3D Object Detection Models

The Dream Team: Shahriar Hossain, Sanjay Mohan, Sean Steinle, Joseph Sepich

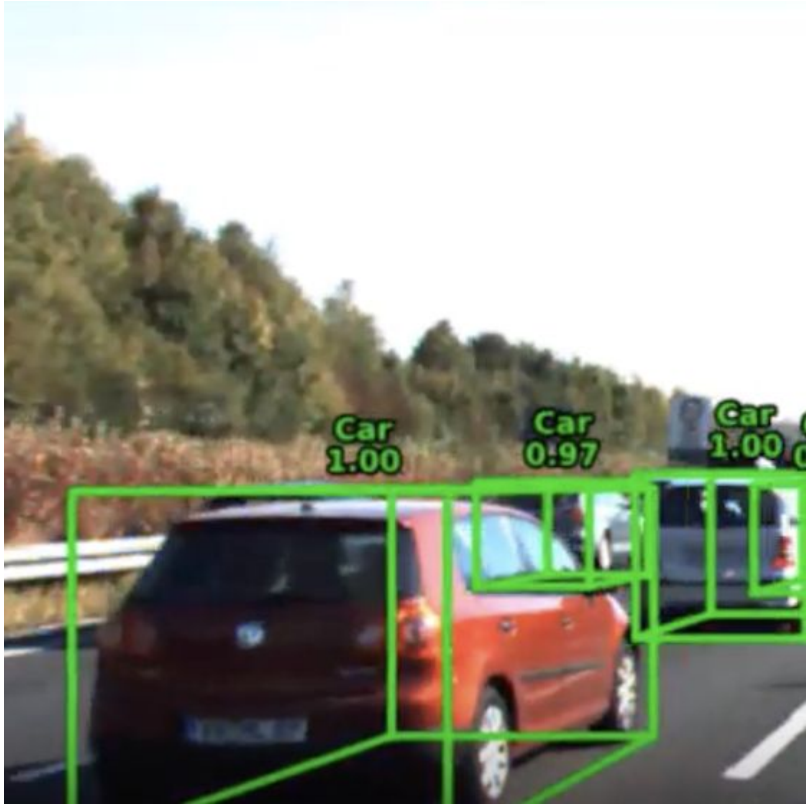
Overview

Shahriar - Motivation and Introduction

Sanjay - Background on Models

Sean - Experimental Results

Joseph - Practical Notes on Model Training



Sample output from a 3D object detection model

Motivation: Why 3D object detection matters?

Robust Depth Sensing: Direct distance measurements, immune to lighting changes (night, glare, shadows)

Critical for Safety: Precise 3D localization of cars, cyclists, pedestrians down to a few centimeters

Adverse Conditions: Rain, fog, low sun—cameras struggle, LiDAR still returns reliable data

Real-Time Requirements: Autonomous vehicles must process and react in milliseconds

Introduction

- We implemented 3 state of the art 3D object detection models
- These are: PointPillars, SECOND and PointRCNN
- All the models were implemented on KITTI dataset consisting of 29GB of pointcloud data and 12 GB of image data along with some calibration files and label files
- GPU used: Nvidia RTX 3080

PointPillar

- **Pillar-based Discretization:** Organizes the LiDAR point cloud into a grid of vertical pillars, creating a structured 2D representation from unstructured 3D data.
- **PointNet Feature Encoding:** Extracts features from points within each pillar using a simplified PointNet, capturing local geometric information efficiently.
- **2D CNN Backbone:** Processes the resulting 2D feature map with standard 2D convolutional neural networks, leveraging their speed and optimization.
- **Single-stage Detection:** Predicts 3D bounding boxes, class labels, and orientations in a single forward pass, enabling real-time performance.

SECOND

- **Voxelization:** Transforms the raw point cloud into a 3D voxel grid, enabling structured spatial processing.
- **Sparse 3D CNN:** Applies a sparse 3D convolutional neural network to efficiently process only non-empty voxels.
- **Region Proposal Network (RPN):** Uses a 2D convolutional backbone on a bird's-eye view (BEV) feature map to propose 3D regions.
- **Refinement Head:** Fine-tunes proposals to predict precise 3D bounding boxes, orientations, and class labels.

PointRCNN

- **Point-based Proposal:** Generates 3D object proposals directly from raw point clouds using a PointNet++ backbone.
- **Region-wise Feature Extraction:** Extracts local point features for each proposal to capture detailed geometric context.
- **Two-stage Pipeline:** Refines proposals in a second stage to predict accurate 3D bounding boxes, orientations, and classes.
- **Foreground Segmentation:** Enhances proposal quality by segmenting foreground points, improving detection robustness.

PointRCNN Dominates Pedestrian Detection

Though SECOND is more robust to challenging examples

PointPillar is not a serious competitor

Pedestrians are the most challenging object to classify

PointPillar

Specificity	Easy	Moderate	Hard
2D BBox AP	64.40%	61.43%	57.62%
AOS	49.35%	46.73%	43.84%
BEV AP	59.11%	54.32%	50.50%
3D AP	51.35%	47.98%	43.80%

SECOND

Specificity	Easy	Moderate	Hard
2D BBox AP	69.21%	**66.12%**	**63.43%**
AOS	65.40%	61.91%	58.93%
BEV AP	63.11%	56.77%	53.83%
3D AP	58.68%	53.90%	49.75%

PointRCNN

Specificity	Easy	Moderate	Hard
2D BBox AP	**73.54%**	65.83%	62.27%
AOS	**71.29%**	**63.17%**	**59.48%**
BEV AP	**67.51%**	**60.27%**	**54.09%**
3D AP	**61.84%**	**57.02%**	**51.15%**

PointRCNN Also Dominates Cyclist Detection

All models have serious degradation with respect to both skill and specificity

Cyclists are also difficult to detect

PointPillar

Specificity	Easy	Moderate	Hard
2D BBox AP	86.24%	73.06%	70.17%
AOS	85.02%	69.08%	66.28%
BEV AP	84.41%	67.13%	63.74%
3D AP	81.76%	63.66%	60.90%

SECOND

Specificity	Easy	Moderate	Hard
2D BBox AP	86.66%	76.53%	72.67%
AOS	86.33%	75.98%	72.12%
BEV AP	83.95%	69.92%	66.34%
3D AP	80.72%	66.56%	62.22%

PointRCNN

Specificity	Easy	Moderate	Hard
2D BBox AP	**89.75%**	**77.67%**	**75.27%**
AOS	**89.67%**	**77.20%**	**74.70%**
BEV AP	**88.36%**	**74.44%**	**71.01%**
3D AP	**87.72%**	**72.57%**	**69.94%**

All Models Succeed at Car Detection

Cars are the easiest category to detect

All models do very well, performing within 1-2% on all specificity-difficulty pairs

PointPillar

Specificity	Easy	Moderate	Hard
2D BBox AP	90.65%	89.33%	86.66%
AOS	90.48%	88.68%	85.73%
BEV AP	**89.96%**	**87.88%**	85.77%
3D AP	86.63%	76.74%	74.17%

SECOND

Specificity	Easy	Moderate	Hard
2D BBox AP	**90.81%**	**89.98%**	89.18%
AOS	**90.80%**	**89.90%**	**89.01%**
BEV AP	89.88%	87.83%	**86.47%**
3D AP	**88.52%**	**78.61%**	77.33%

PointRCNN

Specificity	Easy	Moderate	Hard
2D BBox AP	90.52%	89.23%	**88.94%**
AOS	90.52%	89.13%	88.78%
BEV AP	89.81%	87.12%	85.84%
3D AP	88.38%	78.19%	**77.72%**

PointRCNN Passes the Eye Test

PointRCNN is perfect

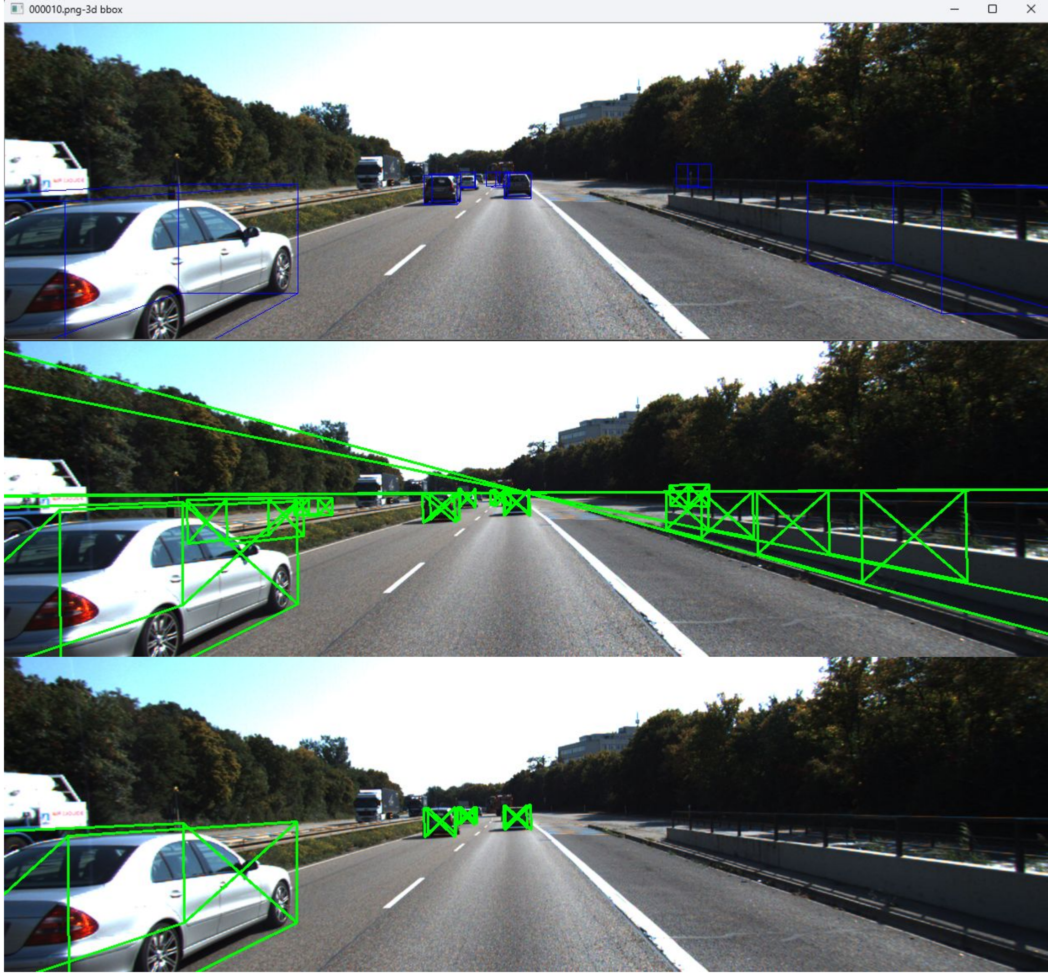
PointPillar has 2 false positives

SECOND has many false positives

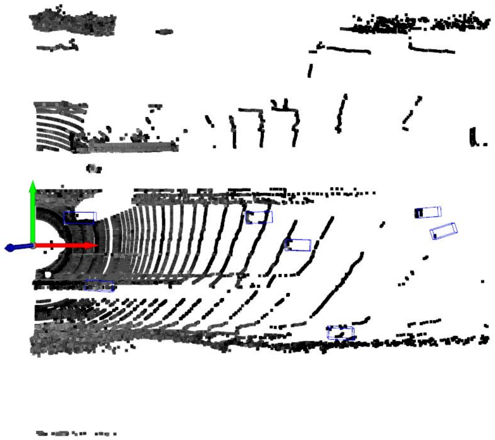
PointPillar

SECOND

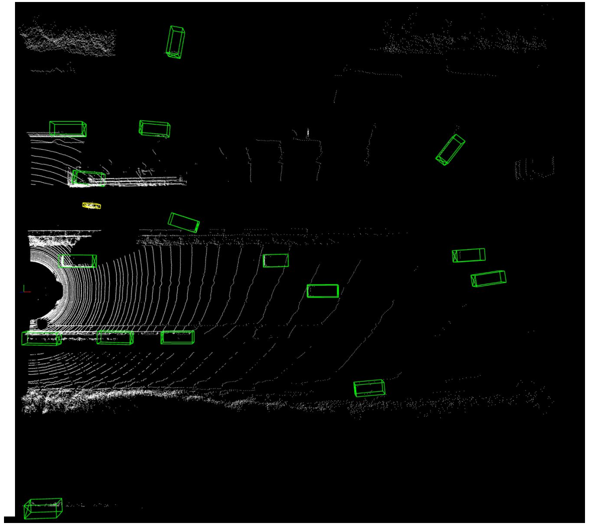
PointRCNN



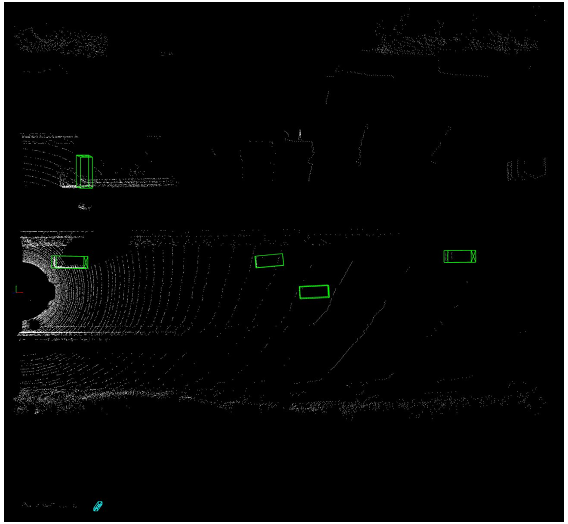
PointRCNN Passes the (Bird's) Eye Test



PointPillar



SECOND



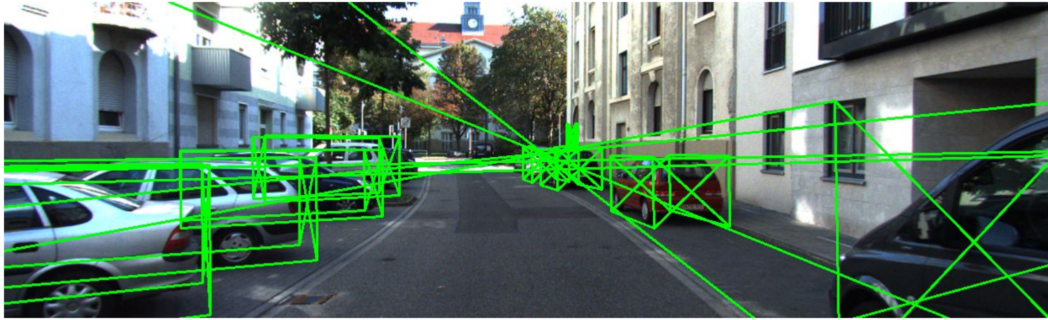
PointRCNN

Some Scenes Are Harder Than Others

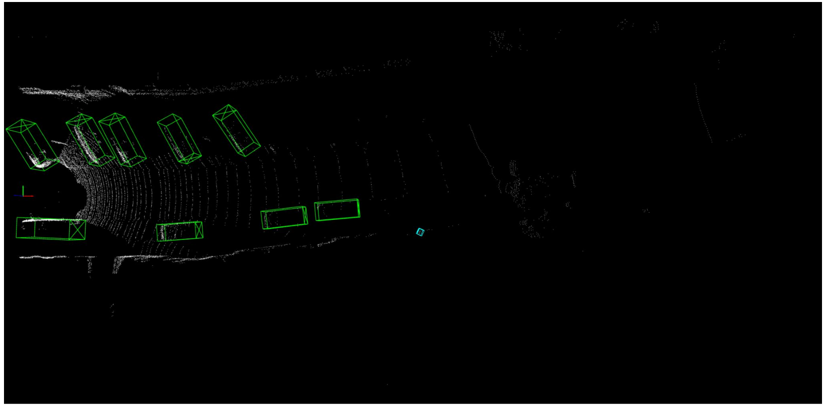
Occlusion is one challenge among many

Highlights need for sensor fusion

3D Bounding Boxes



Bird's Eye View (BEV)



Training Setup

PointPillars	SECOND	PointRCNN
Custom - mmdet	OpenPCDet	OpenPCDet
Windows	WSL Ubuntu 22.04	WSL Ubuntu 22.04
3080 GPU	3080Ti GPU	3080Ti GPU
~3 hours	4 hrs 43 minutes	7 hrs 46 minutes
Cross Entropy/Smooth L1/Focal Loss	Cross Entropy/Smooth L1/Focal Loss	IoU Based Loss

Reproducibility Challenges

Data Set Preparation

- Directory Layout
- Preprocessing

Environment Difficulties

- OS – Windows vs Linux (System Library Support)
- System Libraries – CUDA
- Language Packages
  - pytorch
  - spconv
- Language Version

Conclusion

3D Object Detection

PointPillars

SECOND

PointRCNN