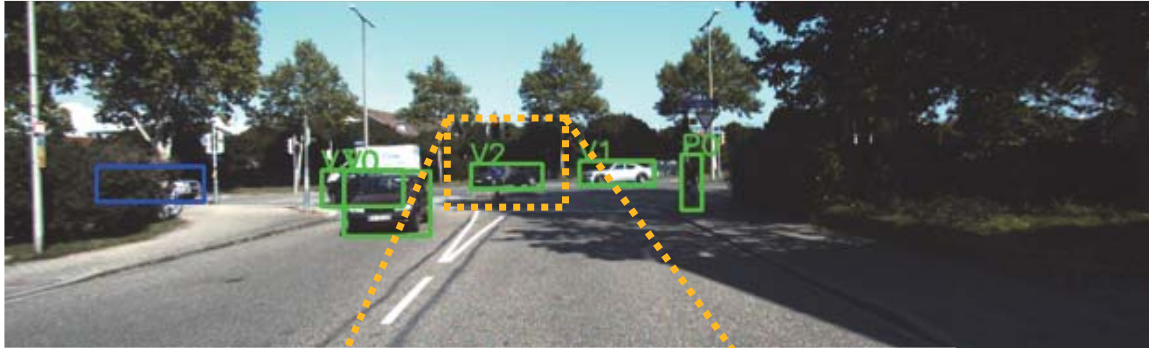
Qualitative Results (on KITTI and SUN-RGBD)

KITTI Results: Qualitative

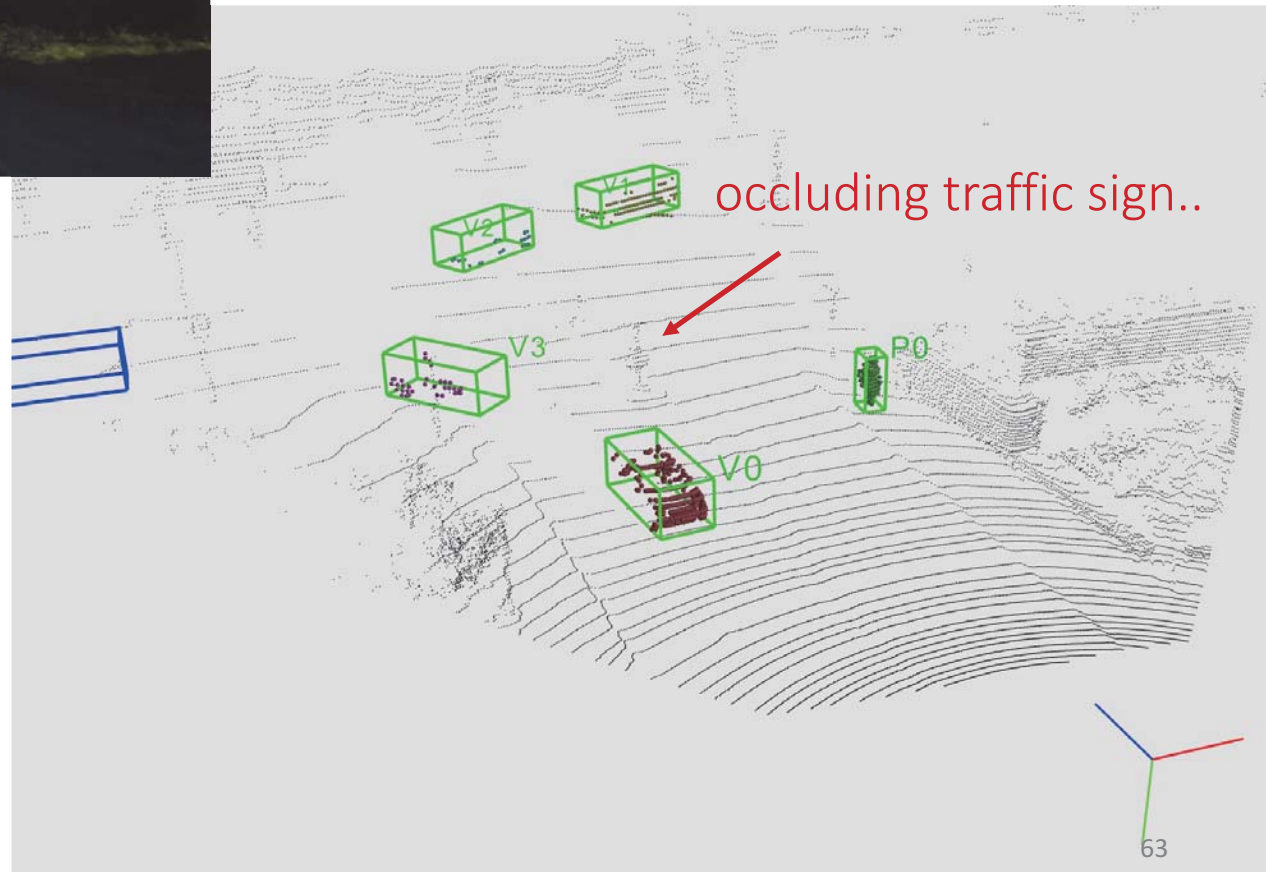


Remarkable box estimation accuracy even with a dozen of points or with very partial point clouds.

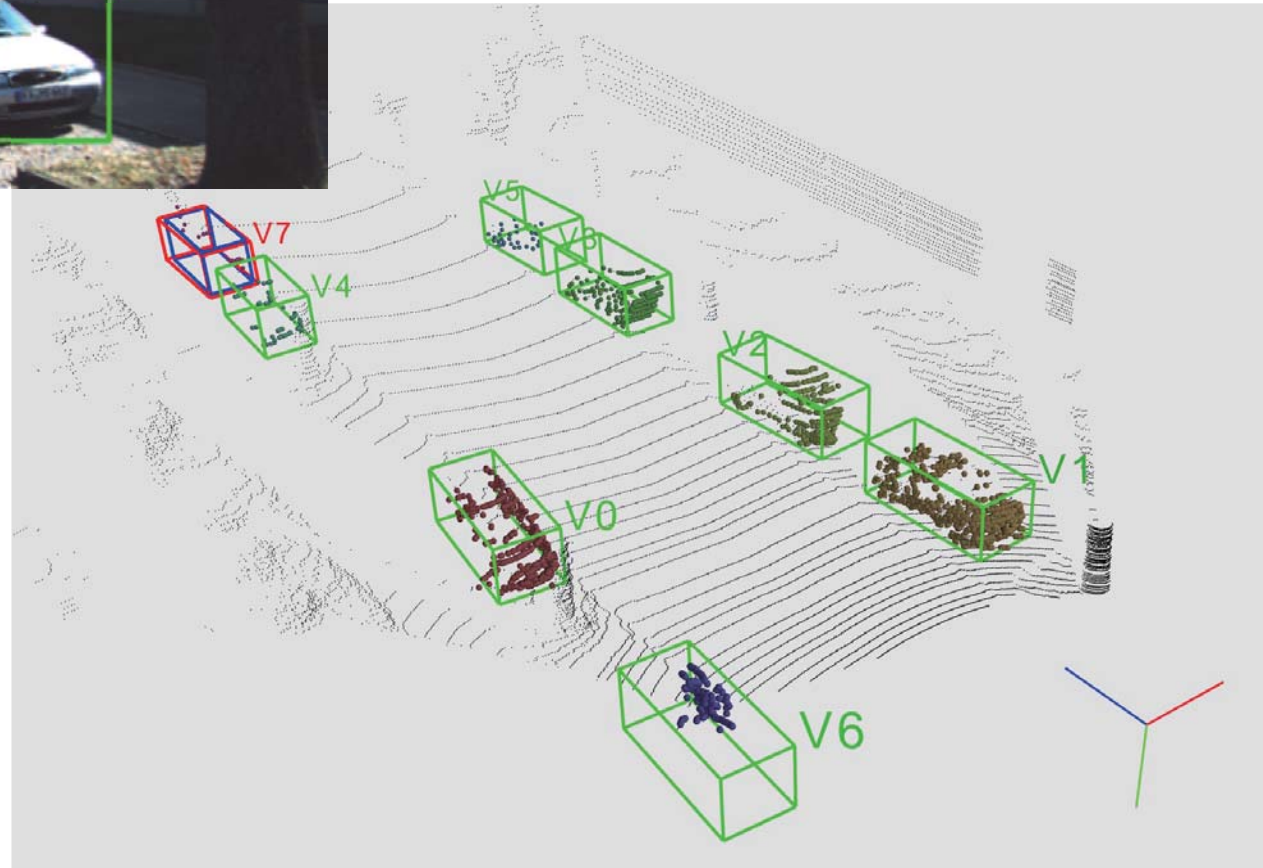
KITTI Results: Qualitative



Correct segmentation in point clouds with heavy occlusion.



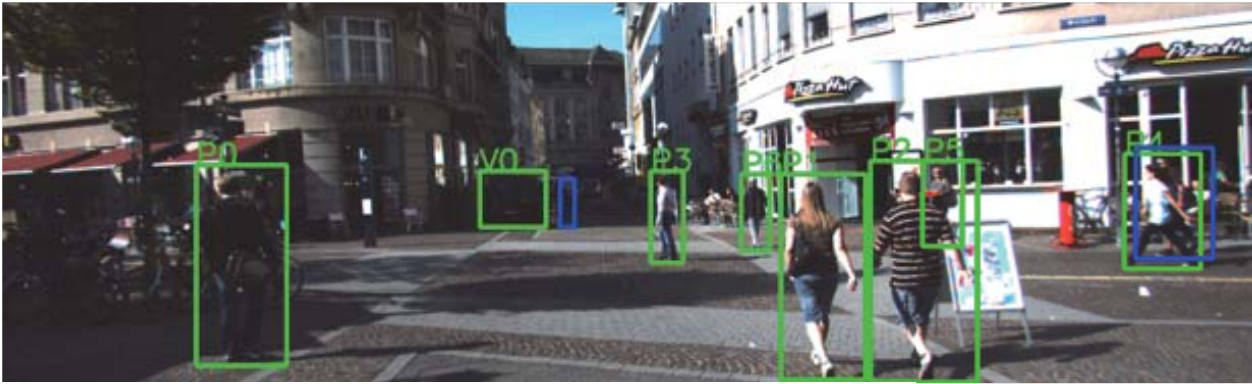
KITTI Results: Qualitative



In-accurate box regression with too few LiDAR points

Image features could help.

KITTI Results: Qualitative

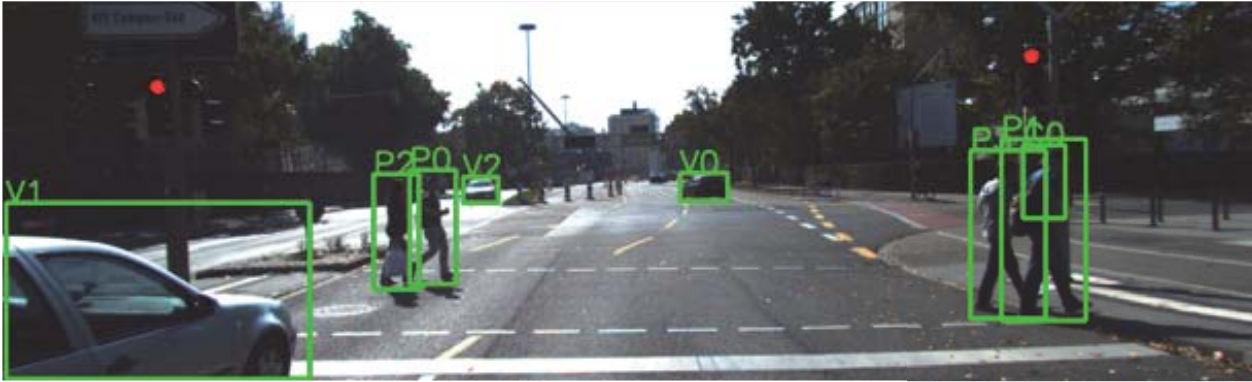


Missing 2D detection results in no 3D detection

Multiple ways for proposal could help (e.g. bird's eye view, multiple 2D proposal networks)

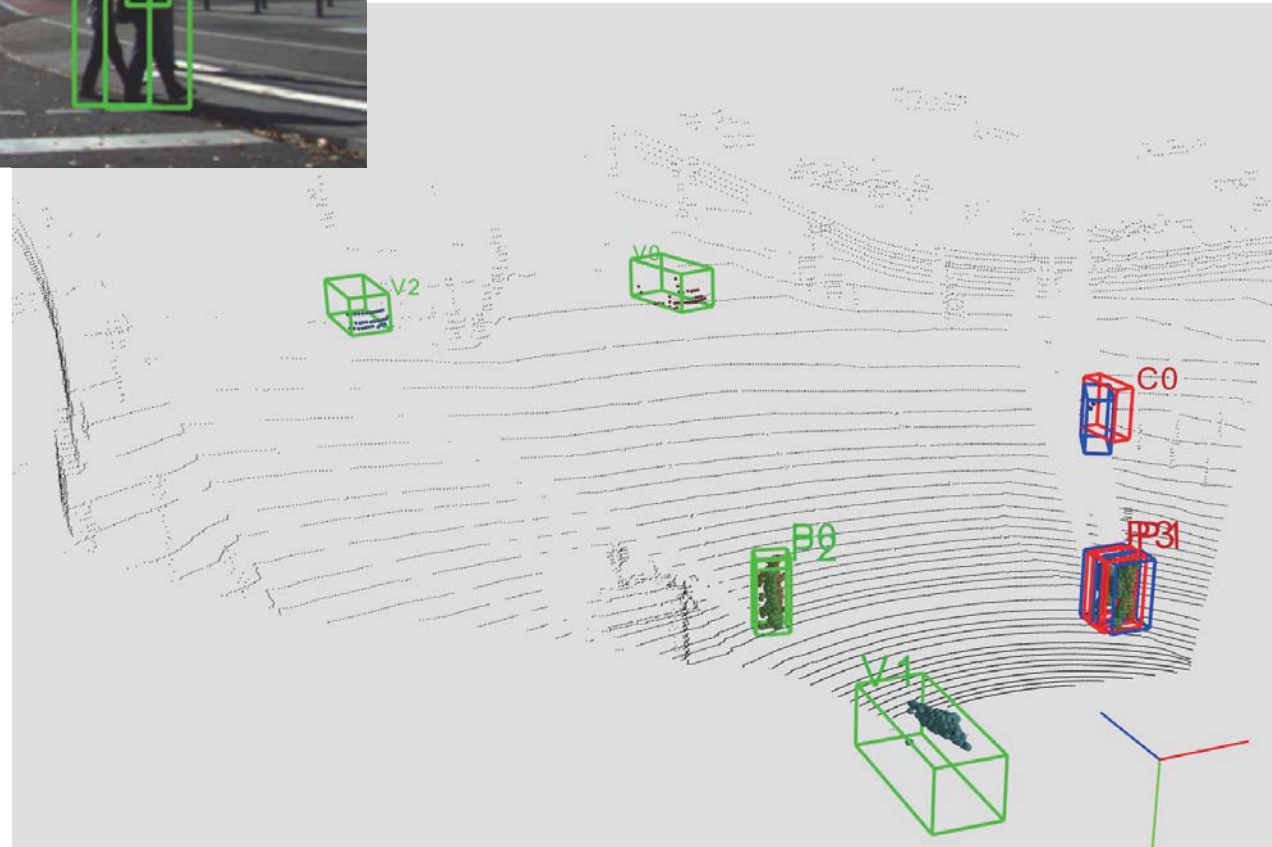


KITTI Results: Qualitative



Strong occlusion. Just 4 LiDAR points..

Challenging case for instance segmentation (multiple closeby objects in a single frustum)

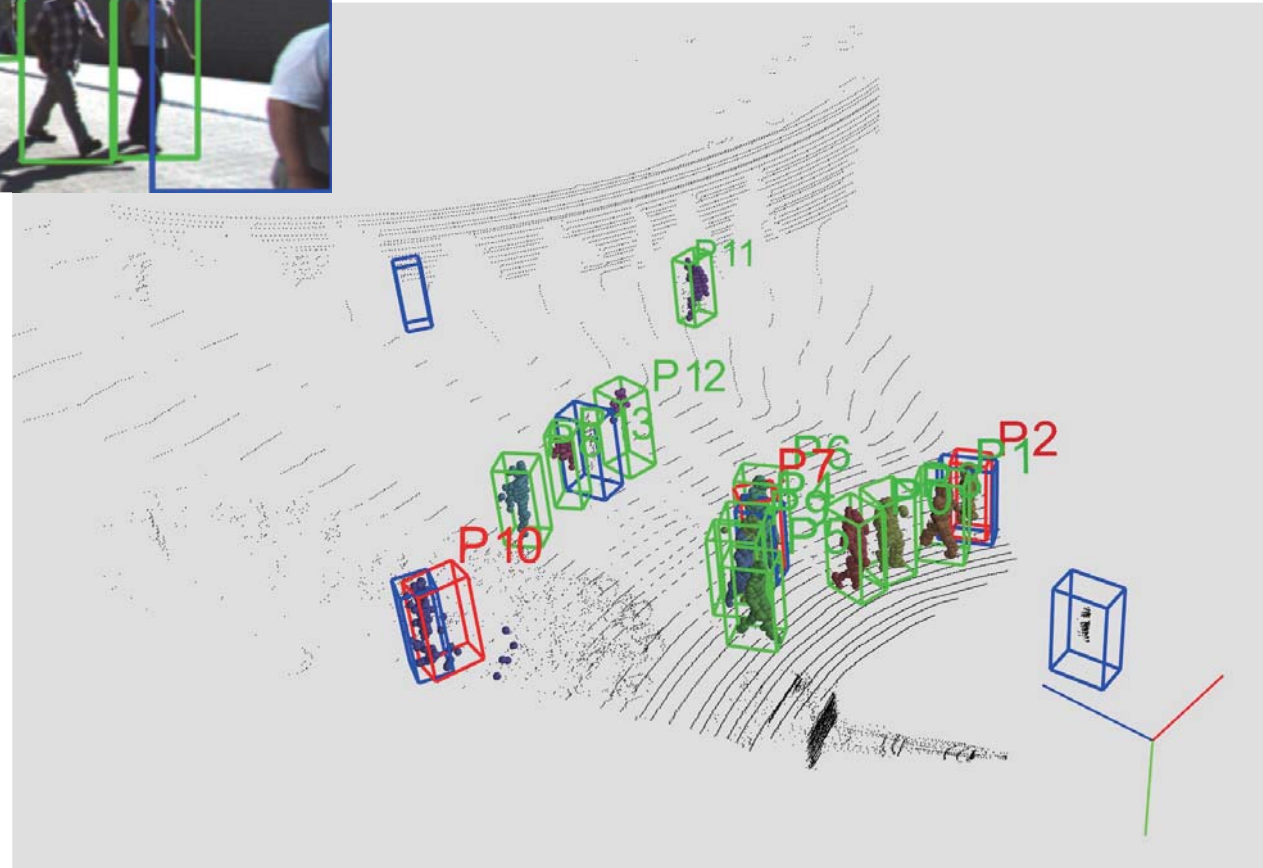


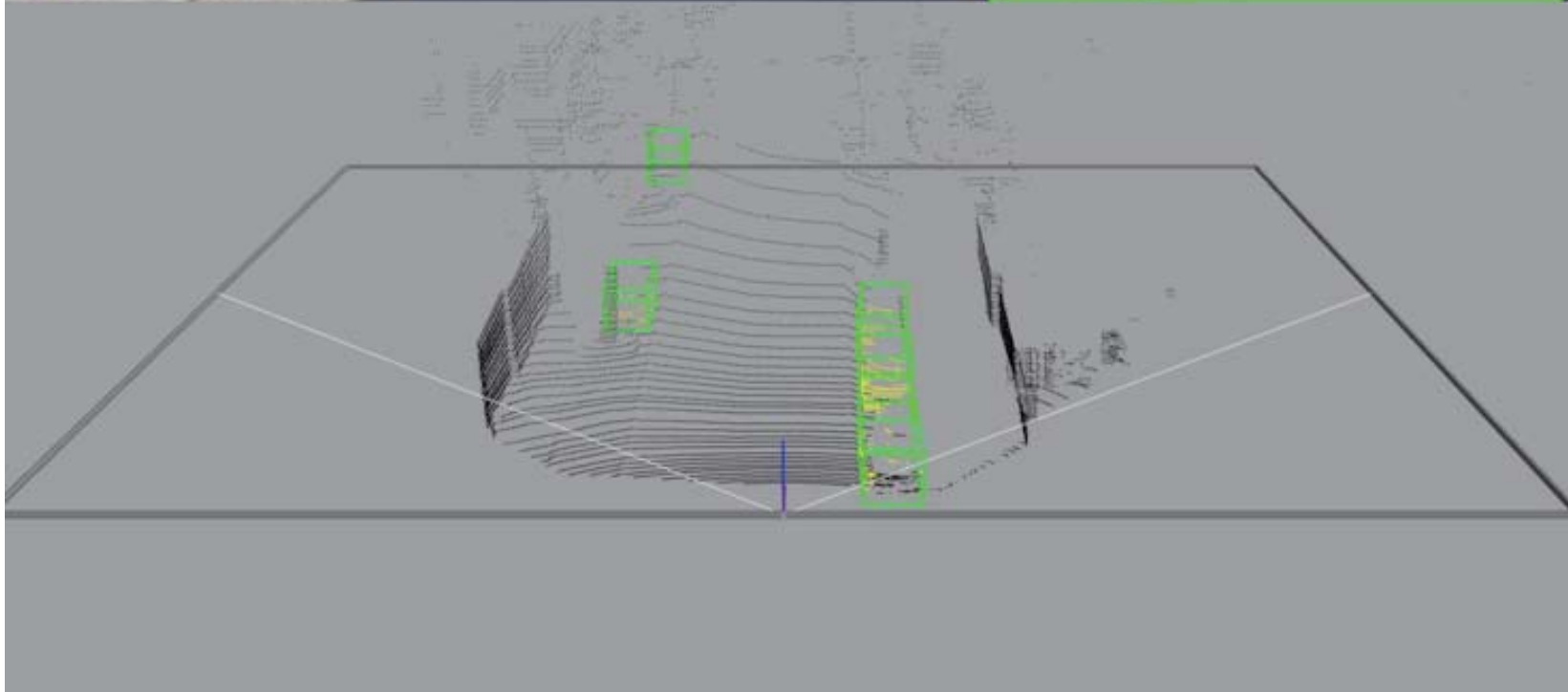
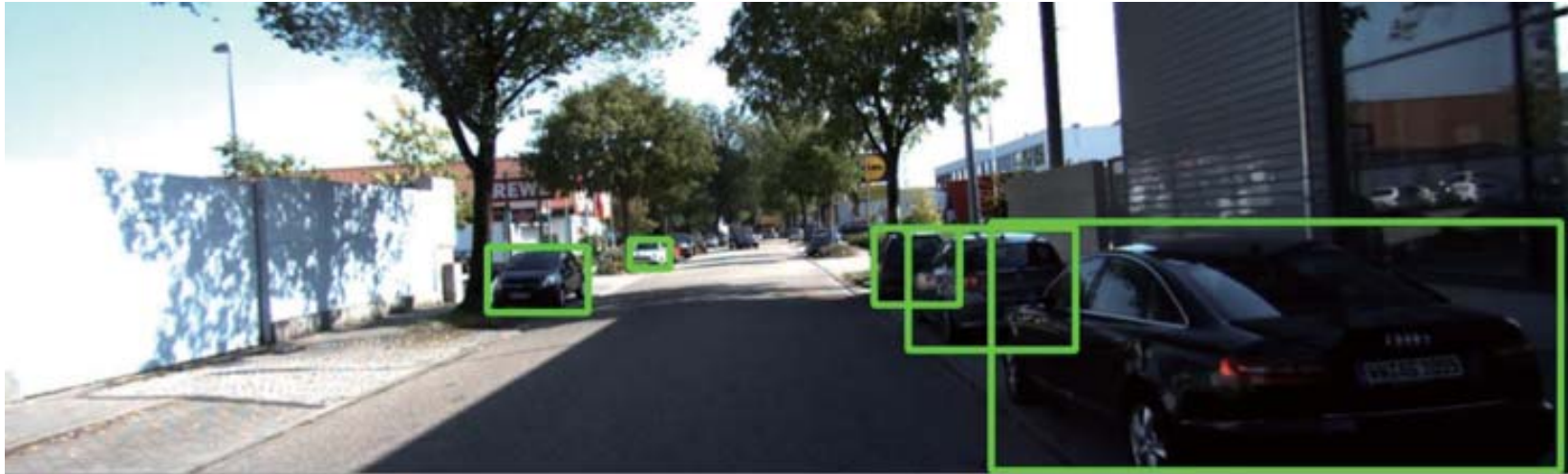
KITTI Results: Qualitative



Missed 2D detection in a complicated scene with strong occlusions

Challenging segmentation cases





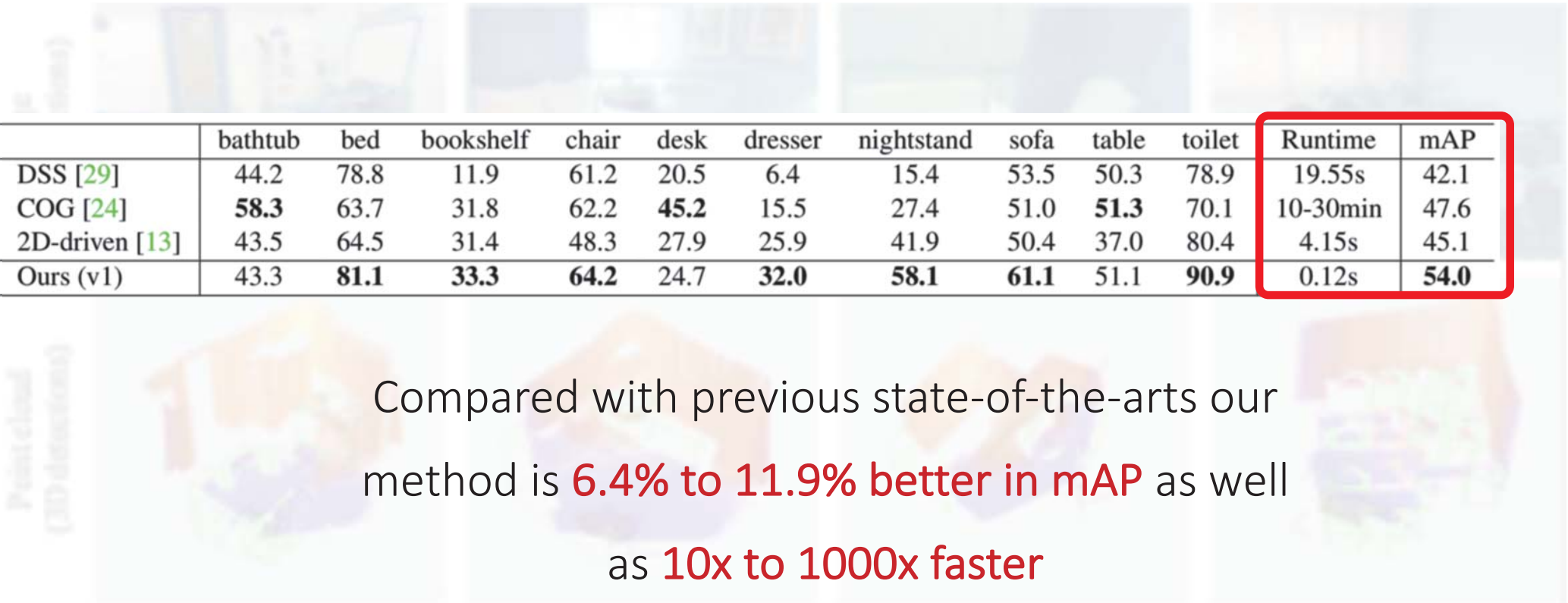
SUN RGB-D Results

Our method can be directly applied to indoor RGB-D data



SUN RGB-D Results

Our method can be directly applied to indoor RGB-D data



	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	Runtime	mAP
DSS [29]	44.2	78.8	11.9	61.2	20.5	6.4	15.4	53.5	50.3	78.9	19.55s	42.1
COG [24]	58.3	63.7	31.8	62.2	45.2	15.5	27.4	51.0	51.3	70.1	10-30min	47.6
2D-driven [13]	43.5	64.5	31.4	48.3	27.9	25.9	41.9	50.4	37.0	80.4	4.15s	45.1
Ours (v1)	43.3	81.1	33.3	64.2	24.7	32.0	58.1	61.1	51.1	90.9	0.12s	54.0

Compared with previous state-of-the-arts our method is **6.4% to 11.9% better in mAP** as well as **10x to 1000x faster**

Conclusion

- We propose Frustum PointNets – a novel framework for 3D object detection with 3D deep learning.
- We show how we can train 3D object detectors under our framework which achieve state-of-the-art performance on standard 3D object detection benchmarks.
- We provide extensive quantitative evaluations to validate our design choices as well as rich qualitative results for understanding the strengths and limitations of our method.

Code on GitHub:

<https://github.com/charlesq34/frustum-pointnets>

