Temporal LiDAR Frame Prediction for Autonomous Driving

David Deng and Avideh Zakhor
Video and Image Processing Lab

3DV 2020

Problem Statement

Given previous LiDAR frames, we aim to train a neural network to predict future ones.

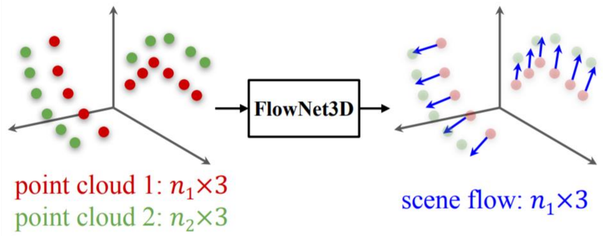
Background

Motivation

- Prediction is an essential task for autonomous driving
  - Cars need to predict future states to avoid collisions
  - Useful for reinforcement learning approaches
- Newly released autonomous driving datasets like nuScenes [1] have rich temporal LiDAR data.

Related Work

- Deep learning on point sets: PointNet [2], PointNet++ [3], EdgeConv [4]
- Advances in temporal point cloud processing: FlowNet3D [5]



Overview

- We propose a *class* of neural network architectures for prediction
- Architectures are inspired by FlowNet3D
- Model can be applied autoregressively to predict frames in the more distant future
- Use similarity between predicted point cloud and ground truth as loss function
- Ground truth is simply the next frame, so training is self-supervised
- Train and evaluate on nuScenes dataset

Architecture Modulation

We modulate the architecture framework in two ways:

1. Choice of feature extractor
2. Downsampling or not downsampling

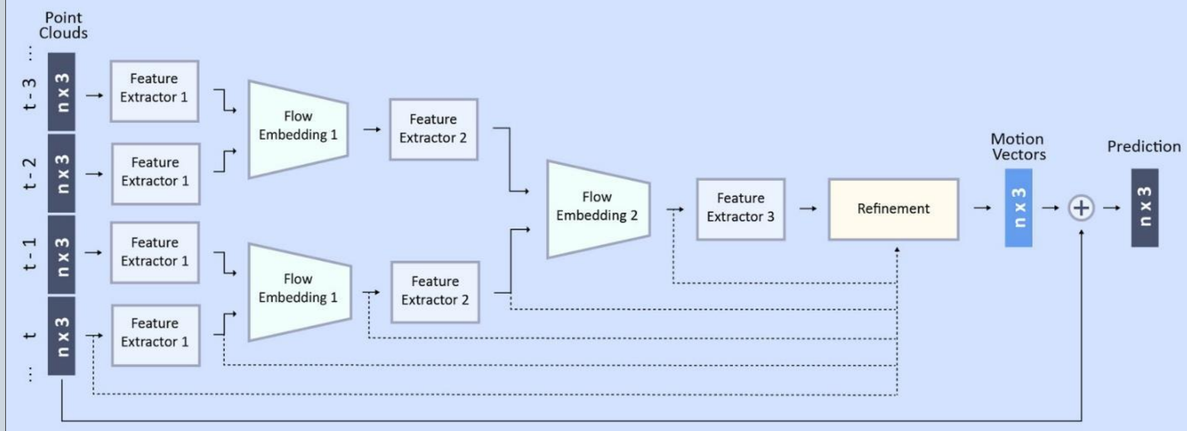
	Downsampling	No Downsampling
PointNet++	FlowNet3D, PN++ w/ DS	PN++ w/o DS
EdgeConv	EC w/ DS	EC w/o DS

Architecture classification. Primary architectural differences between our proposed architectures and FlowNet3D.

References

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. Computer Vision and Pattern Recognition (CVPR), 2020.
[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. Computer Vision and Pattern Recognition (CVPR), 2017.
[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Neural Information Processing Systems (NeurIPS), 2017.
[4] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG), 2019.
[5] X. Liu, C. R. Qi, and L. J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. CVPR, 2019.

Method



Generic architecture framework. Our framework takes in the past 4 frames and generates motion vectors to predict the next frame. The specific architecture is determined by which feature extractor is used and whether or not downsampling is used. The refinement module may use any of the previous learned features, as indicated by the dashed lines.

Module	PN++ w/ DS	PN++ w/o DS	EC w/ DS	EC w/o DS
Feature Extraction 1	**PointNet++**: r = 0.5, SR=0.25×, mlp=[128,128]	**PointNet++**: r=0.7, SR=1×, mlp=[32,32]	**EdgeConv**: k=16, SR=0.25×, mlp=[128,128]	**EdgeConv**: k=16, SR=1×, mlp=[32,32]
Flow Embedding 1	**PointNet++**: r = 1.5, SR=1×, mlp=[128]	**PointNet++**: r=1, SR=1×, mlp=[32]	**EdgeConv**: k = 16, SR=1×, mlp=[128]	**EdgeConv**: k = 16, SR=1×, mlp=[32]
Feature Extraction 2	**PointNet++**: r = 1, SR=0.25×, mlp=[256,256]	**PointNet++**: r = 0.7, SR=1×, mlp=[64,64]	**EdgeConv**: k = 16, SR=0.25×, mlp=[256,256]	**EdgeConv**: k = 16, SR=1×, mlp=[64,64]
Flow Embedding 2	**PointNet++**: r = 3, SR=1×, mlp=[256]	**PointNet++**: r = 1, SR=1×, mlp=[64]	**EdgeConv**: k = 16, SR=1×, mlp=[256]	**EdgeConv**: k = 16, SR=1×, mlp=[64]
Feature Extraction 3	**PointNet++**: r = 2, SR=0.2×, mlp=[512]	**PointNet++**: r = 0.7, SR=1×, mlp=[128]	**EdgeConv**: k = 16, SR=0.2×, mlp=[512]	**EdgeConv**: k = 16, SR=1×, mlp=[128]
Refinement	**Upconv1**: k = 16, SR=5×, SS=XYZ, mlp1=[512], mlp2=[512] **Upconv2**: k = 16, SR=4×, SS=XYZ, mlp1=[512], mlp2=[512] **FeatProp**: SR=4×, SS=XYZ, mlp=[256] **mlp**: [256, 128, 3]	**mlp**: input = (feat1, feat2, feat3), widths = [512, 256, 128, 3]	**Upconv1**: k = 16, SR=5×, SS=flow2, mlp1=[512], mlp2=[512] **Upconv2**: k = 16, SR=4×, SS=flow1, mlp1=[512], mlp2=[512] **FeatProp**: SR=4×, SS=XYZ, mlp=[256] **mlp**: [256, 128, 3]	**mlp**: input = (feat1, feat2, feat3), widths = [512, 256, 128, 3]

Architecture details. r = ball query radius, k = k for KNN grouping, SR = sampling rate, SS = sampling space, feat and flow refer to the output of the corresponding layer.

PointNet++ [3] vs **EdgeConv [4]**

$$f_i' = \max_{j\ ||p_j - p_i|| \le r} h_\theta(f_j, p_j - p_i) \qquad f_i' = \max_{j=1\ldots k} h_\theta(f_i^j - f_i, f_i)$$

p: point, f: feature, h_θ: multilayer perceptron, r: radius, f_i^k: kth nearest neighbor of f_i in feature space

- Similar operations, but EdgeConv groups points dynamically in feature space
- EdgeConv achieves higher classification and segmentation accuracy

Downsampling

•Pros
- Reduces computational complexity
- Encourages hierarchical feature learning

•Cons
- Reduces feature content; need more features at bottleneck to resolve ambiguity during upsampling

Loss Functions

$$L_{CD}(P,Q) = \sum_{p\in P} \min_{q\in Q} ||q-p||_2^2 + \sum_{q\in Q} \min_{p\in P} ||p-q||_2^2$$

Chamfer Distance

$$L_{EMD}(P,Q) = \min_{\phi\in P\to Q} \sum_{p\in P} ||p-\phi(p)||_2$$

Earth Mover's Distance

Quantitative Results

Baselines

x_{t+1}^* indicates prediction, x_t indicates point cloud at time t

Identity:

$$x_{t+1}^* = x_t$$

FlowNet3D Out of the Box (FN3DOOB):

$$x_{t+1}^* = x_t - FlowNet3D(x_t, x_{t-1})$$

FN3D Adjusted (FN3DA): retrain the FlowNet3D architecture using our loss functions.

Model	CD (m^2)	EMD (m)	Model Size (MB)	Runtime (s)	Max Memory Allocated (MB)
Identity	.2472	34.88	-	-	-
FN3DOOB [11]	1.2084	91.68	14.9	.2946	**669.6**
FN3DA	.1399	33.94	14.9	.2946	**669.6**
PN++ w/ DS	**.1381**	**33.14**	22.1	.2635	783.5
PN++ w/o DS	.1813	37.63	**3.7**	.5079	1007.2
EC w/ DS	.1837	35.49	10.2	**.2591**	907.9
EC w/o DS	.1450	34.23	3.9	.6198	721.3

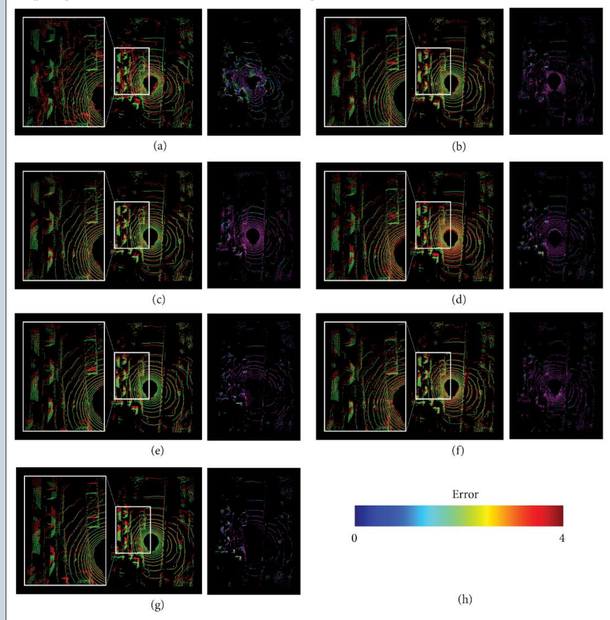
Accuracy and complexity of methods. The table shows the average CD and EMD across the first 5 future frames, as well as the size, runtime, and memory usage of the models.

Observations

- Downsampling models are ~2x faster but need to be ~4x larger than their non downsampling counterparts
- PointNet++ w/ Downsampling has the best accuracy and fast runtime.
- EdgeConv w/o Downsampling has strong accuracy and a smaller model.

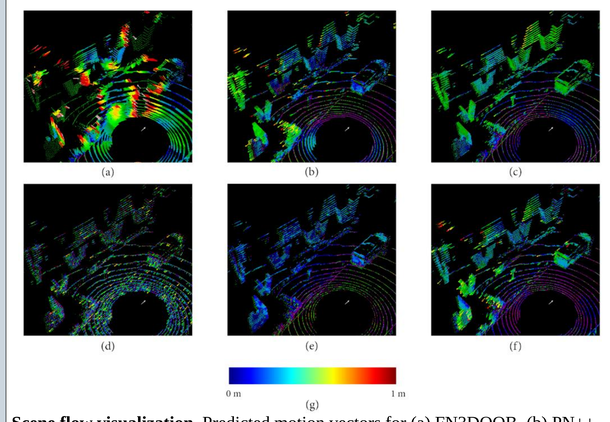
Visualizations

In this scene, the ego vehicle is driving past a truck parked next to a line of v-shaped columns. In the magnified region, the truck is in the top right, and the columns are along the left side.



Visualizations. Error visualizations for (a) FN3DOOB, (b) FN3DA, (c) PN++ w/ DS, (d) PN++ w/o DS, (e) EC w/ DS, (f) EC w/o DS, (g) Identity on t + 5. For each method, the middle picture shows the ground truth in green and the prediction in red; the left picture zooms in on a region of interest in the middle picture; the right picture shows the error between the prediction and the ground truth.

Observations: PN++ w/ DS and EC w/o DS accurately capture the motion of the truck and pillars, shown in the zoomed in region. They exhibit close to 0 error, indicated by their purple error visualizations.



Scene flow visualization. Predicted motion vectors for (a) FN3DOOB, (b) PN++ w/ DS, (c) EC w/ DS, (d) FN3DA, (e) PN++ w/o DS, (f) EC w/o DS.

Observations: PN++ w/ DS and EC w/o DS produce smooth, accurate scene flow compared to the other methods.

Conclusion

We design an architecture framework that accurately predicts future point clouds and scene flow in a self-supervised fashion. We found PN++ w/ DS and EC w/o DS to be the most viable architectures.