2021 embedded vision summit
VIRTUAL | MAY 25-28

Efficient Deep Learning for 3D Point Cloud

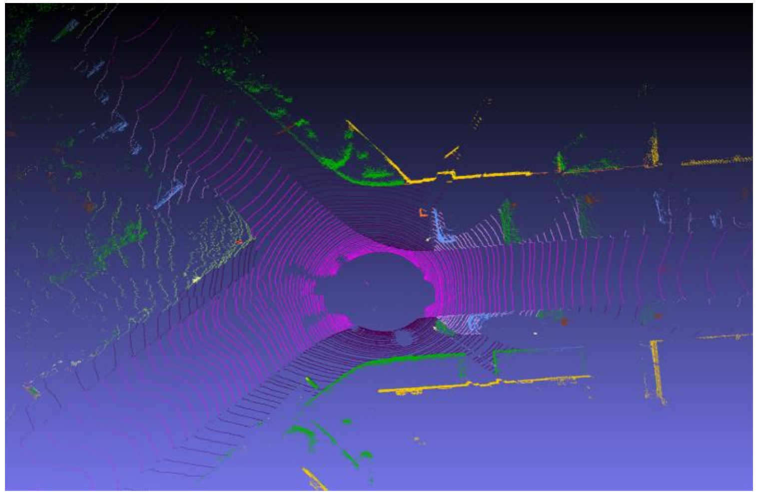
Dr. Bichen Wu
Research Scientist
Facebook Reality Labs

Outline

- Background
- Review of challenges
  - Modeling challenge
  - Data challenge
- Tackling the modeling challenge
- Tackling the data challenge
- Summary

Background

Applications Powered by 3D Point Cloud



Autonomous driving

Robotics

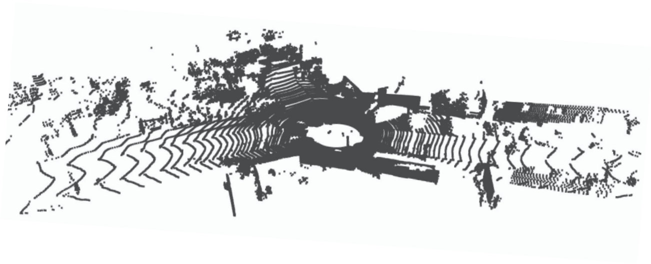
AR/VR

Adapted from:
https://www.flickr.com/photos/94549193@N00/4519088620

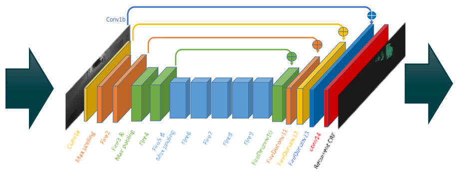
Point Cloud Understanding

- What does point cloud understanding mean?
  - Point-cloud understanding through semantic segmentation

Efficient Deep Learning for Point Cloud Understanding



LiDAR point cloud



Deep Neural Net

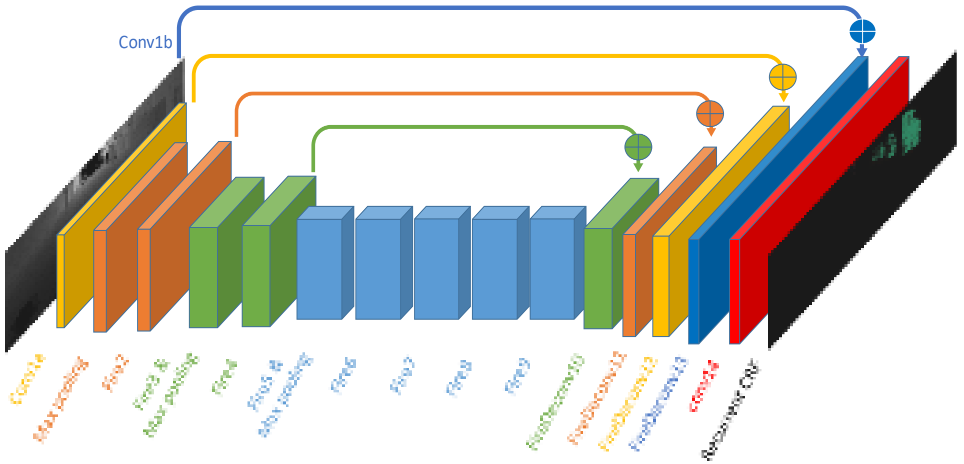
Point-wise object labels (car, person, etc.)

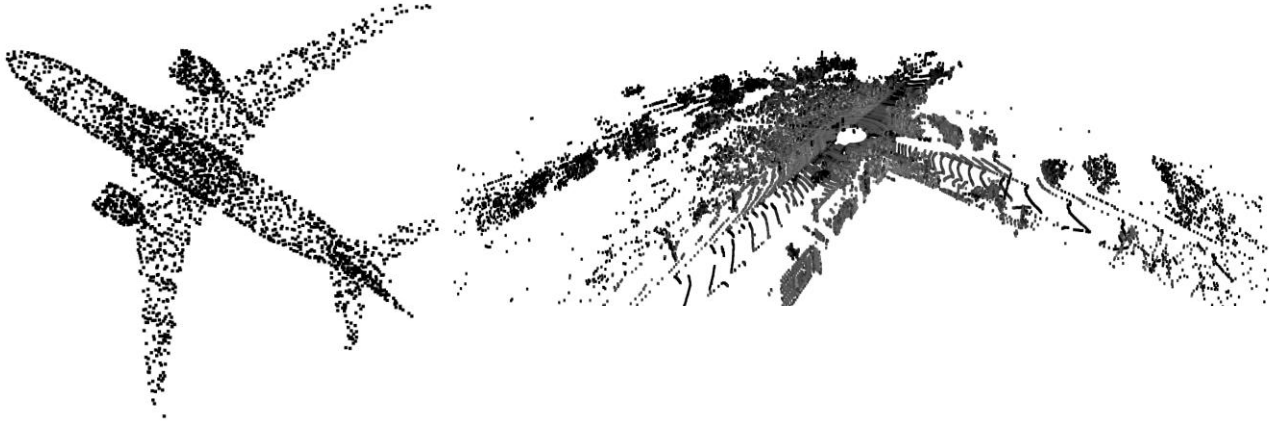
- Key metrics:
  - Accuracy: essential for applications such as autonomous driving, AR/VR, etc.
  - Efficiency: Real-time speed, low energy on embedded processors

Challenges

Modeling Challenges

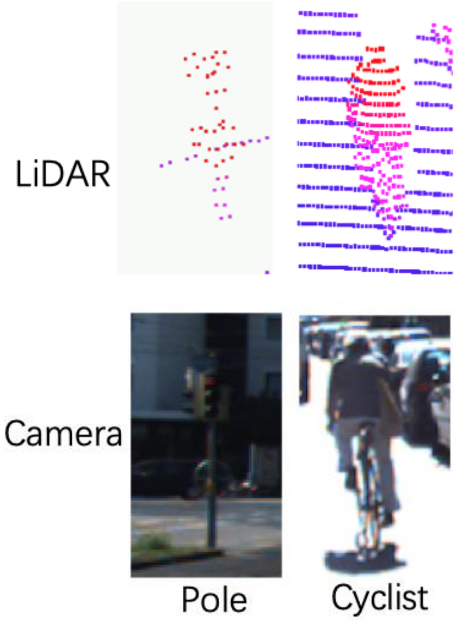
- A point cloud consists a set of points
  - Sparse
  - Irregularly distributed in the 3D space
  - Unordered
- While ConvNets are great for images, they are not suitable for point clouds.
- **What kind of neural network models can process 3D point cloud?**



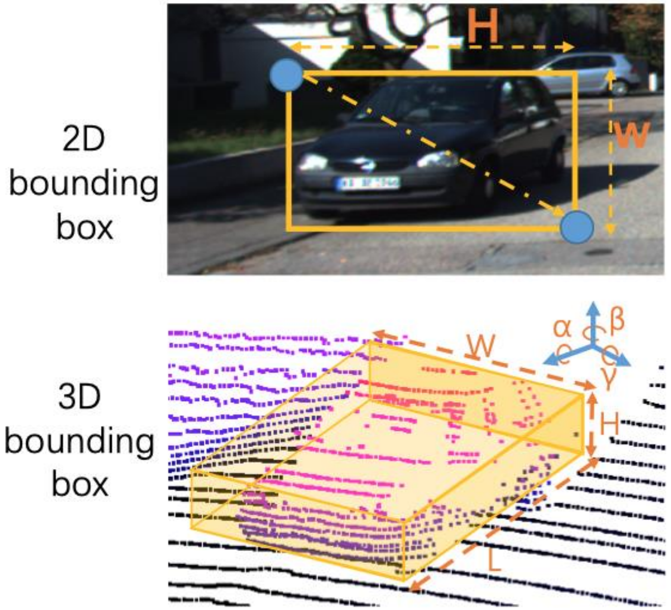


Data Challenges

- Deep learning requires a large amount of data, but annotating point cloud is challenging
  - Low resolution: Point-cloud sensors (such as LiDAR) have much lower resolution
  - Complex annotation operation: annotating objects in point cloud is harder than in images



(a) Low resolution



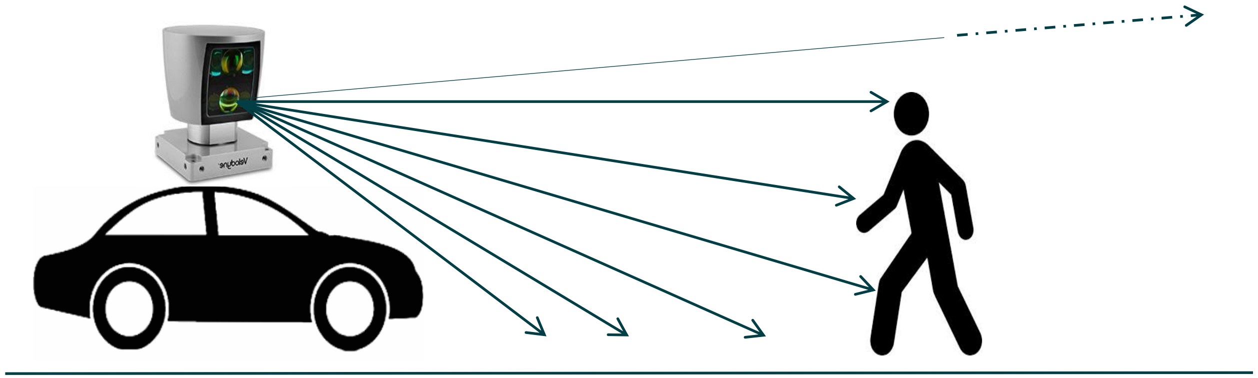
(b) Complex annotating operations

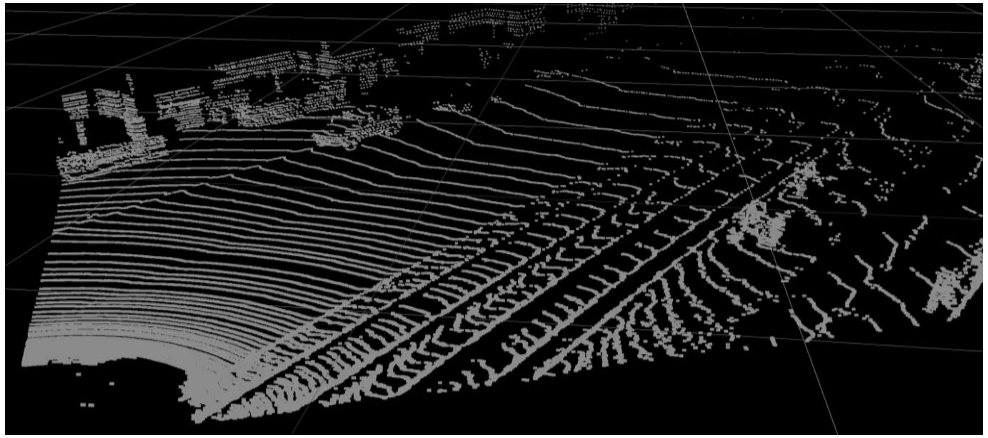
Tackling the modeling challenge

Projection-based methods

LiDAR Point Cloud

- LiDAR (**Li**ght **D**etection **A**nd **R**anging) is an important sensor for autonomous driving
- An example: a Velodyne-64 LiDAR
  - Emitting lasers, and measure distances through time-of-flight
  - Emitting 64 rays per pulse, 2000 pulses per rotation, and 10 rounds per second

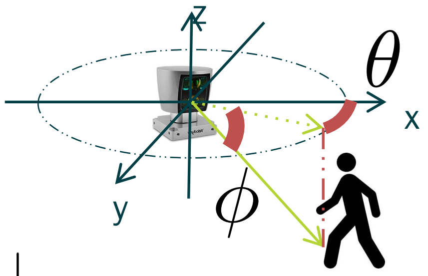




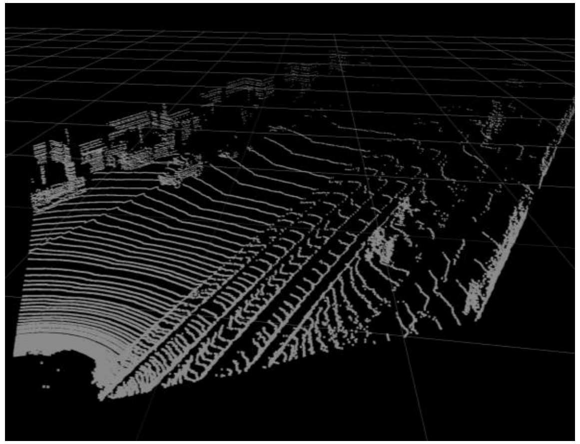
Projecting a 3D point cloud to a 2D sphere

$$i = \left\lfloor \frac{\arcsin\left(\frac{y}{\sqrt{x^2+y^2}}\right)}{\delta\theta} \right\rfloor$$

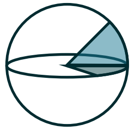
$$j = \left\lfloor \frac{\arcsin\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right)}{\delta\phi} \right\rfloor$$



i, j – "pixel" coordinates



Spherical projection



RGB

Intensity

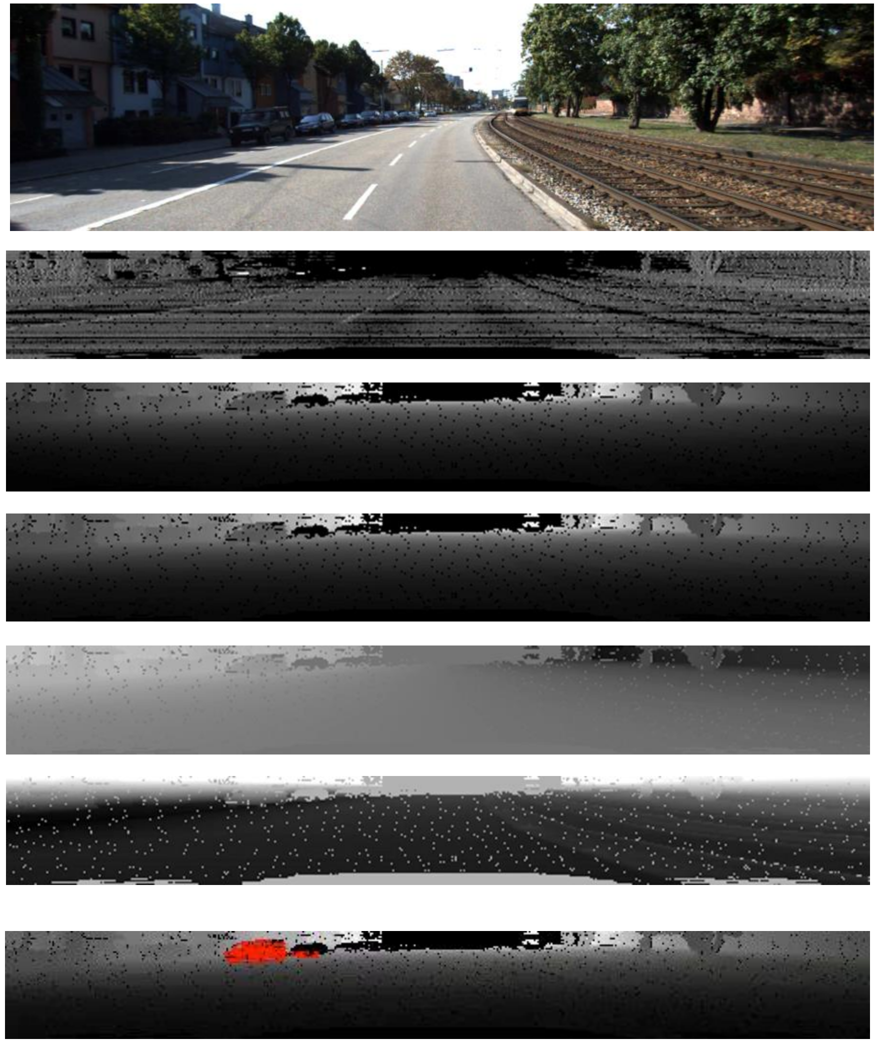
$\sqrt{x^2+y^2+z^2}$

x

y

z

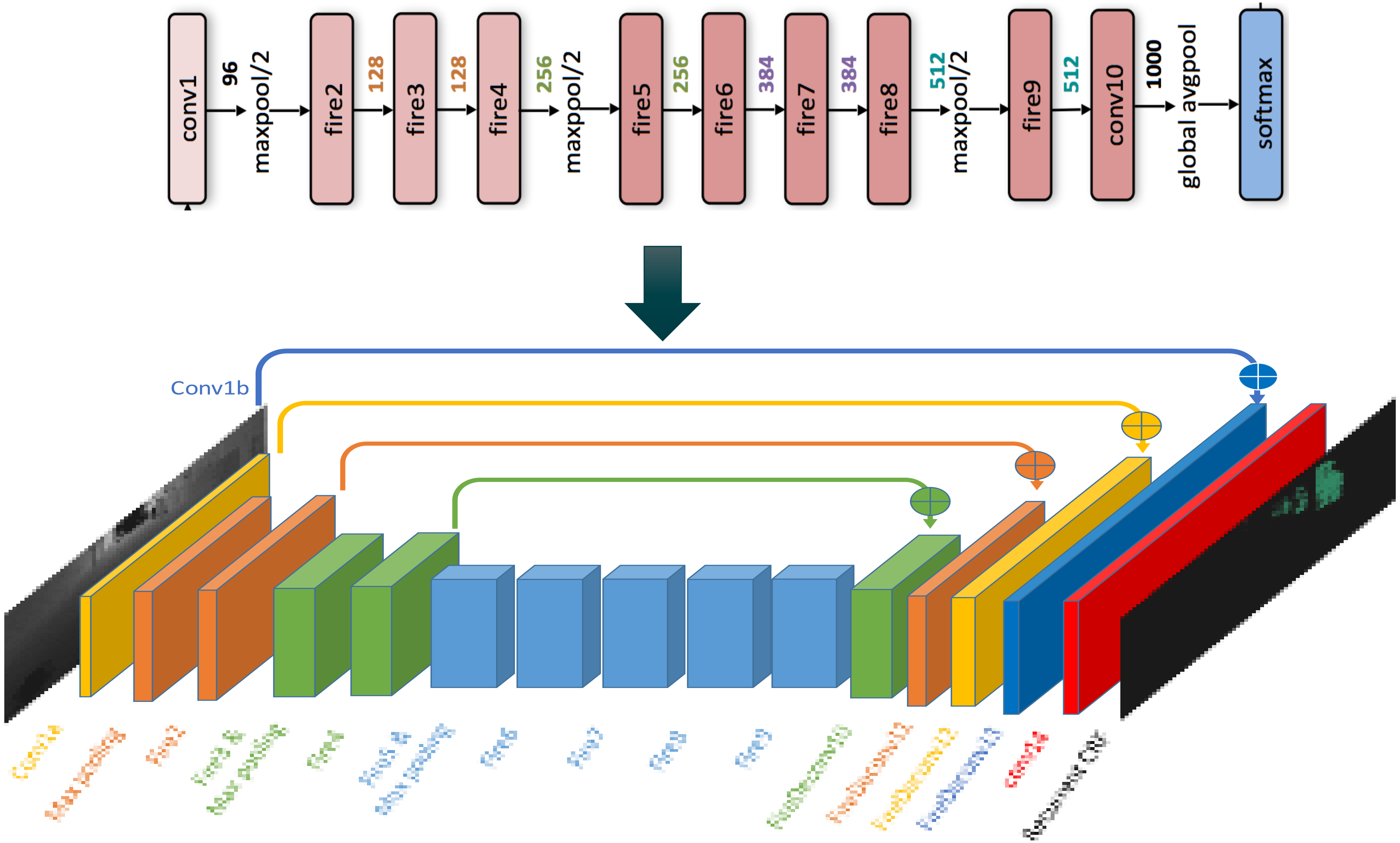
Label



SqueezeSeg: a 2D ConvNet for 3D Point-cloud

2021 embedded VISION summit

- Processing projected point cloud as 2D images
- Use an efficient 2D ConvNet (SqueezeNet) to predict point-wise labels
- Extremely fast:
  - >100 FPS on desktop GPU
  - >25 FPS on an embedded GPU

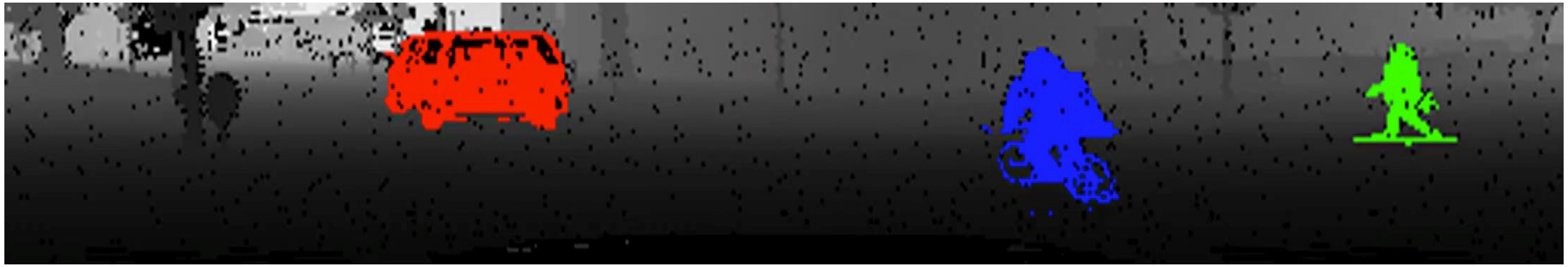


FACEBOOK REALITY LABS

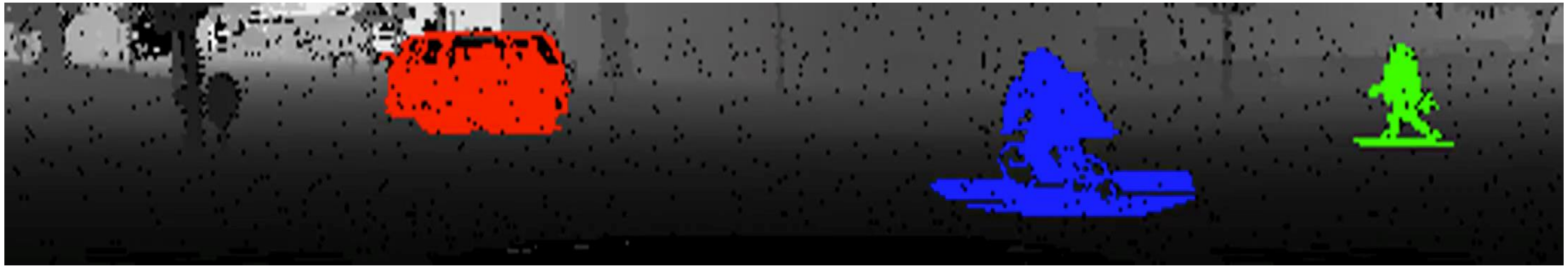
Result Visualization

Video reference

Ground truth label map



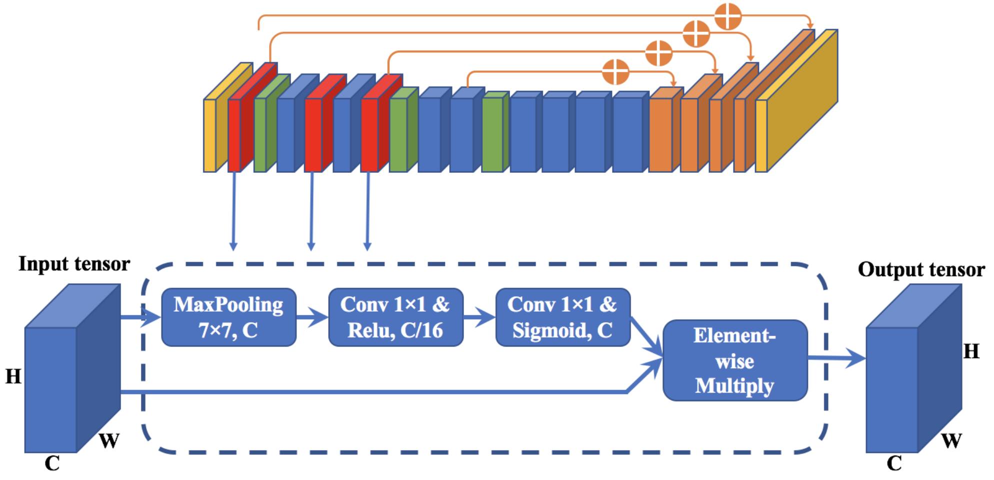
Predicted label map



FACEBOOK REALITY LABS

SqueezeSegV2 & V3

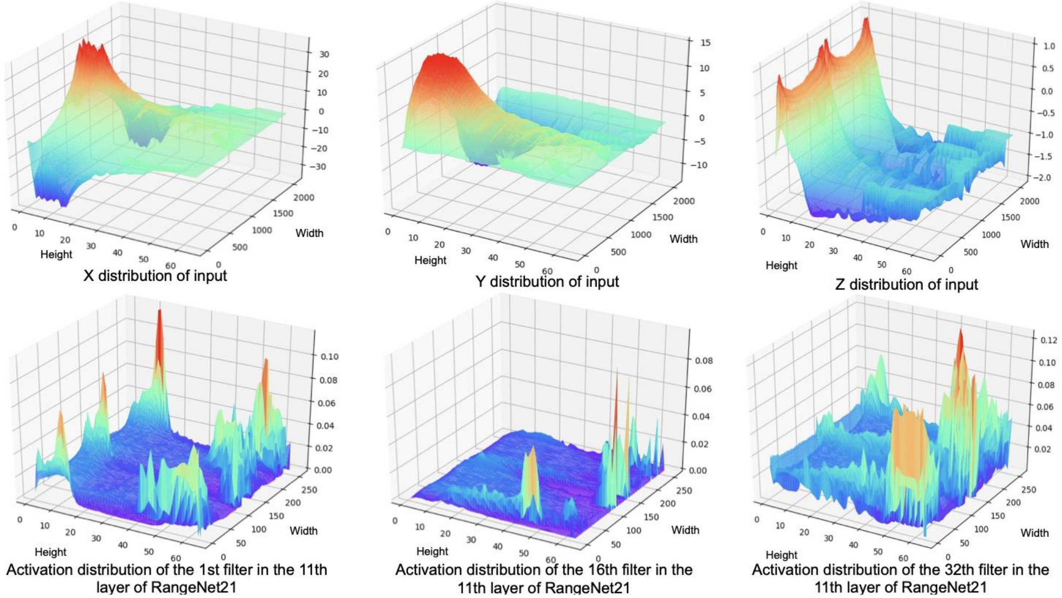
- SqueezeSegV2:
  - Context Aggregation Module for better dealing with dropout noise



Context Aggregation Module

Bichen Wu, et al. ICRA19

- SqueezeSegV3:
  - Spatially-Adaptive Convolution to deal with spatial variance in projected point clouds

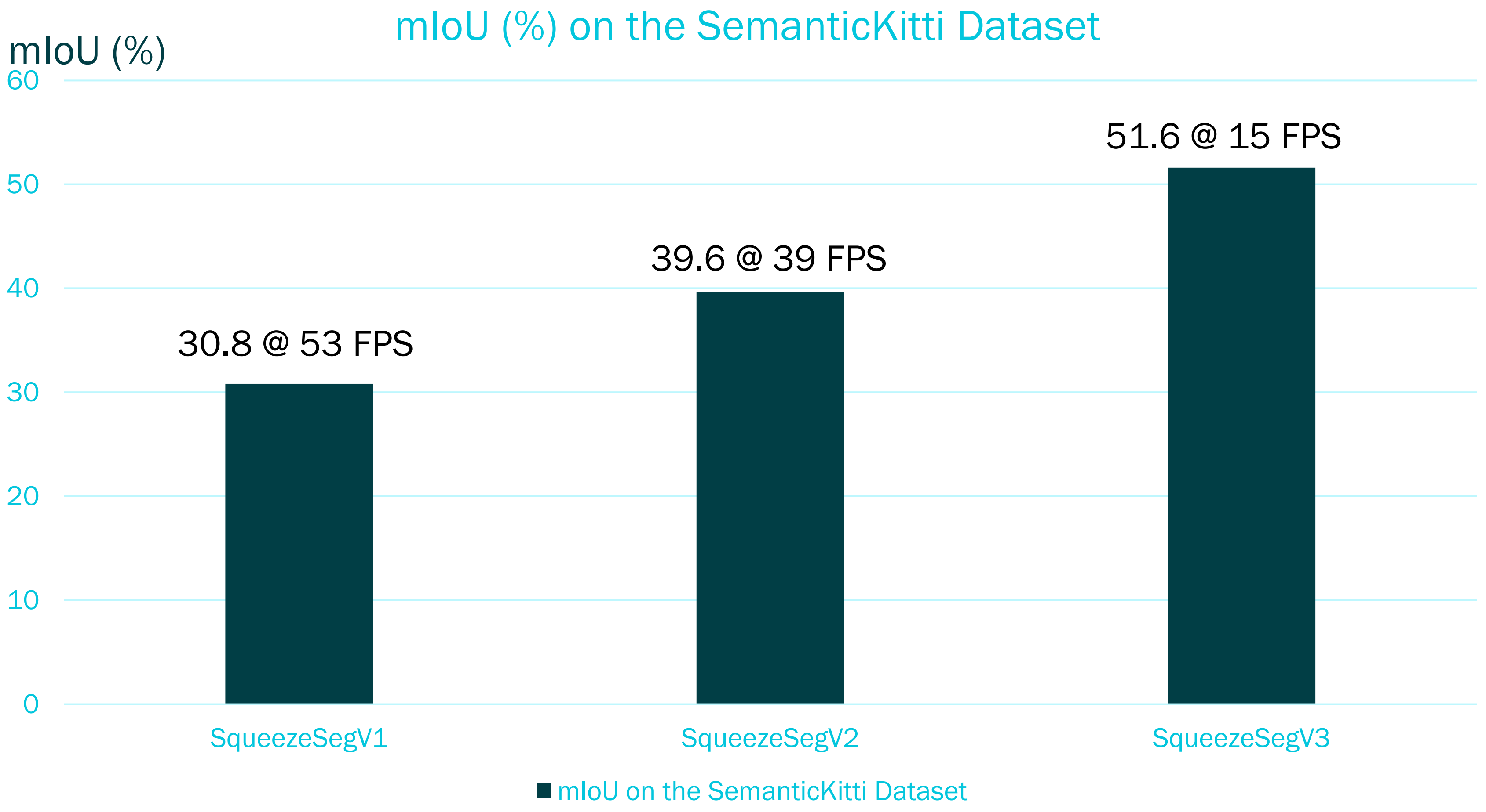


Chenfeng Xu, et al. ECCV20

Results

From SqueezeSegV1 to SqueezeSegV3:

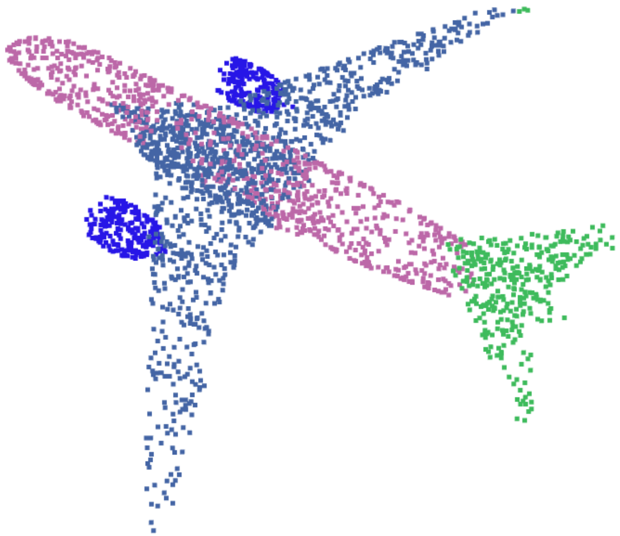
- +20.8 pts accuracy
- Slower inference speed, but still faster than real-time (15 FPS)
  - Measured on Nvidia Titan X GPU, w/o speed optimization



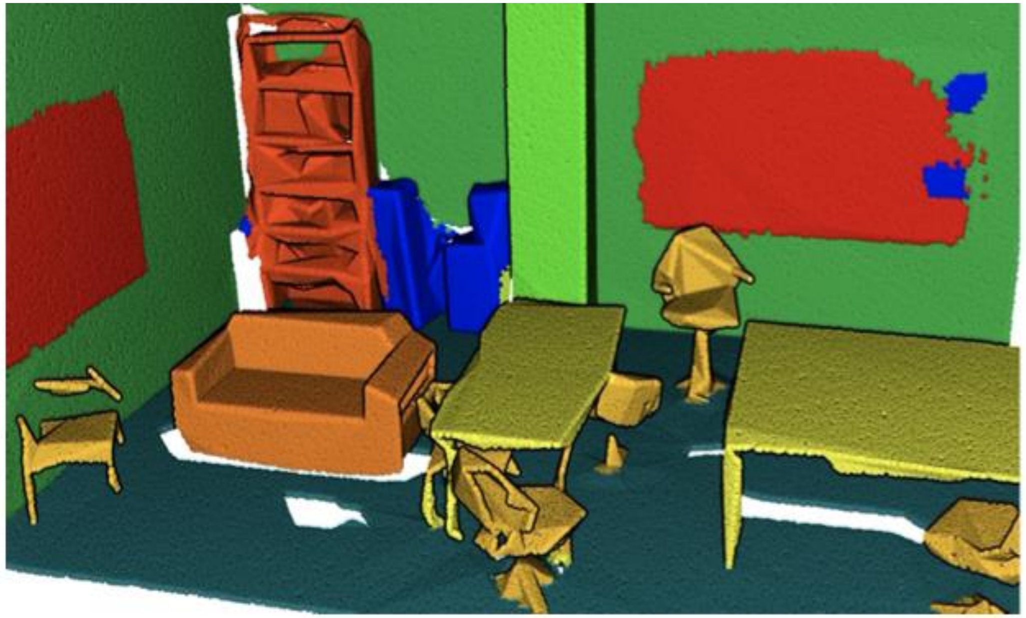
Transformer-based methods

What About Point Cloud That Cannot Be Projected?

- Many point clouds cannot be conveniently projected to 2D
- Can we process point cloud directly as a set of points?



3D CAD models

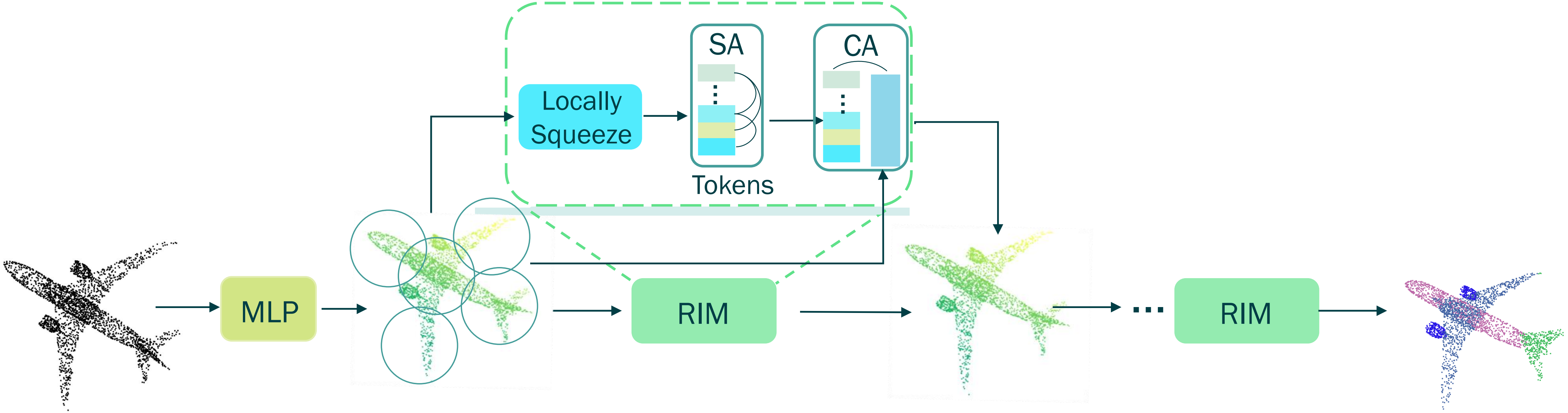


Accumulated LiDAR scans

Image credit: S3DIS dataset

YOGO: Processing Point-cloud Using Transformers

- Divide a point cloud evenly into sub-regions using the farthest-point sampling
- Process each point using multi-layer perceptron (MLP), locally aggregate features
- Use self-attention to exchange information across local regions



SA: self-attention CA: cross attention RIM: Relation Inference Module

YOGO Results

- Accuracy on-par with previous SOTA, but at least 3x faster

Method	Mean IoU	Latency	GPU Memory
PointNet [2]	83.7	21.4 ms	1.5 GB
RSNet [39]	84.9	73.8 ms	0.8 GB
SynSpecCNN [40]	84.7	-	-
PointNet++ [3]	85.1	77.7 ms	2.0 GB
PointNet++* [3]	85.4	236.7 ms	0.9 GB
DGCNN [41]	85.1	86.7 ms	2.4 GB
SpiderCNN [42]	85.3	170.1 ms	6.5 GB
SPLATNet [14]	85.4	-	-
SO-Net [33]	84.9	-	-
PointCNN [4]	86.1	134.2 ms	2.5 GB
YOGO (KNN)	85.2	25.6 ms	0.9 GB
YOGO (Ball query)	85.1	21.3 ms	1.0 GB

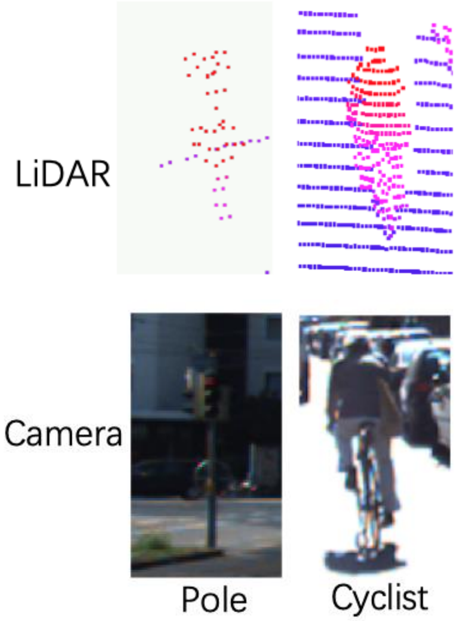
Method	Mean IoU	Latency	GPU Memory
PointNet [2]	42.97	24.8 ms	1.0 GB
DGCNN [41]	47.94	174.3 ms	2.4 GB
RSNet [39]	51.93	111.5 ms	1.1 GB
PointNet++* [3]	50.7	501.5 ms	1.6 GB
TangentConv [43]	52.6	-	-
PointCNN [4]	57.26	282.43 ms	4.6 GB
YOGO (KNN)	54.0	27.7 ms	2.0 GB
YOGO (Ball query)	53.8	24.0 ms	2.0 GB

Chenfeng Xu, et al. 2021

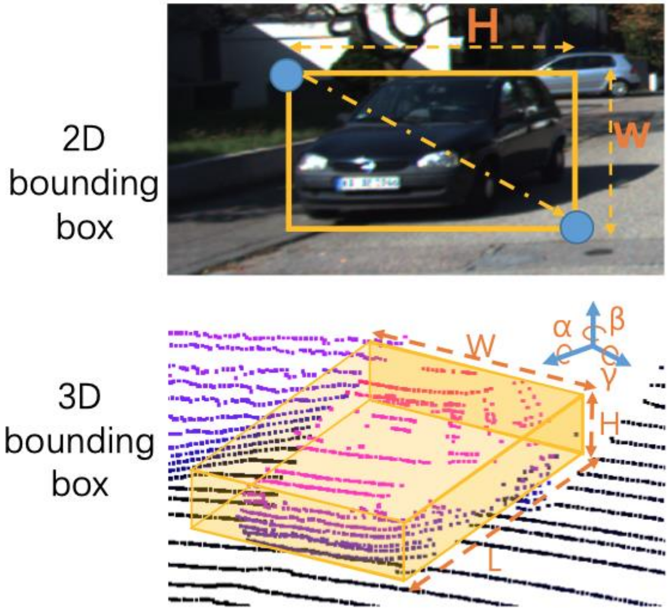
Tackling the data challenge

Data Challenges

- Deep learning requires a large amount of data, but annotating point cloud is challenging
  - Low resolution: Point-cloud sensors (such as LiDAR) have much lower resolution
  - Complex annotation operation: annotating objects in point cloud is harder than in images



(a) Low resolution

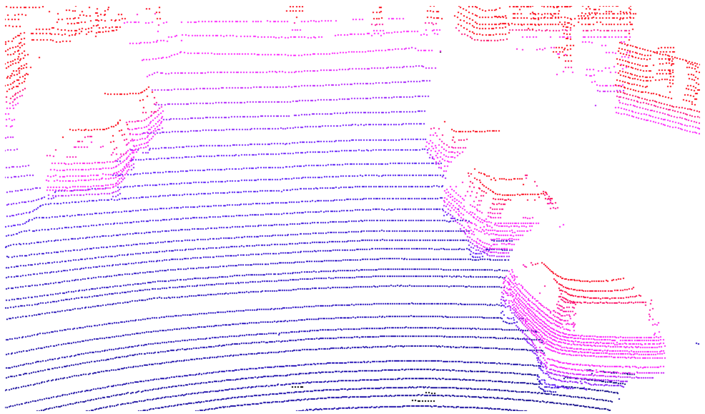


(b) Complex annotating operations

Building better annotation tools

Improving Annotation Efficiency: Sensor Fusion

- LiDAR point cloud has low resolution
- Solution: Use image-based detection to label LiDAR point cloud



Unlabeled LiDAR point cloud

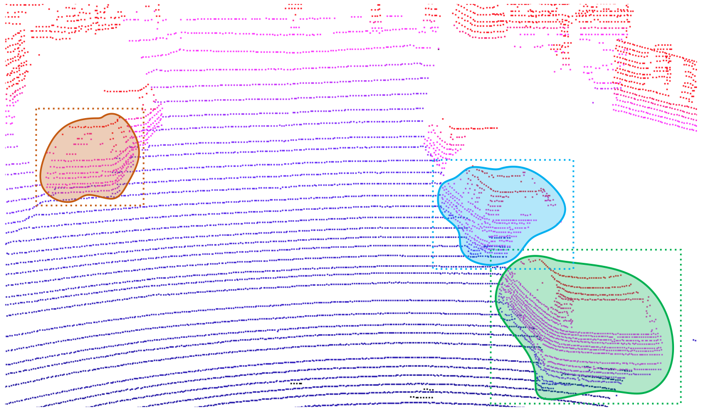
Projection

Projecting point cloud to the image

Query

Mask-RCNN pre-labeled image

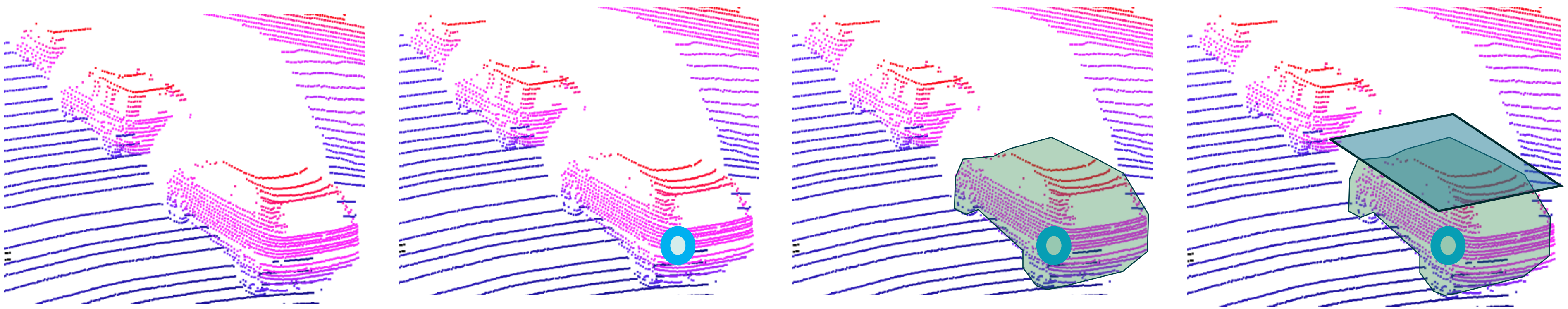
Label



Auto pre-labeled LiDAR point cloud

Improving Annotation Efficiency: One-click Annotation

- Annotating 3D point cloud is operationally complex
- Solution: Reducing the annotation operation to one-click

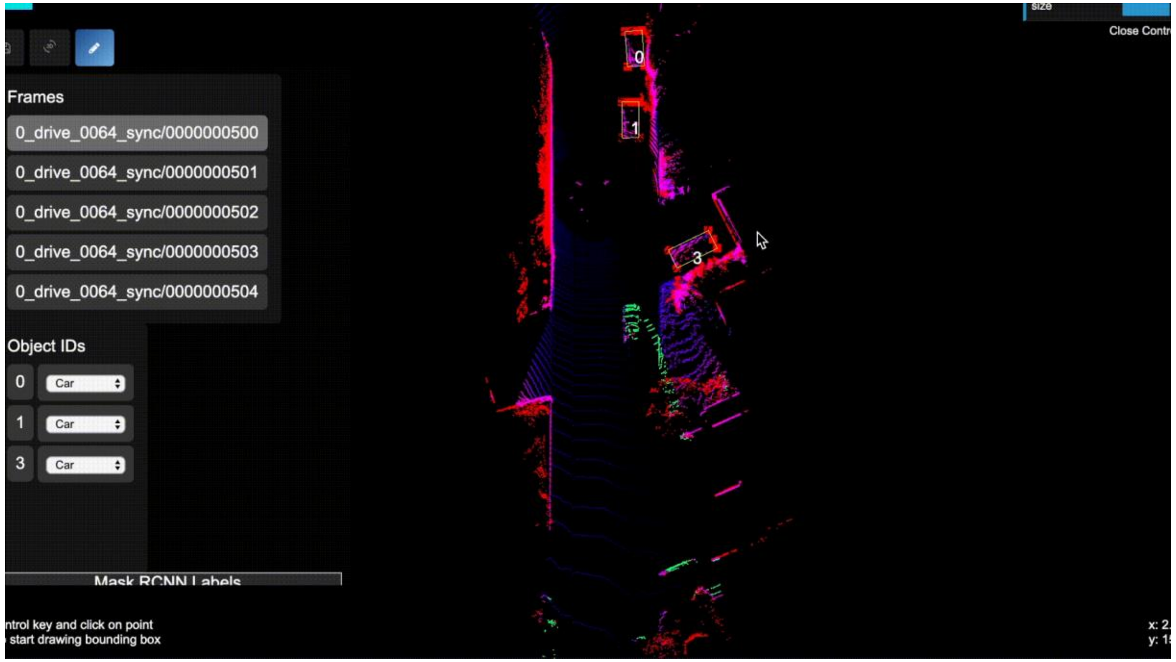


Original point cloud | Click | Grow | Bounding box estimation

LATTE: Accelerating LiDAR Point Cloud Annotation

LATTE: accelerated LiDAR annotation

- **Sensor fusion**: using images to assist annotation LiDAR
- **One-click annotation**: point-wise labels -> 3D bbox -> 2D top-view bbox -> one-click
- **Tracking:** using previous annotations to predict future ones
- 6.2x speedup in annotation!
- Paper published at ITSC2019



Open-sourced: https://github.com/bernwang/latte

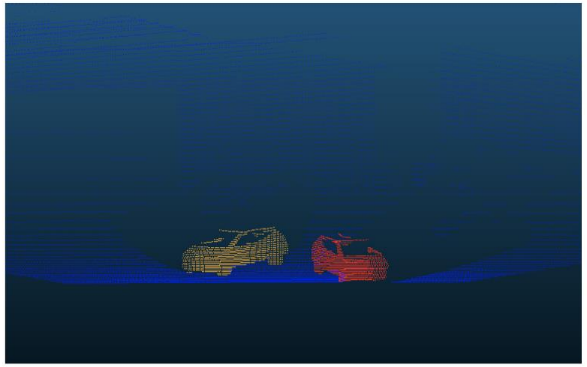
Bernie Wang et al., ITSC2019

Training with simulated data

Training Using Simulated Data?

Can we obtain unlimited training data from simulation?

Image

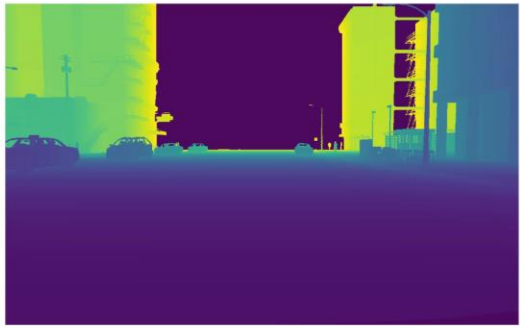


Point Cloud

Xiangyu Yue et al, ICMR 2018

Training Using Simulated Data?

Images



Depth map

Labels

- Accuracy drops significantly due to domain shift!

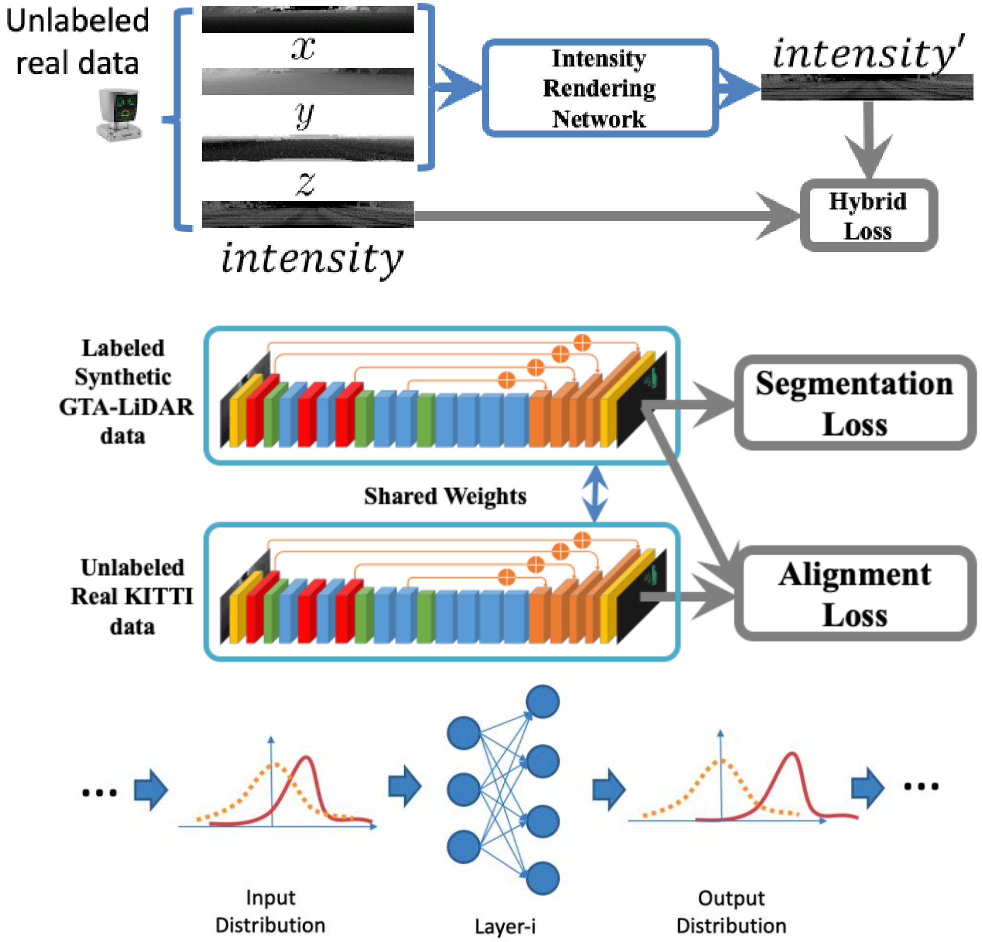
	Car accuracy (IoU - %)
Trained on real data	57.1
Trained on simulated data	30.0 (-27.1)

Domain Adaptation

Domain adaptation: techniques to bridge the domain gap between simulated data and real-world data:

- Learned Intensity rendering
- Feature alignment
- Batch statistics alignment

	IoU (%)
SQSGv1 on real data	57.1
SQSGv1 on sim data	30.0 (-27.1)
SQSGv2 on sim data w/ domain adaptation	57.4 (+0.3)



Bichen Wu, et al. " SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud", under review for ICRA19

Summary

Summary

- Increasingly more applications are powered by computer vision on 3D point cloud
- In this talk, we discuss two challenges for CV for point cloud:
  - Modeling challenge: difficult to process sparse, un-ordered 3D points
  - Data challenge: difficult to annotate enough data
- Our solution:
  - Modeling:
    - SqueezeSeg-V{1, 2, 3} efficient point cloud modeling based on spherical projection
    - YOGO: processing point-cloud using transformers
  - Data:
    - LATTE (efficient annotation tool)
    - Domain adaptation (training with simulated data)

Resources

Paper & code:

Modeling:

SqueezeSegV1: https://github.com/BichenWuUCB/SqueezeSeg

SqueezeSegV2: https://github.com/xuanyuzhou98/SqueezeSegV2

SqueezeSegV3: https://github.com/chenfengxu714/SqueezeSegV3

YOGO: https://github.com/chenfengxu714/YOGO

Data:

LATTE: https://github.com/bernwang/latte

Data synthesis paper: https://arxiv.org/abs/1804.00103

Thank you!