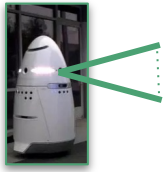
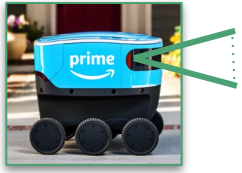
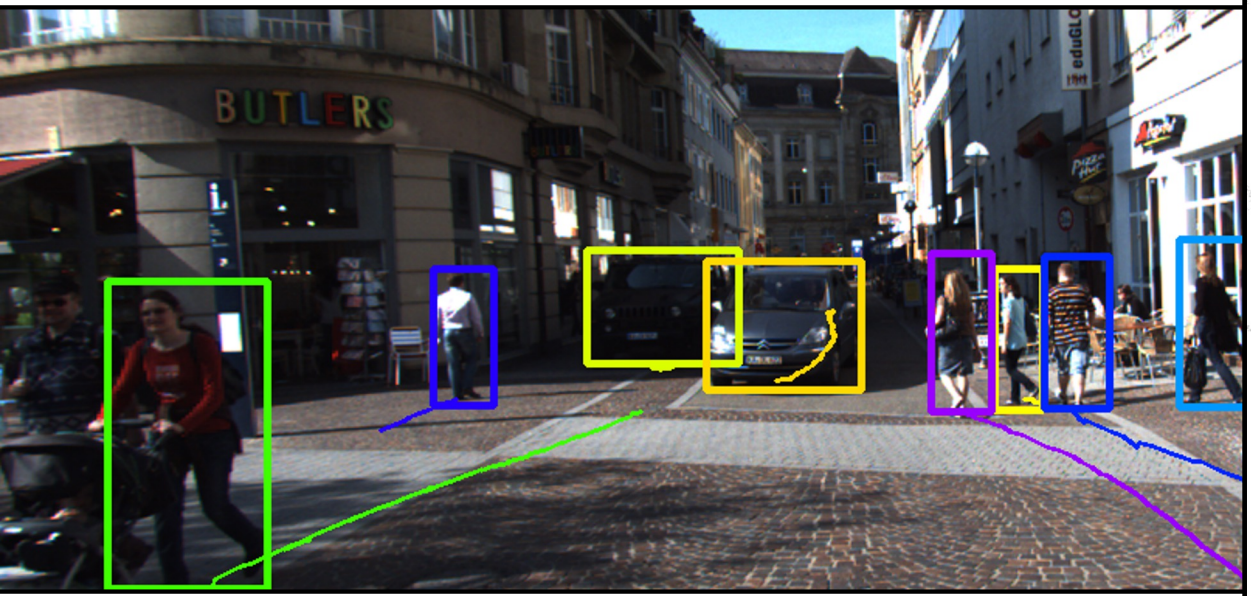
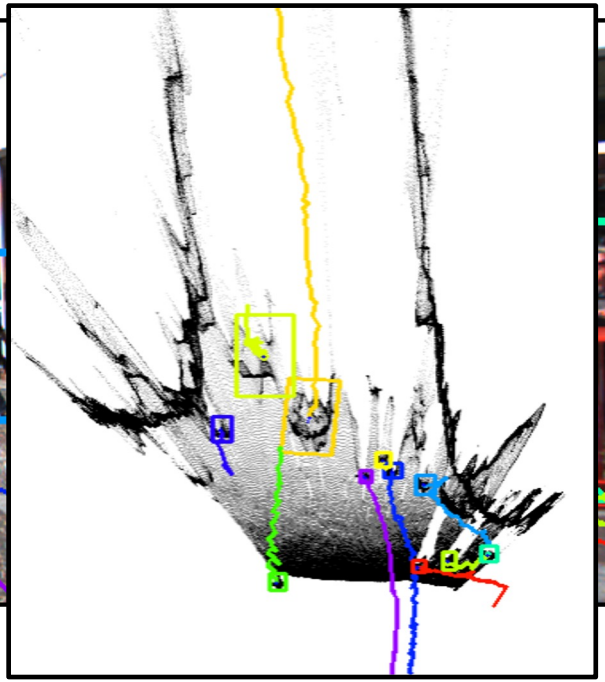
3D Computer Vision: Detection, Tracking and Segmentation

Motivation









A Brief History

1984: NavLab Project

- So far: indoor robotics, laboratory settings (left: Shakey the robot)
- NavLab project takes robotic perception outdoors! (right: NavLab 1)

Toward Autonomous Driving: The CMU Navlab

Part I — Perception

Charles Thorpe, Martial Hebert, Takeo Kanade, and Steven Shafer
Carnegie Mellon University

Designing an outdoor mobile robot that follows flat, straight, well-illuminated, and clearly marked roads is one thing. Operating such a robot in a realistic environment with bad weather, bad lighting, and bad or changing roads is another.

1986: NavLab 1

- NavLab 1: 3D vision!
- Stereo? ERIM scanner => early LiDAR sensor!
  - 50kg,
  - 40m range, 2 frames/sec

Toward Autonomous Driving: The CMU Navlab

Part I — Perception

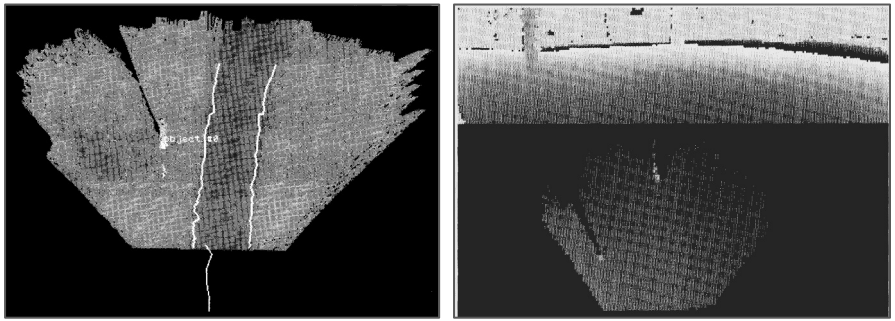
Charles Thorpe, Martial Hebert, Takeo Kanade, and Steven Shafer
Carnegie Mellon University

Clearly, one color camera is not enough to collect 3D data. An alternative is to use passive techniques for recovering 3D data, such as stereo vision, but these techniques have significant drawbacks, including high computational demand, difficulty in ranging bland surfaces, and reliance on ambient lighting. Instead, we use an active sen-

	ERIM
EYE SAFE	Yes (?)

1986: NavLab 1

- 3D vision
  - Terrain mapping and vehicle localization
  - Road segmentation
  - Obstacle "detection"



3D measurements from imaging laser radars: how good are they?

1992

Martial Hebert and Eric Krotkov

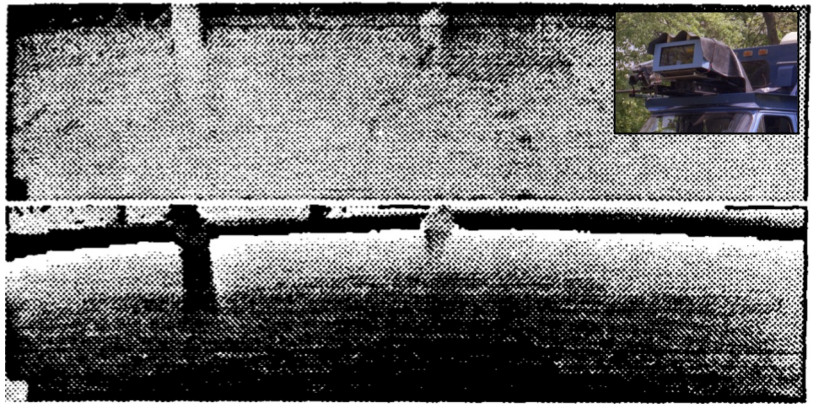
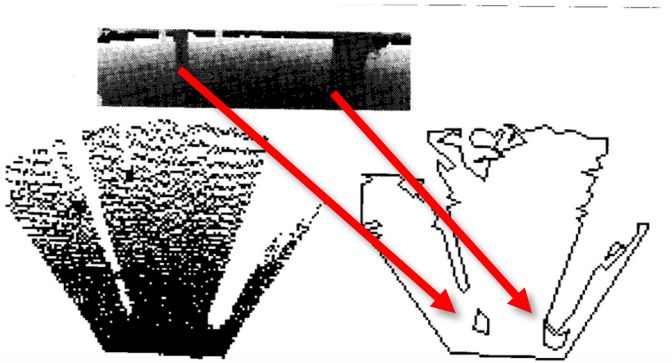


Figure 3. Erim intensity (top) and range images. The scene contains a tree (visible to the left), and a person (visible in the upper centre) on a path

Early Obstacle Detection!

- Grid + cell classification
  - Estimate per-cell normal
  - Deviates from "up" vec?
- Cluster obstacle cells to obtain regions (obstacles)



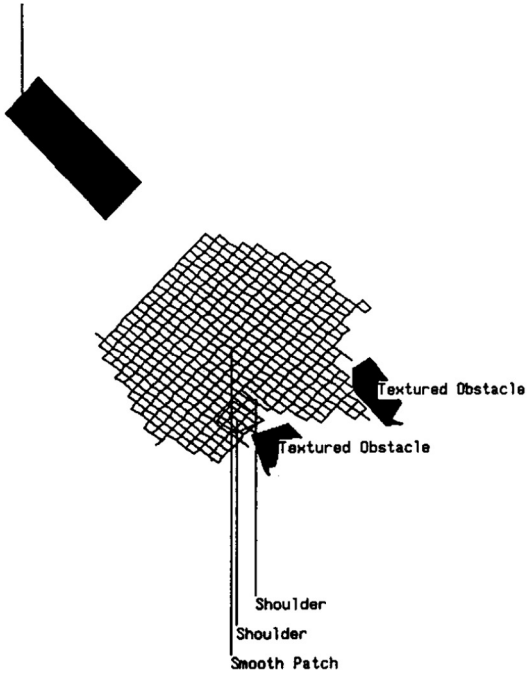


Fig. 19. The resultant description of 3-D terrain and obstacles from the image in Fig. 18. The navigable area is shown as a mesh, and the two trees are detected as "textured obstacles" and shown as black polygons.

1995: NavLab 5

- The first autonomous coast-to-coast drive!
  - 2,849 mi (Pittsburgh -> San Diego), 2,797 mi autonomous (98.2%), avg. 63.8 mph
- Detection + tracking using line laser [Zhao&Thrope, CVPR'98]

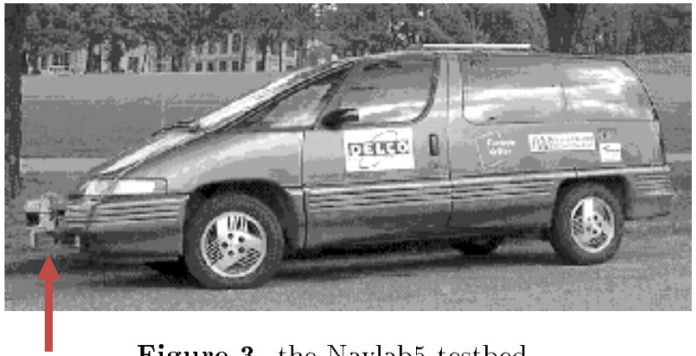
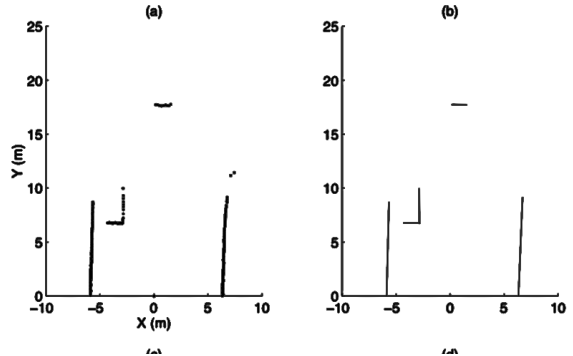


Figure 3. the Navlab5 testbed

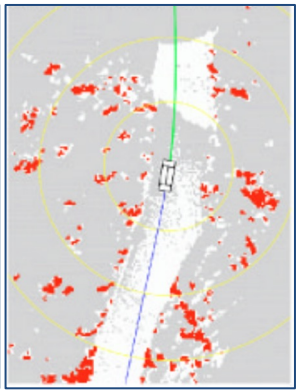


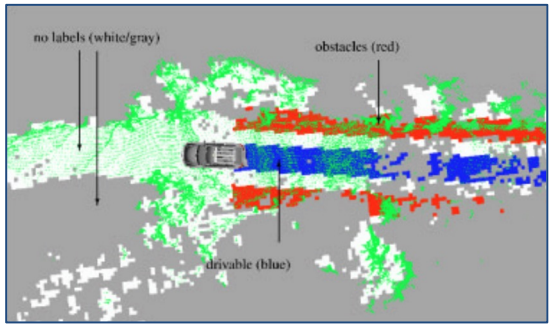
DARPA 2005 Urban Challenge

- 142-mile long course, Mojave desert (10h)
  - **2004**: 107 teams registered and 15 raced
    - None navigates for more than 5% of the entire course!
  - **2005**: 195 teams registered and 23 raced.
    - Five teams finished.
    - Winner: Stanford's robot "Stanley",
      - 6 h, 53 min
    - CMU team 2nd (NavLab 11)

DARPA 2005 Urban Challenge

- Vehicle state estimation:
  - GPS, IMU, wheel encoders
  - Extended Kalman filter
- 3D Vision: lidar-based terrain segmentation and mapping: detect non-drivable terrain ahead!





DARPA 2005 Urban Challenge

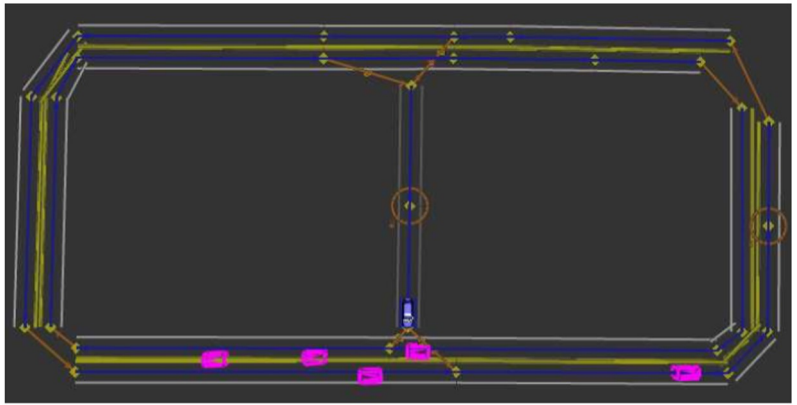
"While the DARPA Grand Challenge was a milestone in the quest for self-driving cars, it left open a number of important problems. Most important among those was the fact that the **race environment was static**. Stanley is **unable to navigate in traffic**. For autonomous cars to succeed, robots, such as Stanley, **must be able to perceive and interact with moving traffic**. While a number of systems have shown impressive results [...], further research is needed to achieve the level of reliability necessary for this demanding task.

Thrun et al., Stanley: The Robot That Won the DARPA Grand Challenge. The 2005 DARPA Grand Challenge. Springer Tracts in Advanced Robotics'07

DARPA 2007 Urban Challenge

- 60 mi urban area course, 6 h
- Obey traffic laws
- CMU team 1st, Stanford 2nd

(a) test conditions on course A at the UGC



(b) Junior at intersection on course A

3D Mobile Perception Landscape

- 2005: static world
- 2007: 60 mi urban area course, 6 h
  - Competing robots + human professional drivers
  - Early object detection and tracking!

RSS'04

Model Based Vehicle Tracking for Autonomous Driving in Urban Environments

Anna Petrovskaya and Sebastian Thrun
Computer Science Department
Stanford University
Stanford, California 94305, USA
{ anya, thrun }@cs.stanford.edu

***Abstract*— Situational awareness is crucial for autonomous driving in urban environments. This paper describes moving vehicle tracking module that we developed for our autonomous driving robot Junior. The robot won second place in the Urban Grand Challenge, an autonomous driving race organized by the U.S. Government in 2007. The tracking module provides reliable tracking of moving vehicles from a high-speed moving platform using laser range finders. Our approach models both dynamic and geometric properties of the tracked vehicles and estimates them using a single Bayes filter per vehicle. We also show how to build efficient 2D representations out of 3D range data and how to detect poorly visible black vehicles. Experimental validation includes the most challenging conditions presented at the UGC as well as other urban settings.**

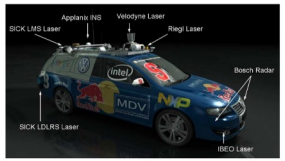


Fig. 1. Junior, our entry in the DARPA Urban Challenge. Junior is equipped with five different laser measurement systems, a multi-radar assembly, and a multi-signal inertial navigation system, as shown in this figure.

I. Introduction

Autonomously driving cars have been a long-lasting dream of robotics researchers and enthusiasts. Self-driving cars promise to bring a number of benefits to society, including prevention of road accidents, optimal fuel usage, comfort and

to which we will also refer as the ego-vehicle (Fig. 1). In

ICRA'11

Towards 3D Object Recognition via Classification of Arbitrary Object Tracks

Alex Teichman, Jesse Levinson, Sebastian Thrun
Stanford Artificial Intelligence Laboratory
{teichman, jessel, thrun}@cs.stanford.edu

***Abstract*— Object recognition is a critical next step for autonomous robots, but a solution to the problem has remained elusive. Prior 3D-sensor-based work largely classifies individual point cloud segments or uses class-specific trackers. In this paper, we take the approach of classifying the tracks of all visible objects. Our new track classification method, based on a mathematically principled method of combining log odds estimators, is fast enough for real time use, is non-specific to object class, and performs well (98.5% accuracy) on the task of classifying correctly-tracked, well-segmented objects into car, pedestrian, bicyclist, and background classes.**

We evaluate the classifier's performance using the Stanford Track Collection, a new dataset of about 1.3 million labeled point clouds in about 14,000 tracks recorded from an autonomous vehicle research platform. This dataset, which we make publicly available, contains tracks extracted from about one hour of 360-degree, 10Hz depth information recorded both while driving on busy campus streets and parked at busy intersections.

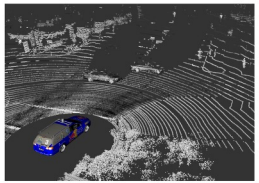


Fig. 1: Example scan of two other cars and a bicyclist at an intersection. Points are colored by return intensity.

I. Introduction

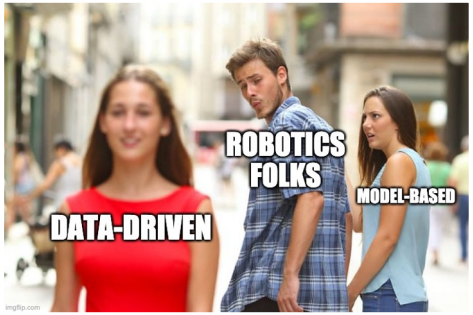
Object recognition in dynamic environments is one of the primary unsolved challenges facing the robotics commu-

None of the algorithms described here are specifically

3D Object Recognition via Classification of Object Tracks

Teichman et al., ICRA'11:

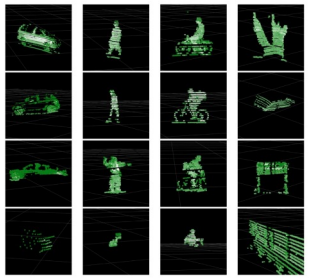
- "***Object recognition*** is a critical next step for autonomous robots"

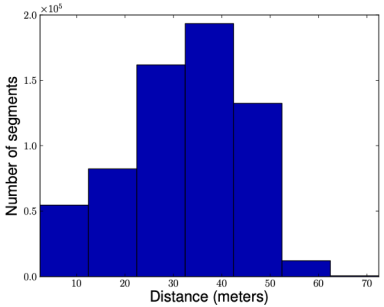


- "A solution would include segmentation, tracking and ***classification*** components, and would allow for the addition of new object classes without the need for an expert to specify new models"

3D Object Recognition via Classification of Object Tracks

- Data: Stanford Track Collection
  - Bottom-up point cloud segmentation
  - Tracking (Kalman filter)
  - Label tracks!





Number of tracks

Set	Car	Pedestrian	Bicyclist	Background	All
Training	904	205	187	6585	7881
Testing	847	112	140	4936	6035
Total	1751	317	327	11521	13916

Number of segments

Set	Car	Pedestrian	Bicyclist	Background	All
Training	92255	32281	31165	532760	688461
Testing	59173	22203	25410	530917	637703
Total	151428	54484	56575	1063677	1326164

3D Object Recognition via Classification of Object Tracks

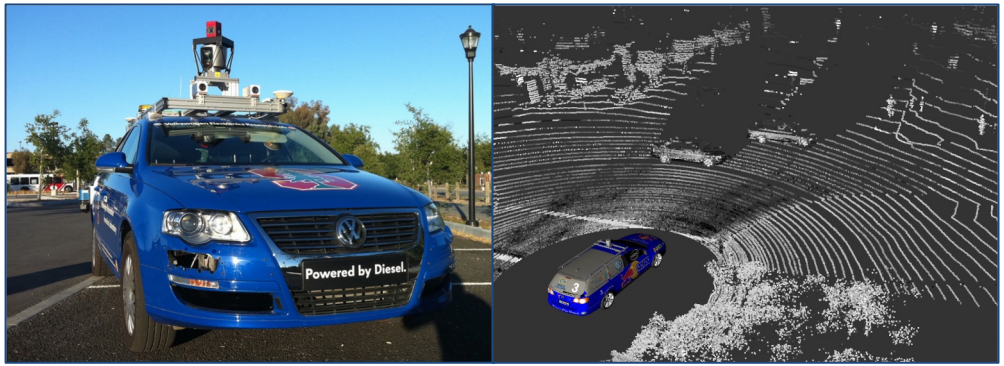
- Extract features + classify
- Results:

Fig. 7: Examples of side, top, and front views of objects using virtual orthographic camera intensity images. Pixel intensities reflect the average LIDAR return intensity. Several HOG descriptors are computed on each view. Alignment into a canonical orientation enables seeing consistent views of objects.

Method	Car	Pedestrian	Bicyclist	Overall
ADBF	98.7%	99.8%	99.9%	98.5%
Naïve DBF	98.3%	99.6%	99.8%	97.7%
Segment classifier only	98.3%	99.5%	99.8%	97.6%
Holistic classifier only	94.3%	99.2%	99.2%	93.0%
Prior only	86.0%	98.1%	97.7%	81.8%

3D Object Recognition via Classification of Object Tracks

Tracking-Before-Detection



Segment & Track

Classify

Teichman et al., Tracking-Based Semi-Supervised Learning, RSS'11

Segmentation is Difficult!

- Interacting objects, crowded scenes
- Sensor resolution decreasing with distance from the sensor, “holes” due to reflective and low-albedo surfaces

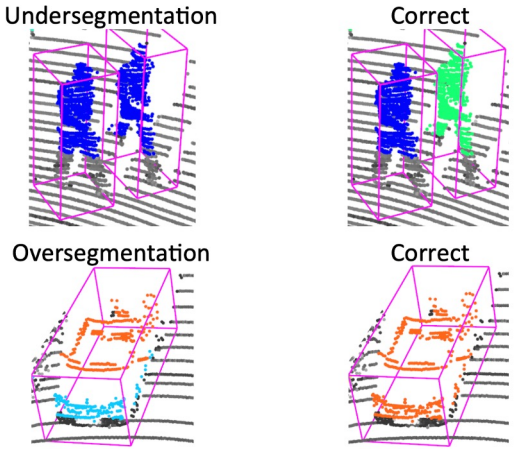


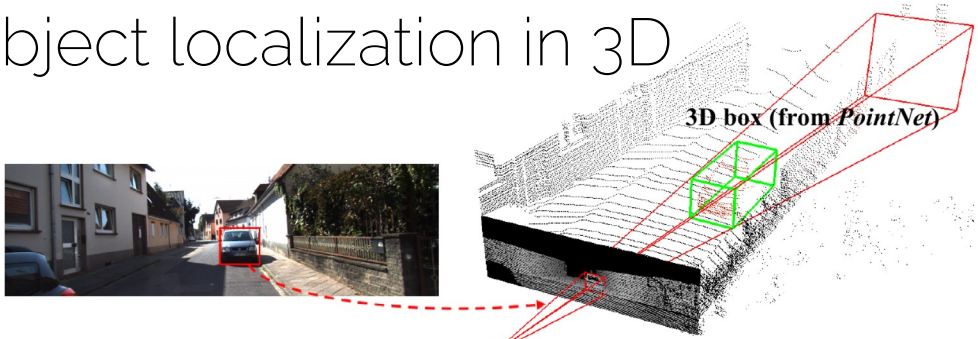
Figure from Held et al., A Probabilistic Framework for Real-time 3D Segmentation using Spatial, Temporal, and Semantic Cues, RSS'16

3D Computer Vision in the Era of Deep Learning

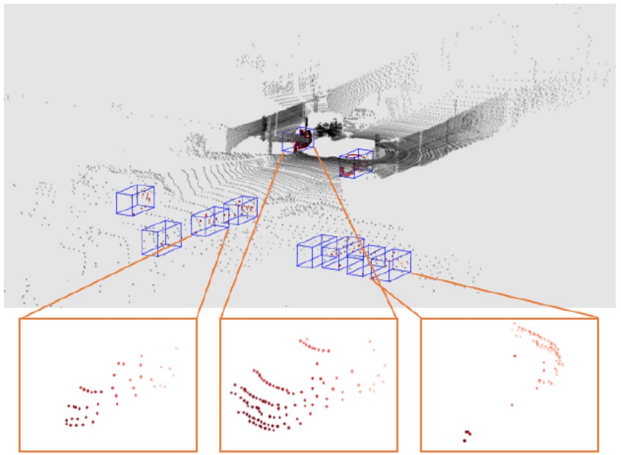
Learning Representations from 3D Data

Challenges

- Depth sensor characteristics
  - Limited scan range
  - ``Non-cooperative`` materials
  - Sparse and unstructured signal
- Mobile platform
- Object localization in 3D



Source: Qi et al., CVPR'18



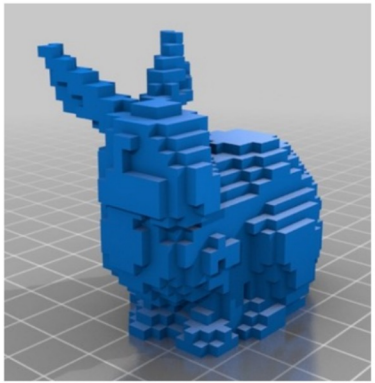
Source: Yuan et al., 3DV'19

Deep Learning on Point Clouds

- Signal representation?

Point Cloud

Mesh



Volumetric

Projected View RGB(D)

Slides adapted from Charles Qi CVPR presentation slides (https://web.stanford.edu/~rqi/pointnet/docs/cvpr17_pointnet_slides.pdf)

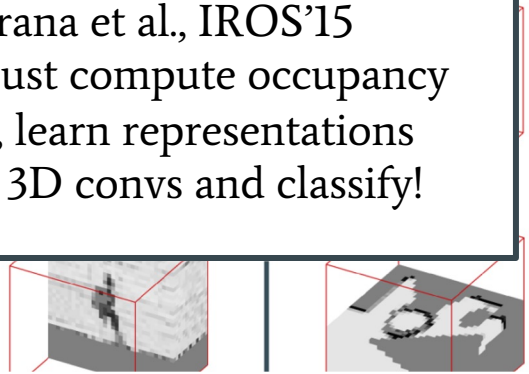
Voxel Grids + 3D Convolutions?

VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition

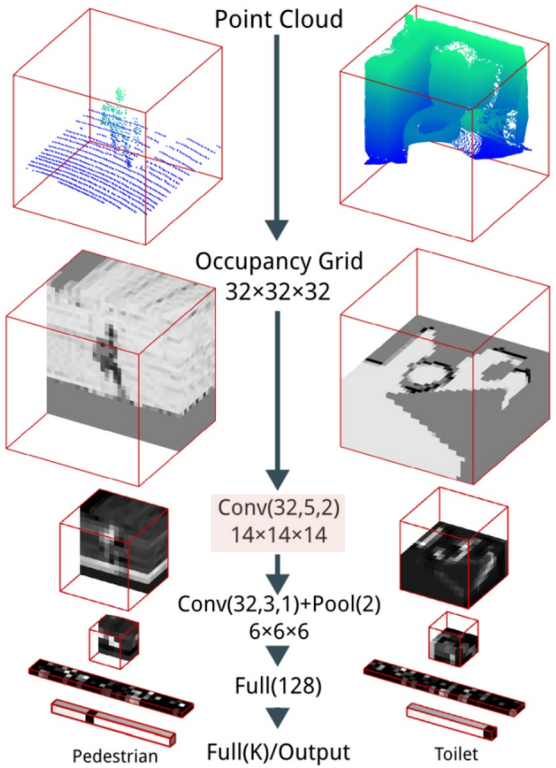
Daniel Maturana and Sebastian Scherer

***Abstract*— Robust object recognition is a crucial s... fo... robots operating autonomously in real world environ... Range sensors such as LiDAR and RGBD cameras are... creasingly found in modern robotic systems, providing a ri... source of 3D information that can aid in this task. Howeve... many current systems do not fully utilize this information an... have trouble efficiently dealing with large amounts of poin... cloud data. In this paper, we propose *VoxNet*, an architectur... to tackle this problem by integrating a volumetric Occupanc... Grid representation with a supervised 3D Convolutional Neura... Network (3D CNN). We evaluate our approach on publicl... available benchmarks using LiDAR, RGBD, and CAD data. *VoxNet* achieves accuracy beyond the state of the art while labeling hundreds of instances per second.**

Maturana et al., IROS'15
Y'all just compute occupancy maps, learn representations using 3D convs and classify!



Voxel Grids + 3D Convolutions?



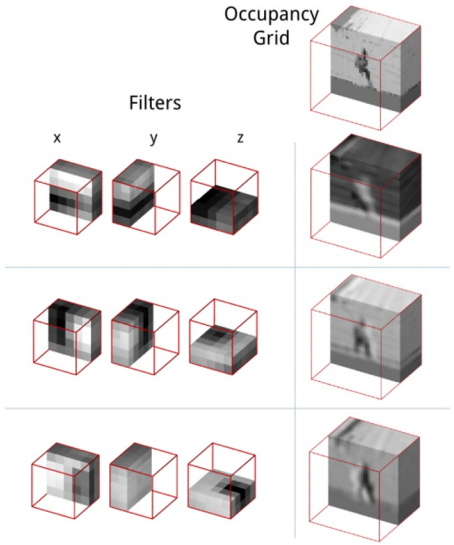


Fig. 3. Cross sections of three $5 \times 5 \times 5$ filters from the first layer of VoxNet in the Sydney Objects Database, with corresponding feature map on the right.

Deep Learning on Point Clouds

- Signal representation?

Point Cloud

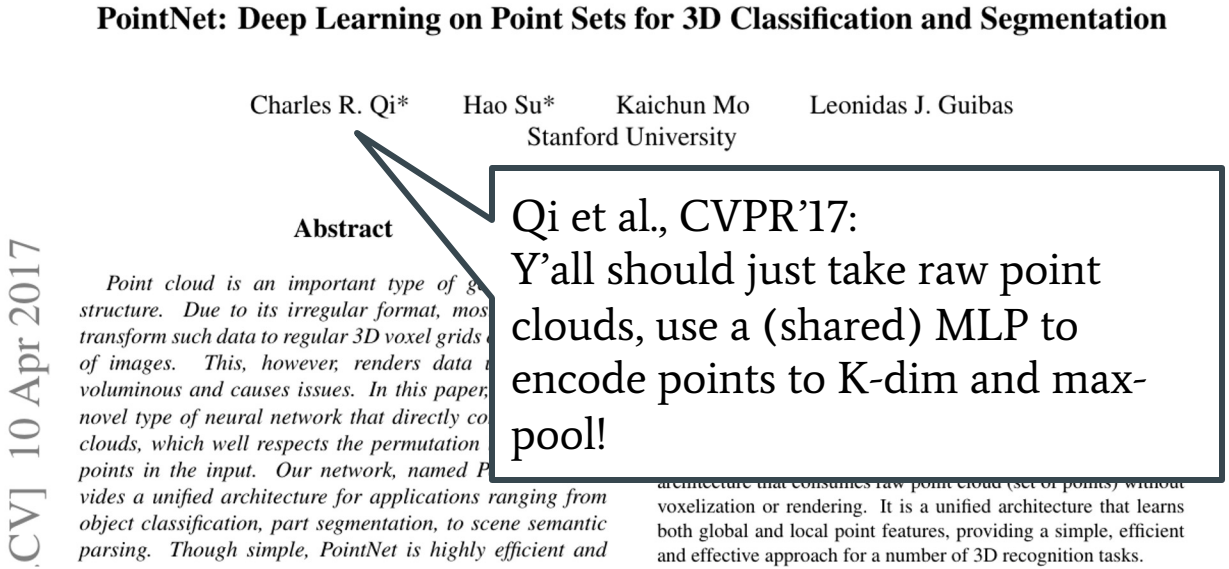
Mesh

Volumetric

Projected View RGB(D)

Slides adapted from Charles Qi CVPR presentation slides (https://web.stanford.edu/~rqi/pointnet/docs/cvpr17_pointnet_slides.pdf)

Deep Learning on Unordered Sets



PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

Charles R. Qi* Hao Su* Kaichun Mo Leonidas J. Guibas
Stanford University

Abstract

Point cloud is an important type of g structure. Due to its irregular format, mos transform such data to regular 3D voxel grids of images. This, however, renders data voluminous and causes issues. In this paper, novel type of neural network that directly co clouds, which well respects the permutation points in the input. Our network, named P vides a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing. Though simple, PointNet is highly efficient and

voxelization or rendering. It is a unified architecture that learns both global and local point features, providing a simple, efficient and effective approach for a number of 3D recognition tasks.

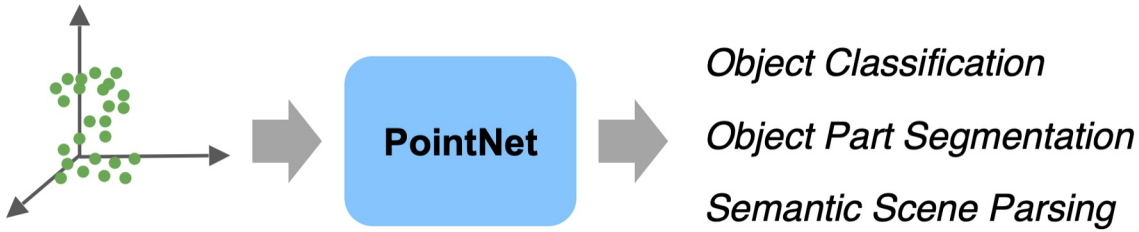
CVJ 10 Apr 2017

Qi et al., CVPR'17:
Y'all should just take raw point clouds, use a (shared) MLP to encode points to K-dim and max-pool!

- Seminal paper by Qi et al., CVPR'17
- Game-changer

Deep Learning on Point Clouds

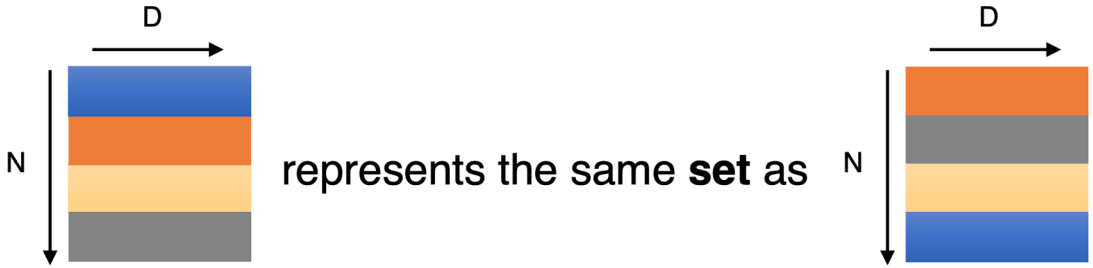
- End-to-end learning for unordered point data



- Challenges:
  - Unordered: need repr. invariant to N! permutations!
  - Invariance under geometric transformations:
    - Rotation/translation should not alter classification results!

Slides adapted from Charles Qi CVPR presentation slides (https://web.stanford.edu/~rqi/pointnet/docs/cvpr17_pointnet_slides.pdf)

Permutation Invariance



$$f(x_1, x_2, \ldots, x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}), \quad x_i \in \mathbb{R}^D$$

Examples:

$$f(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$$
$$f(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$
$$\ldots$$

- How can we construct a family of symmetric functions by neural networks?

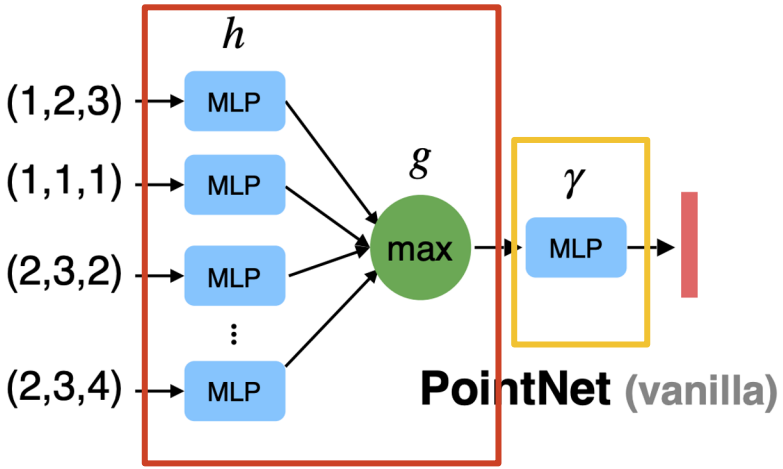
Slides adapted from Charles Qi CVPR presentation slides (https://web.stanford.edu/~rqi/pointnet/docs/cvpr17_pointnet_slides.pdf)

Vanilla PointNet

- Observe:

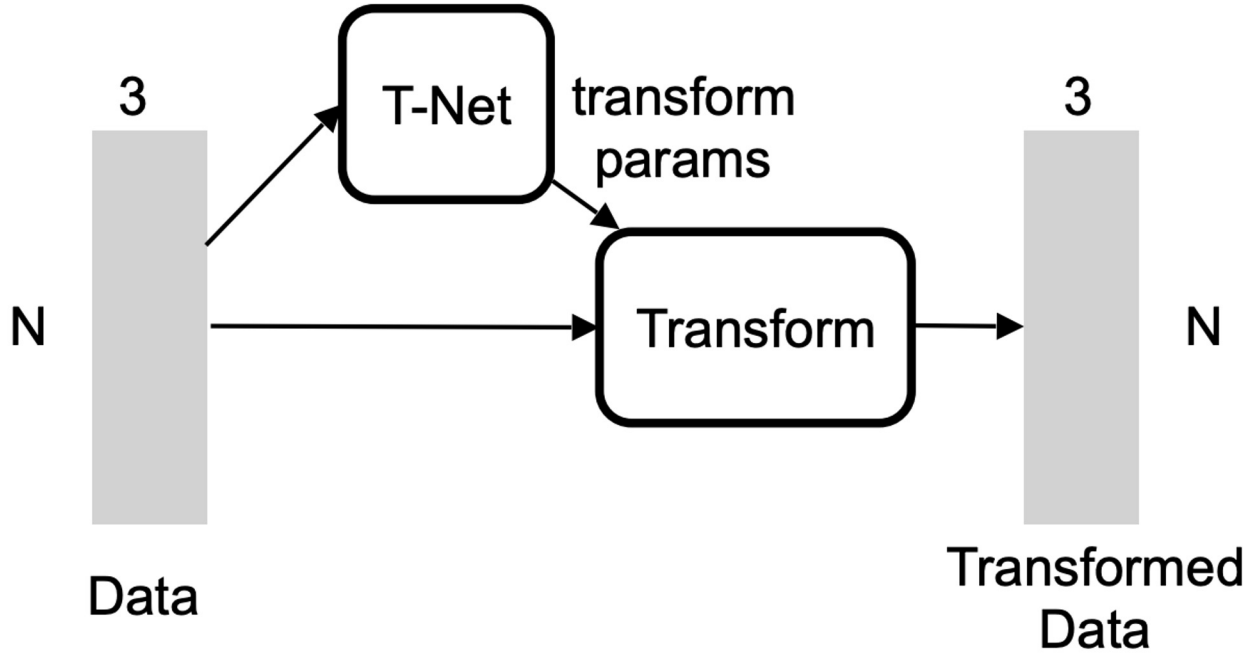
$f(x_1, x_2, \ldots, x_n) = \gamma \circ g(h(x_1), \ldots, h(x_n))$ is symmetric if g is symmetric

- PointNet: MLP + max pooling



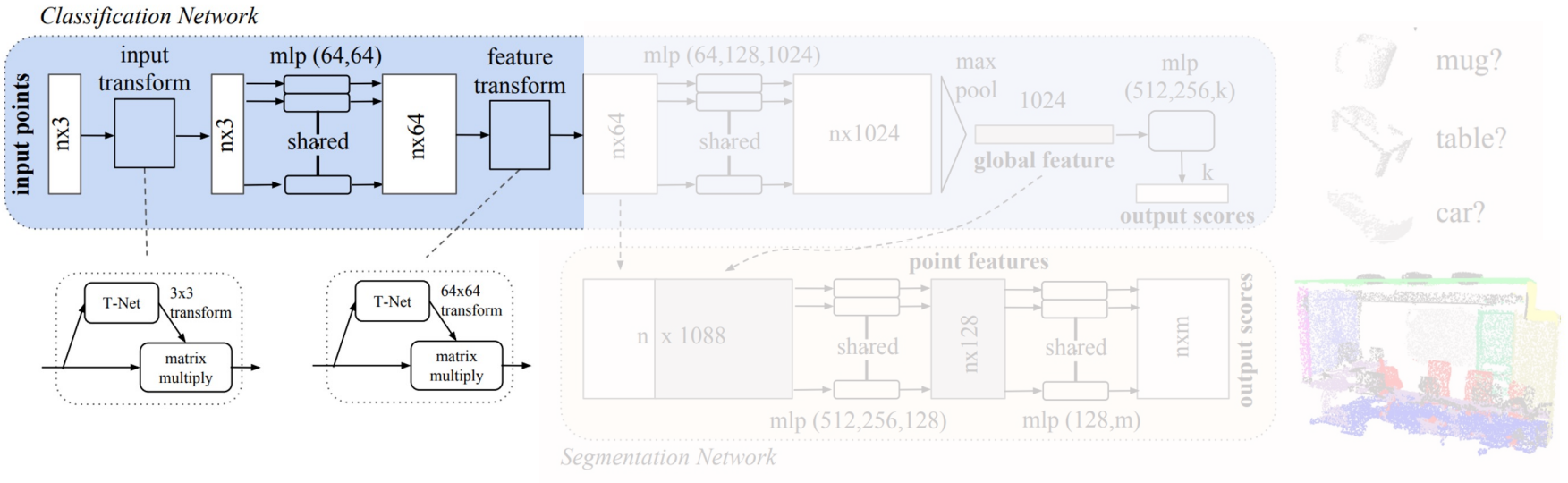
Slides adapted from Charles Qi CVPR presentation slides (https://web.stanford.edu/~rqi/pointnet/docs/cvpr17_pointnet_slides.pdf)

Invariance to Transformations



Slides adapted from Charles Qi CVPR presentation slides (https://web.stanford.edu/~rqi/pointnet/docs/cvpr17_pointnet_slides.pdf)

Network Architecture



Kernel Point Convolution: KPConv

KPConv: Flexible and Deformable Convolution for Point Clouds

Hugues Thomas[1] Charles R. Qi[2] Jean-Emmanuel Deschaud[1] Beatriz Marcotegui[1]
François Goulette[1] Leonidas J. Guibas[2,3]

[1]Mines ParisTech [2]Facebook AI Research [3]Stanford University

Abstract

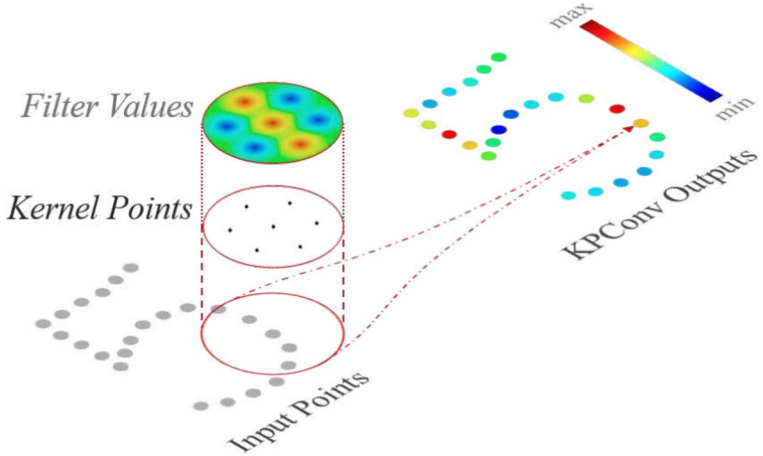
We present Kernel Point Convolution[1] (KPConv), a new design of point convolution, i.e. that operates on point clouds without any intermediate representation. The convolution weights of KPConv are located in Euclidean space by kernel points, and applied to the input points close to them. Its capacity to use any number of kernel points gives KPConv more flexibility than fixed grid convolutions. Furthermore, these locations are continuous in space and can be learned by the network. Therefore, KPConv can be extended to deformable convolutions that learn to adapt kernel points to local geometry. Thanks to a regular subsampling strategy, KPConv is also efficient and robust to varying densities. Whether they use deformable KPConv for complex tasks, or rigid KPconv for simpler tasks, our networks outperform state-of-the-art classification and segmentation approaches

are coupled with corresponding features like colors. In this work, we will always consider a point cloud as those two elements: the points $\mathcal{P} \in \mathbb{R}^{N \times 3}$ and the features $\mathcal{F} \in \mathbb{R}^{N \times D}$. Such a point cloud is a *sparse* structure that has the property to be *unordered*, which makes it very different from a grid. However, it shares a common property with a grid which is essential to the definition of convolutions: it is *spatially localized*. In a grid, the features are localized by their index in a matrix, while in a point cloud, they are localized by their corresponding point coordinates. Thus, the points are to be considered as structural elements, and the features as the real data.

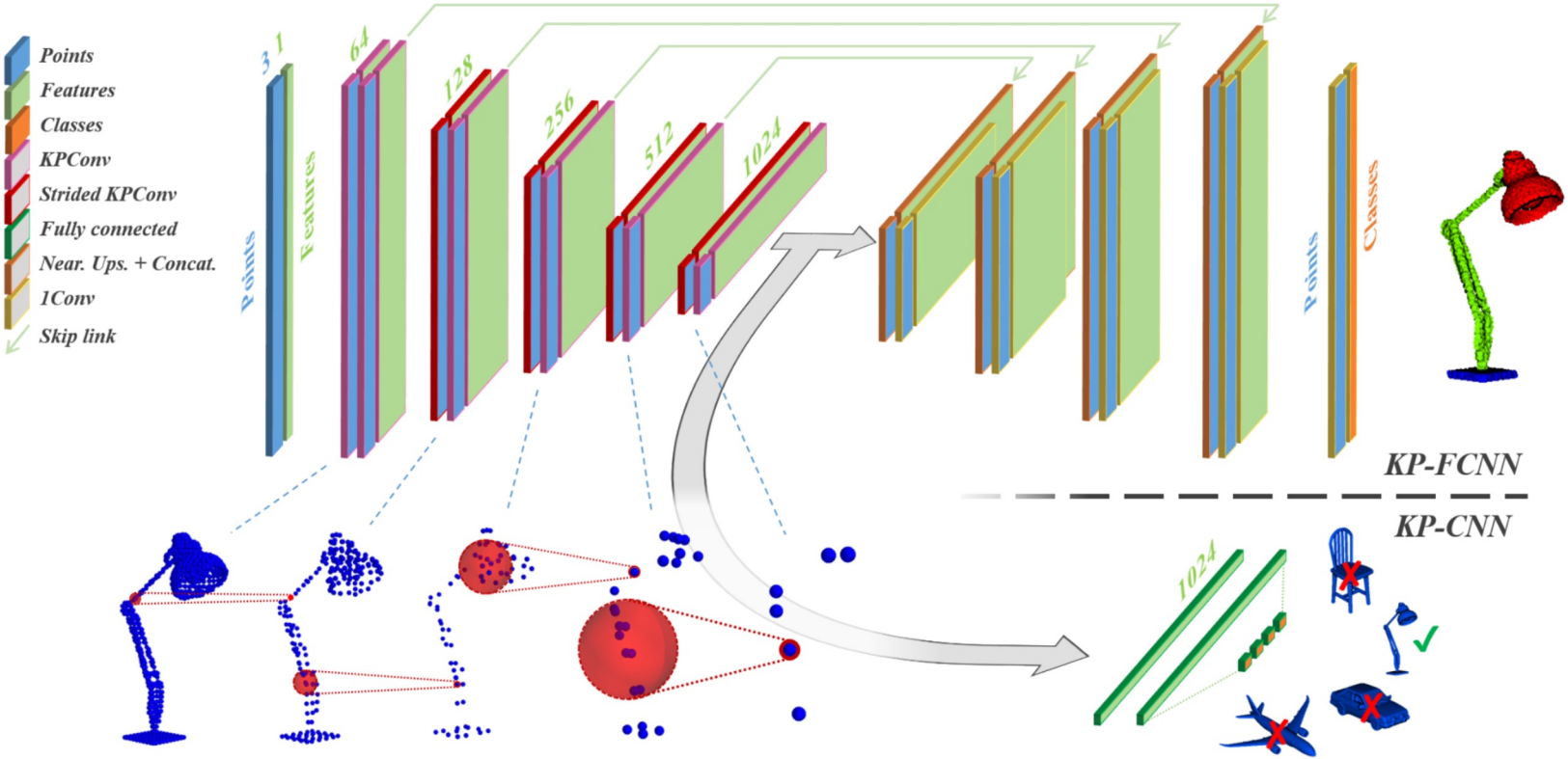
Various approaches have been proposed to handle such data, and can be grouped into different categories that we will develop in the related work section. Several methods fall into the grid-based category, whose principle is to project the sparse 3D data on a regular structure where a

Kernel Point Convolution: KPConv

- Kernel Point Convolution
  - Convolve input points with KPConv (radius-nbhd)
  - KPConv: kernel points + (learnable) weights



Network Architecture



Deep Learning on Point Clouds

- Signal representation?

Point Cloud

Mesh

Volumetric

Projected View RGB(D)

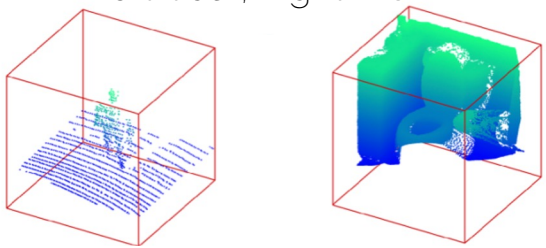
Slides adapted from Charles Qi CVPR presentation slides (https://web.stanford.edu/~rqi/pointnet/docs/cvpr17_pointnet_slides.pdf)

Sparse 3D Convolutions

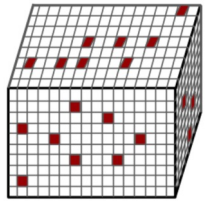
VoxelNet: point cloud -> occupancy map -> 3D conv

- Pros:
  - Convolutional networks are effective and *well-consolidated*!
- Cons:
  - Expensive!
  - Observe:
    - 3D space is "mostly empty"
    - Lidar: sparse!
- Can we get *best* of both worlds?

Left: lidar; Right: RGB-D



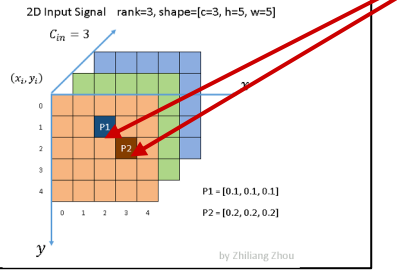
Example sparse 3D tensor



Signal Representation

- Regular image signals: dense *matrices*, *tensors*
  - Conv. op.: dense matrix multiplication
- Sparse signals: *data list* + index list

5x5 image with three channels, 5x5x3 tensor



Active sites

Sparse form:
- Data list is `[[0.1, 0.1, 0.1], [0.2, 0.2, 0.2]]`
- Index list is `[[1,2], [2, 3]]`
- YX order

- Conv. op. that leverages such a sparse representation to save computation?

Figure credits: Zhiliang Zhou

First Try: Sparse Convolutions

Conv. kernel:

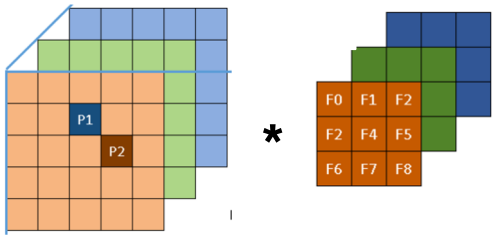
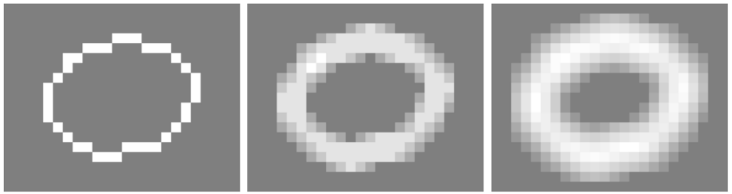


Figure credits: Zhiliang Zhou

Output (data/index list):

- Sparse convolutions: as 2D, but skip "empty space"!
  - Issue: submanifold dilation:



M. Engelcke et al., Vote3Deep: Fast Object Detection in 3D Point Clouds using Efficient Convolutional Neural Networks. ICRA'17.
B. Graham. Sparse 3D Convolutional Neural Networks. BMVC'15.

Submanifold Convolutions!

Conv. kernel:

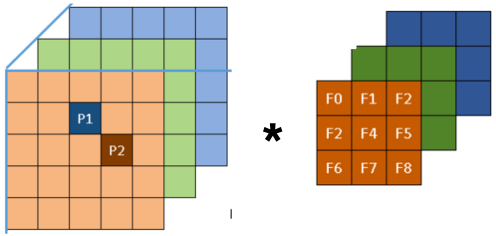
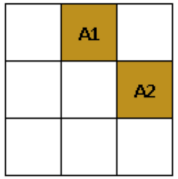


Figure credits: Zhiliang Zhou

Output (data/index list):

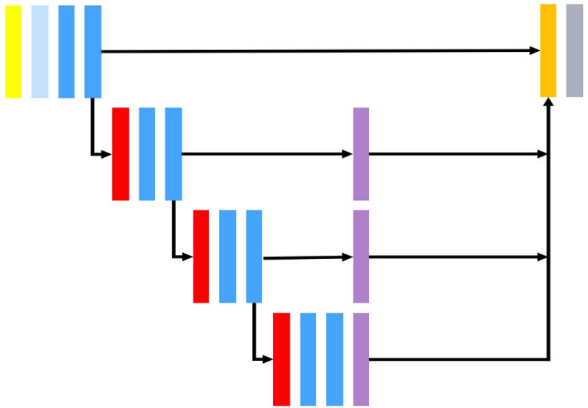


- Submanifold convolutions: *Do not increase active sites!*
- Output: kernel *must cover* input site!
  - Maintains sparsity!

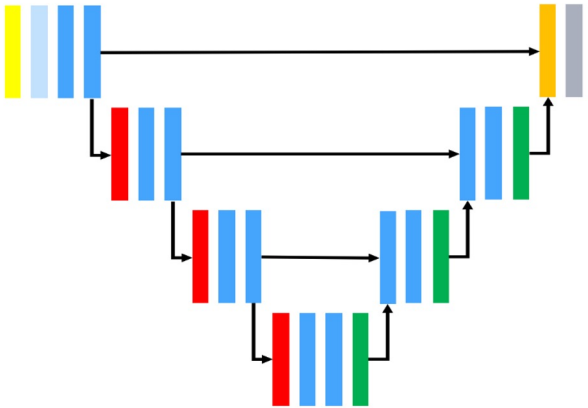
Graham et al, 3d semantic segmentation with submanifold sparse convolutional networks. CVPR'18

Network Architecture

- SSCN networks: FCN, U-Net
  - Combo of SSC and strided SC layers



(a) Submanifold sparse FCN.



(b) Submanifold sparse U-Net.

Applied to "active" voxels

Graham et al, 3d semantic segmentation with submanifold sparse convolutional networks. CVPR'18

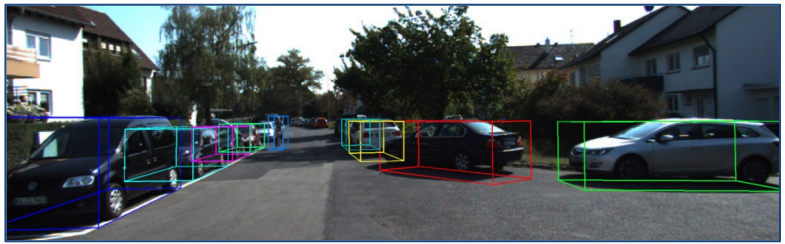
3D Computer Vision in the Era of Deep Learning

Object Detection and Tracking

Datasets

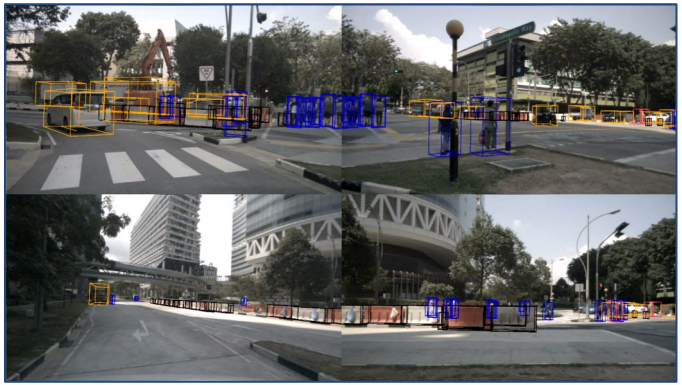
KITTI [Geiger et. al., CVPR'12]

- 64 beam lidar, 10 Hz
- Karlsruhe, Germany
- 2 classes



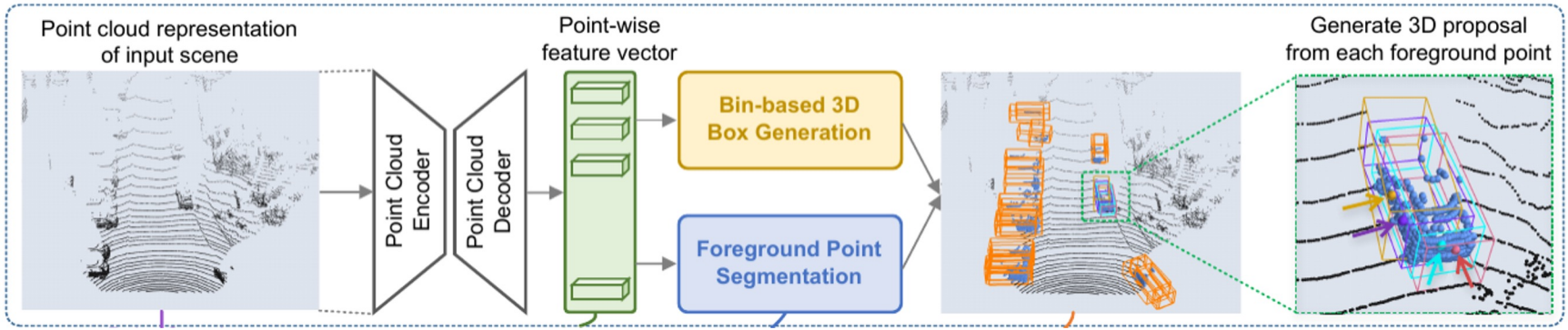
nuScenes [Caesar et al., CVPR'20]

- 32 beam lidar, 2 Hz
- Boston, MA; Singapore
- 7 classes



Object Detection: Point RCNN

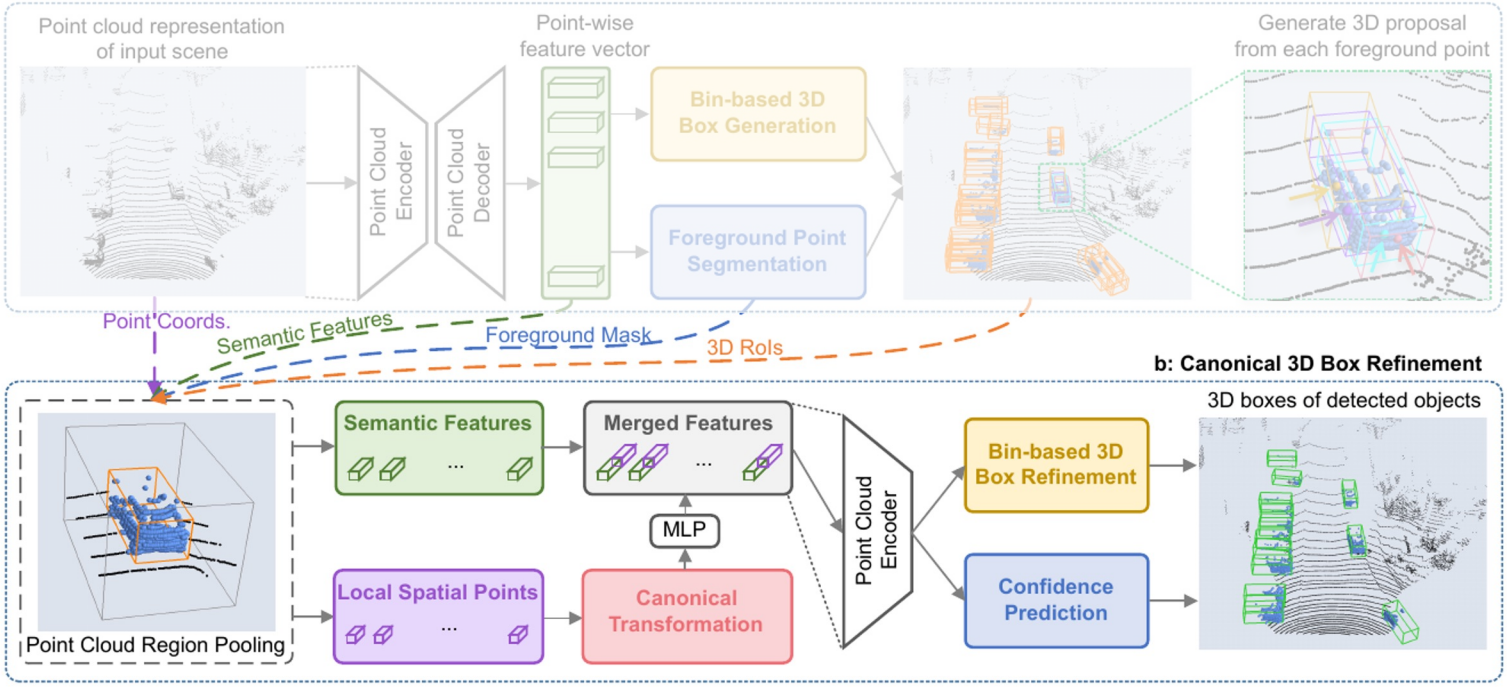
- Two-stage detector (Faster R-CNN!)
- Stage-I: proposal generation



Shi et al., PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud, CVPR'19

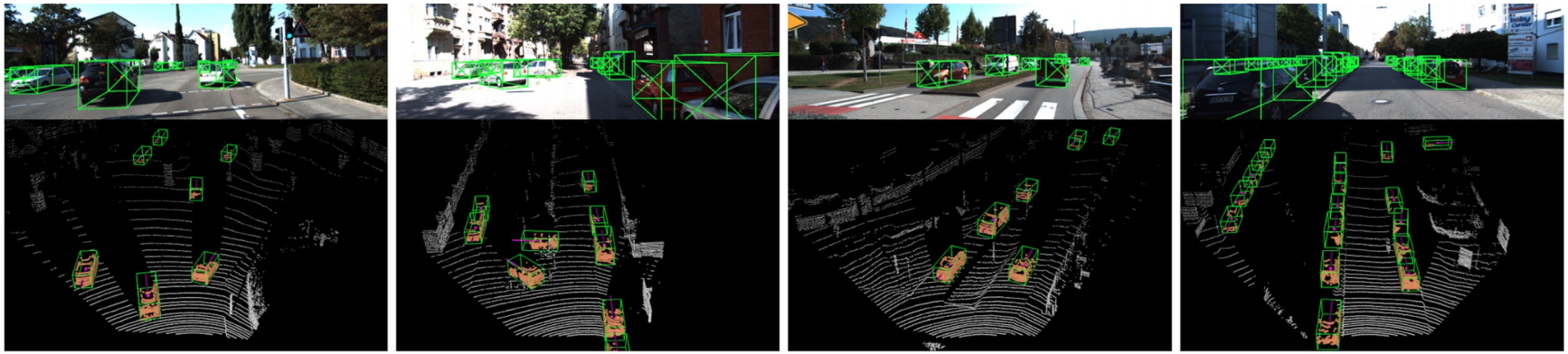
Object Detection: Point RCNN

- Stage-II



Shi et al., PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud, CVPR'19

Point RCNN



Method	Modality	Car (IoU=0.7)			Pedestrian (IoU=0.5)			Cyclist (IoU=0.5)		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D [4]	RGB + LiDAR	71.09	62.35	55.12	-	-	-	-	-	-
UberATG-ContFuse [17]	RGB + LiDAR	82.54	66.22	64.04	-	-	-	-	-	-
AVOD-FPN [14]	RGB + LiDAR	81.94	71.88	66.38	50.80	42.81	**40.88**	64.00	52.18	46.61
F-PointNet [25]	RGB + LiDAR	81.20	70.39	62.19	**51.21**	**44.89**	40.23	71.96	56.77	50.39
VoxelNet [43]	LiDAR	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
SECOND [40]	LiDAR	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
Ours	LiDAR	**85.94**	**75.76**	**68.32**	49.43	41.78	38.63	**73.93**	**59.60**	**53.59**

Shi et al., PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud, CVPR'19

3D Multi-Object Tracking

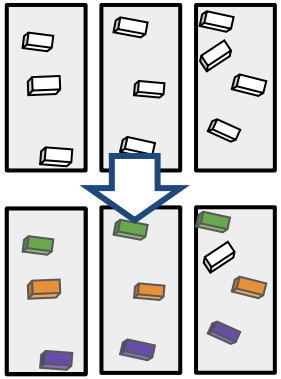
In:

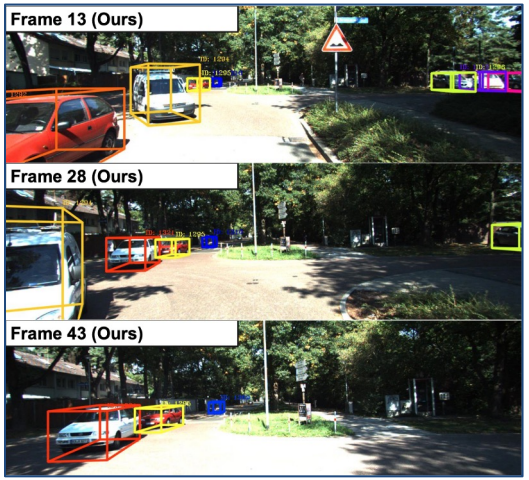
Object detections ~ 3D boxes

=> position + orientation

Out:

Temporally linked detections



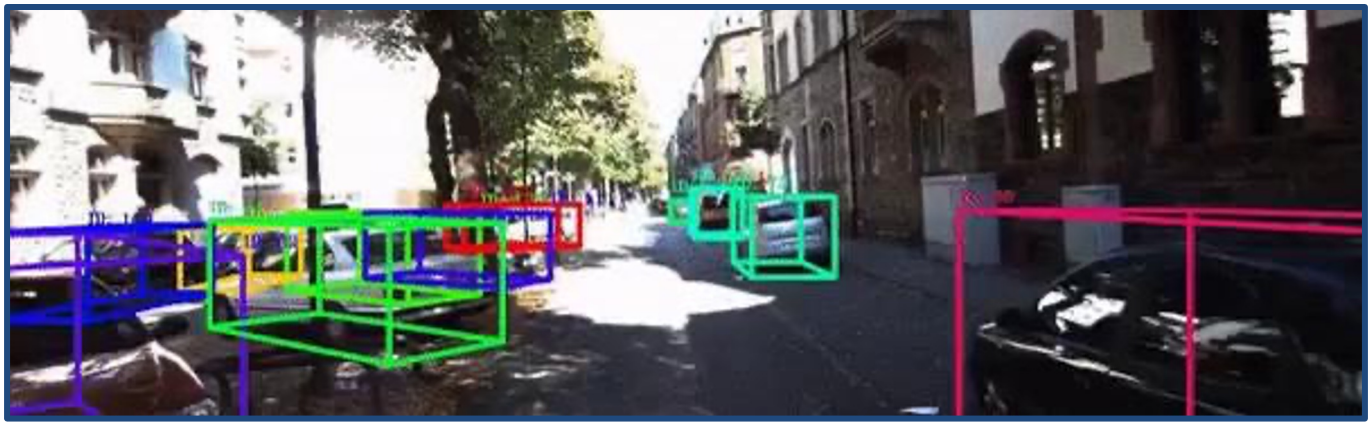


Reminder: Tracking-by-detection

- Train object detectors for target classes
- Detect + associate over time
- Challenges: noisy detections, occlusions!

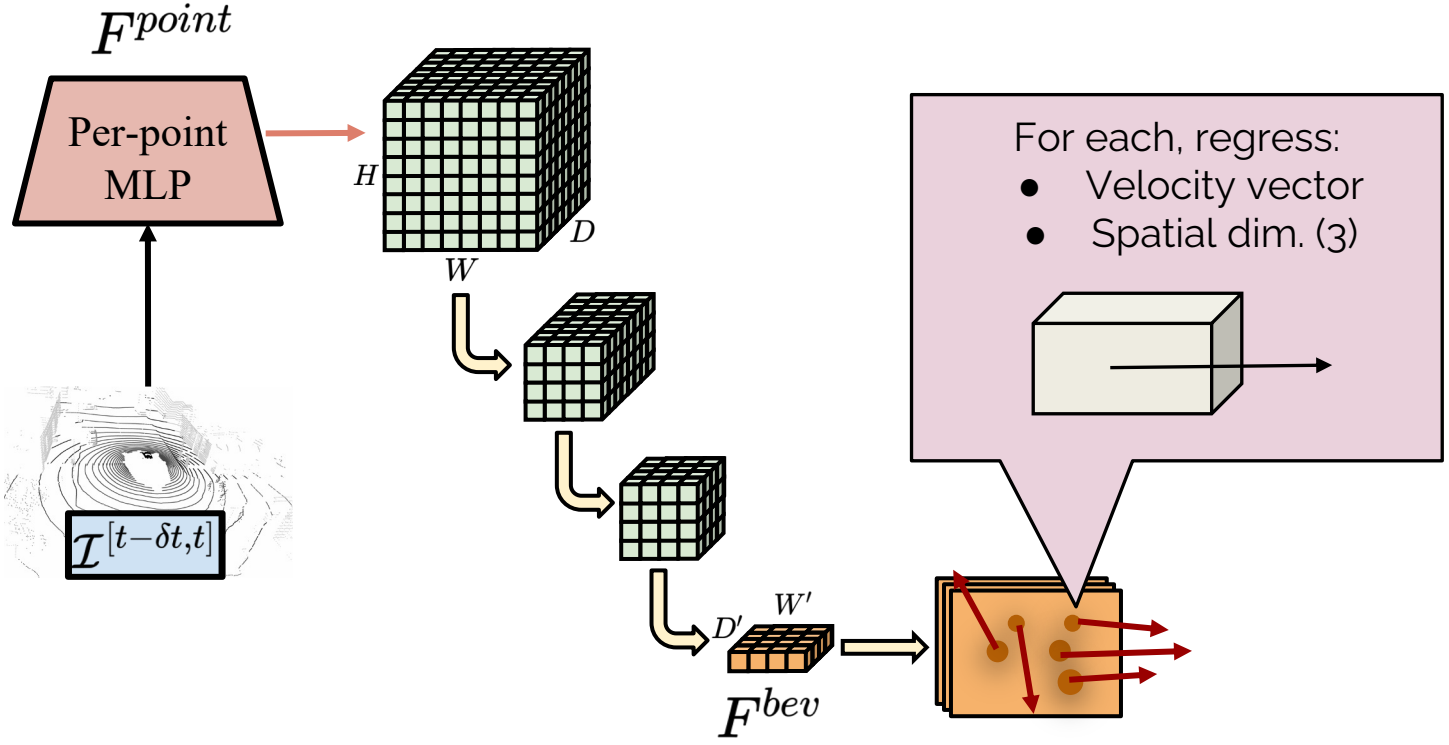
Object Tracking: AB3D-MOT

- ``Embarrassingly simple``, great performance!
  - Dynamics model: const-velocity ***Kalman Filter***
  - ***Hungarian algorithm***, ***3D IoU*** as matching cost
  - Why does this simple approach work so well in this case?



Weng et al., A Baseline for 3D Multi-Object Tracking, IROS'20

End-to-End Detection + Tracking

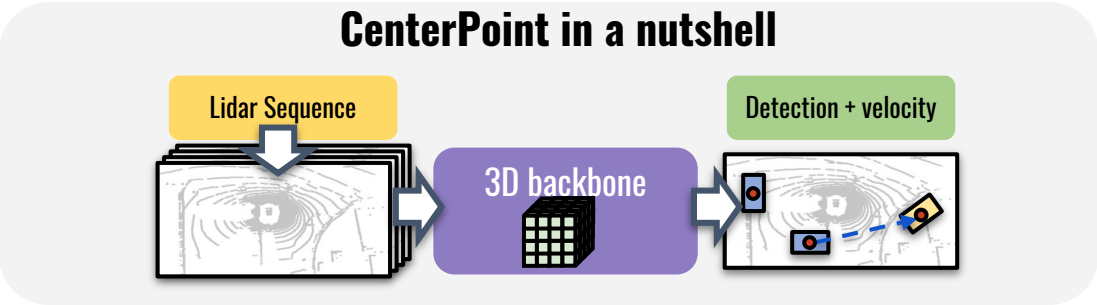


T. Yin et al., Center-based 3D object detection and tracking from point clouds, CVPR'21

Summary So Far

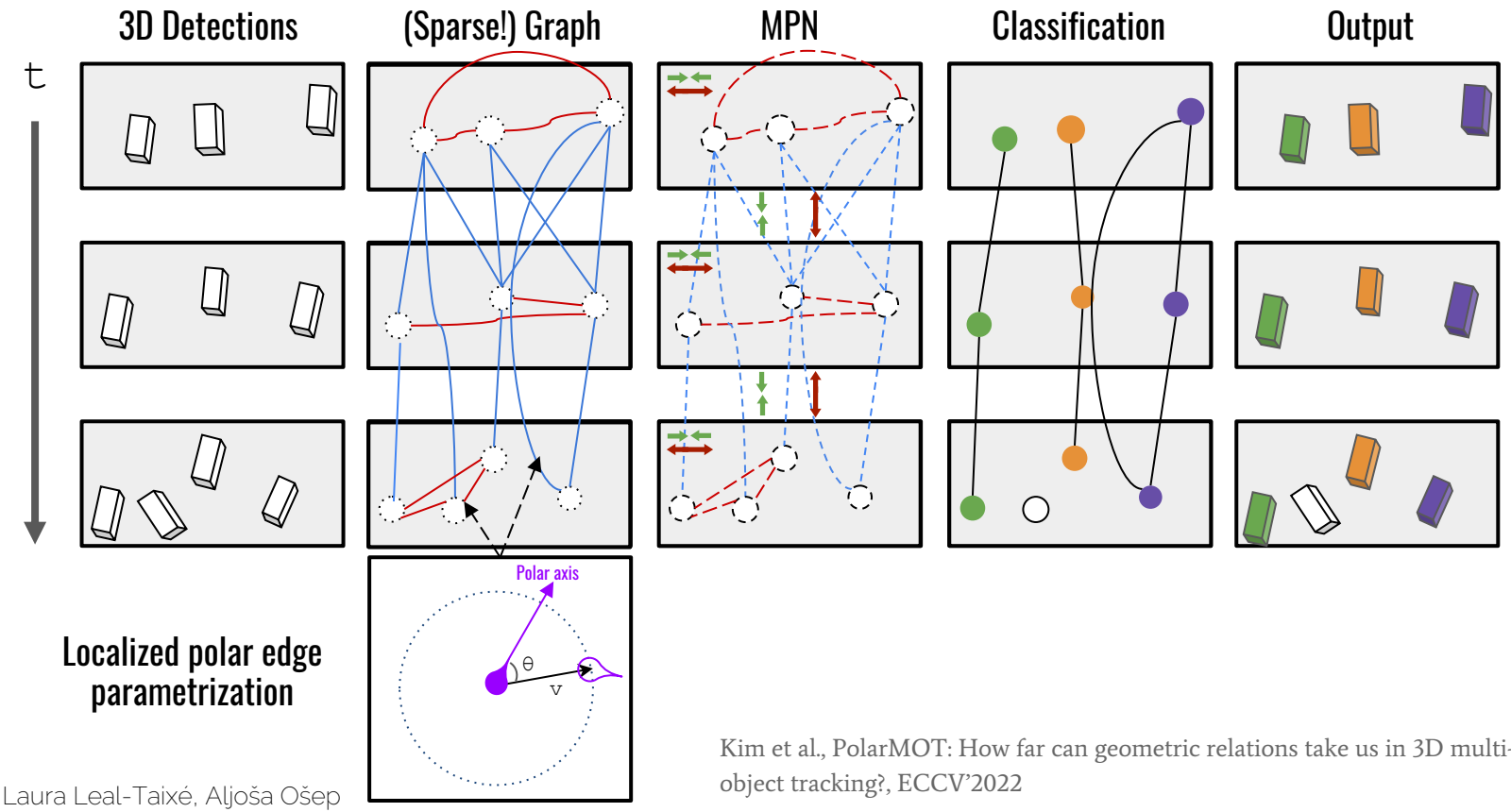
Simple (yet effective!) approaches:

- Detect + Kalman filter (Weng et al, IROS'20),
- Detect + regress e2e (CenterPoint, Yin et al., CVPR'21)
- Associate (Greedy, Hungarian; 3D IoU or Euclid. dist.)



- Modeling long-range geometric relations?

PolarMOT: Graph Neural Networks!



Kim et al., PolarMOT: How far can geometric relations take us in 3D multi-object tracking?, ECCV'2022

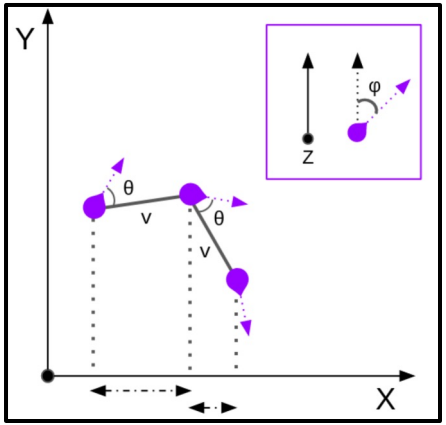
How to Parametrize?

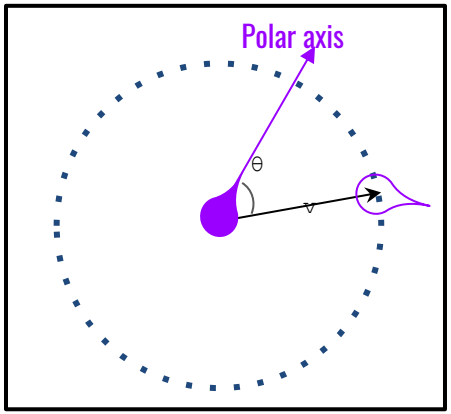
- Cartesian coordinates?
  - Works but ...

$$h_{i,j}^{(0)} = \Delta(o_i, o_j) = \begin{bmatrix} \Delta x \\ \Delta y \\ \cdots \end{bmatrix} = \begin{bmatrix} o_i^x - o_j^x \\ o_i^y - o_j^y \\ \cdots \end{bmatrix}$$

- Localized polar coordinates!
  - Detections ~ two *oriented* points
    - Velocity magnitude
    - Polar angle
- Why?
  - Invariant to global ref. frame
  - Non-holonomic motion prior

$$h_{i,j}^{(0)} = \Delta(o_i, o_j) = \begin{bmatrix} v \\ \theta_{i,j} \\ \Delta\phi \\ \Delta t \end{bmatrix} = \begin{bmatrix} \frac{\|o_i - o_j\|_2}{\Delta t} \\ \angle\left(\vec{o}_i^{\,head}, \vec{o}_j - \vec{o}_i\right) \\ o_i^{\phi} - o_j^{\phi} \\ o_i^{t} - o_j^{t} \end{bmatrix}$$





PolarMOT: Key Results

Online vs. offline

- Our method uses CenterPoint 3D detections as input

Method name	Input modality	IDs ↓ total	Recall ↑ average	AMOTA ↑ average	class-specific AMOTA ↑ car	ped	bicycle	bus	motor	trailer	truck
Ours *offl.*	3D	**213**	75.14	**71.14**	**85.83**	**81.70**	54.10	**87.36**	72.32	48.67	**68.03**
Ours *onl.*	3D	439	72.46	67.27	81.26	78.79	49.38	82.76	67.19	45.80	65.70
CenterPoint *onl.*	3D	562	70.62	65.91	84.23	77.29	43.70	80.16	59.16	**51.47**	65.39

- Offline (batch) variant is a top-performer (*online:* same model, different graph construction)
- Ablations on (sparse) online graph construction => extra slides!

$$\text{MOTA} = 1 - \frac{\text{FP} + \text{FN} + \text{IDS}}{\text{num}_{\text{gt}}}$$

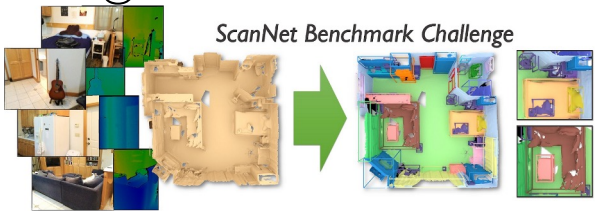
Kim et al., PolarMOT: How far can geometric relations take us in 3D multi-object tracking?, ECCV'2022

3D Computer Vision in the Era of Deep Learning

4D Segmentation

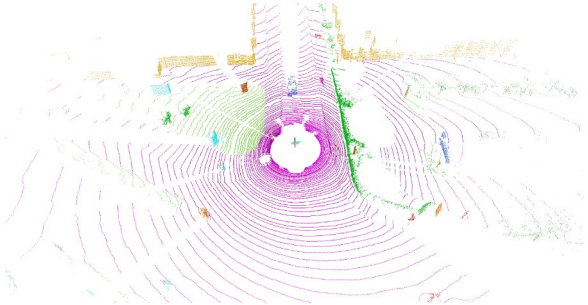
3D Semantic Segmentation

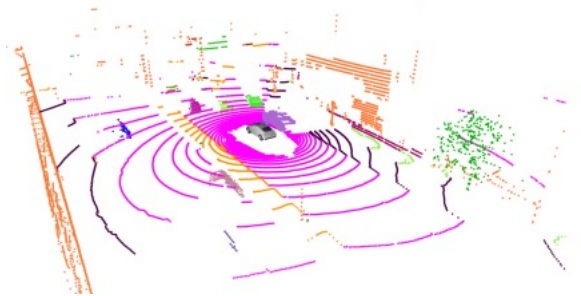
- Existing datasets (Dense, pre-aligned RGB-D)



Dai et al., ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes, CVPR'17

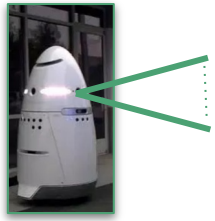
- How about sparse LiDAR scans?

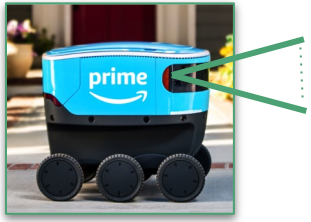


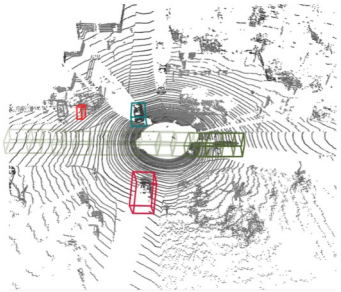


Behley et al., SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences, ICCV'19
Fong et al.,. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking, IEEE RAL 2022

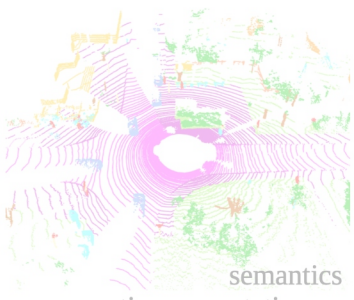
3D Vision

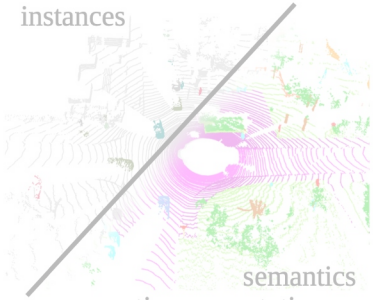


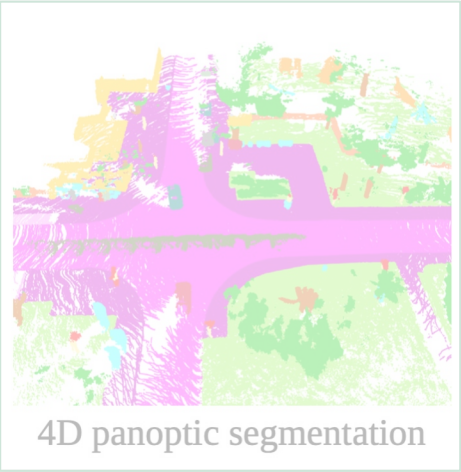




multi-object tracking







4D panoptic segmentation

4D Lidar Panoptic Segmentation

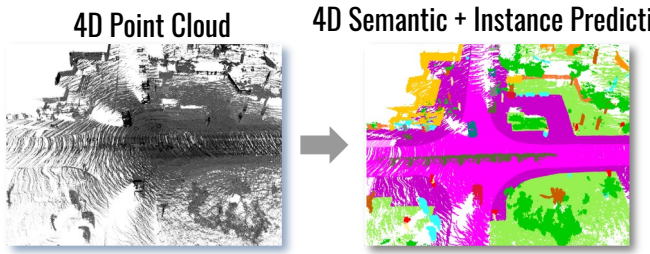
- *Input*: a point cloud sequence
- *Output*: per-point semantics + identity (space + time!)

- 1st benchmark!
  - (Lidar) segmentation and tracking quality (*point-centric*)

$$STQ = (AQ \times SQ)^{\frac{1}{2}}$$

Association Quality Segmentation Quality

- First models / baselines:
  - Detect + segment + track-by-det
  - End-to-end!

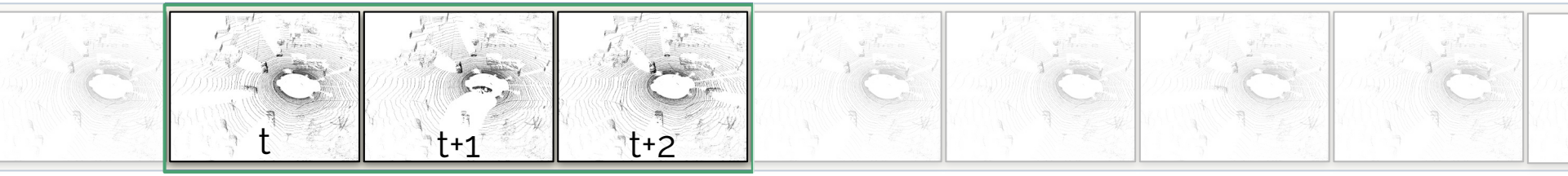


Ayguen et al., 4D Lidar Panoptic Segmentation, CVPR'21

4D LiDAR Panoptic Segmentation

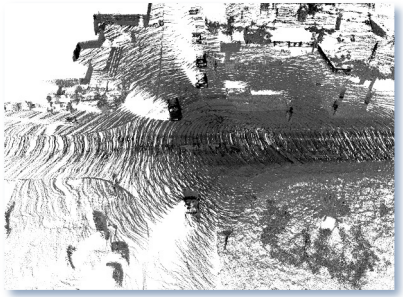
Ayguen et al., 4D Lidar Panoptic Segmentation, CVPR'21

4D Panoptic Lidar Segmentation



Point sampling

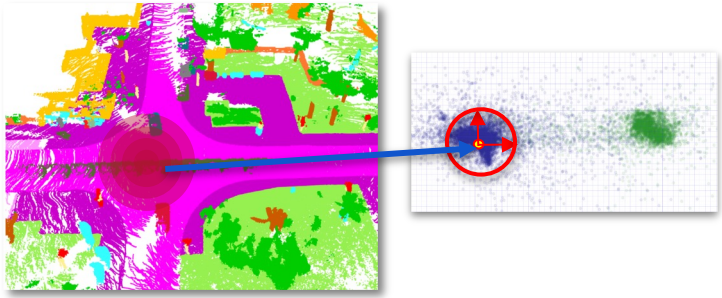
4D Point Cloud



Encoder-Decoder Network

S
O
Σ
ε

4D Semantic + Instance Predictions



S Semantic head

O Objectness head

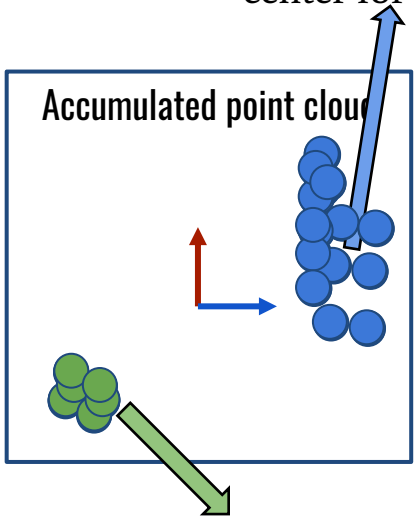
Σ Point variance head

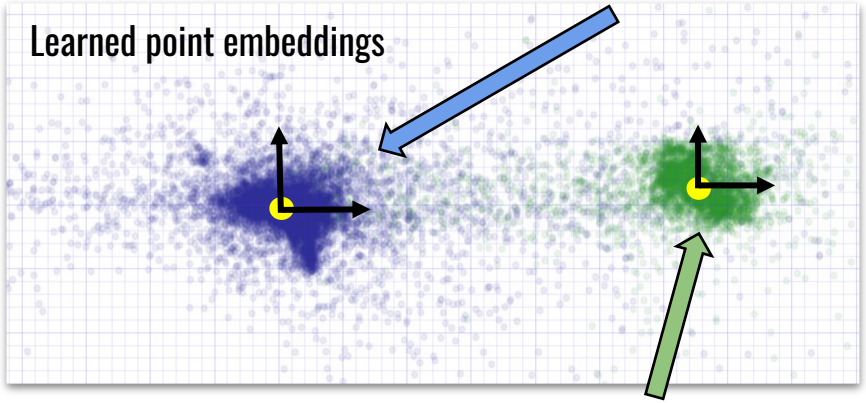
ε Point embeddings

Ayguen et al., 4D Lidar Panoptic Segmentation, CVPR'21

4D Panoptic Lidar Segmentation

center for object 1 (c_1) → $c_1 = (x_1, y_1, z_1, t_1)$ → μ_1 : lookup $\boldsymbol{\varepsilon}$-map at (x_1, y_1, t_1)
$\sigma_{1,x}$, $\sigma_{1,y}$: lookup $\boldsymbol{\Sigma}$-map at (x_1, y_1, t_1)





center for object 2 (c_2) → $c_2 = (x_2, y_2, z_2, t_2)$ → μ_2 : lookup $\boldsymbol{\varepsilon}$-map at (x_2, y_2, t_2)
$\sigma_{2,x}$, $\sigma_{2,y}$: lookup $\boldsymbol{\Sigma}$-map at (x_2, y_2, t_2)

Query point embedding

Variance

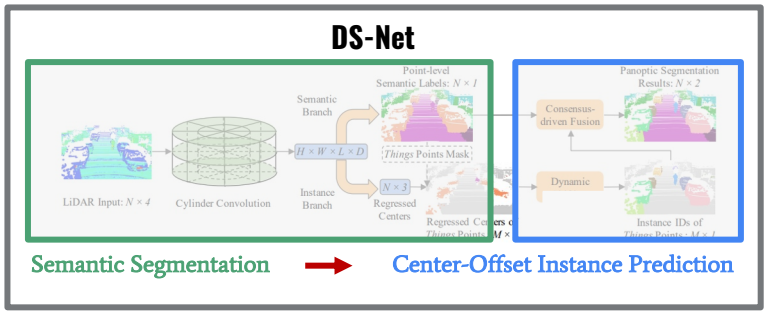
$$p_{ij} = \frac{1}{(2\pi)^{\frac{E}{2}} |\boldsymbol{\Sigma}_j|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{e}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{e}_i - \boldsymbol{\mu}_j)\right)$$

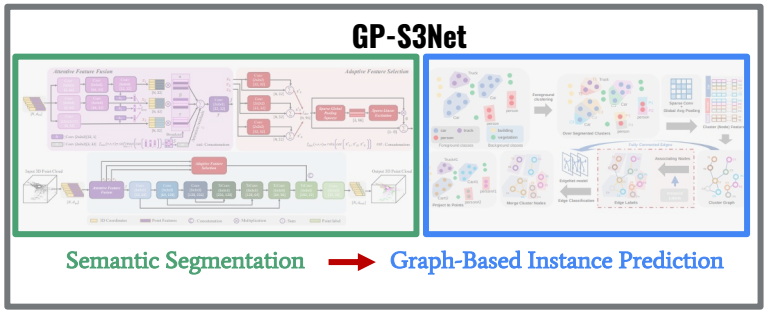
Instance mean embedding

Ayguen et al., 4D Lidar Panoptic Segmentation, CVPR'21

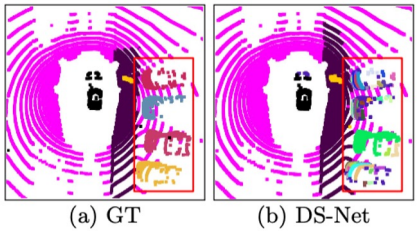
(4D) Panoptic Lidar Segmentation

- Lidar panoptic segmentation community: bottom-up
  - *Segmentation-centric* methods
  - Instance segmentation as *point-grouping*



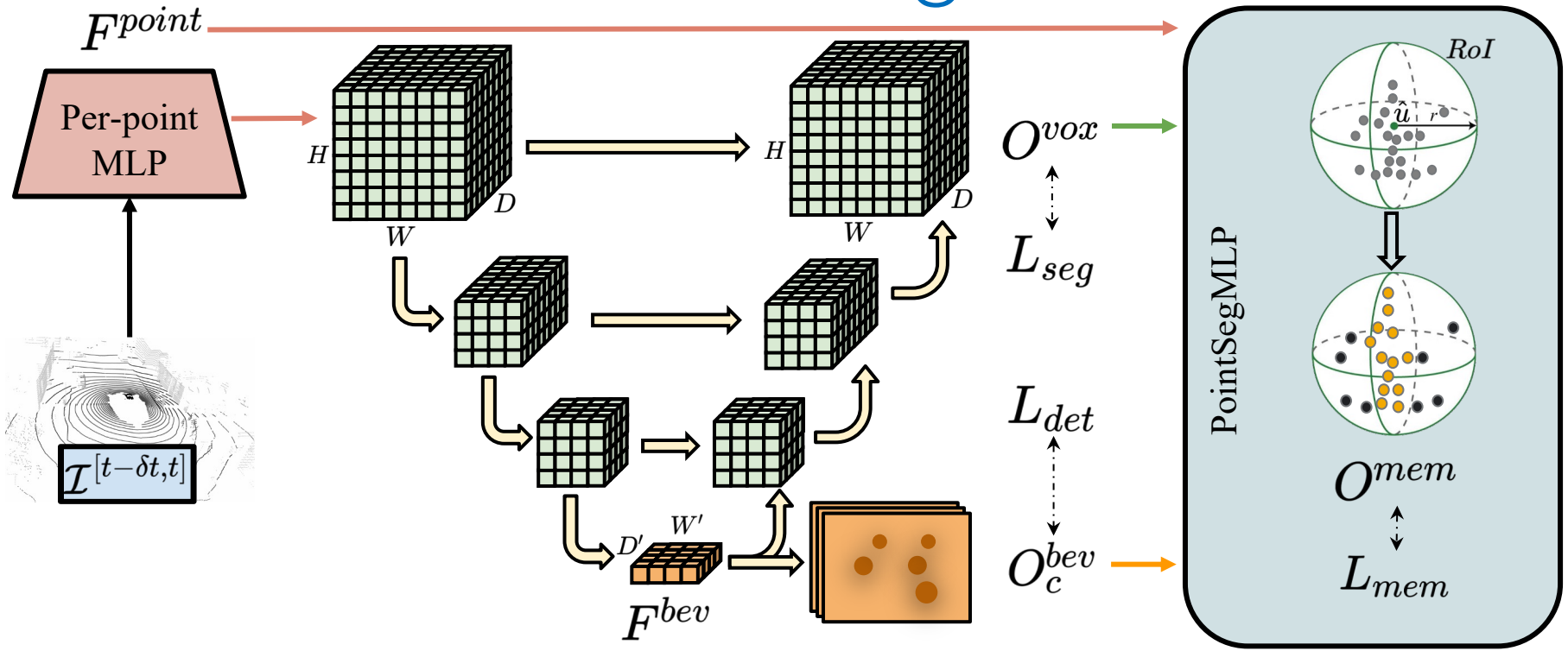


- Bottom-up approach pitfalls:
  - Instance prediction is very local
    => Over & under segmentation!



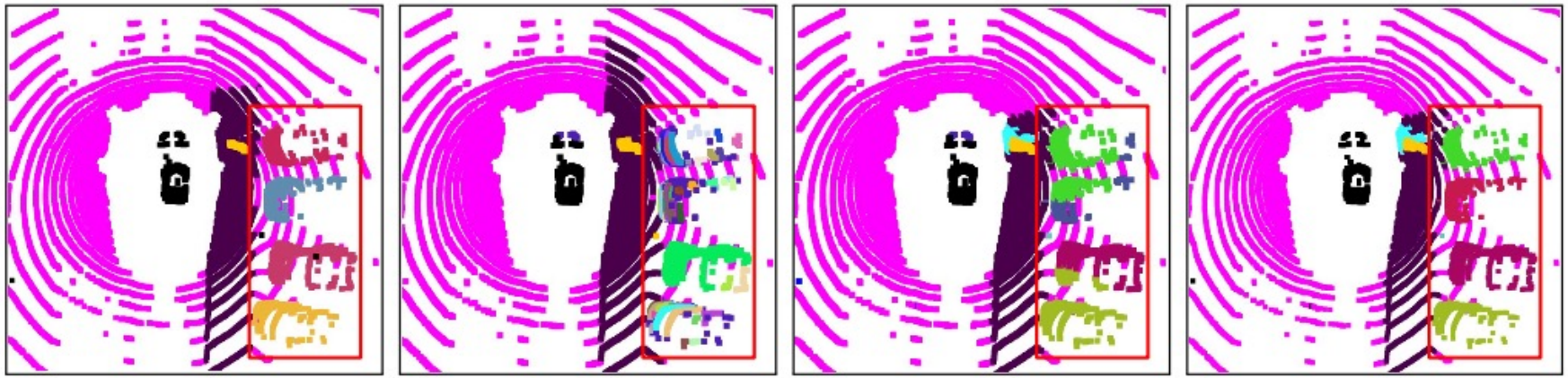
F. Hong, H. Zhou, X. Zhu, H. Li, Z. Liu: ***LiDAR-based Panoptic Segmentation via Dynamic Shifting Network,*** CVPR'21
R. Razani, R. Cheng, E. Li, E. Taghavi, Y. Ren, L. Bingbing: ***GP-S3Net: Graph-based Panoptic Sparse Semantic Segmentation Network,*** ICCV'21

MOST: Modal Segmentation and Tracking



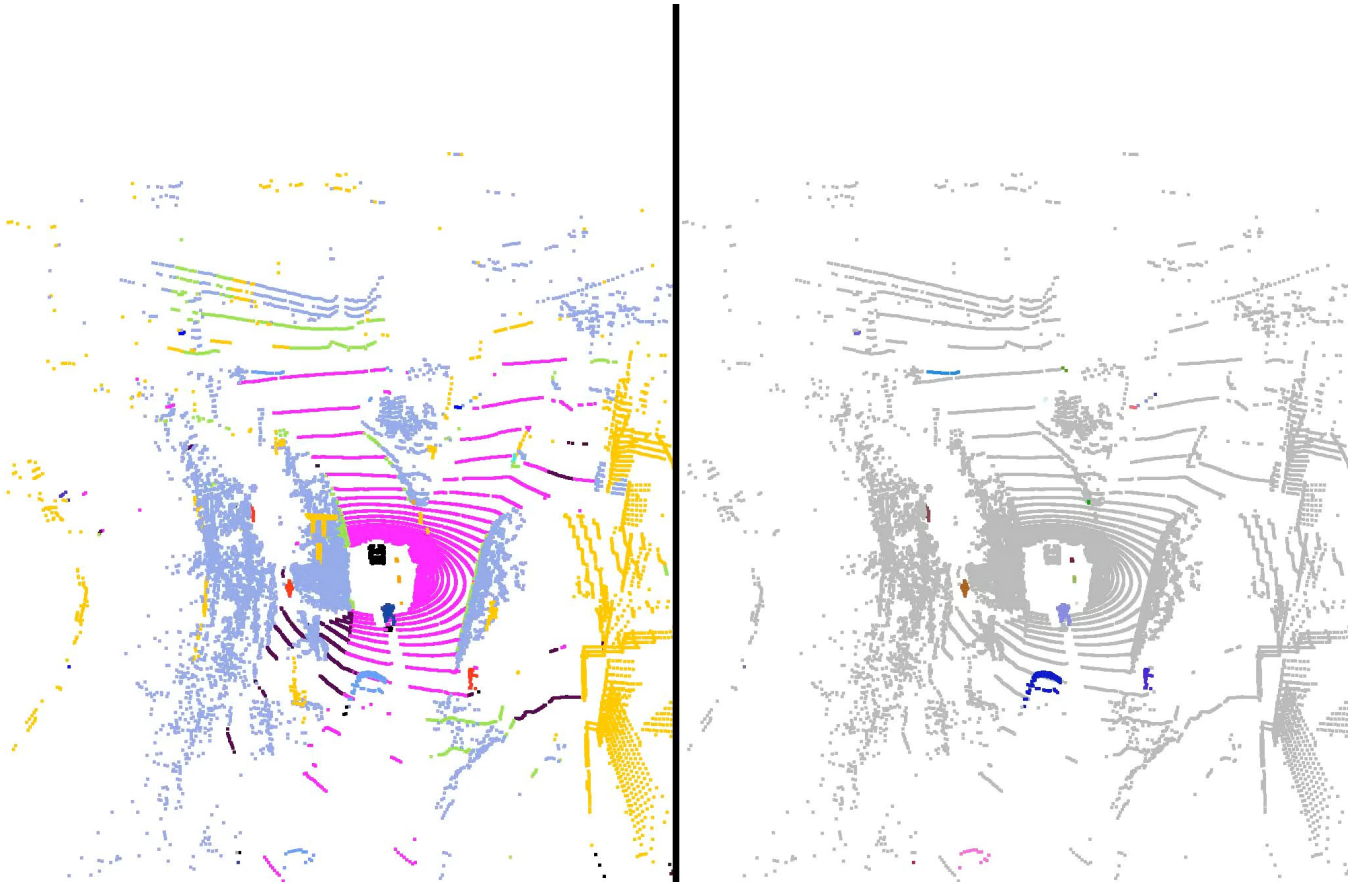
Agarwalla et al., Modal Recognition for Lidar Panoptic Segmentation and Tracking, arXiv 2022 (soon!)

MOST: Results



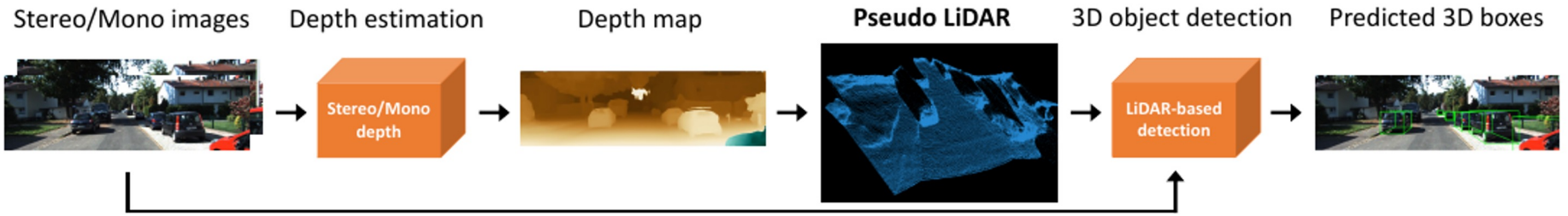
Agarwalla et al., Modal Recognition for Lidar Panoptic Segmentation and Tracking, arXiv 2022 (soon!)

MOST: Results



Agarwalla et al., Modal Recognition for Lidar Panoptic Segmentation and Tracking, arXiv 2022 (soon!)

Hey, How About Stereo?



Detection algorithm	Input signal	IoU = 0.5			IoU = 0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard
MONO3D [4]	Mono	30.5 / 25.2	22.4 / 18.2	19.2 / 15.5	5.2 / 2.5	5.2 / 2.3	4.1 / 2.3
MLF-MONO [33]	Mono	55.0 / 47.9	36.7 / 29.5	31.3 / 26.4	22.0 / 10.5	13.6 / 5.7	11.6 / 5.4
AVOD	Mono	61.2 / 57.0	45.4 / 42.8	38.3 / 36.3	33.7 / 19.5	24.6 / 17.2	20.1 / 16.2
F-POINTNET	Mono	70.8 / 66.3	49.4 / 42.3	42.7 / 38.5	40.6 / 28.2	26.3 / 18.5	22.9 / 16.4
3DOP [5]	Stereo	55.0 / 46.0	41.3 / 34.6	34.6 / 30.1	12.6 / 6.6	9.5 / 5.1	7.6 / 4.1
MLF-STEREO [33]	Stereo	-	53.7 / 47.4	-	-	19.5 / 9.8	-
AVOD	Stereo	89.0 / 88.5	77.5 / 76.4	68.7 / 61.2	74.9 / 61.9	56.8 / 45.3	49.0 / 39.0
F-POINTNET	Stereo	89.8 / 89.5	77.6 / 75.5	68.2 / 66.3	72.8 / 59.4	51.8 / 39.8	44.0 / 33.5
AVOD [17]	LiDAR + Mono	90.5 / 90.5	89.4 / 89.2	88.5 / 88.2	89.4 / 82.8	86.5 / 73.5	79.3 / 67.1
F-POINTNET [25]	LiDAR + Mono	96.2 / 96.1	89.7 / 89.3	86.8 / 86.2	88.1 / 82.6	82.2 / 68.8	74.0 / 62.0

Wang et al., Pseudo-LiDAR from Visual Depth Estimation, CVPR'19

Takeaways

- 3D vision: robots operate in 3D world! 3D scene understanding is crucial.
- Nowadays, we know how to **learn representations** from unstructured point clouds, yay!
  - => 3D object detection, semantic/instance segmentation, tracking!
- 3D detection/tracking/segmentation vibrant and **exciting** area of research!

Thank you for your attention!