An Analysis on the Competitive Capabilities of Stereo Vision Cameras: Stereo Vision for 3D Object Localization

Kristian Gonzalez
Masters Mechatronics Thesis Presentation

12 September 2019

Institut für
Künstliche Intelligenz

Introduction: Context (1/2)

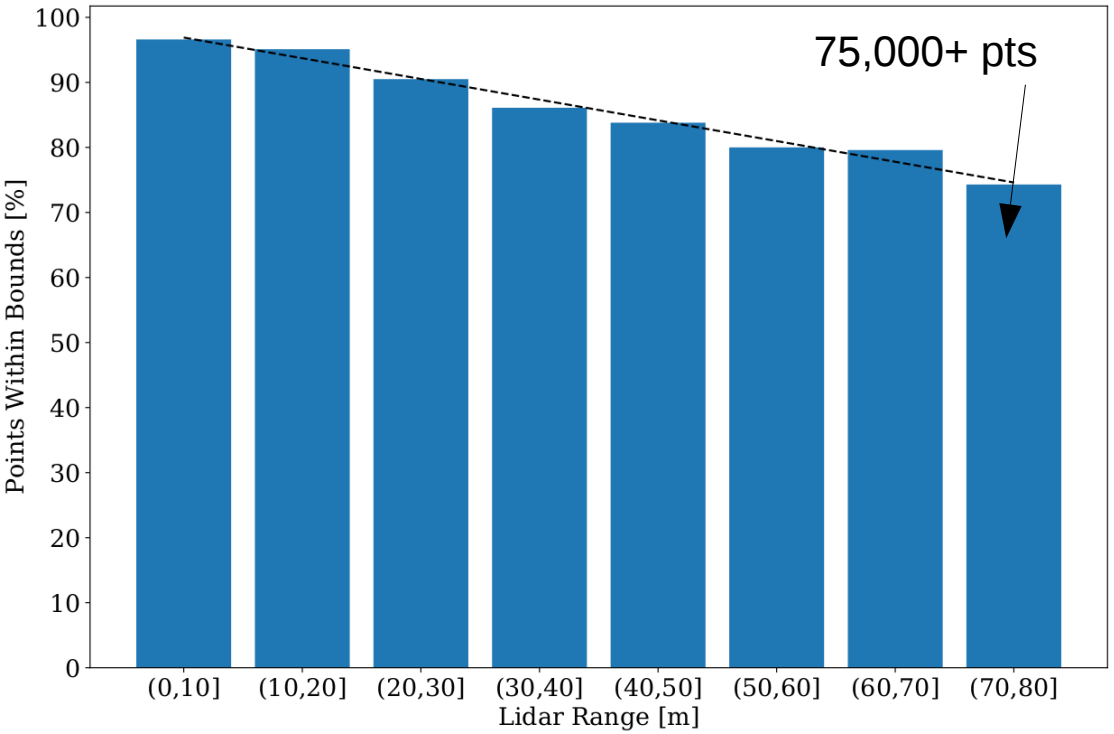
- SMART3D origin
- Related Work:
  - Pyramid Stereo Matching Network
  - iResNet
  - Semi-Global Block Matching

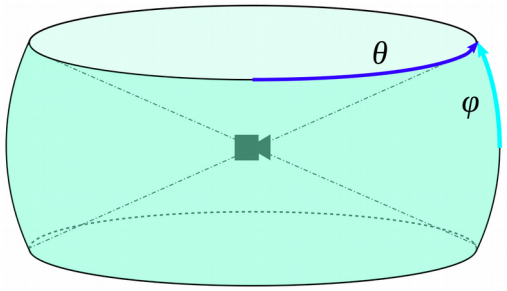
  - Frustum Pointnet
  - Sparse-to-Dense net
  - Real-Time 3D net
  - Stereo R-CNN
  - Pseudo-Pointcloud net

  (others as cited)

Introduction: Motivation (2/2)

- Lidar:
  - Linear error
  - 360 degree vision
- Stereo cameras:
  - Affordable
  - High refresh rate
  - Passive sensor





Property	Lidar	Stereo (x1)	Stereo (x4)
hFOV (deg)	360	81	324
vFOV (deg)	20.0	31.5	31.5
Cost (USD)	75,000	1,400	5,600
Points/Scan	476,898	453,376	1,813,504
Angular Area (rad^2)	2.18	0.77	3.07
Points/Price (pts/USD)	6.36	323.84	323.84
Points/Area (pts/rad^2)	218,761	588,800	588,800

Network Architecture

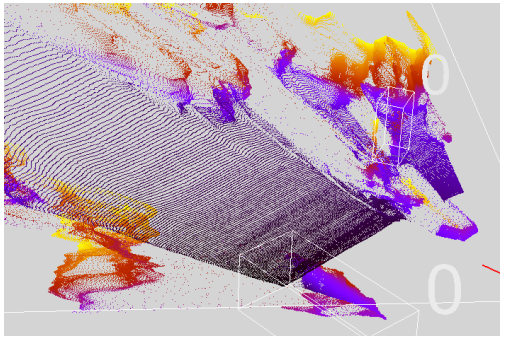
Left Image

Right Image

Pyramid Stereo Matching Network

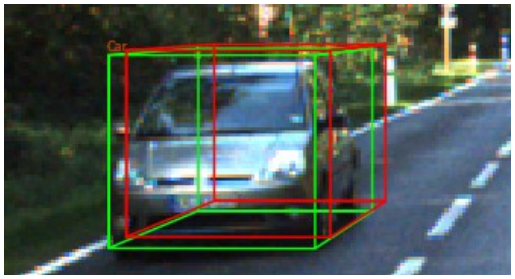
Disparity Map

3D Reconstruction



Stereo Point Cloud

Frustum Pointnet



3D Box Estimations*

** different image shown for convenience*

About: Disparity Maps

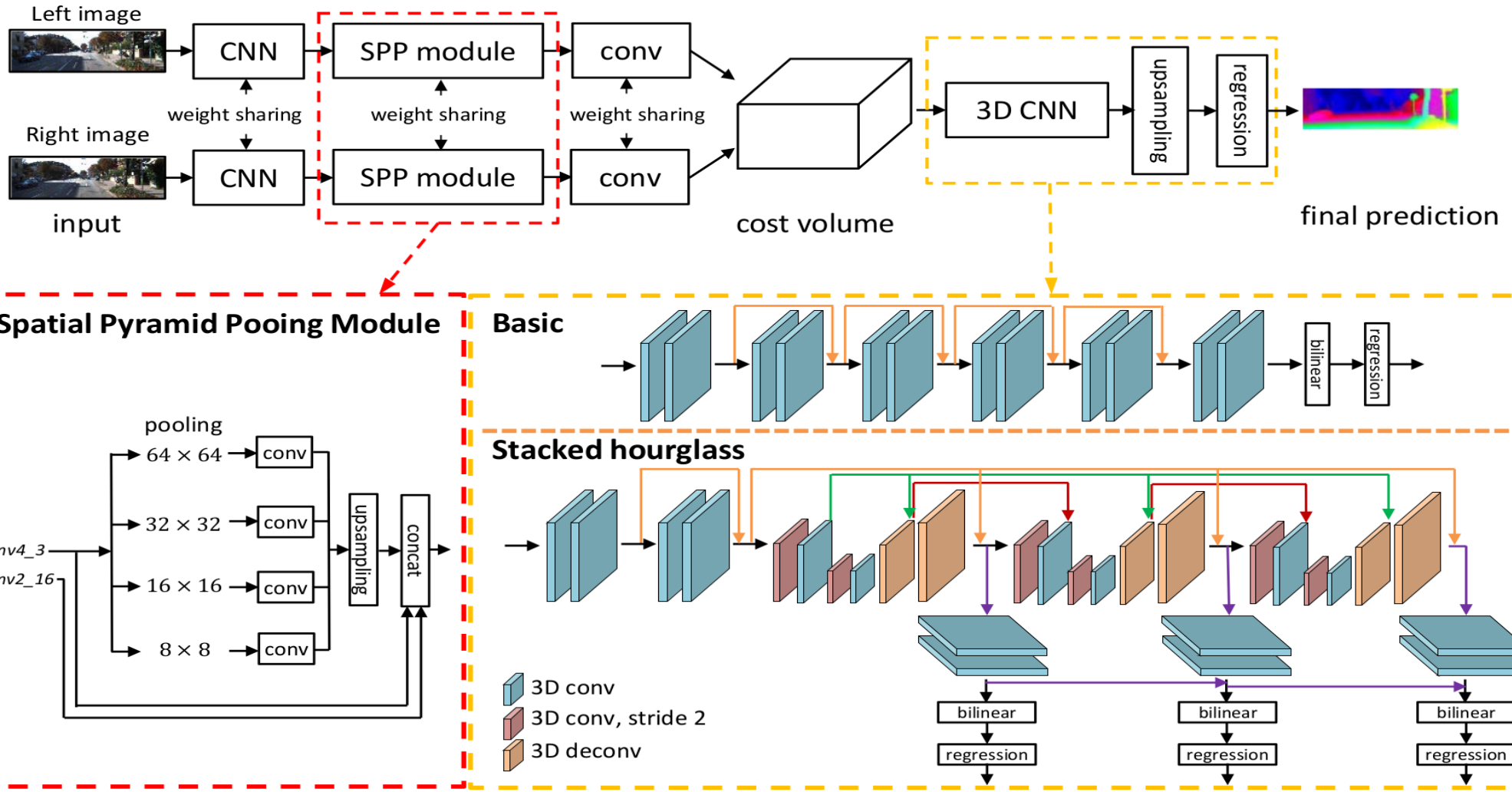
- Goal: pixel difference between matching items

SGBM disparity map

PSMnet disparity map

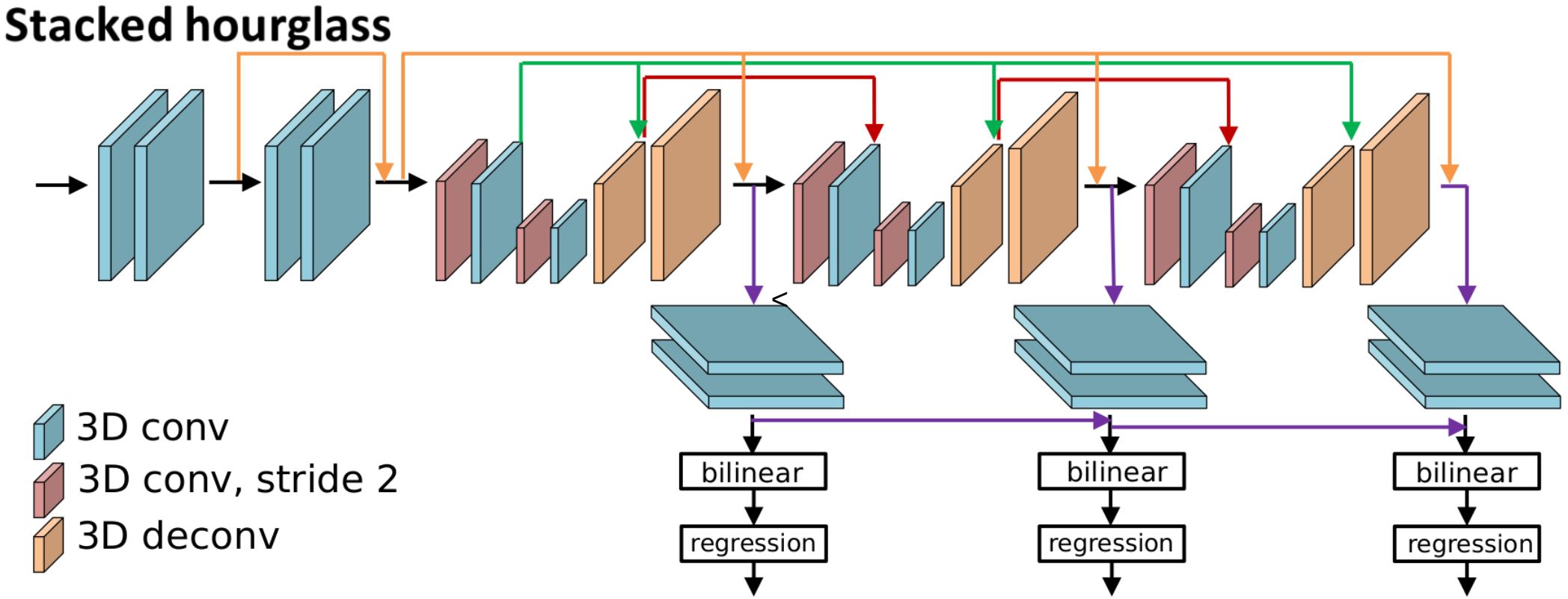
PSMnet: Stereo Vision (1/2)

- Training: Pretrained on Freiburg SceneFlow, finetuned on KITTI



PSMnet: Stacked Hourglass (2/2)

- Hourglass network: conv. + transposed conv. (deconv.)
- Skip layers: preserve info
- Supervision at three layers



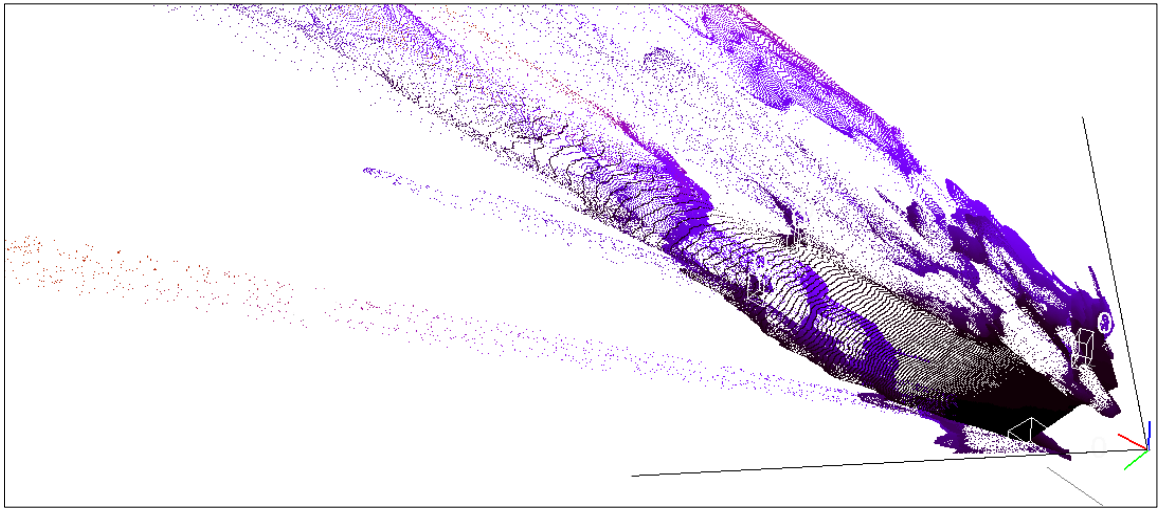
3D Reconstruction

Disparity Map

Depth Map

$$z = \frac{f_U b}{D}$$

Point Cloud

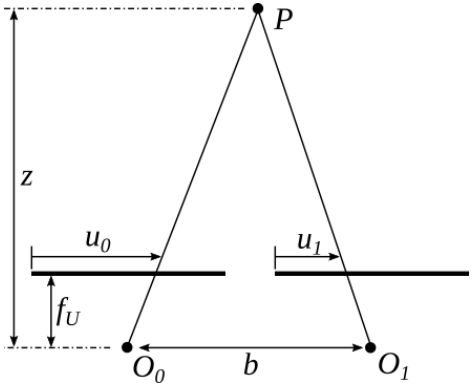


$$x = \frac{z(u - c_U)}{f_U}$$

$$y = \frac{z(v - c_V)}{f_V}$$

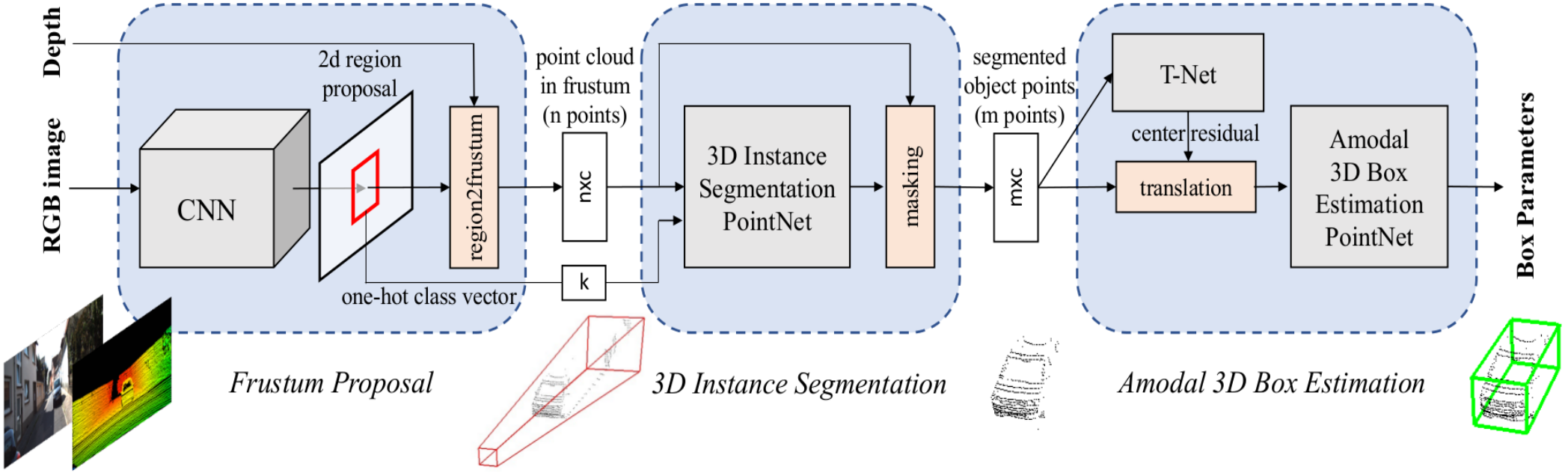
Assumptions:

1) Two cameras, identical in performance
2) Aligned camera centers
3) Images are rectified
4) Horizontal baseline between camera centers



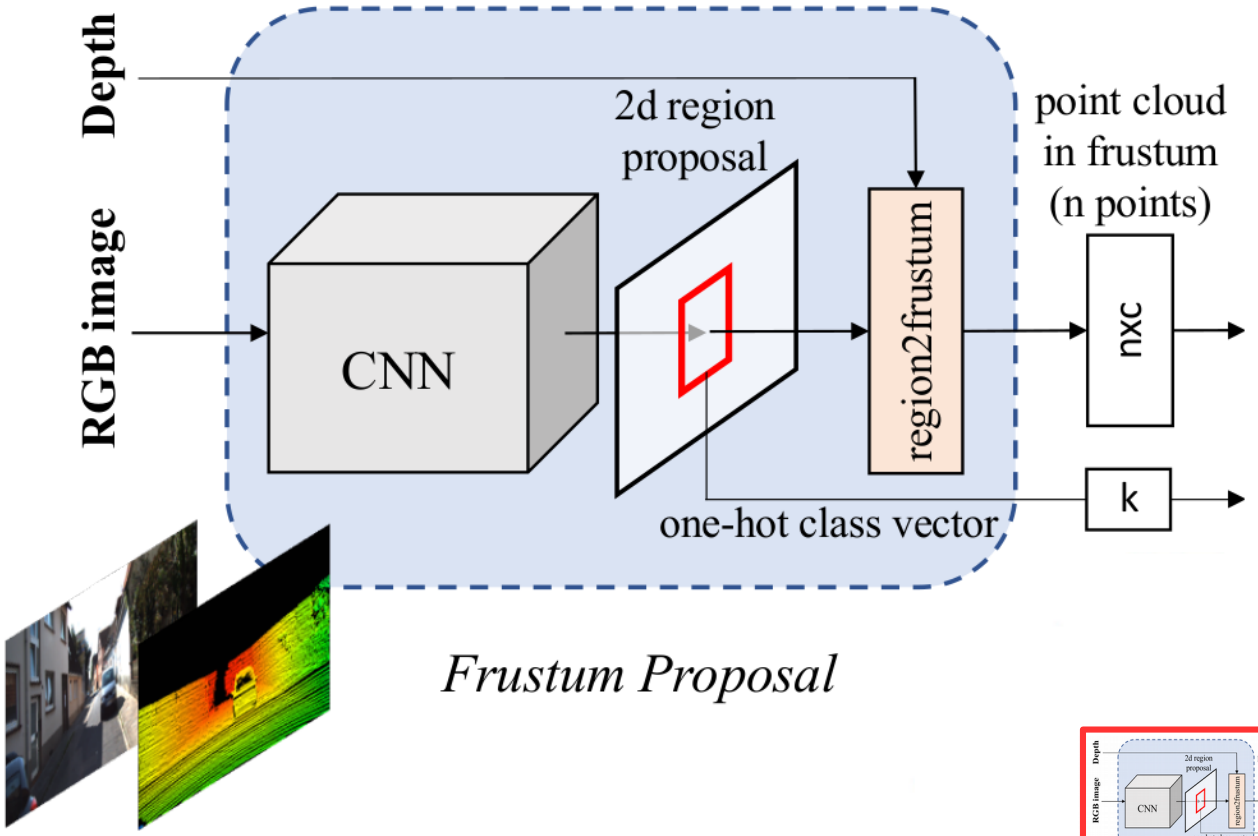
Frustum Pointnet: 3D Bboxes (1/4)

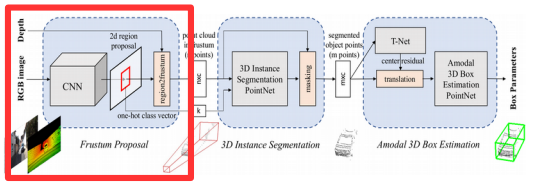
- Stereo training: preprocess both train and validation with PSMnet



Frustum Pointnet: Proposal (2/4)

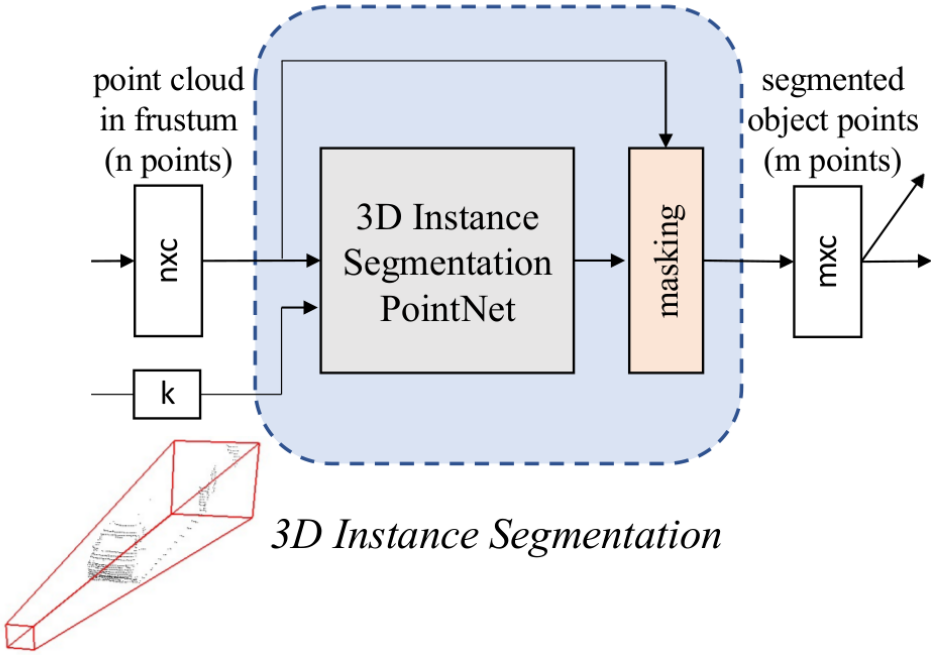
- Any 2D detector possible, default Feature Pyramid Network used

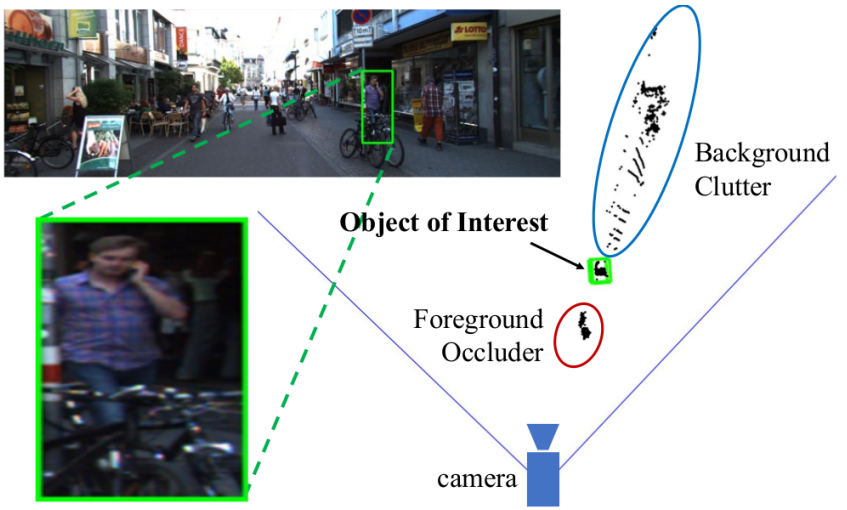


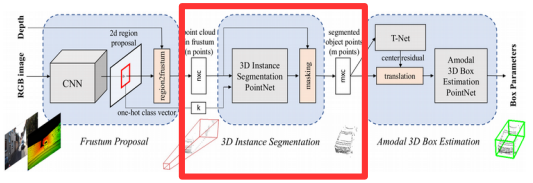


Frustum Pointnet: Segmentation (3/4)

- Pointnet: learn to segment the point cloud (supervised)

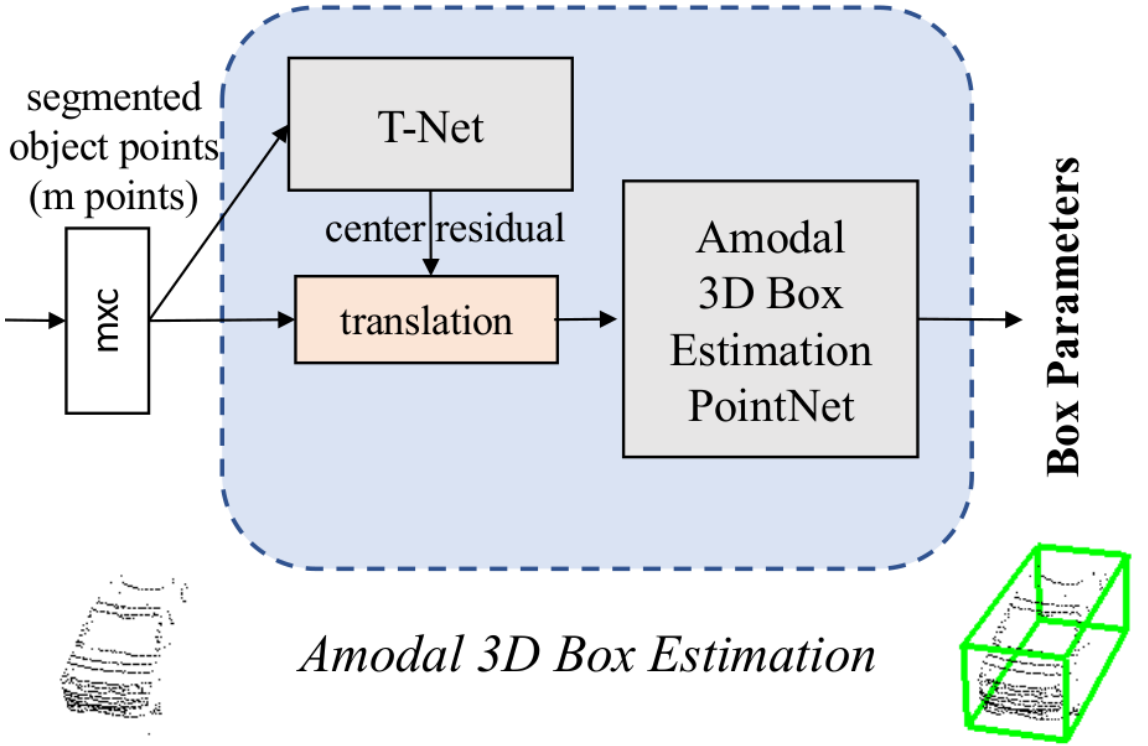


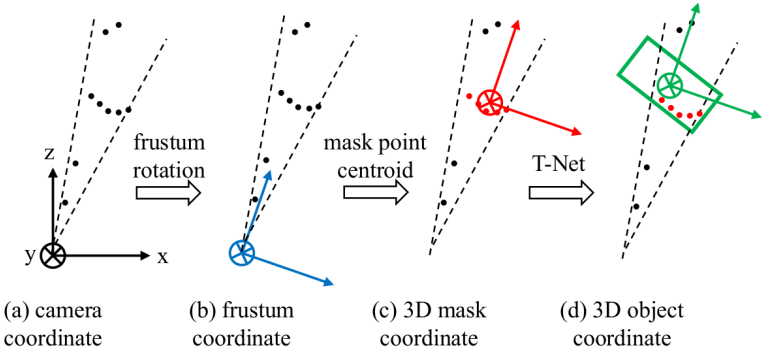




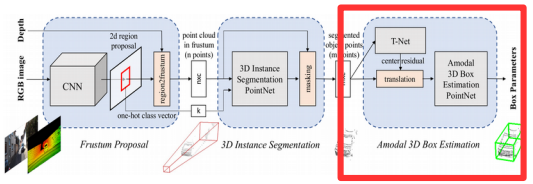
Frustum Pointnet: Estimation (4/4)

- T-Net: estimate “true center” (supervised)
- 2nd Pointnet: Estimate orientation (supervised)

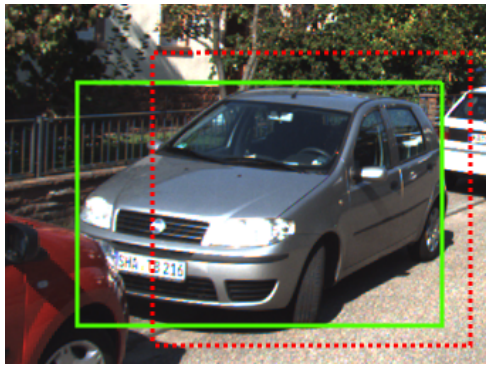


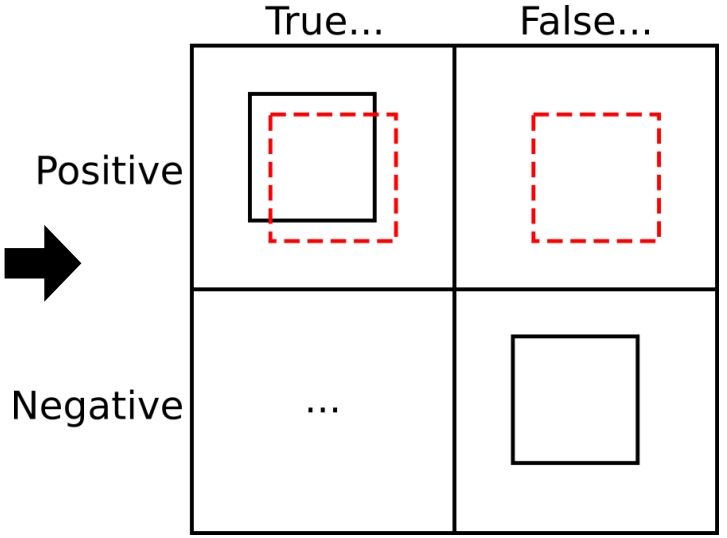


Amodal 3D Box Estimation



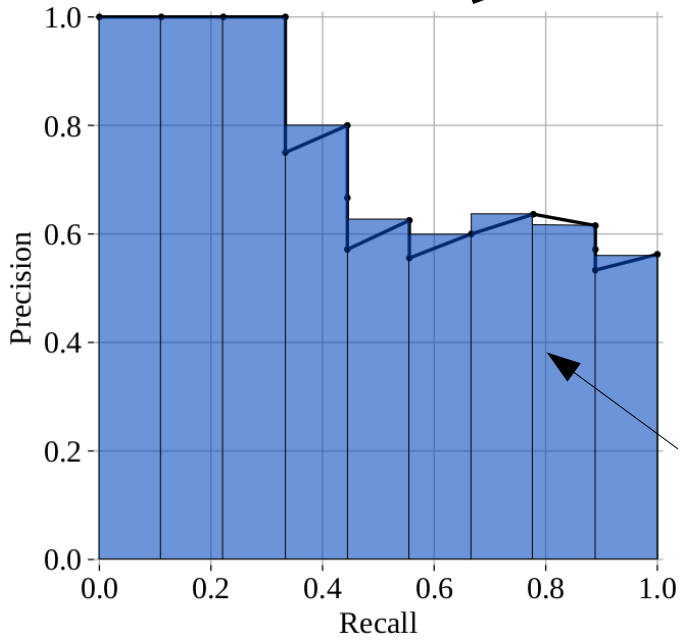
Performance Metrics





$$IOU = \frac{|A \cap B|}{|A \cup B|}$$

$$AP = \sum_{i=1}^{n-1} (re_{i+1} - re_i)(pr_{i+1})$$



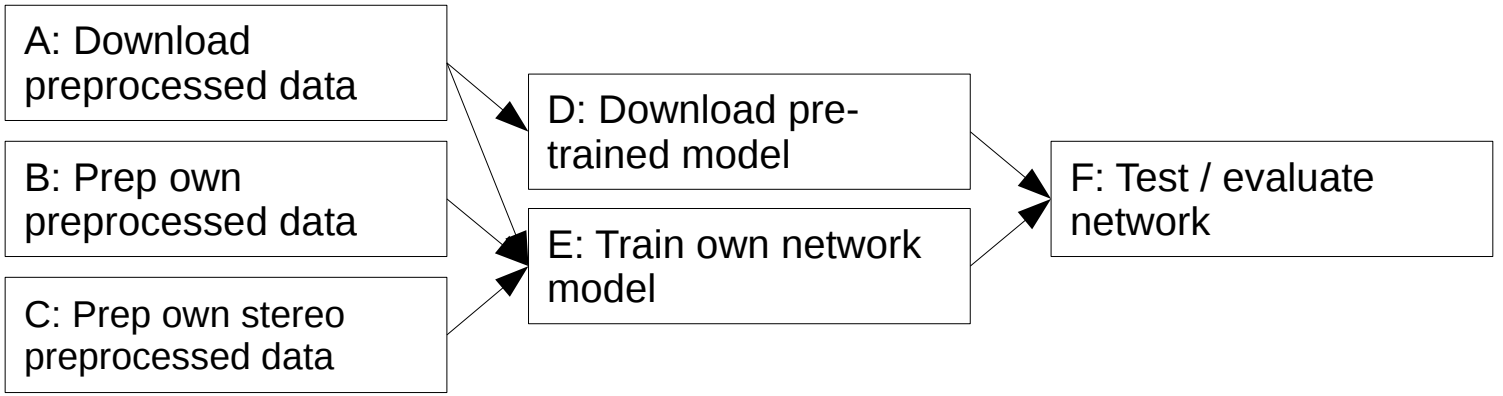
Typical Average Precision

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

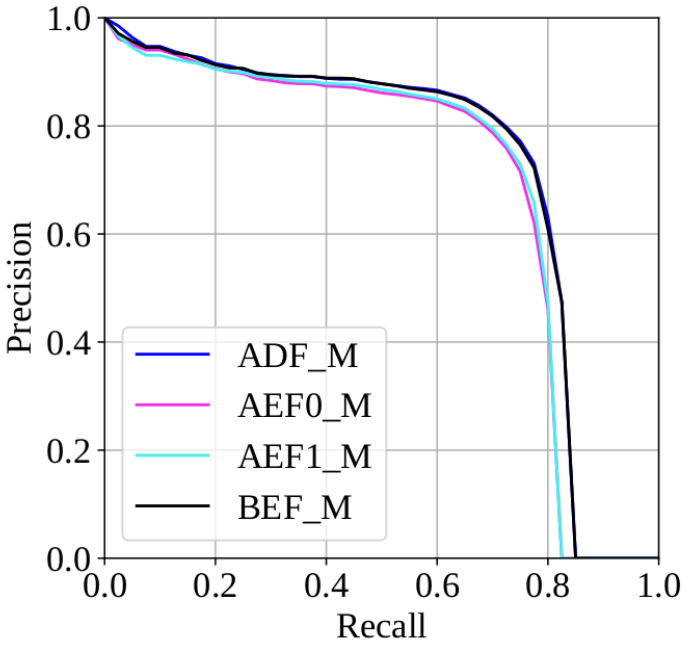
$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

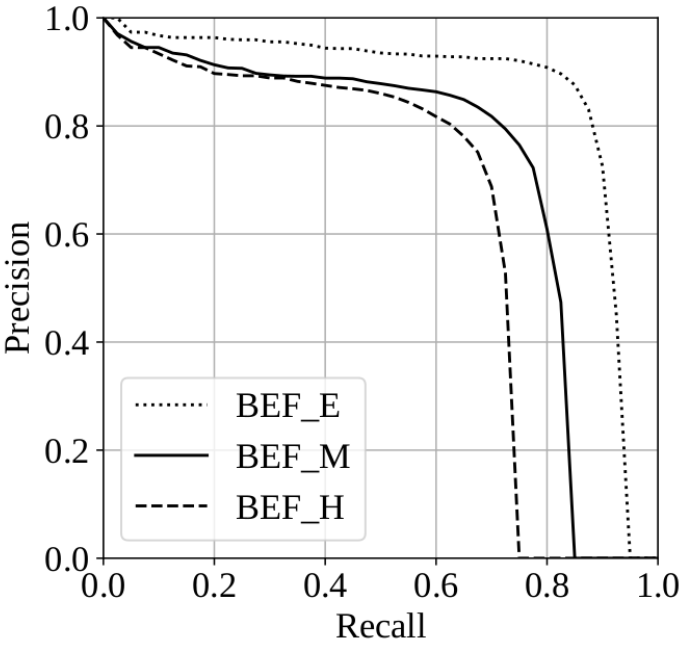
Experiment 1: Lidar Baseline

"Paths" for experimentation



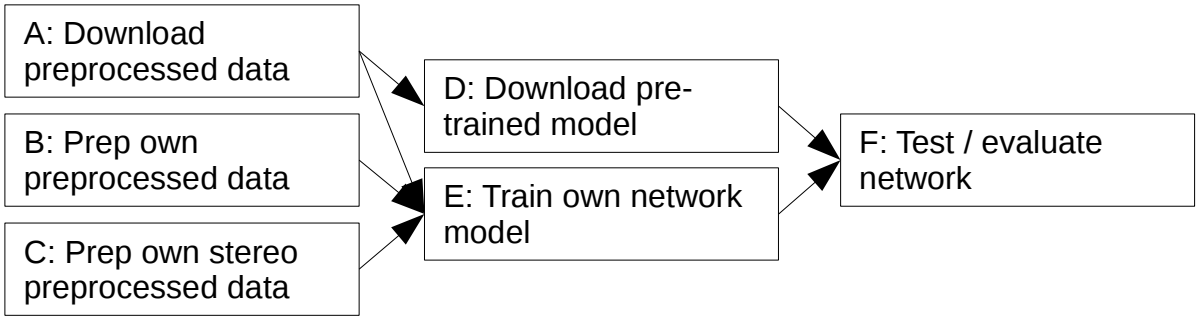
Path	AP_Easy	AP_Med	AP_Hard
ADF	84.07	71.28	63.35
AEF0	82.59	68.72	61.25
AEF1	83.51	68.99	61.04
BEF	84.12	71.00	63.27



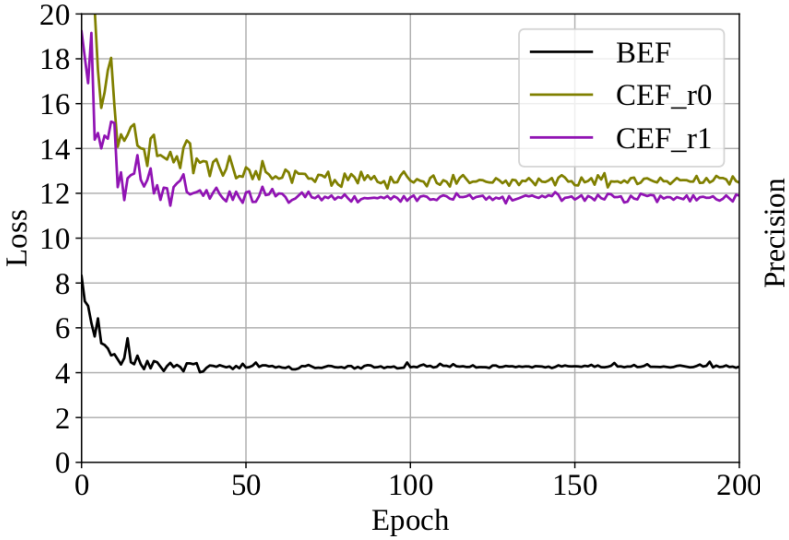


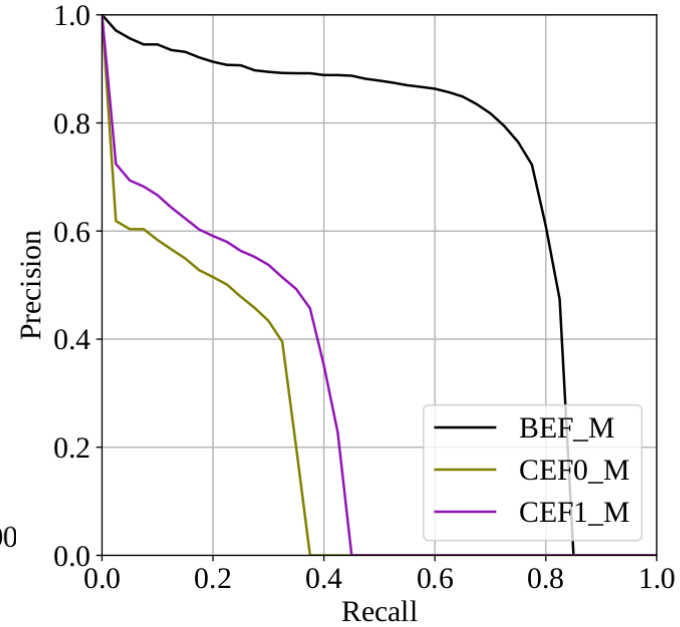
Experiment 2: Stereo-Point Cloud Network

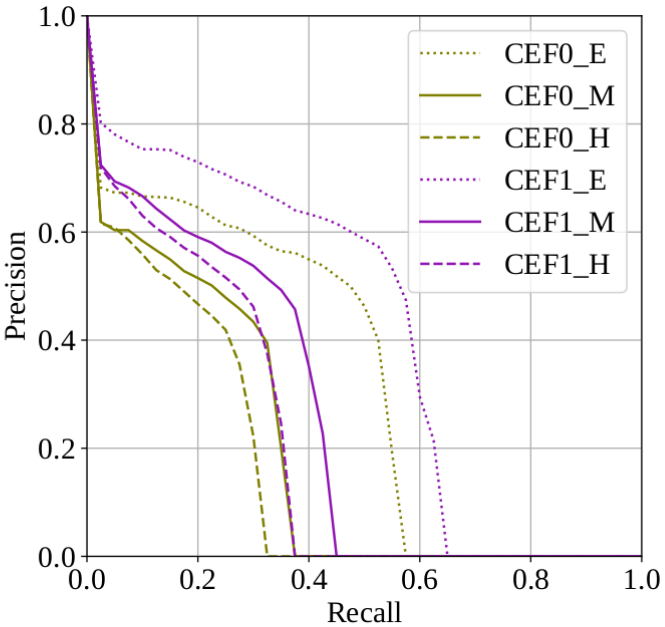
- CEF0: Initial attempt at stereo network
- CEF1: improvement on stereo alignment & reconstruction



Path	AP_Easy	AP_Med	AP_Hard
CEF0	35.61	23.02	20.43
CEF1	42.57	28.60	24.07



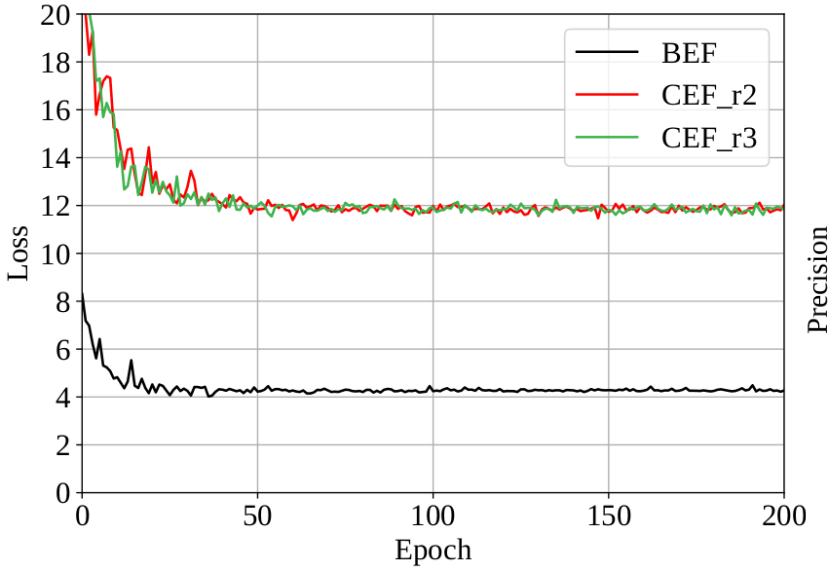


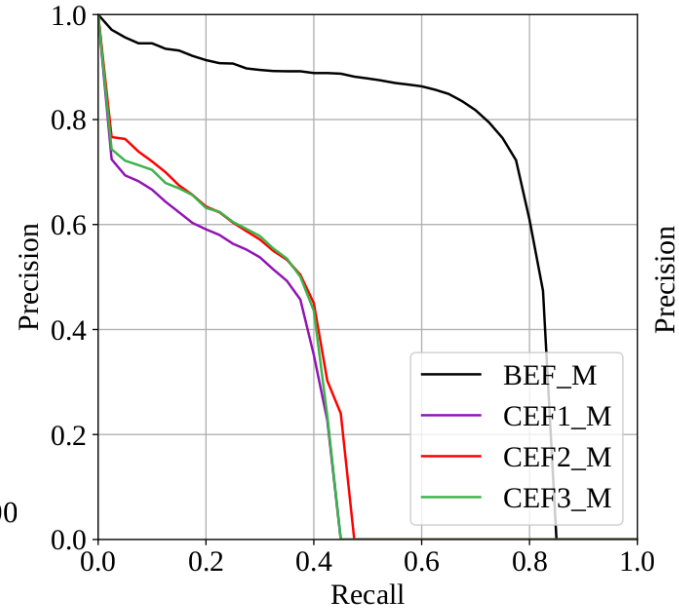


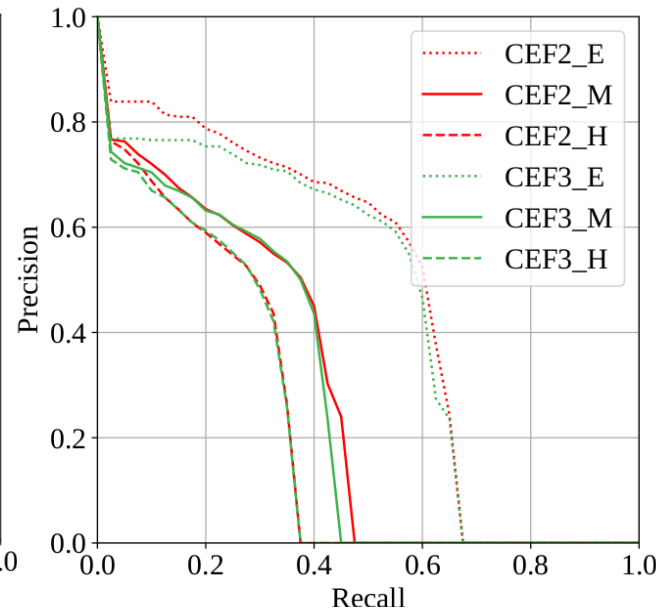
Experiment 3: SPCLnet v2

- CEF2: improvement after PSMnet retraining
- CEF3: reducing data (remove 1m above origin) not helpful

Path	AP_Easy	AP_Med	AP_Hard
CEF2	47.41	30.69	25.15
CEF3	45.43	30.45	24.29

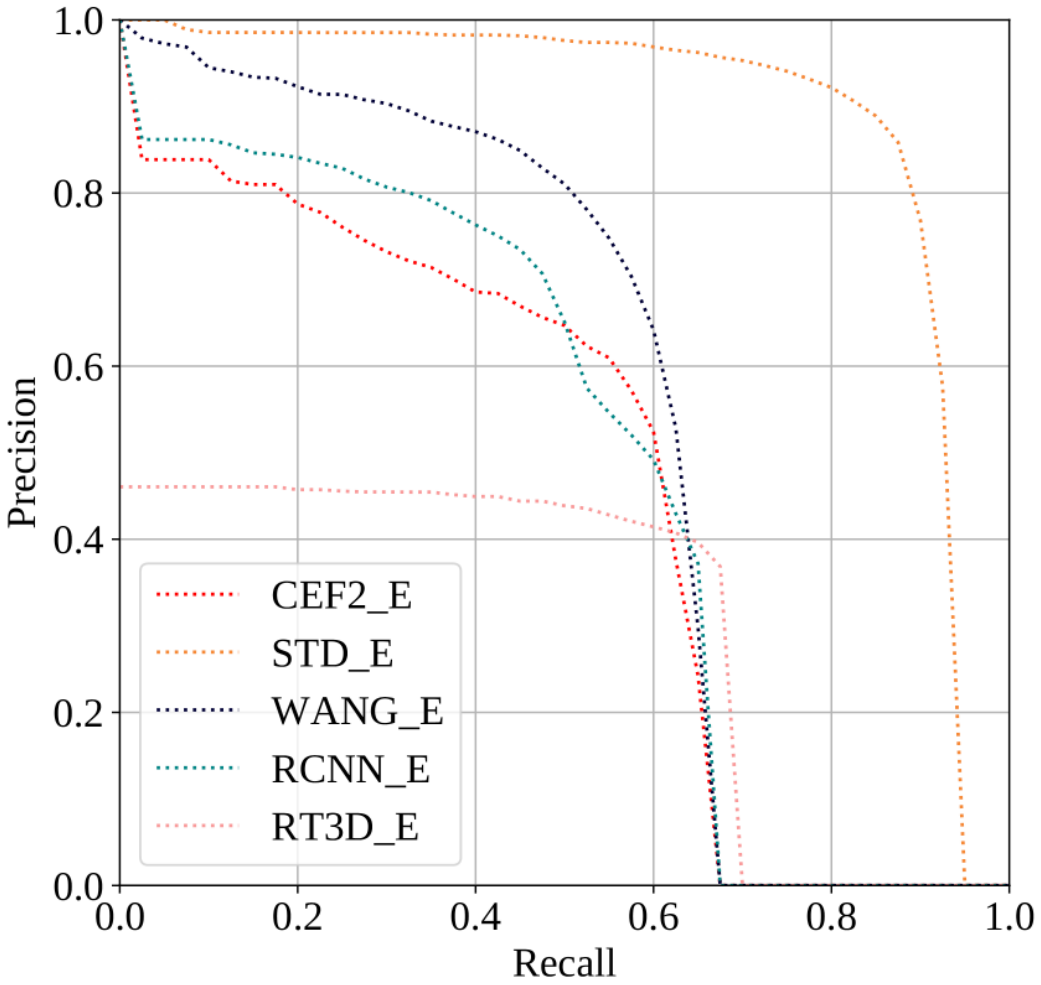


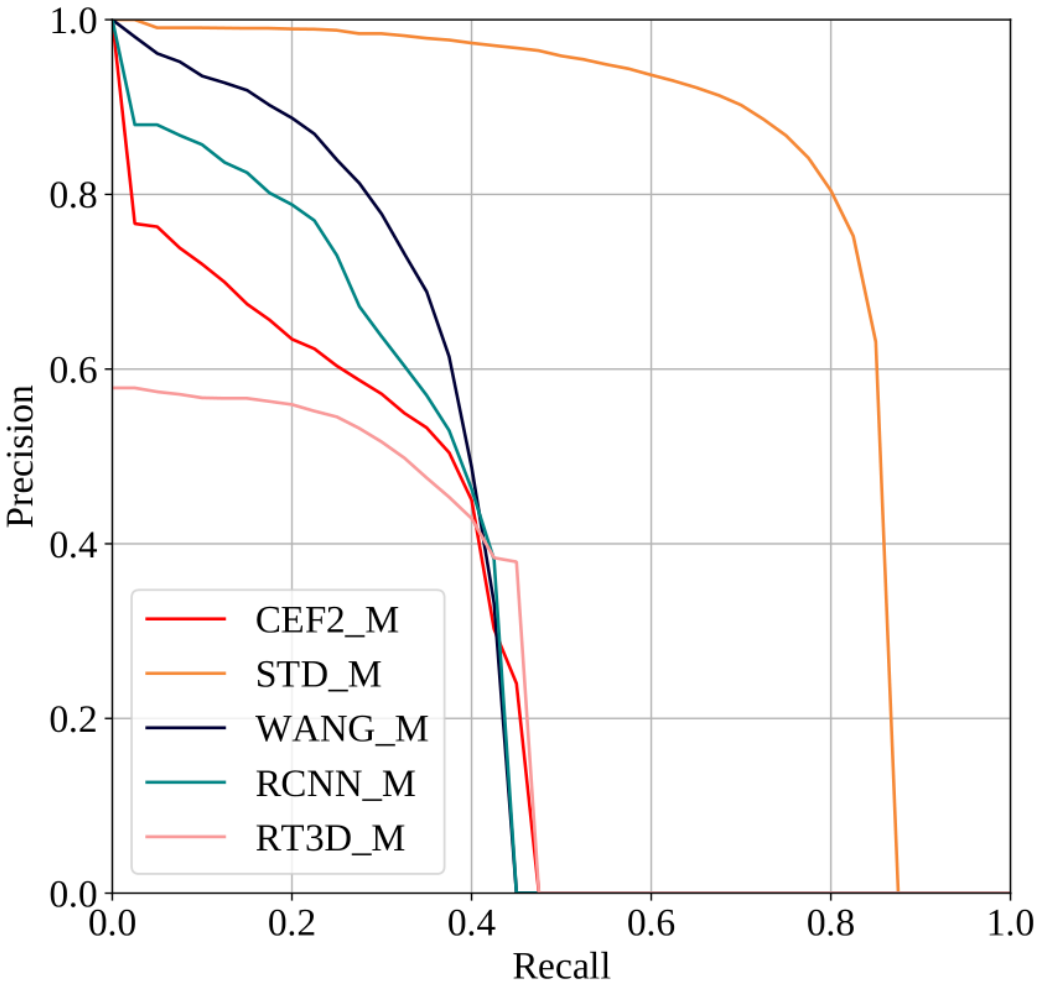




Comparison with Other Networks

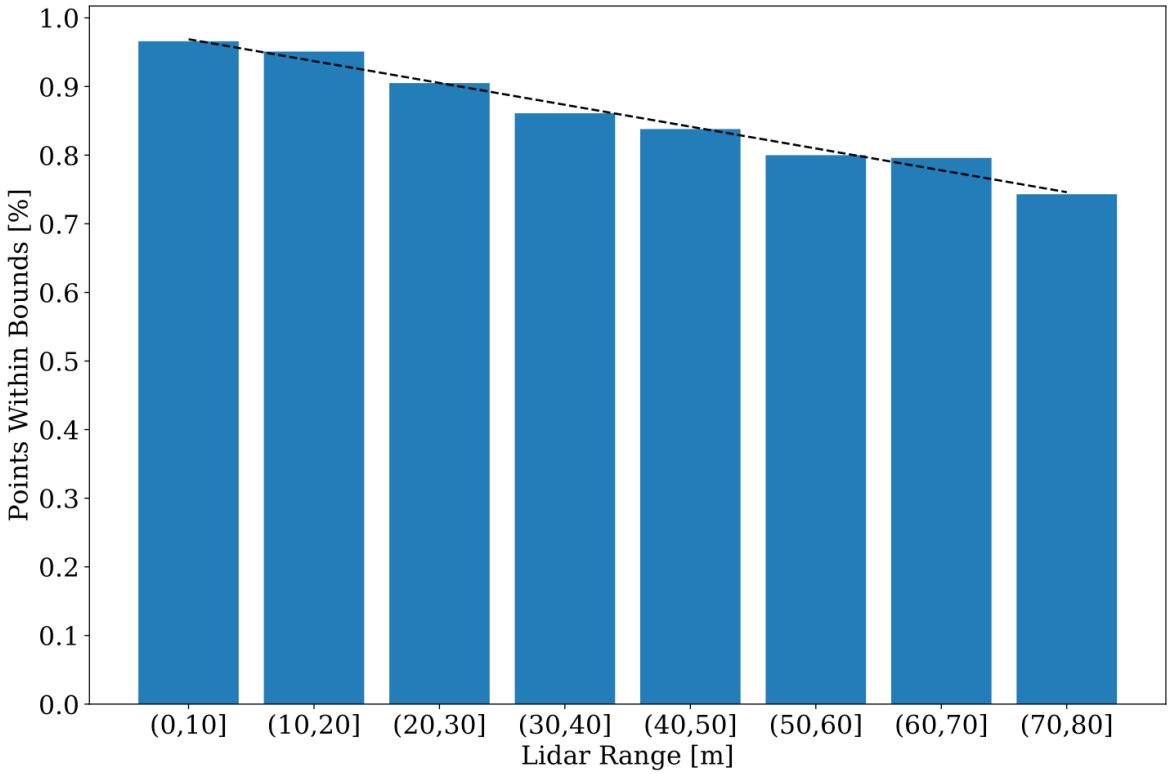
- CEF2: own network. STD: lidar “sparse2dense”. WANG: “Pseudo-Lidar”. RCNN: “Stereo R-CNN”. RT3D: “Real-Time 3D (stereo)”





Conclusions

- Stereo vision can potentially compete, at near range.
- Disparity map quality is key, and this is affected by the input data. The KITTI dataset may not always be the best choice.
- Real-time stereo is approaching reality, but not without simplification.



51x19 px, 24 meters away

Thank you!

Backup: Why point cloud representation?

- Natural question: why not use a depth map?

Wang et al, “Pseudo-Lidar”:

*… The central assumption [of 2D convolutions] is two fold: (a) local neighborhoods in the image have meaning … and (b) all neighborhoods can be operated upon in an **identical manner**…*

*… two pixels can be co-located next to each other in the depth map, yet can be very far away in 3D space. Second, objects that occur at multiple depths project to **different scales** in the depth map.*

Qi et al, “Frustum Pointnets”:

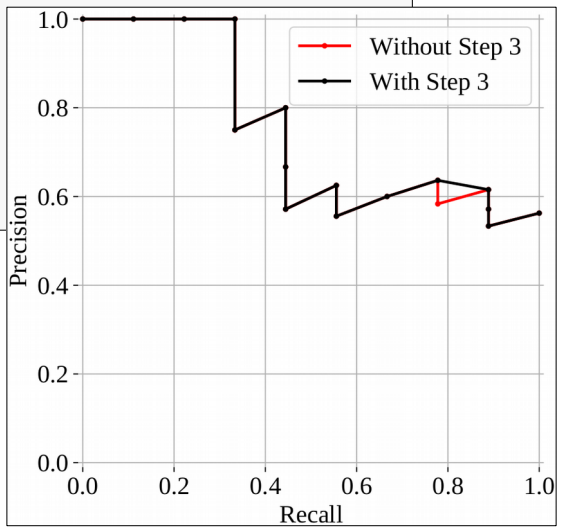
*One straightforward solution is to directly regress 3D object locations from a depth map using 2D CNN’s. However, … **occluding objects** and **background clutter** is common in natural scenes, which might severely distract the 3D localization task…*

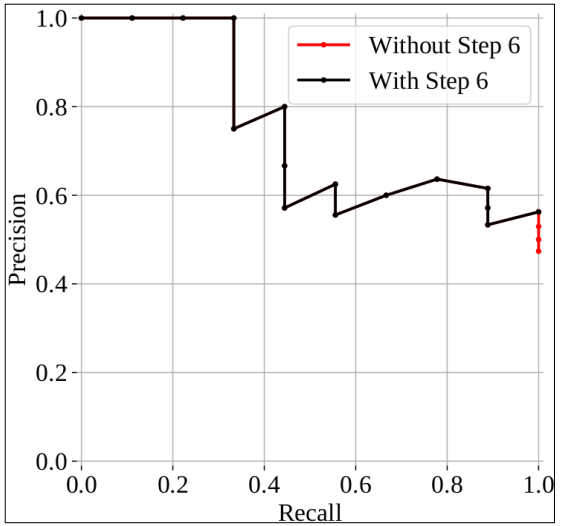
*[segmentation is easier in 3D than] in images where **pixels from distant objects can be near-by to each other**.*

Backup: Precision-Recall Curve

```python
# 1. obtain conf, outcomes vectors
# 2. sort in order of descending confidence
indices=np.argsort(conf)[::-1]
conf=conf[indices]
outcomes=outcomes[indices]

# 3. optional get indices of unique values plus final index location
distinct_value_indices = np.where(np.diff(conf))[0]
threshold_idxs = np.append(distinct_value_indices,outcomes.size-1)

# 4. generate aggregate tp/fp vectors
tps = np.cumsum(outcomes)[threshold_idxs]
fps = 1+threshold_idxs - tps

# 5. calculate precision / recall
precision = tps/(tps+fps)
precision[np.isnan(precision)]=0
recall = tps/tps[-1]

# 6. remove indices after tps stops increasing, and add a starting point
lastind=tps.searchsorted(tps[-1])+1
recall = np.append(0,recall[:lastind])
precision = np.append(1,precision[:lastind])

#7. optional: sort prec/rec in DECREASING recall
recall=recall[::-1]
precision=precision[::-1]
```

Outcomes	Confidence
1	1.000
1	0.994
1	0.981
0	0.932
1	0.919
0	0.879
0	0.854
1	0.846
0	0.768
1	0.763
1	0.752
0	0.728
1	0.728
0	0.706
0	0.653
1	0.653
0	0.502
0	0.309
0	0.298
0	0.223





Backup: Stereo & ObjDet Similarity

Stereo (above) and Object Detection (below) dataset sample images.

Backup: Other Datasets

- Difficult to find other datasets with both stereo and lidar
- Others examined:
  - Apolloscape: growing, based off KITTI dataset
  - Cityscapes: oriented towards segmentation, but high quality images
  - Berkeley DeepDrive: oriented towards lidar usage, but also has high quality images

Backup: FPnet loss

4.4. Training with Multi-task Losses

We simultaneously optimize the three nets involved (3D instance segmentation PointNet, T-Net and amodal box estimation PointNet) with multi-task losses (as in Eq. 2). L_{c1-reg} is for T-Net and L_{c2-reg} is for center regression of box estimation net. L_{h-cls} and L_{h-reg} are losses for heading angle prediction while L_{s-cls} and L_{s-reg} are for box size. Softmax is used for all classification tasks and smooth-l_1 (huber) loss is used for all regression cases.

$$\begin{aligned} L_{multi-task} = & L_{seg} + \lambda(L_{c1-reg} + L_{c2-reg} + L_{h-cls} + \\ & L_{h-reg} + L_{s-cls} + L_{s-reg} + \gamma L_{corner}) \end{aligned} \tag{2}$$

$$L_{corner} = \sum_{i=1}^{NS} \sum_{j=1}^{NH} \delta_{ij} \min\{\sum_{k=1}^{8} \|P_k^{ij} - P_k^*\|, \sum_{i=1}^{8} \|P_k^{ij} - P_k^{**}\|\}$$

References

[1] Yiming Zeng, Yu Hu, Shice Liu, Jing Ye, Yinhe Han, Xiaowei Li, and Ninghui Sun. Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving. *IEEE Robotics and Automation Letters*, 3(4):3434–3440, 2018.

[2] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. *arXiv preprint arXiv:1907.10471*, 2019.

[3] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Weinberger. Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3d Object Detection for Autonomous Driving. In CVPR, 2019.

[4] Richard Szeliski. Computer Vision: Algorithms and Applications. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[5] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3d Object Detection from RGB-D Data. CoRR, abs/1711.08488, 2017.

[6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125, 2017.

[7] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo R-CNN based 3d Object Detection for Autonomous Driving. arXiv preprint arXiv:1902.09738, 2019.

[8] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The ApolloScape Dataset for Autonomous Driving. ArXiv: 1803.06184, 2018.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[10] Jia-Ren Chang and Yong-Sheng Chen. Pyramid Stereo Matching Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5410–5418, 2018.

Xx order these by appearance