SPATIAL RECOGNITION

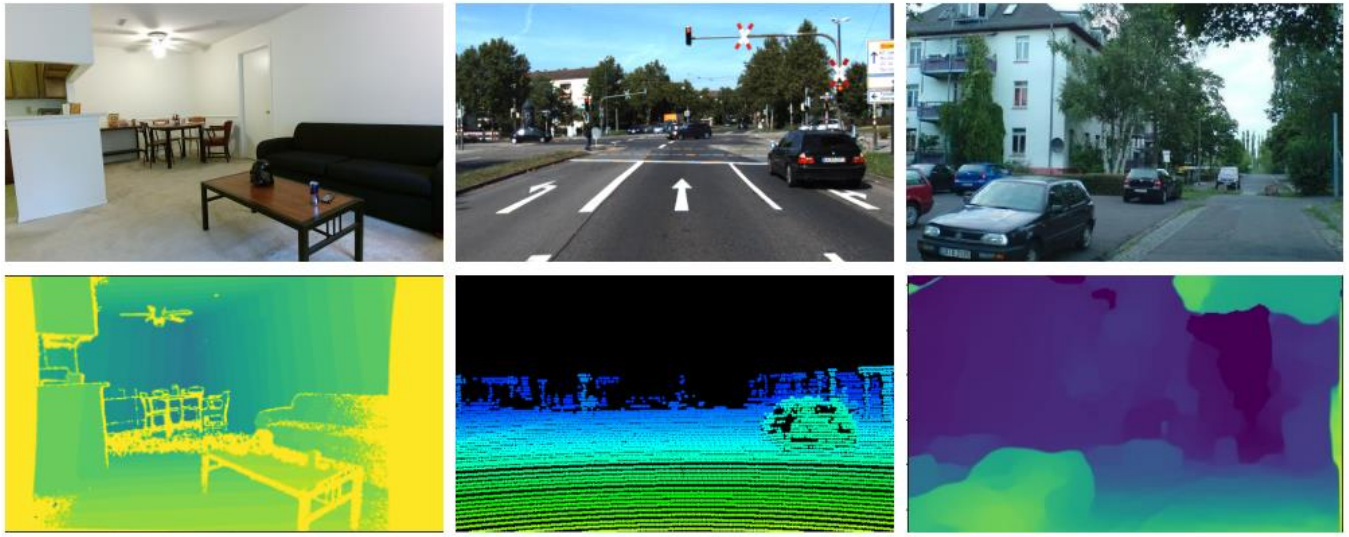
RGB-D AND POINT CLOUD

Dr TIAN Jing

tianjing@nus.edu.sg

3D data sensing

- AR / VR: For sensing real 3D environments and reconstructing them in the virtual world. (Project Tango)
- Robotics: For navigation, localization, mapping, and avoiding collision.
- Facial recognition: For improving convenience while preventing fraud.
- Gesture and proximity detection: For gaming, security. (Kinect, RealSense)

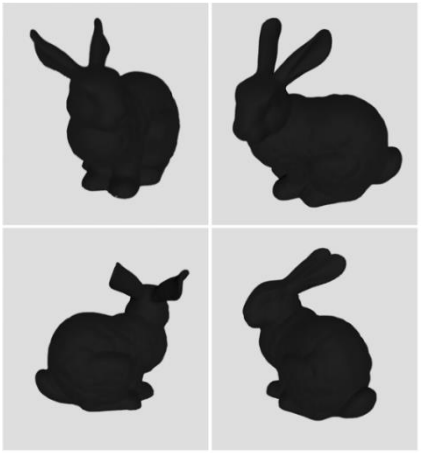


Time of Flight LiDAR Stereo camera

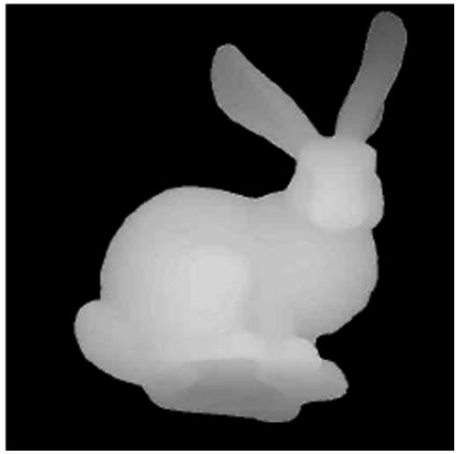
Reference
1. P. Ammirato, P. Poirson, E. Park, J. Kosecka, A. C. Berg, A Dataset for Developing and Benchmarking Active Vision, ICRA 2017.
2. A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision Meets Robotics: The KITTI Dataset, IJRR 2013.
3. K. Xian, C. Shen, Z. Cao, H. Lu, Y. Xiao, R. Li, Z. Luo, Monocular Relative Depth Perception With Web Stereo Data Supervision, CVPR 2018.

3D data representation

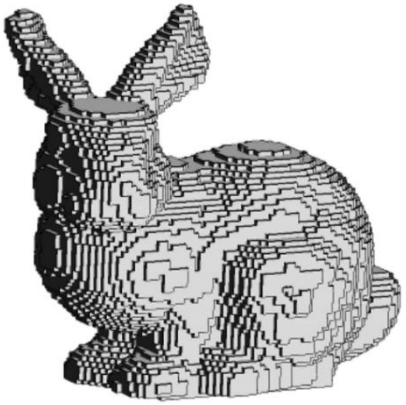
Rasterized form (regular grids)	Multi-view images	A 2D projected view
	Depth image	A 2.5D format
	Volumetric	A occupancy version of 3D points
Geometric form (irregular)	Polygon mesh	A surface version of 3D points
	Point cloud	3D points



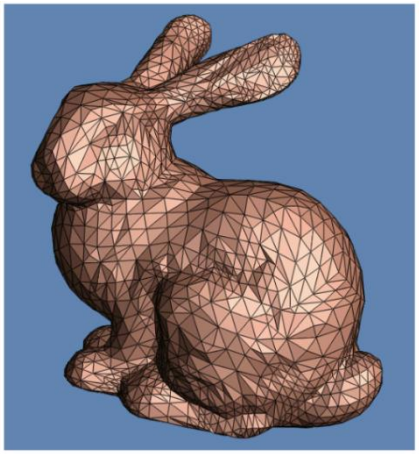
Multi-view images



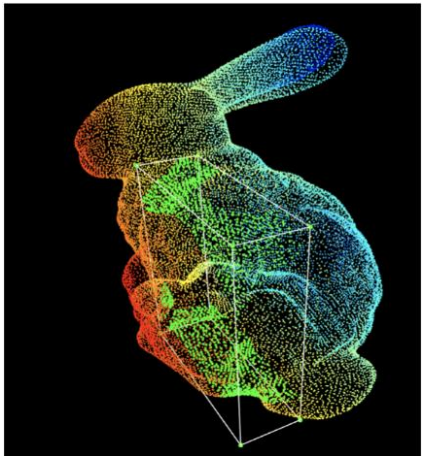
Depth image



Volumetric



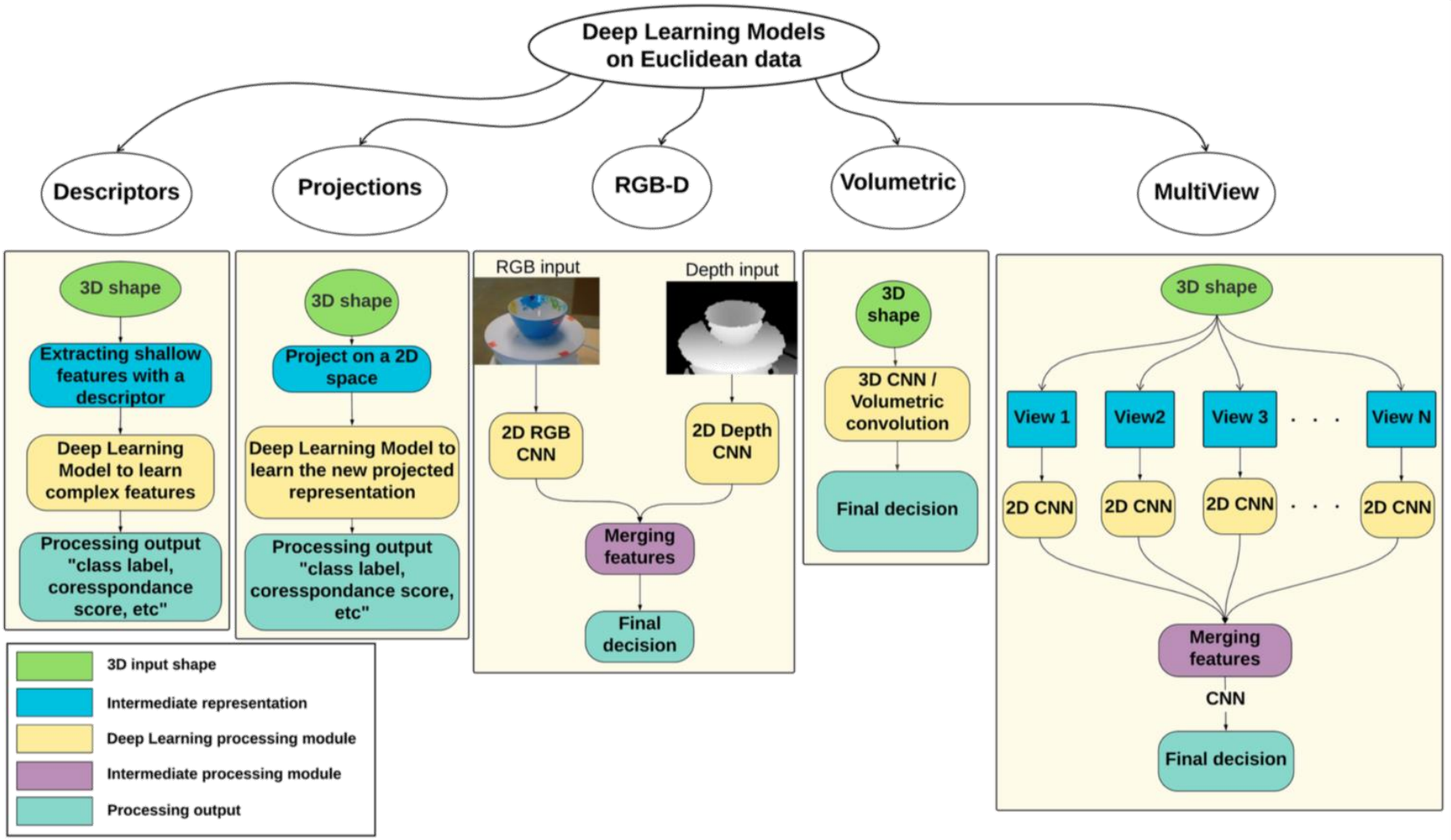
Polygon mesh



Point cloud

Reference: 3D deep learning in geometric forms, https://cseweb.ucsd.edu/~haosu/slides/NIPS16_3DDL.pptx

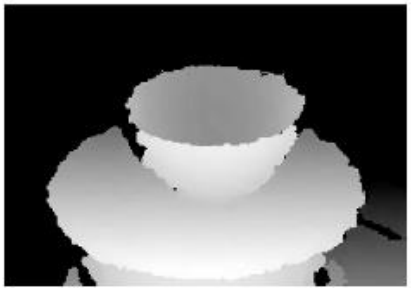
3D data machine learning



Reference: https://github.com/philastotle/pointnet-tutorial

RGB-D datasets

- RGB-D Object Dataset, http://rgbd-dataset.cs.washington.edu/
- NTU RGB+D dataset contains 60 action classes and 56,880 video samples. http://rose1.ntu.edu.sg/Datasets/actionRecognition.asp
- Major datasets are reviewed in “RGB-D Datasets: Past, Present and Future,” https://arxiv.org/pdf/1604.00999.pdf

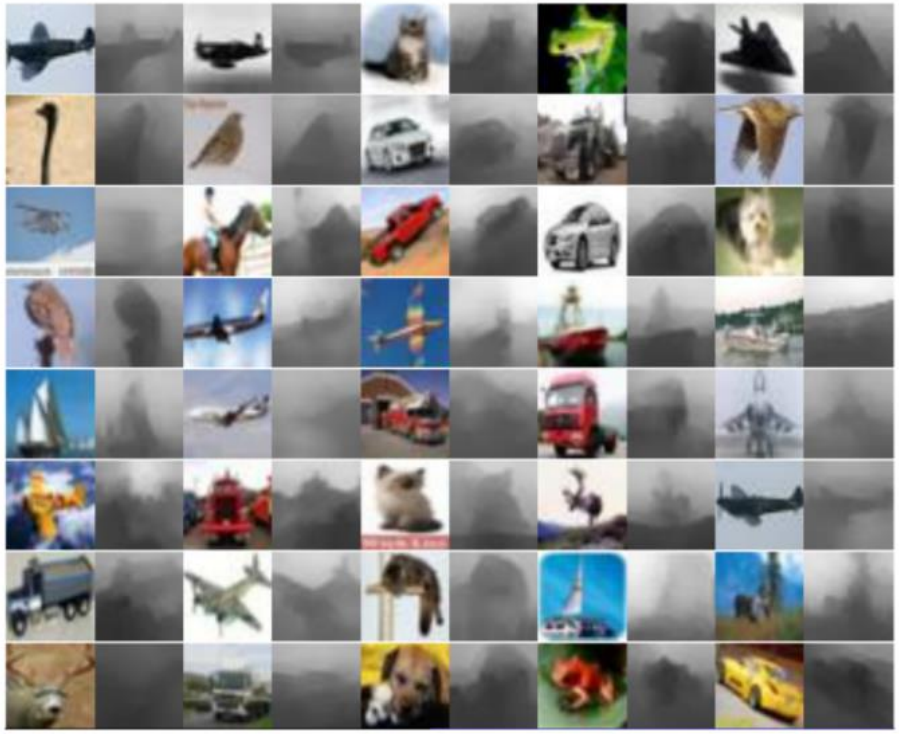


Name	Year	Labeled size	Classes	Resolution	Annotations
Cornell-RGBD-Dataset [1]	2011	550 frames from 52 scenes	17	N/A	per-point clouds annotations
RGB-D Object Dataset [2]	2011	250,000 frames of 300 objects	51	640×480	object annotations
NYU Depth v1 [3]	2011	2347 frames from 64 scenes	13	640×480	dense pixel annotations
NYU Depth v2 [4]	2012	1449 frames from 464 scenes	4/13/40	640×480	dense pixel annotations
SUN3D [5]	2013	415 sequences from 254 scenes	33	640×480	object polygons annotations
Berkeley B3DO [6]	2013	849 frames from 75 scenes	over 50	640×480	bounding box annotations
Kinect RGBD Dataset for Category Modeling [7]	2013	900 frames from 264 scenes	7	640×480	object annotations
SUN RGB-D [8]	2015	10,335 frames across 47 scene classes	37	variable	dense pixel annotations

Depth can help us

1. Assist image classification using RGB-D data.

RGB-D CIFAR10 dataset

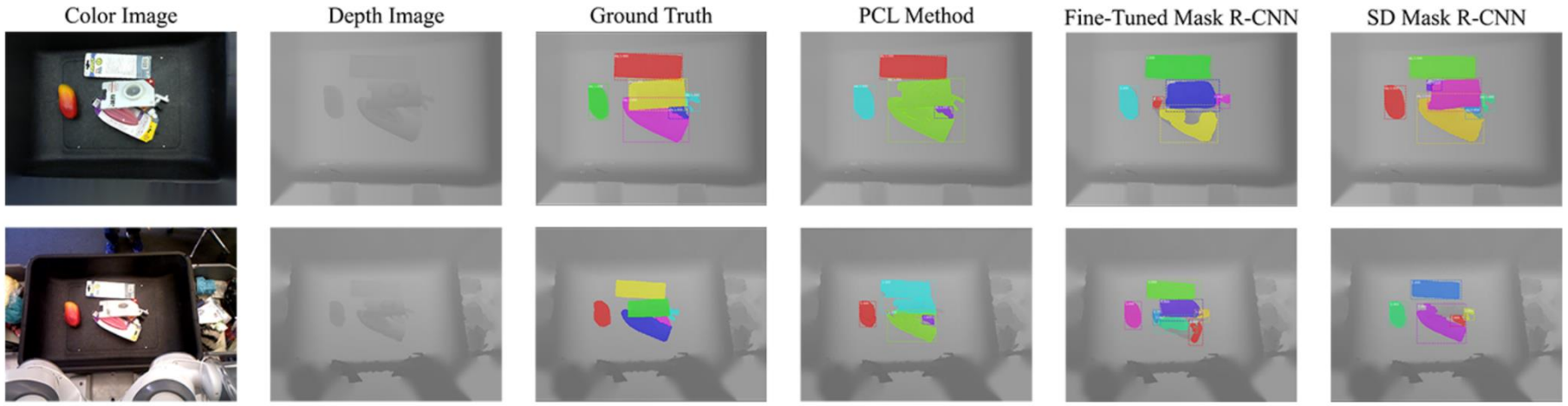


- The dataset only has RGB images with annotation.
- Synthesized D data is generated from the RGB images (via the depth estimation method from the single RGB image).
- Build a new CNN classification model using RGB (in the original dataset) and D (synthesized).

Reference: Estimated depth map helps image classification, https://arxiv.org/abs/1709.07077

Depth can help us

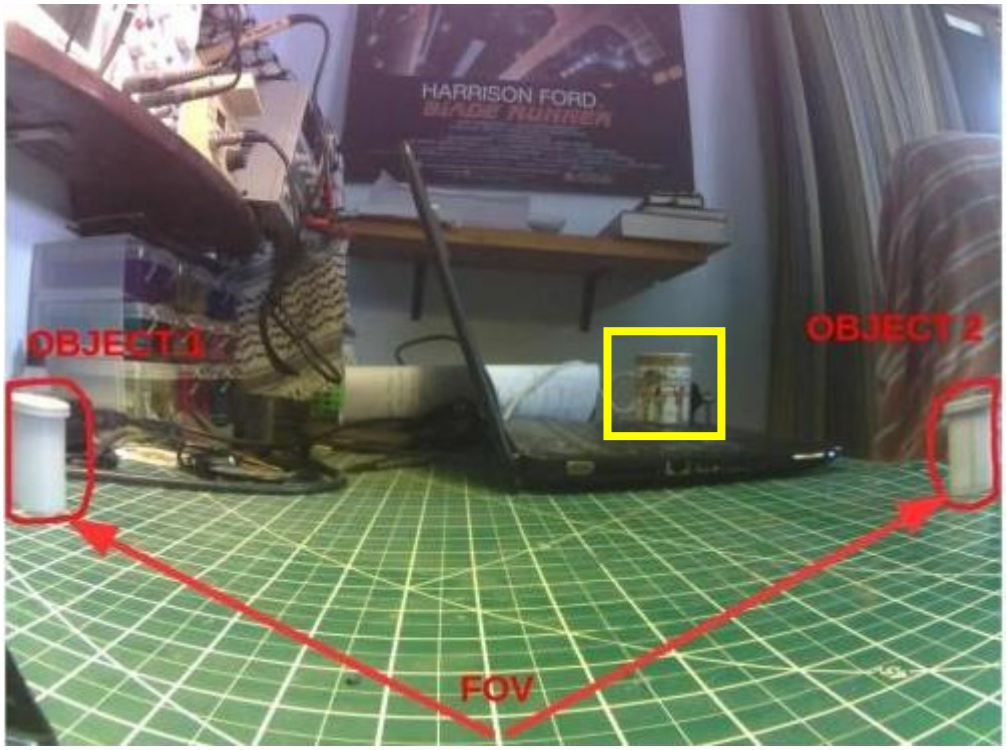
2. Assist Detection/Segmentation: Depth information encodes the geometric cues necessary to separate object instances.



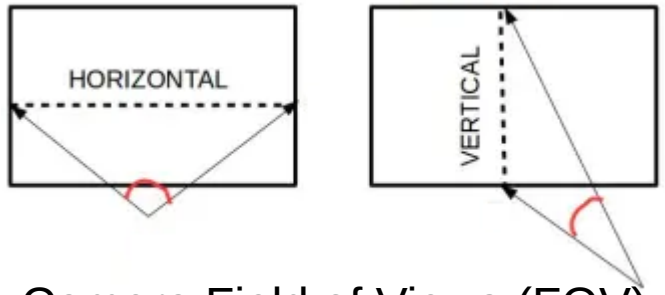
Reference: https://bair.berkeley.edu/blog/2018/10/23/depth-sensing/

Depth can help us

3. Assist to infer the position (angle/distance) of the object with the respect to the camera.



- Detect the target object in the RGB image to obtain its object bounding box center (r, c).
- Use the depth image (value at the position (r, c)) to infer the distance between the object and the camera.
- Use the object center (r, c) and the camera *Field of Views* (FOV) to infer the angle.
- The camera FOVs are provided by the camera manufacture.



Camera Field of Views (FOV)

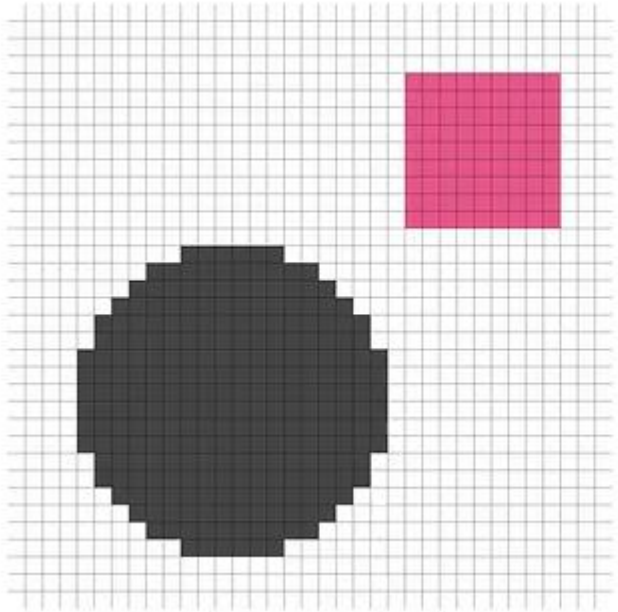
Image: https://bayesianadventures.wordpress.com/2016/09/13/upping-the-raspberry-pis-field-of-view/

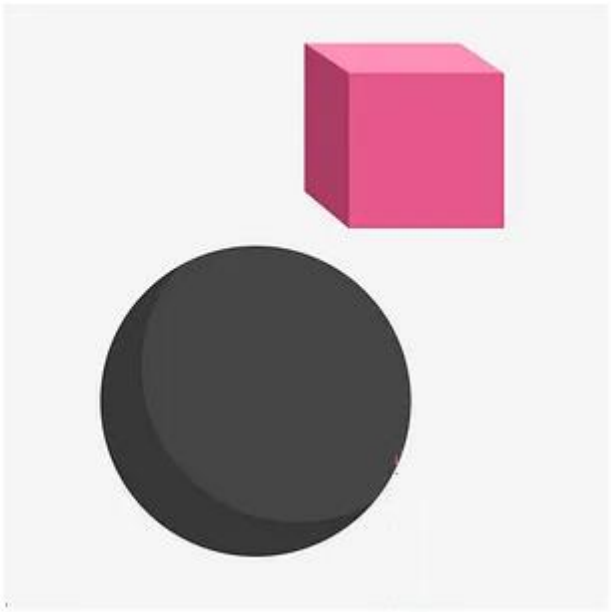
Depth can help us

4. Generate point cloud from RGB image and depth image.

$$z_c = D_{u,v}$$
$$x_c = z_c(u - u_0)/f$$
$$y_c = z_c(v - v_0)/f$$

Calculate the color of the point (x_c, y_c, z_c) in the physical world based on its color value at (u, v) in the RGB image and its depth value $D_{u,v}$ in the depth image.

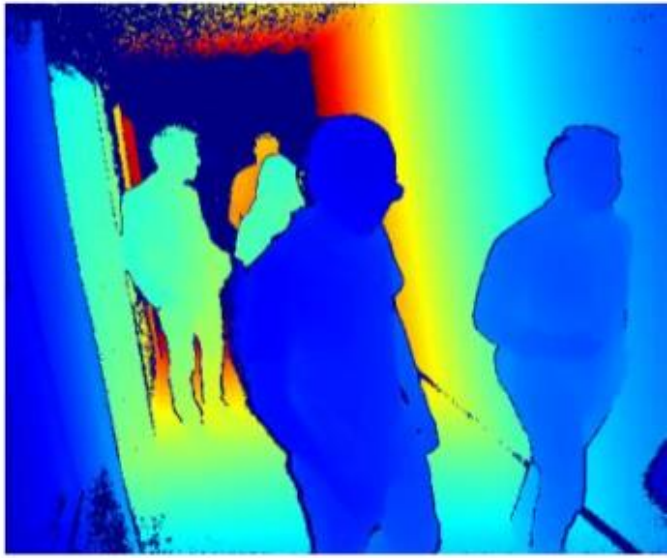




Depth map encoding

Method 1. Gray scale encoding: The depth values are converted to gray-scale intensity values linearly $g(Z_{u,v}) = \frac{Z_{u,v} - Z_{min}}{Z_{max} - Z_{min}} \times 255$, $Z_{u,v}$ is the depth value at the coordinates (u, v) of the depth image, Z_{max}, Z_{min} are the maximum and minimum values of the whole dataset.

Method 2. Color map encoding: Use the grayscale encoding values as indices for a Jet colormap with 256 entries



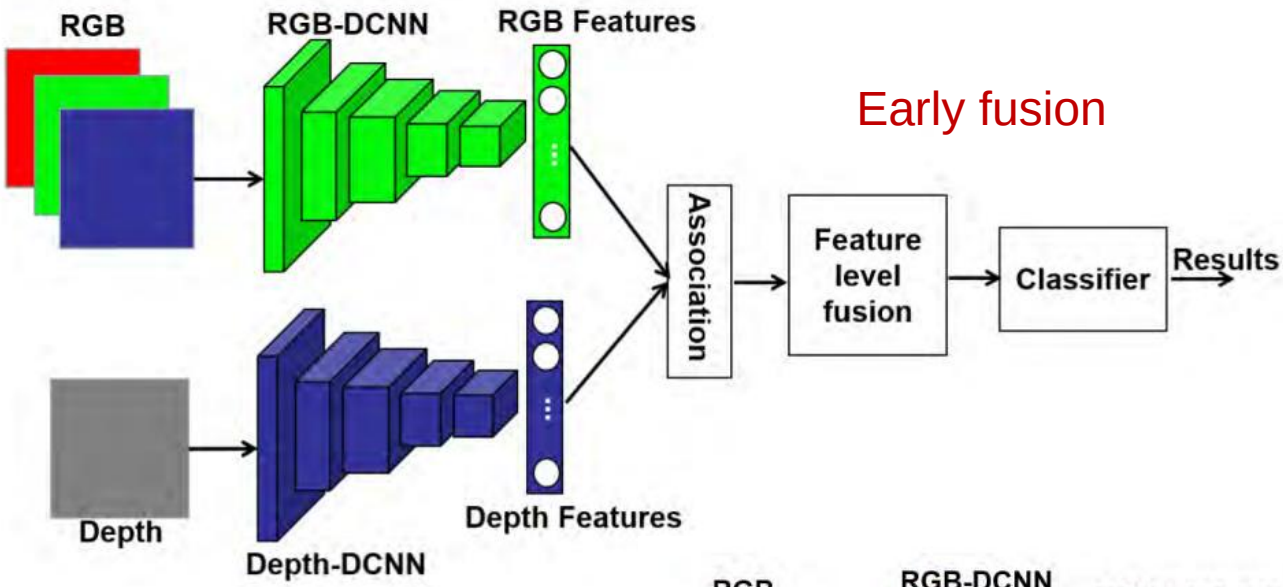
Depth map encoding

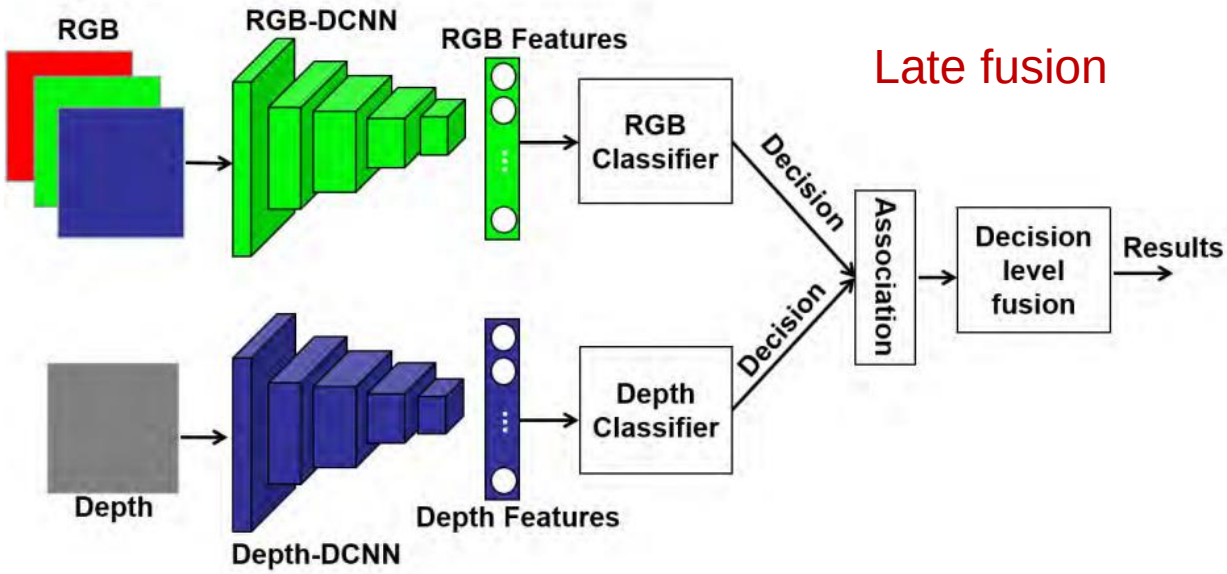
Method 3. HHA: A geocentric embedding with three channels.

- Horizontal disparity (H): $d = \frac{fT}{Z}$, d is the disparity of each pixel position, f and T are the focal length and the baseline of the cameras.
- Height from the ground (H): The 3D point cloud is calculated from the depth map. Then the height value of each pixel is roughly calculated by subtracting the lowest point (with the minimum height) within an image.
- Angle with gravity (A): An iterative procedure is used. An initial estimate for the gravity direction is taken as the vertical axis, with respect to which all surface normal are clustered into surfaces that are approximately parallel or orthogonal to the gravity direction. After clustering, a new gravity direction estimated with respect to the parallel and orthogonal clusters. These steps are iterated to minimize so that the gravity direction is as parallel as possible to the parallel surfaces and as orthogonal as possible to the orthogonal surfaces.

Reference: S. Gupta, Learning Rich Features from RGB-D Images for Object Detection and Segmentation, ECCV 2014, https://link.springer.com/chapter/10.1007/978-3-319-10584-0_23, Python implementation: https://github.com/charlesCXK/Depth2HHA-python

RGB-D data: Image classification

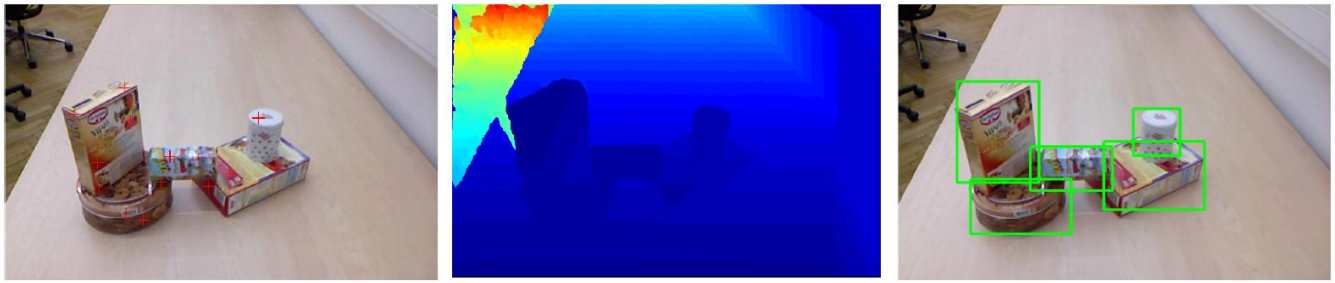


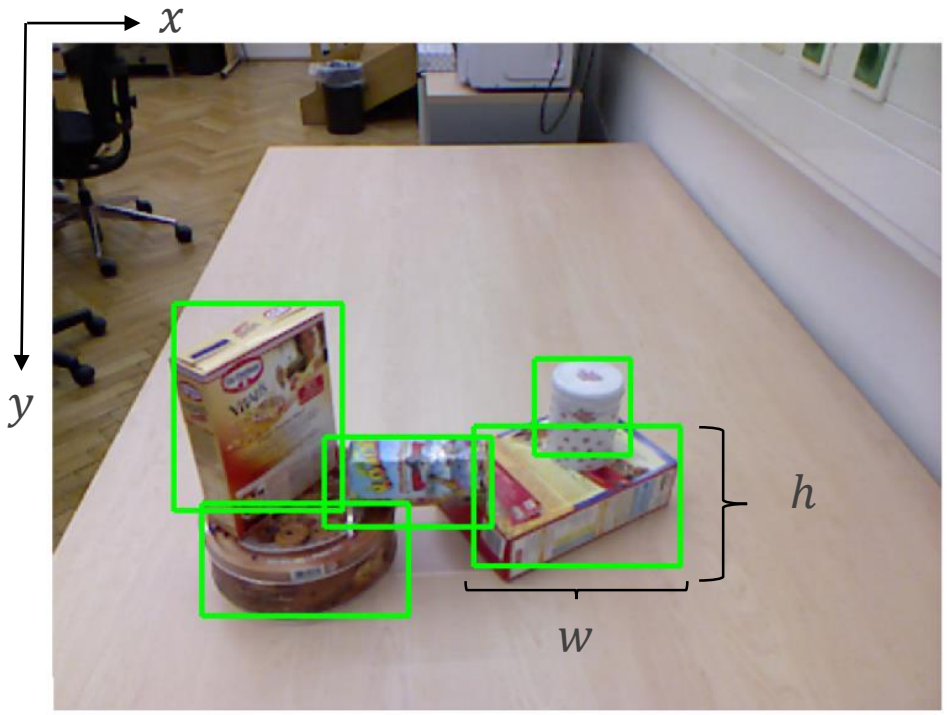


Reference: M. Gao, et al., RGB-D-Based Object Recognition Using Multimodal Convolutional Neural Networks: A Survey, IEEE Access, 2019, https://ieeexplore.ieee.org/document/8683987

RGB-D data: (2D) Object detection

Objective: Perform object detection on a pair of RGB image and depth image.





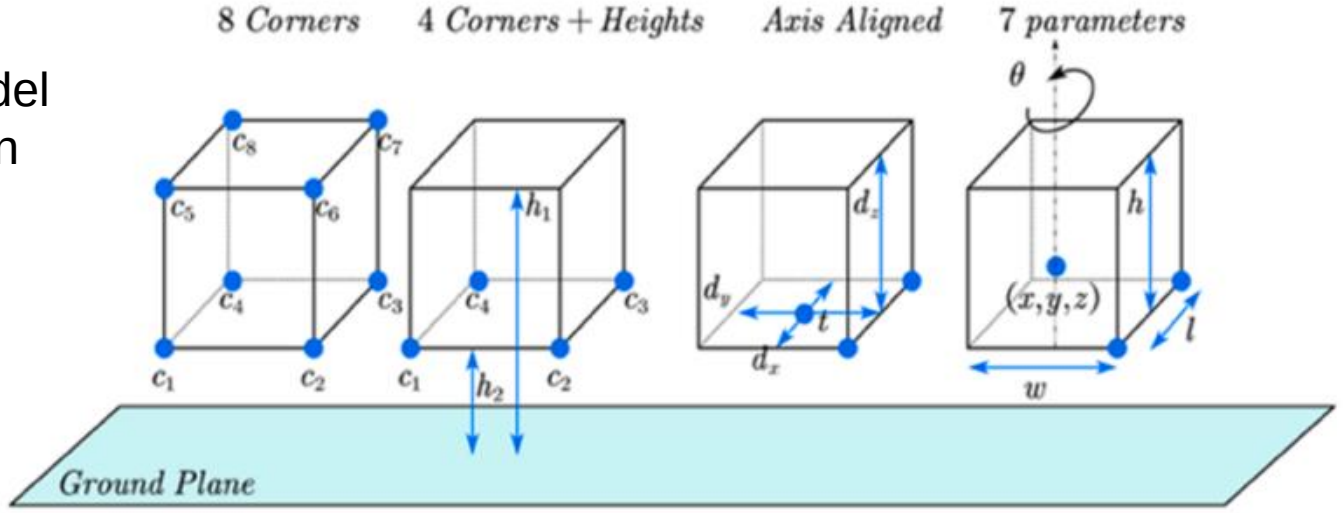
We need to build a model that can detect objects in the 2D bounding box

- Box label (one-hot vector, a classification problem)
- Box coordinates (x, y, w, h) (a regression problem)
  - Column index of top-left corner x
  - Row index of top-left corner y
  - Width of box w
  - Height of box h

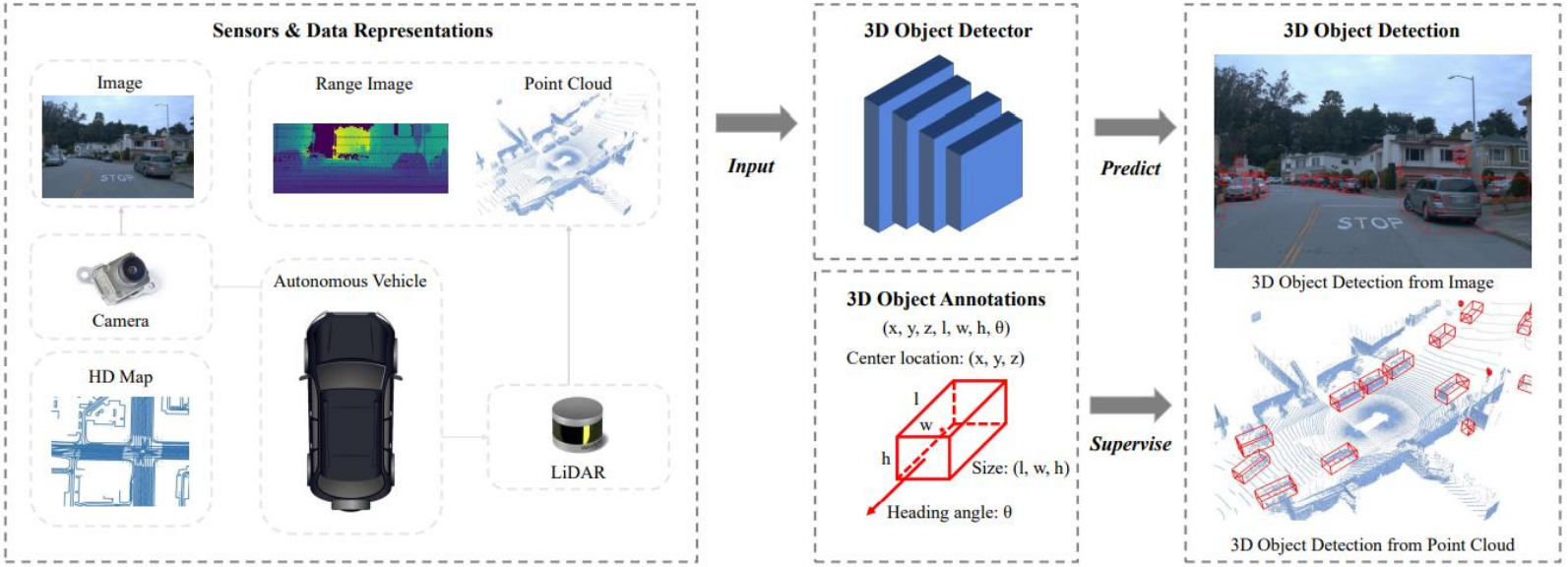
Reference: RGB-D image-based Object Detection: from traditional methods to deep learning techniques, https://arxiv.org/abs/1907.09236

RGB-D data: (3D) Object detection

We also can build a model that can detect objects in the 3D bounding box



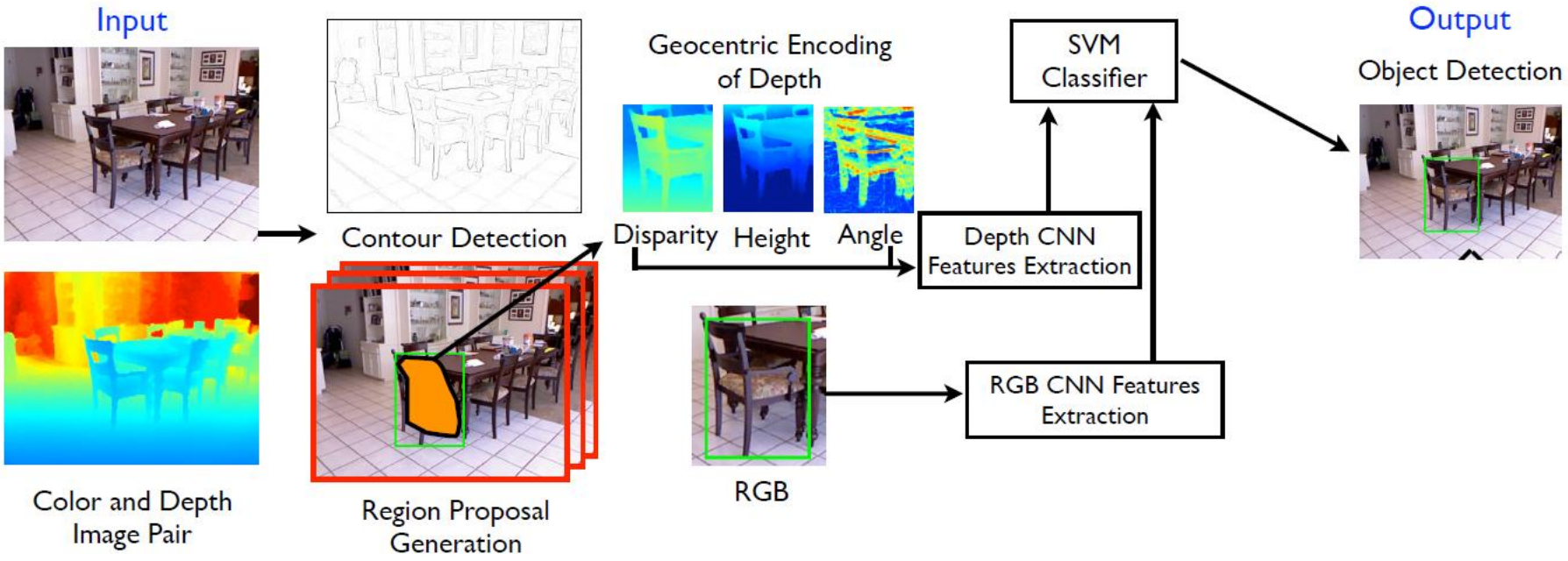
An illustration of 3D object detection in autonomous driving scenarios



Reference: 3D Object Detection for Autonomous Driving: A Comprehensive Survey, IJCV, 2023, https://arxiv.org/abs/2206.09474

RGB-D data: (2D) Object detection

Depth R-CNN
- Perform region proposal from RGB
- Feature extraction (from both RGB and D) from the proposed regions
- Classification for the proposed regions

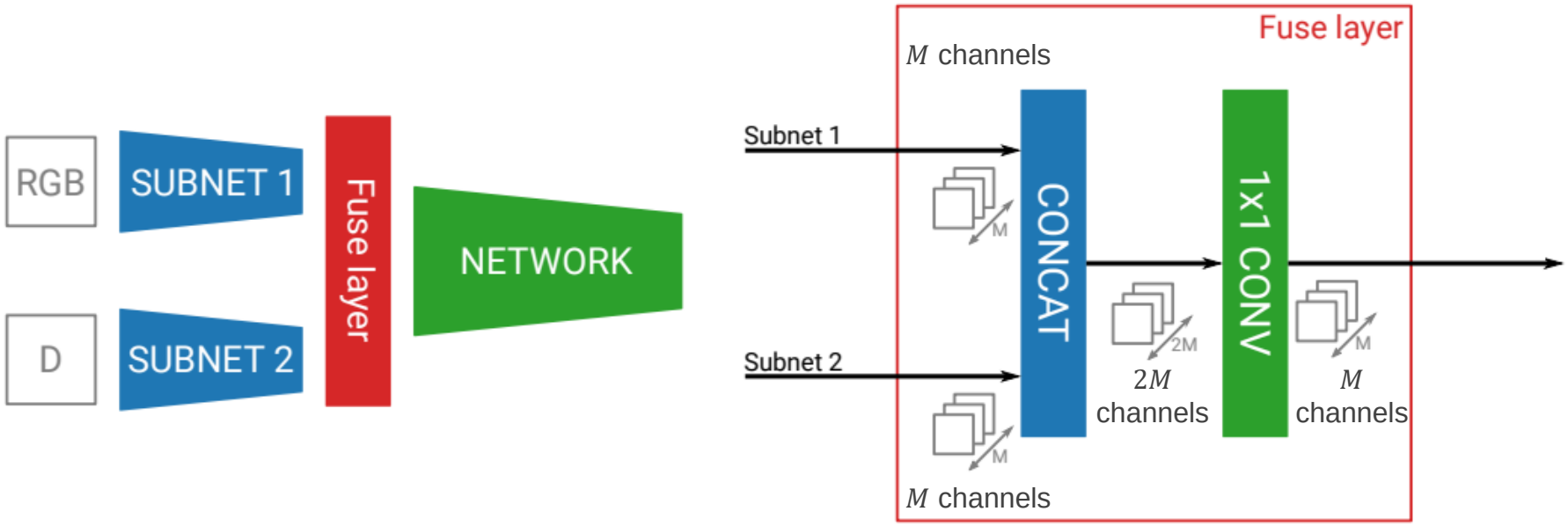


Reference: Learning rich features from RGB-D images for object detection and segmentation, ECCV 2014, https://arxiv.org/abs/1407.5736

RGB-D data: (2D) Object detection

Intuition: How to re-use our powerful model developed for RGB image?

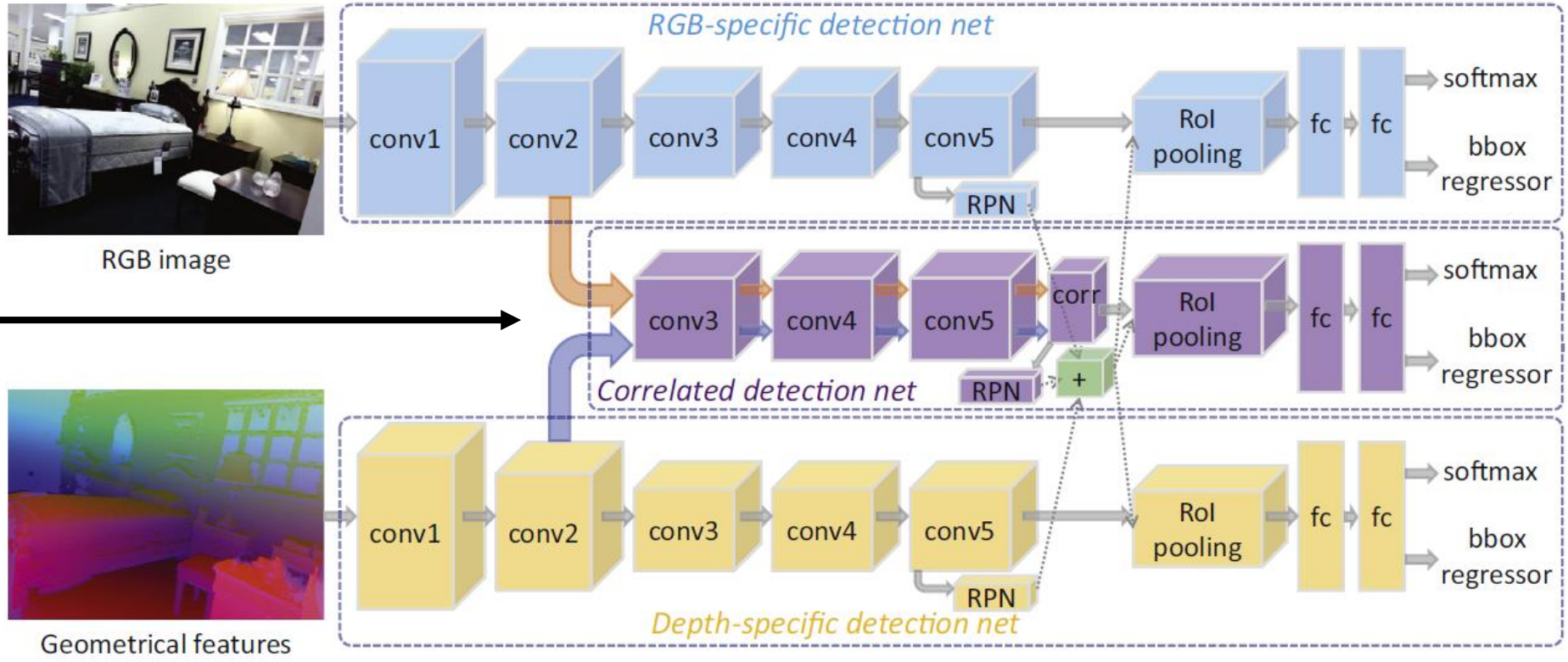
- Apply our powerful model (developed from RGB images) on both RGB and D.
- Apply the fuse layer (1×1 conv) to fuse their features maps (from RGB and D, respectively) to maintain the same feature map structure (e.g., dimension) as that is obtained using the RGB only.
- For example: Subnet1 part and Subnet2 part could be early layers of YOLOv3. Network part could be latter layers of YOLOv3. (refer to the paper for details)



Reference: Exploring RGB+Depth Fusion for Real-Time Object Detection, https://www.mdpi.com/1424-8220/19/4/866

RGB-D data: (2D) Object detection

Intuition: How to consider correlation of multiple modality data (input RGB-D)?



- Shared convolution layers ($conv3$, $conv4$, $conv5$, suggested by the authors)
- $corr$ means a parameter-free correlation layer, which performs multiplicative comparisons (element-wise product) between feature maps of two modalities.
- Geometrical features are 3-channel HHA features calculated from 1-channel depth image.

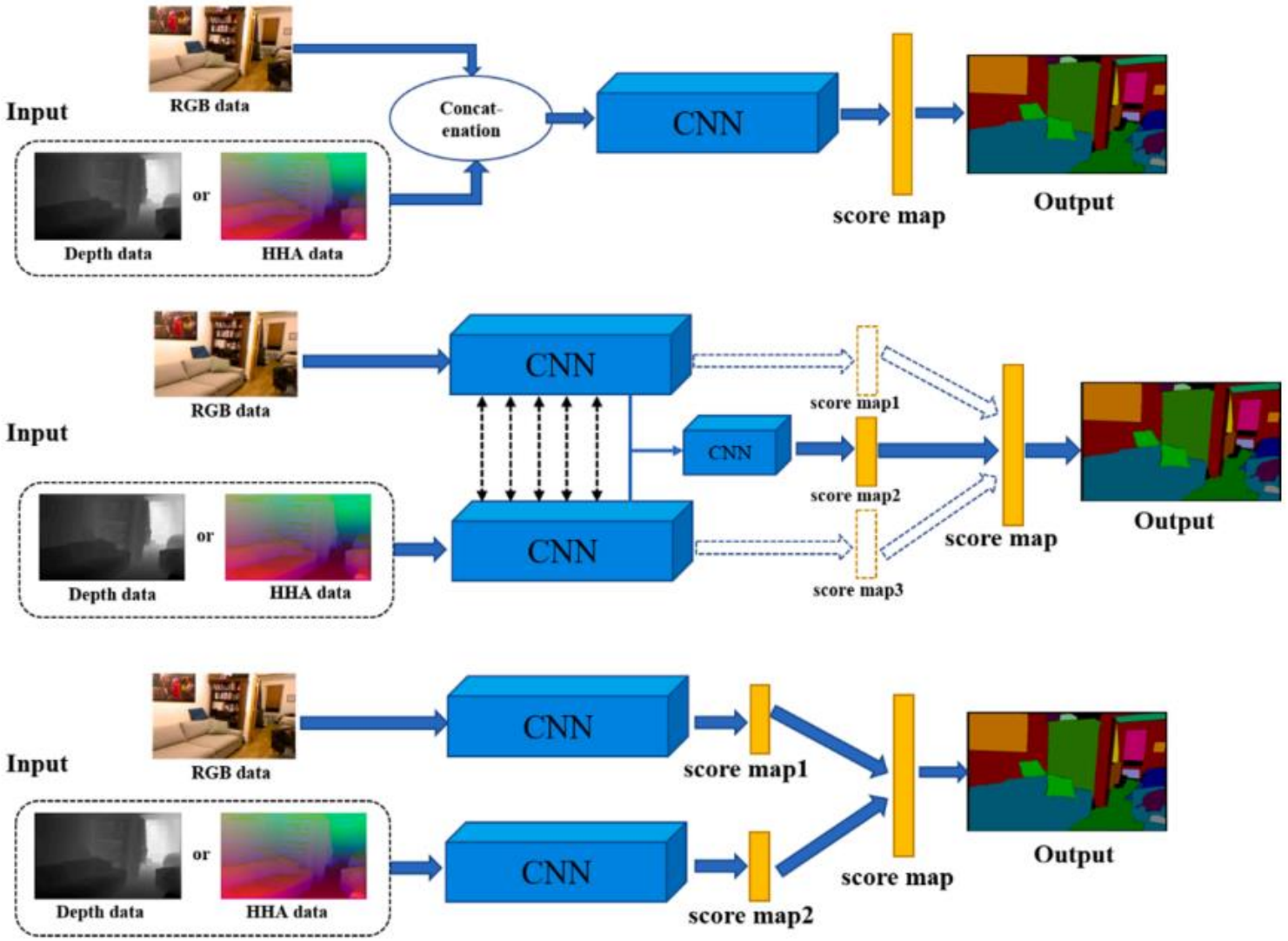
Reference: Multi-modal deep feature learning for RGB-D object detection, https://par.nsf.gov/servlets/purl/10073960

RGB-D data: Segmentation

Image fusion

Feature fusion

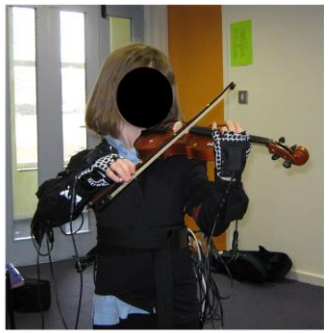
Output fusion



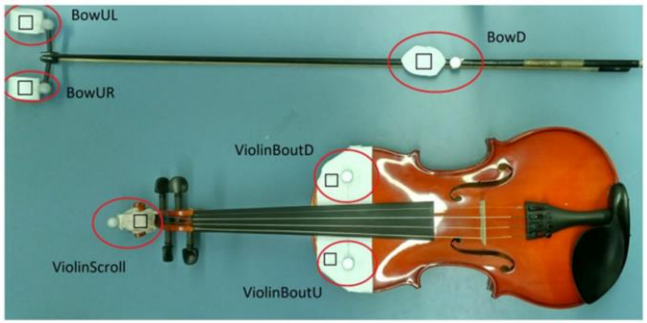
Reference: A brief survey on RGB-D semantic segmentation using deep learning, 2021

Case study

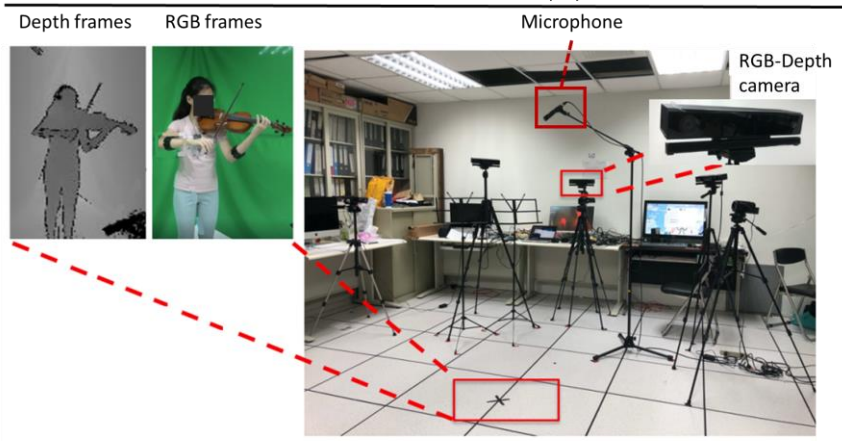
There are three types of intelligent sensing systems, including (i) a smart jacket with embedded wearable sensors for recording the player's body movement, (ii) a marker-based computer vision solution that places markers on the violin and the bow for tracking the player's bowing movement; and (iii) a non-intrusive multiple-modality system based on RGB-Depth camera and microphone. A bowing action recognition model is built using both RGB frame and depth frame. The depth frame in the video-based bowing recognition model is viewpoint-dependent, making recording various scenarios time-consuming.



First-generation system: A smart jacket with embedded wearable sensors



Second generation system: A marker-based computer vision solution. These markers are required to be placed on the bow and the violin to track the bow movement of the player.

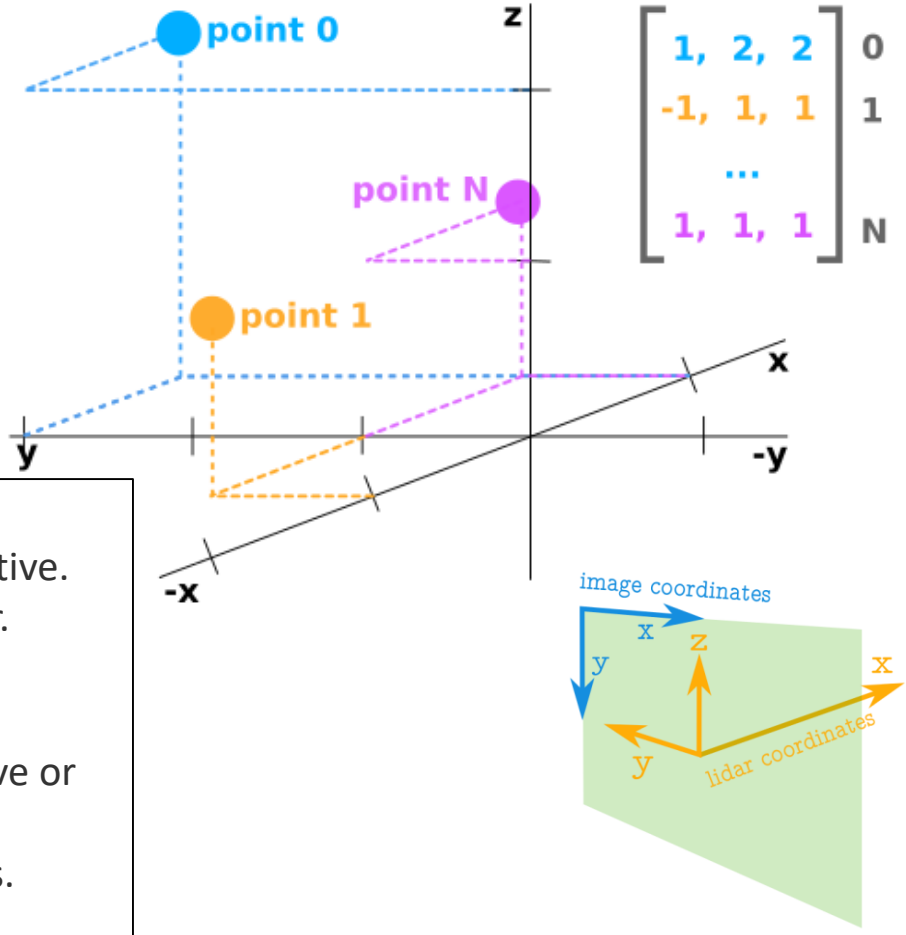


Third-generation system: A non-intrusive multiple-modality system with an RGB-Depth camera and a microphone.

Do you think the traditional RGB image augmentation methods (such as rotation) are suitable for the depth frame? Recommend one augmentation method for the depth map (to simulate various camera viewpoints).

Point cloud: Overview

- The point cloud data can be represented as a numpy array with N rows and 3 columns. Each row corresponds to a single point, which is represented using at least 3 values for its position in space (x, y, z).



Image

- The coordinate values in an image are always positive.
- The origin is located on the upper left hand corner.
- The coordinates are integer values.

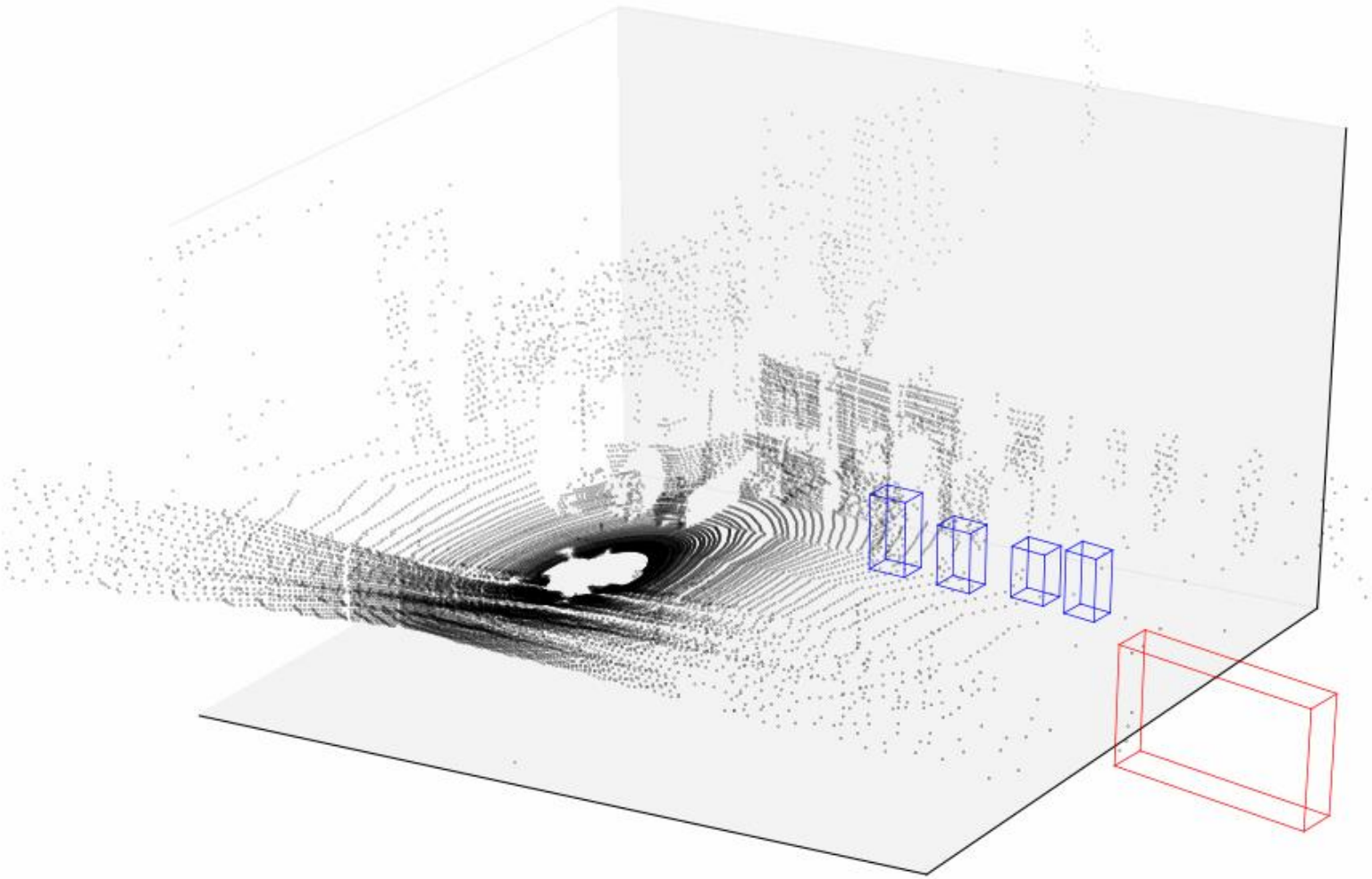
Point cloud

- The coordinate values in point cloud can be positive or negative.
- The coordinates can take on real numbered values.
- The positive x axis represents forward.
- The positive y axis represents left.
- The positive z axis represents up.

Reference: http://ronny.rest/tutorials/module/pointclouds_01/point_cloud_data/

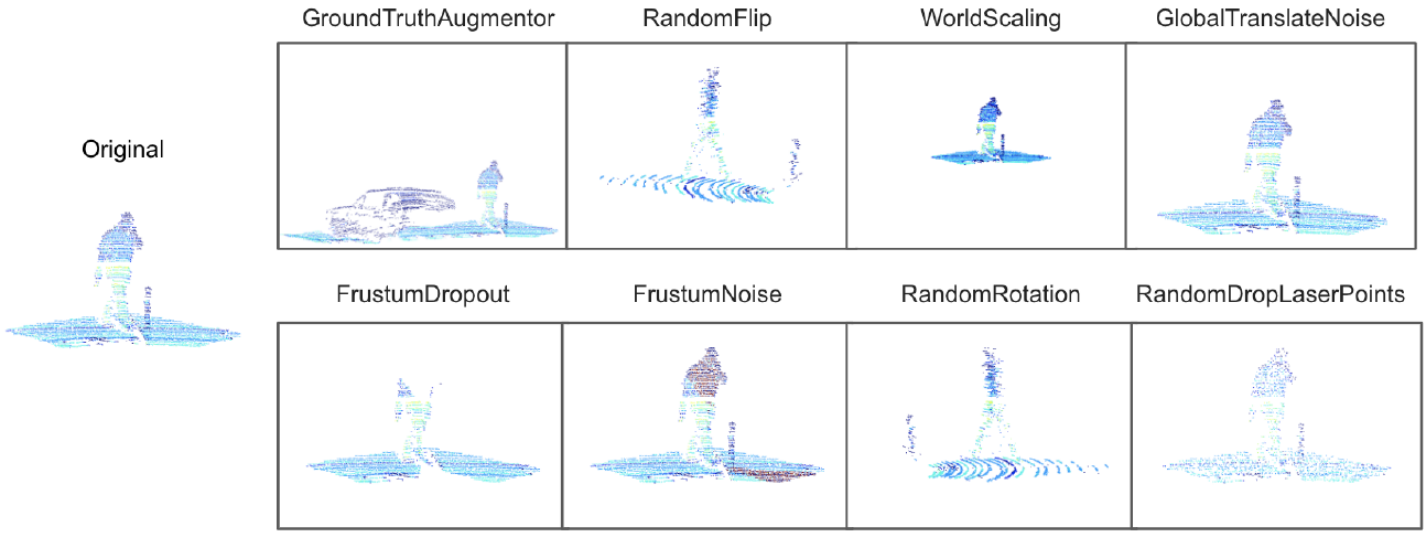
Point cloud: Overview

KITTI dataset: Cars (blue), trams (red), and cyclists (green).



Reference: https://navoshta.com/kitti-lidar/

Point cloud data augmentation

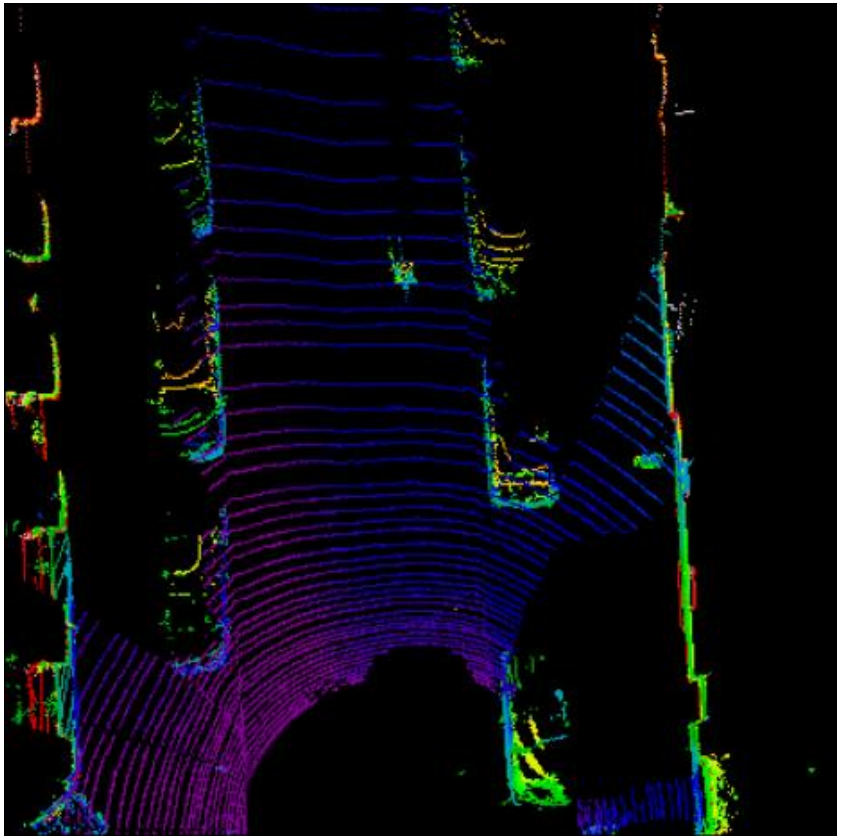


Operation Name	Description
GroundTruthAugmentor [31]	Augment the bounding boxes from a ground truth data base (< 25 boxes per scene)
RandomFlip [33]	Randomly flip all points along the Y axis.
WorldScaling [37]	Apply global scaling to all ground truth boxes and all points.
RandomRotation [37]	Apply random rotation to all ground truth boxes and all points.
GlobalTranslateNoise	Apply global translating to all ground truth boxes and all points along x/y/z axis.
FrustumDropout	All points are first converted to spherical coordinates, and then a point is randomly selected. All points in the frustum around that point within a given phi, theta angle width and distance to the original greater than a given value are dropped randomly.
FrustumNoise	Randomly add noise to points within a frustum in a converted spherical coordinates.
RandomDropout	Randomly dropout all points.

Reference: Improving 3D Object Detection through Progressive Population Based Augmentation, ECCV 2020, https://arxiv.org/abs/2004.00831

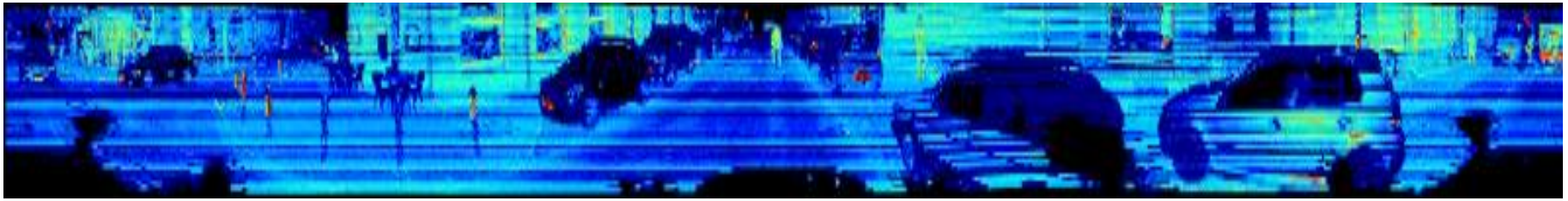
Point cloud data representation

- Bird overview view
- Panoramic view



RPLIDAR A1

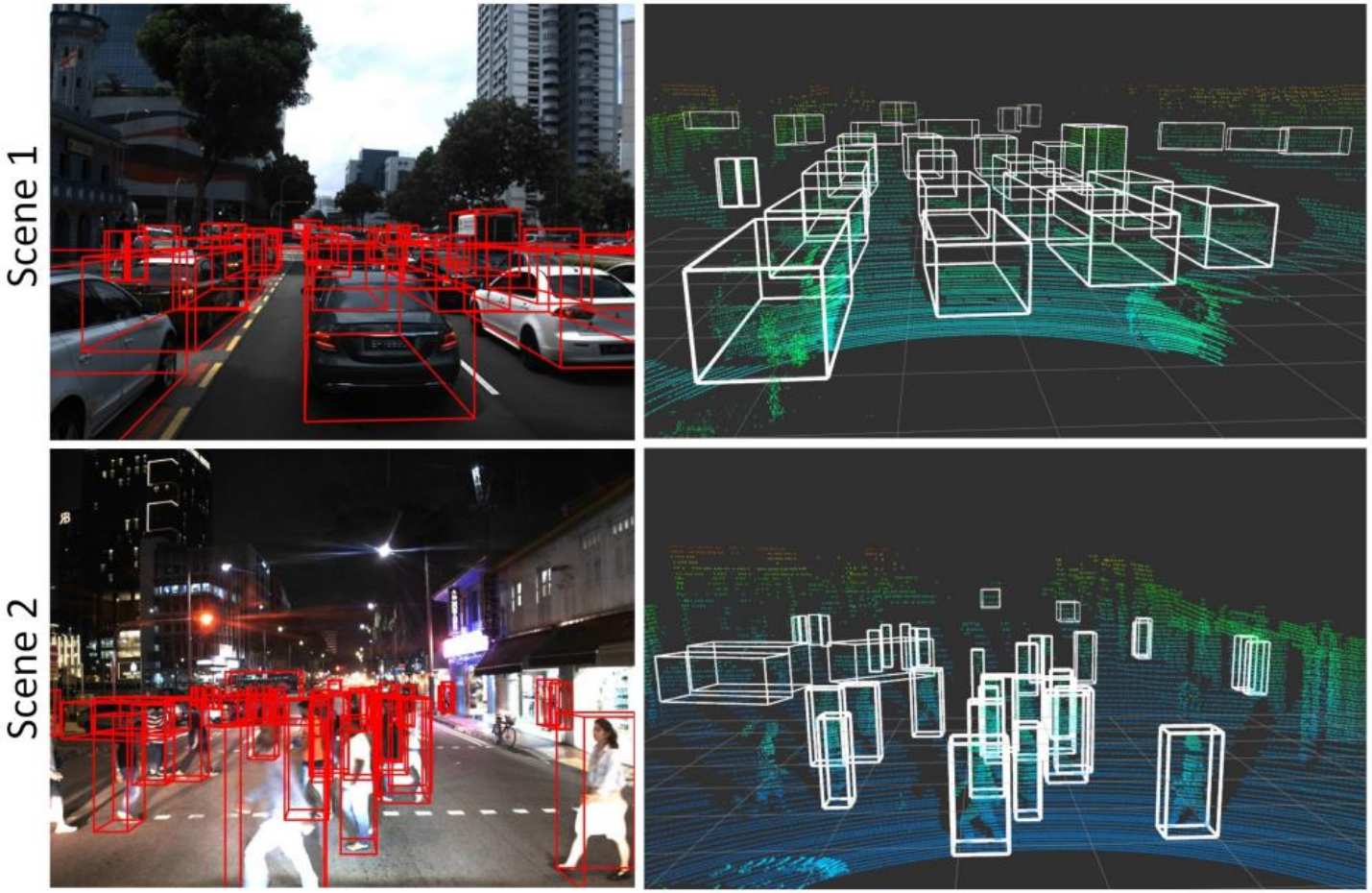
http://www.slamtec.com/en/lidar/a1



Reference: http://ronny.rest/tutorials/module/pointclouds_01/point_cloud_panoramic360/

RGB-LiDAR dataset

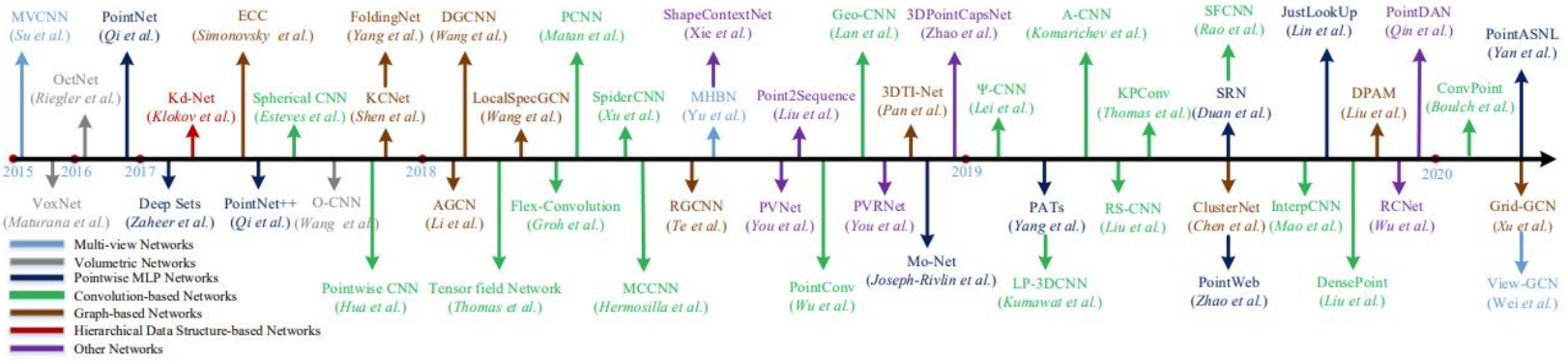
- 230K human-labeled 3D object annotations in 39,179 LiDAR point cloud frames and corresponding frontal-facing RGB images.
- Captured at different times (day, night) and weathers (sun, cloud, rain).



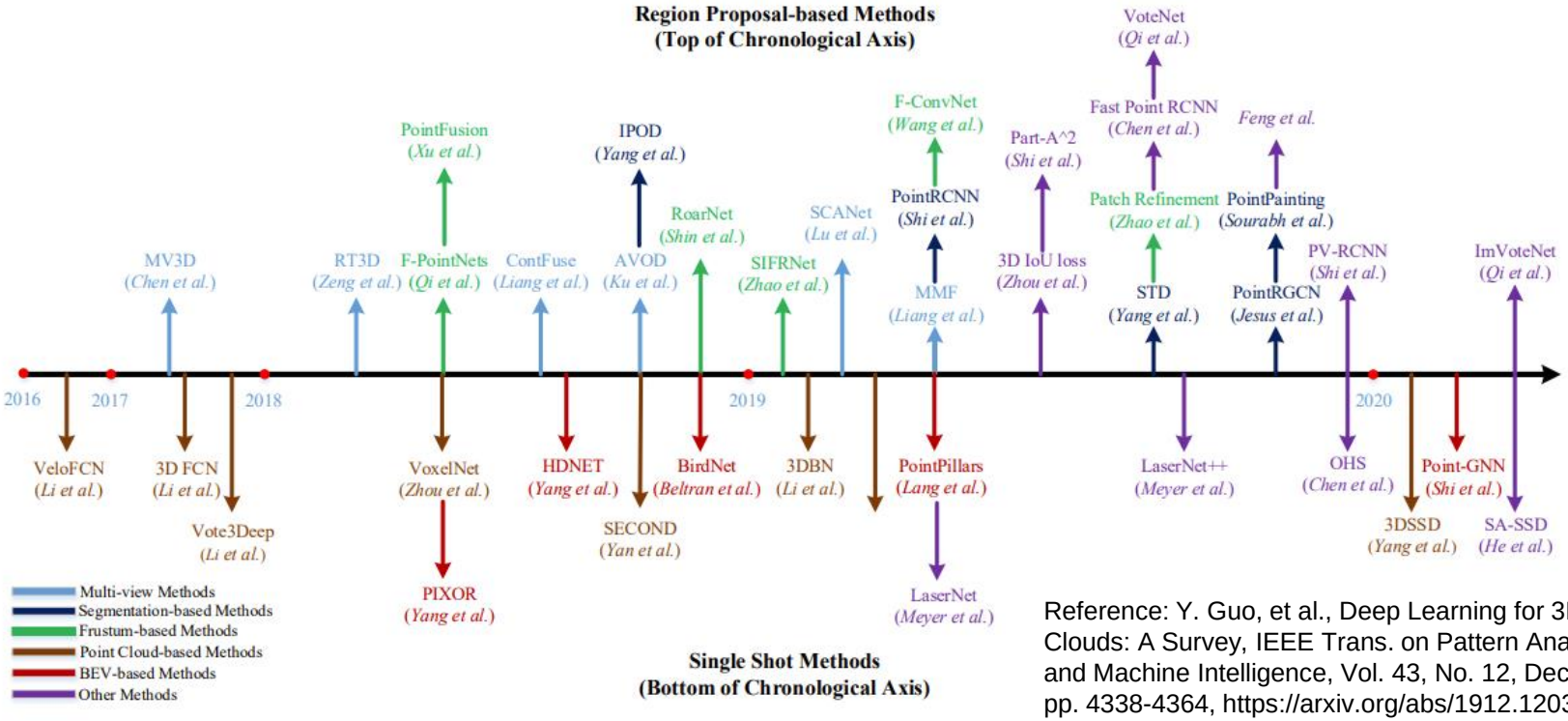
Reference: https://github.com/I2RDL2/ASTAR-3D

Milestones

Classification



Object detection



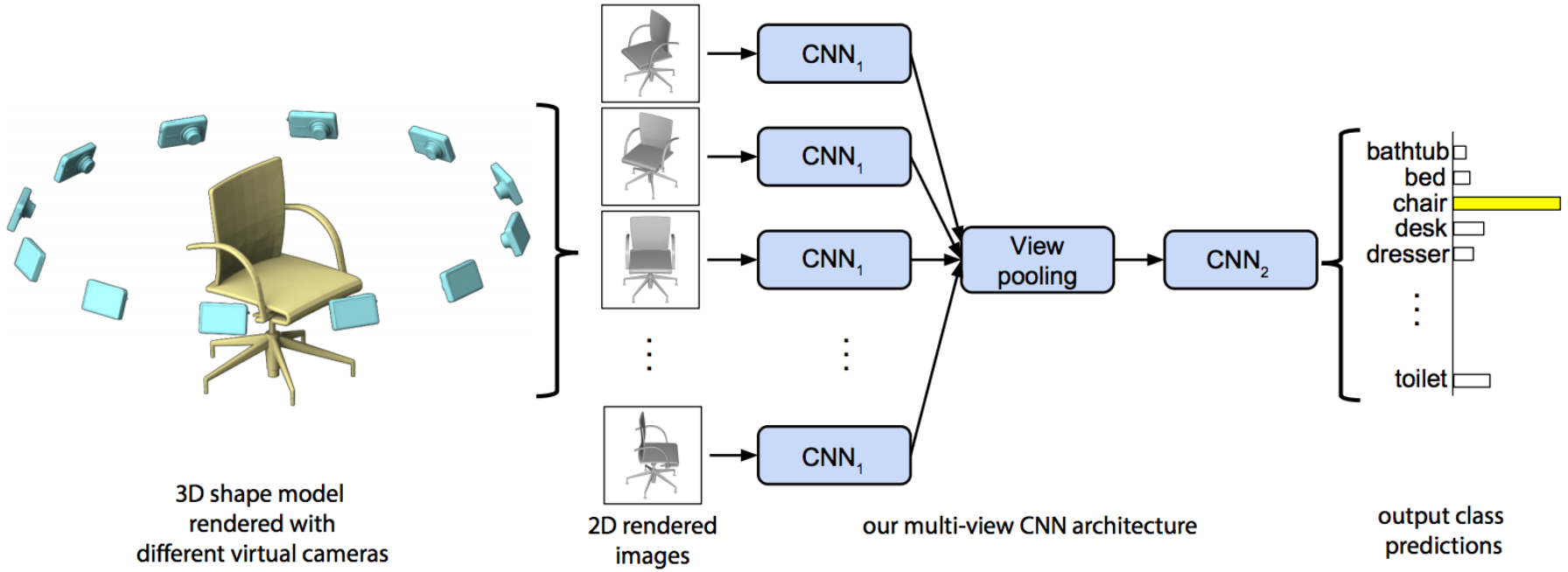
Reference: Y. Guo, et al., Deep Learning for 3D Point Clouds: A Survey, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 43, No. 12, Dec. 2021, pp. 4338-4364, https://arxiv.org/abs/1912.12033

Milestones

Pseudo image-based representation	Classification	MVCNN	H. Su, *et al.*, “Multi-view convolutional neural networks for 3D shape recognition,” ICCV 2015, https://arxiv.org/abs/1505.00880
		VoxNet	D. Maturana, *et al.*, “VoxNet: A 3D convolutional neural network for real-time object recognition,” IROS 2015, https://dimatura.net/research/voxnet/
	Detection	Complex YOLO	M. Simon, *et al.*, “Complex-YOLO: An Euler-region-proposal for teal-time 3D object detection on point clouds,” ECCV 2018, https://arxiv.org/abs/1803.06199
		PointPillars	A. Lang, *et al.*, “PointPillars: Fast encoders for object detection from point clouds,” CVPR 2019, https://arxiv.org/abs/1812.05784
Point-based representation	Classification	PointNet	R. Qi, *et al.*, “PointNet: Deep learning on point sets for 3D classification and segmentation,” CVPR 2017, https://arxiv.org/abs/1612.00593
	Detection	PointRCNN	S. Shi, *et al.*, “PointRCNN: 3D object proposal generation and detection from point cloud,” CVPR 2019, https://arxiv.org/abs/1812.04244
		Pointformer	X. Pan, *et al.*, “3D object detection with pointformer,” CVPR 2021, https://arxiv.org/abs/2012.11409

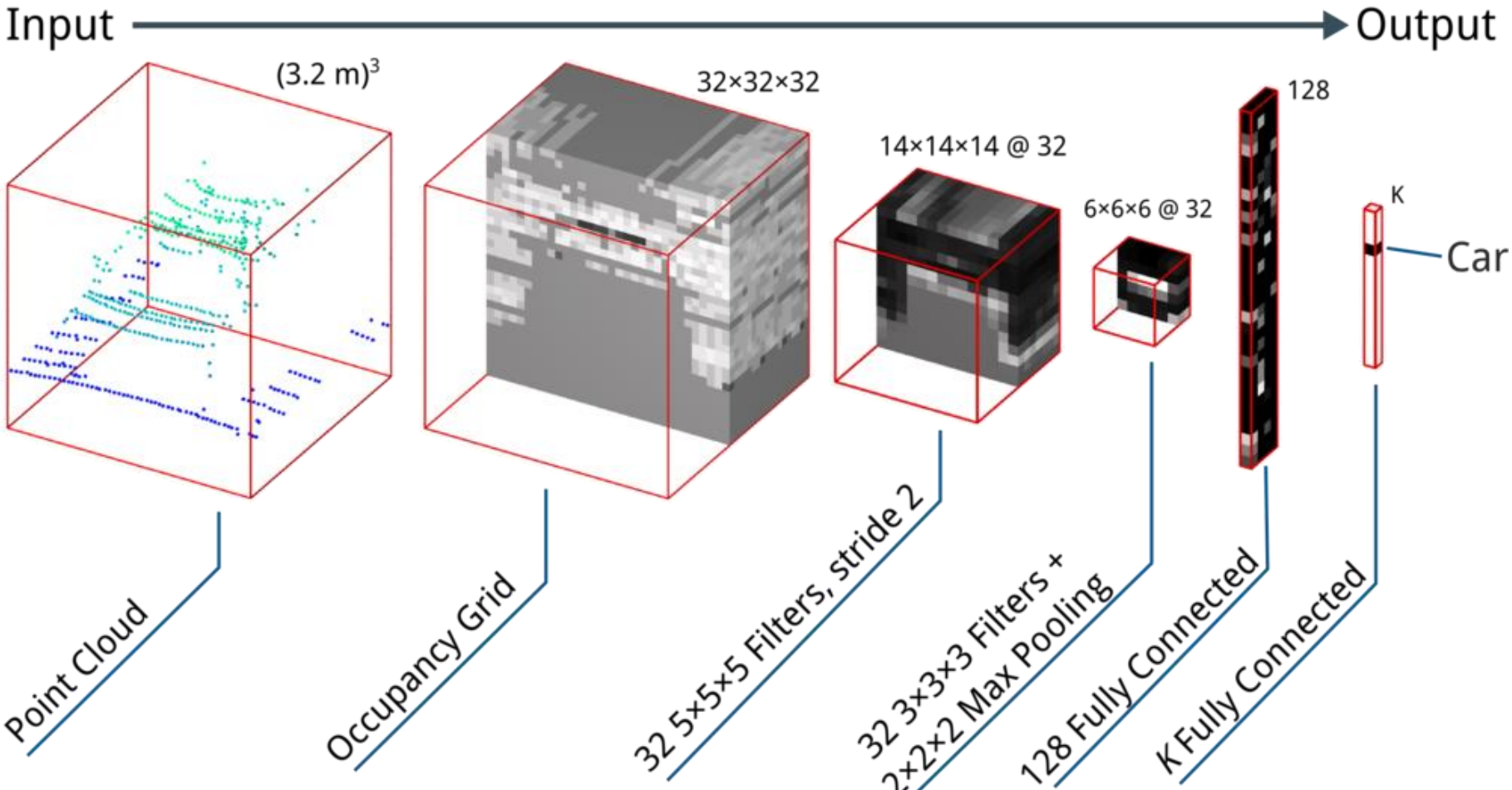
Point cloud: MVCNN

- First CNN: Extract image features from individual view
- View pooling: Element-wise max-pooling across all views
- Second CNN: Extract shape features from pooled image



Reference: H. Su, *et al.*, Multi-view convolutional neural networks for 3D shape recognition, ICCV 2015, https://arxiv.org/abs/1505.00880

Point cloud: VoxNet



Reference: D. Maturana, *et al.*, VoxNet: A 3D convolutional neural network for real-time object recognition, IROS 2015, https://dimatura.net/research/voxnet/

Complex YOLO

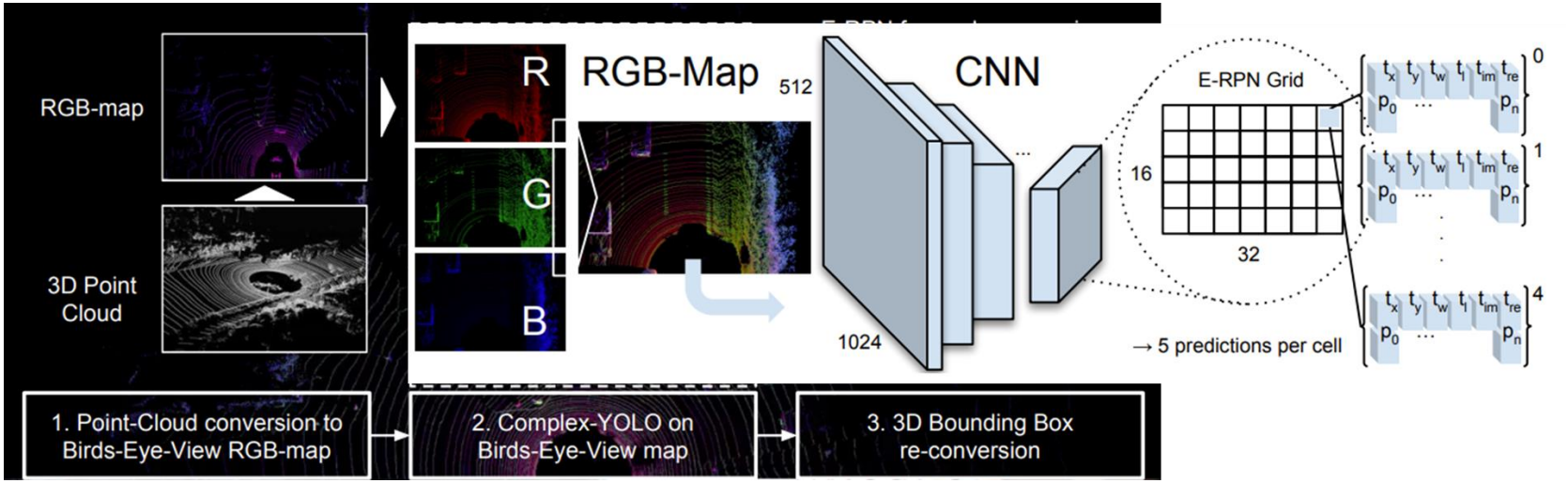


Fig. 2. Complex-YOLO Pipeline. We present a slim pipeline for fast and accurate 3D box estimations on point clouds. The RGB-map is fed into the CNN (see Tab. 1). The E-RPN grid runs simultaneously on the last feature map and predicts five boxes per grid cell. Each box prediction is composed by the regression parameters t (see Fig. 3) and object scores p with a general probability p_0 and n class scores $p_1 ... p_n$.

The bird's eye view representation is encoded by height, intensity and density.

- Height: The maximum height of the points in the cell.
- Intensity: The reflectance value of the point with the maximum height in each cell.
- Density: The number of points in each cell.

Reference: M. Simon, et al., Complex-YOLO: An Euler-region-proposal for teal-time 3D object detection on point clouds, ECCV 2018, https://arxiv.org/abs/1803.06199

PointPillars

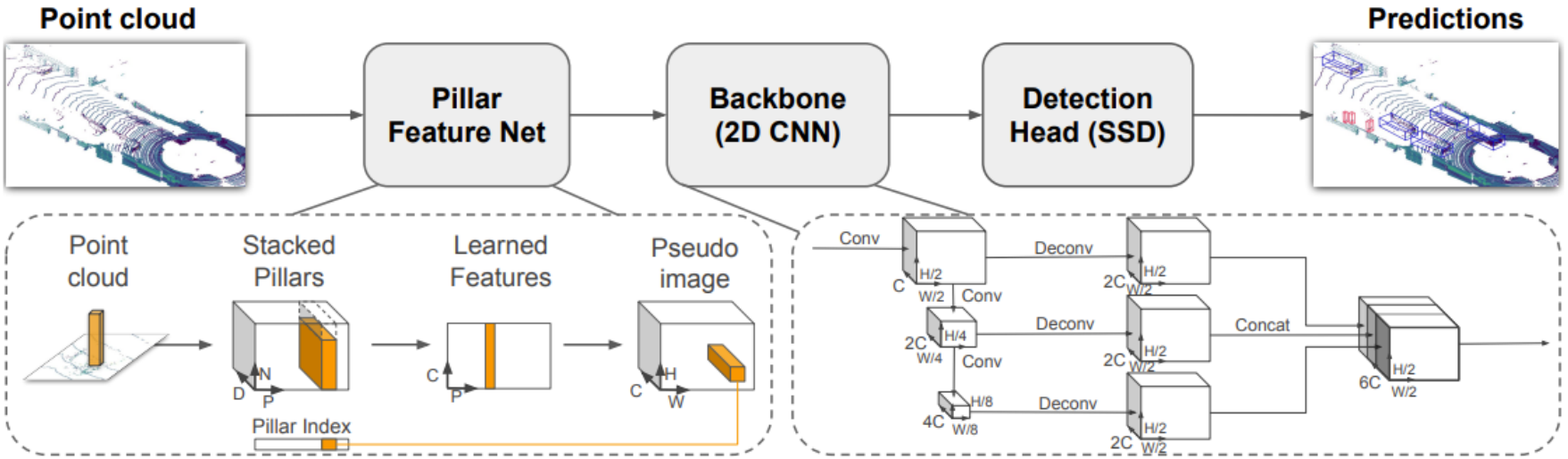
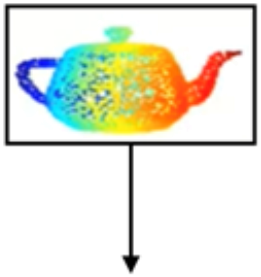


Figure 2. Network overview. The main components of the network are a Pillar Feature Network, Backbone, and SSD Detection Head (see Section 2 for details). The raw point cloud is converted to a stacked pillar tensor and pillar index tensor. The encoder uses the stacked pillars to learn a set of features that can be scattered back to a 2D pseudo-image for a convolutional neural network. The features from the backbone are used by the detection head to predict 3D bounding boxes for objects. Note: we show the car network's backbone dimensions.

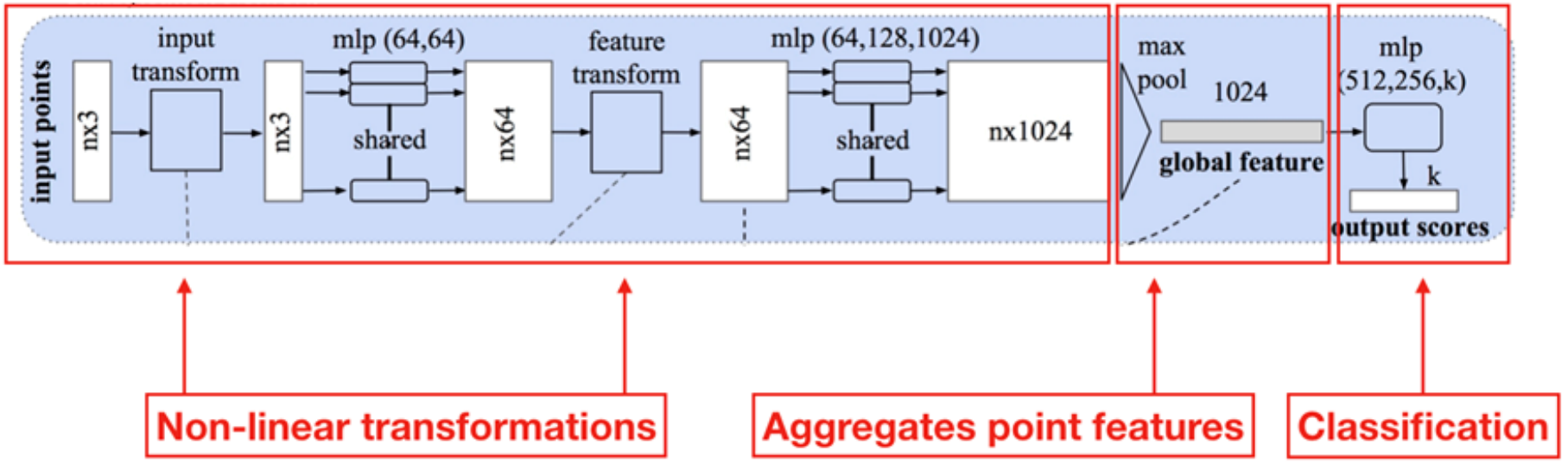
Reference: A. Lang, et al., PointPillars: Fast encoders for object detection from point clouds, CVPR 2019, https://arxiv.org/abs/1812.05784.

PointNet

It uses a shared *multi-layer perceptron* (MLP) to map each of the n points from 3D (x, y, z) to 64 dimensions (64-64 nodes). The mapping is identical on the n points. It is repeated to map the n points from 64 dimensions to 1024 dimensions (64-128-1024 nodes). Max pooling is used to create a global feature vector. Finally, a three-layer fully-connected network is used to map the global feature vector to k output classification scores.



Teapot



Reference: R. Qi, et al., PointNet: Deep learning on point sets for 3D classification and segmentation, CVPR 2017, https://arxiv.org/abs/1612.00593

PointNet (1)

Input point cloud data

$n \times 3$

0.224, 0.325, 0.682

0.129, 0.262, 0.126

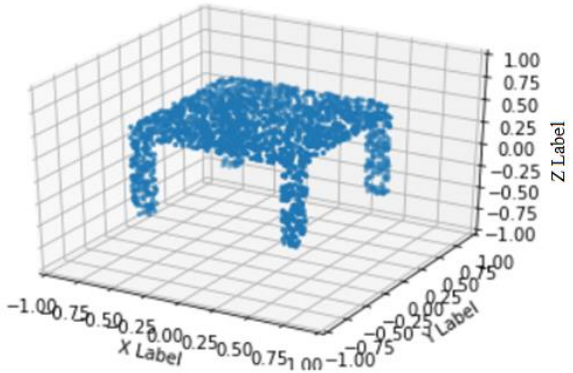
0.287, -0.024, 0.135

-0.331, 0.206, 0.554

…, …, …

Problem statement:
To perform classification based on a set of points.

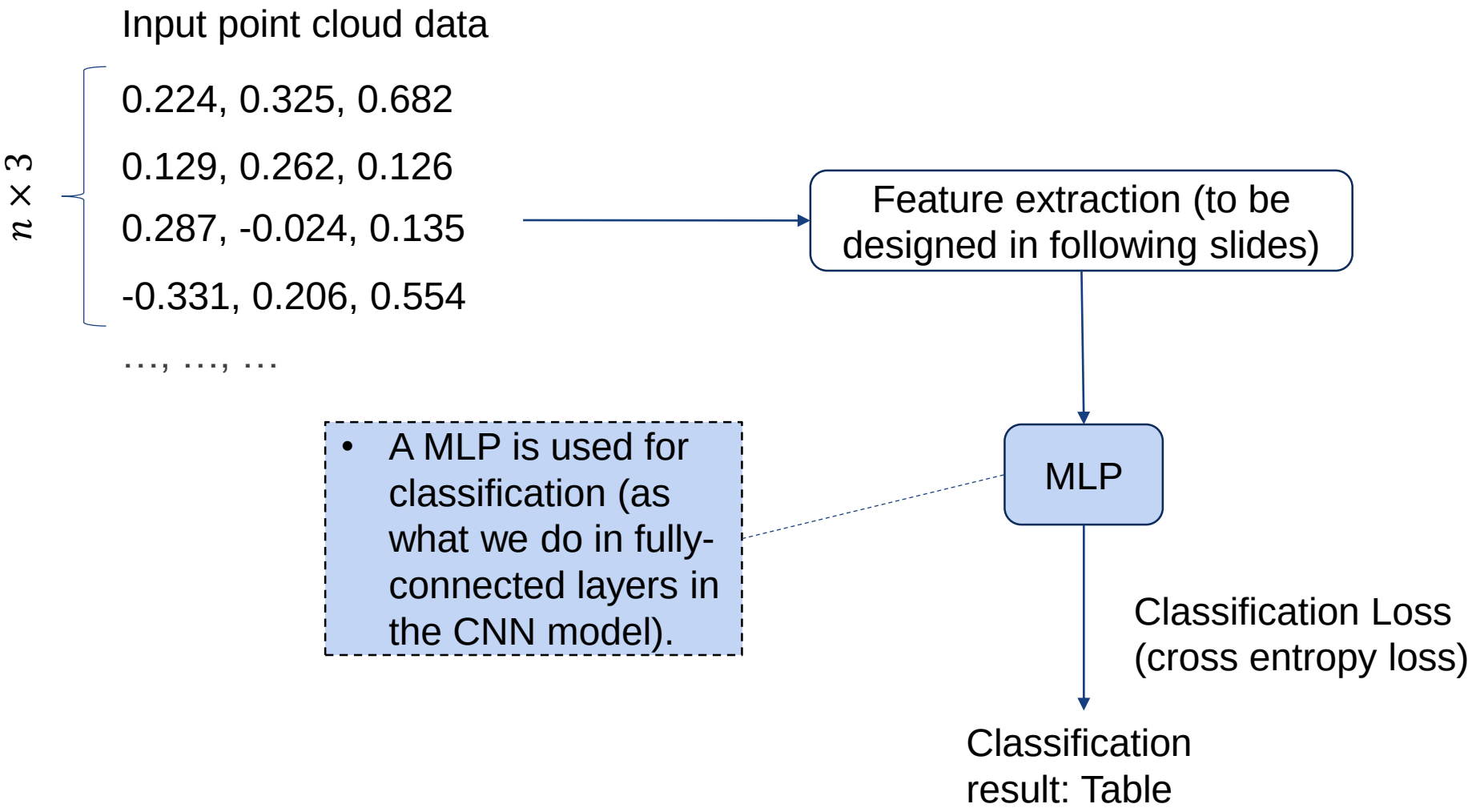
→ Classification result: Table



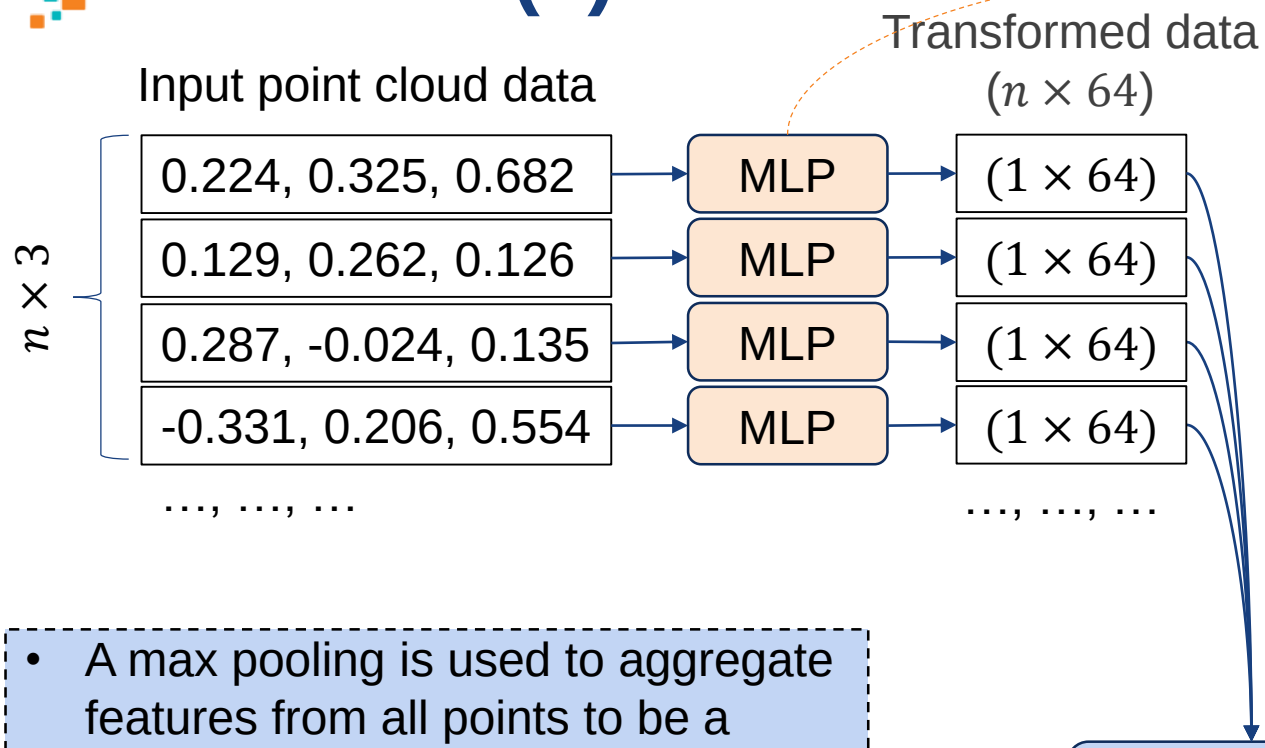
Reference

- [The author's presentation in CVPR 2017], https://www.youtube.com/watch?v=Cge-hot0Oc0
- [A Chinese tutorial video with English slides by the PointNet author in Bilibili], https://www.bilibili.com/video/BV1As411377S?from=search&seid=1823392618 0086373014
- Point cloud classification with PointNet, https://keras.io/examples/vision/pointnet/, created on May 2020.

PointNet (2)



PointNet (3)



- A MLP $(3, 64)$ is applied on each point to transform point to higher dimension space (recall what we have learned in ISSM).
- The MLP is sharable so it doesn't depend on the order of points.
- The # of nodes $(3, 64)$ in MLP is selected in experiments.

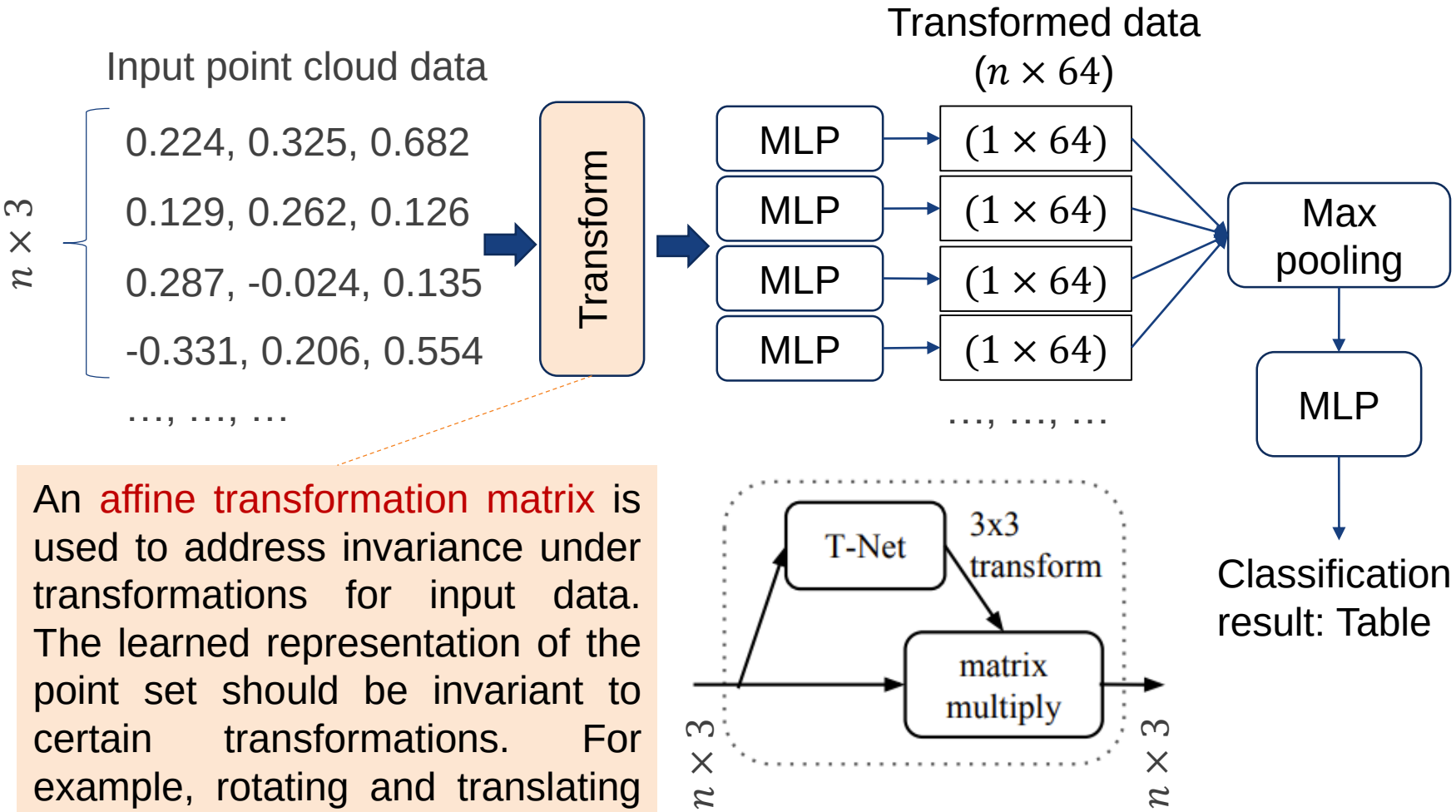
- A max pooling is used to aggregate features from all points to be a global feature for further classification.
- The max pooling function doesn't depend on the order of points.

Max pooling → MLP → Classification result: Table

Discussions: Why not other ideas? (quoted from the original paper)

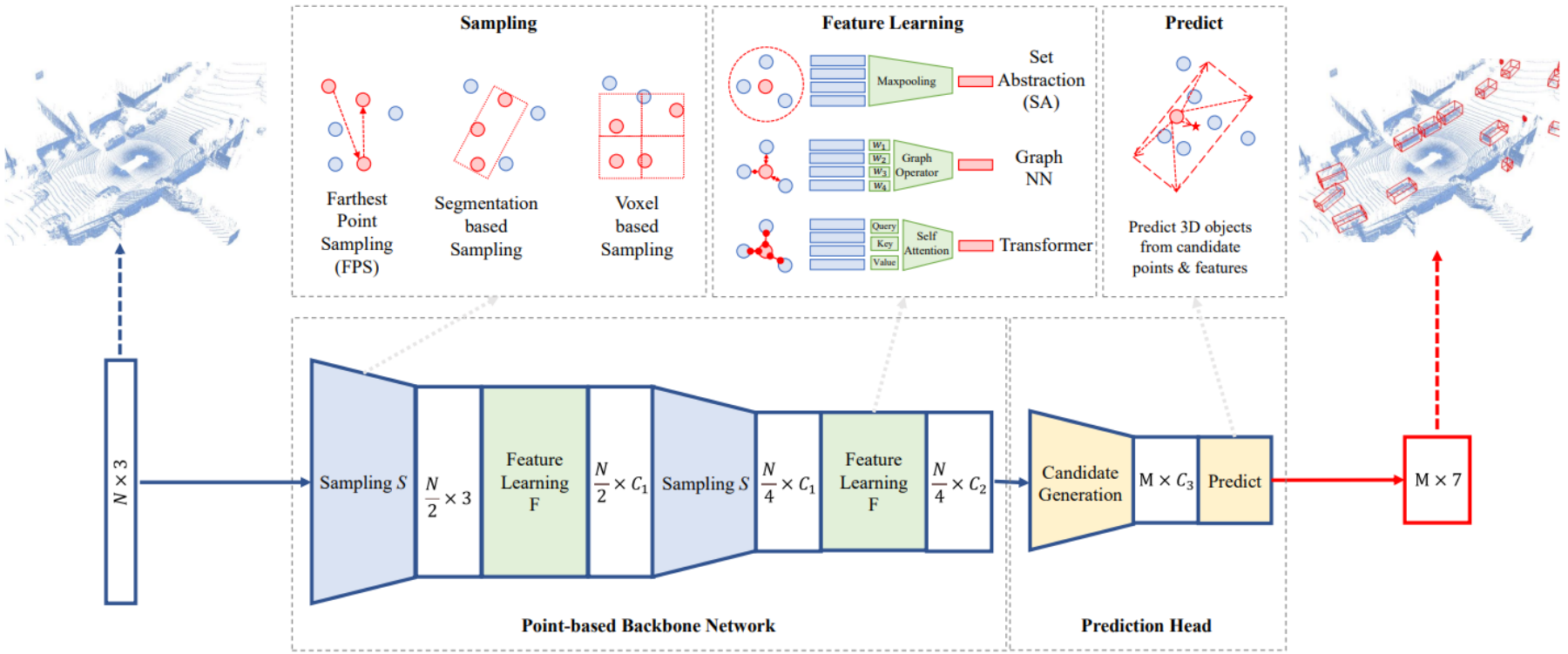
- Sort inputs
- →While sorting sounds like a simple solution, in high dimensional space there in fact does not exist an ordering that is stable w.r.t. point perturbations in the general sense.
- Treat the input as a sequence to train an RNN, but augment the training data by all kinds of permutations.
- → While RNN has relatively good robustness to input ordering for sequences with small length (dozens), it's hard to scale to thousands of input elements, which is the common size for point sets.

PointNet (4)



An affine transformation matrix is used to address invariance under transformations for input data. The learned representation of the point set should be invariant to certain transformations. For example, rotating and translating points all together should not modify the category of the points.

Point-based object detection



Reference: 3D Object Detection for Autonomous Driving: A Comprehensive Survey, IJCV, 2023, https://arxiv.org/abs/2206.09474

PointRCNN

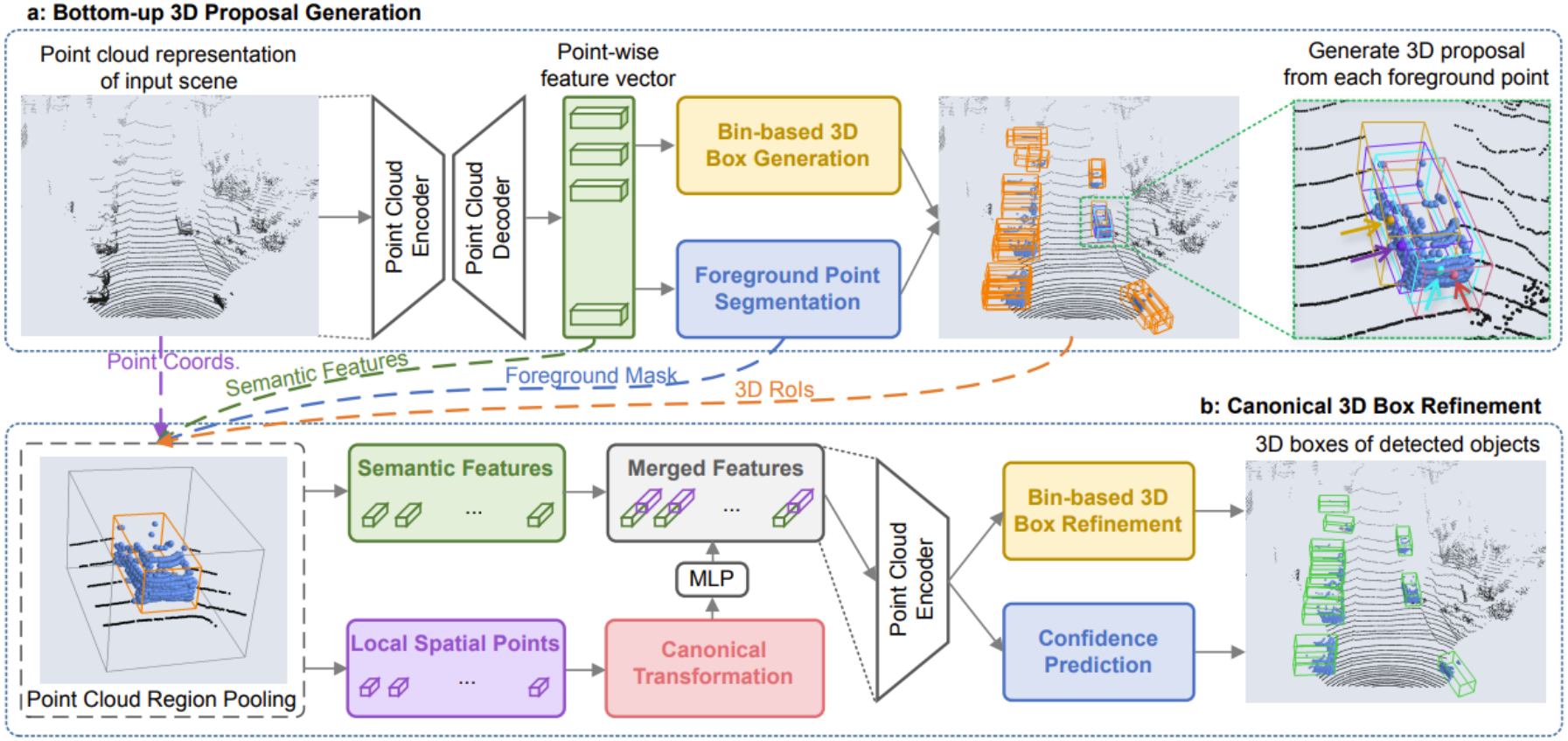
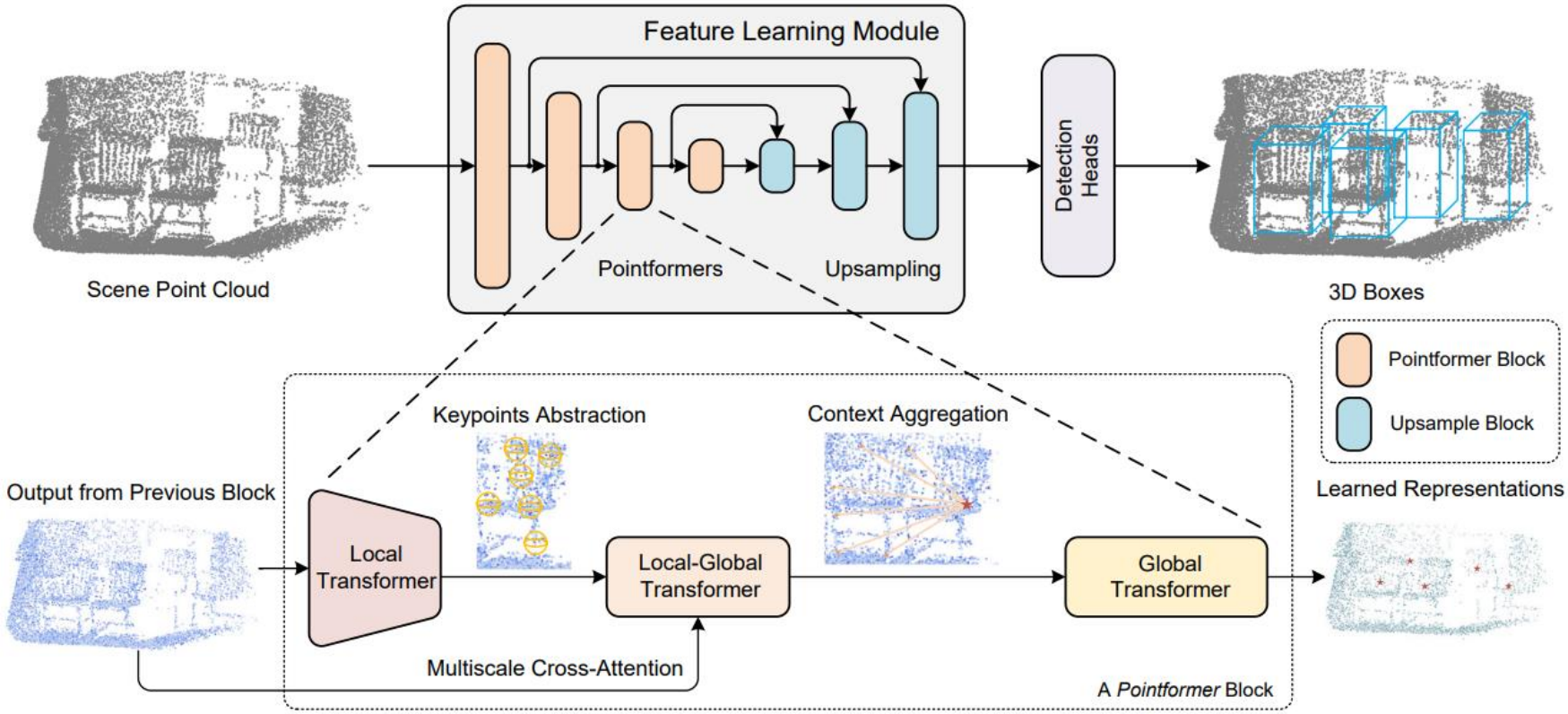


Figure 2. The **PointRCNN** architecture for 3D object detection from point cloud. The whole network consists of two parts: (a) for generating 3D proposals from raw point cloud in a bottom-up manner. (b) for refining the 3D proposals in canonical coordinate.

Reference: S. Shi, *et al.*, "PointRCNN: 3D object proposal generation and detection from point cloud," CVPR 2019, https://arxiv.org/abs/1812.04244

Pointformer

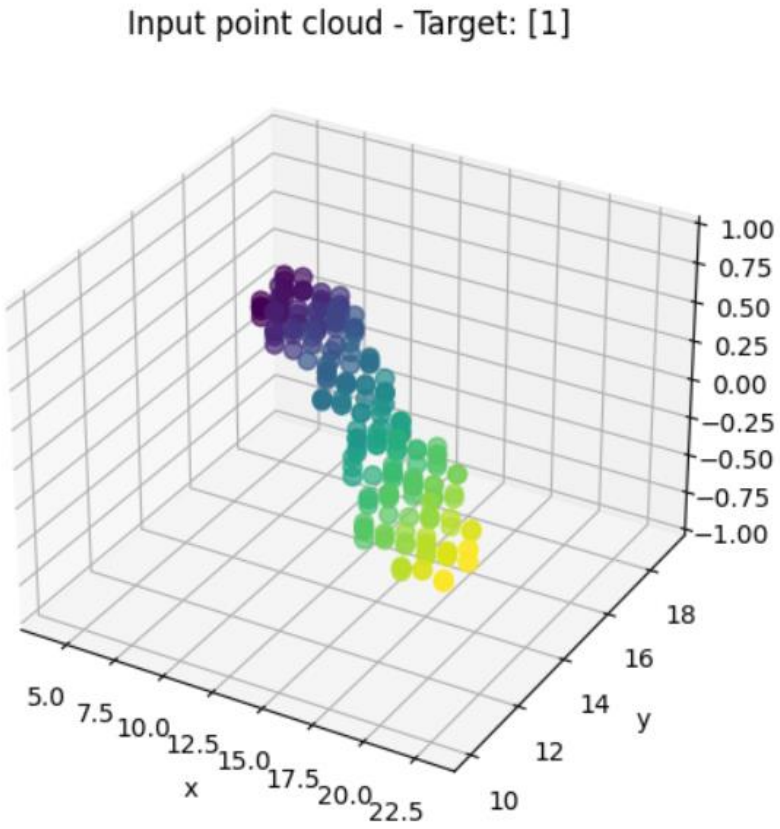
The Pointformer for 3D object detection in point clouds. It consists of three parts: a Local Transformer to model interactions in the local region; a Local-Global Transformer to integrate local features with global information; a Global Transformer to capture context-aware representations at the scene level.

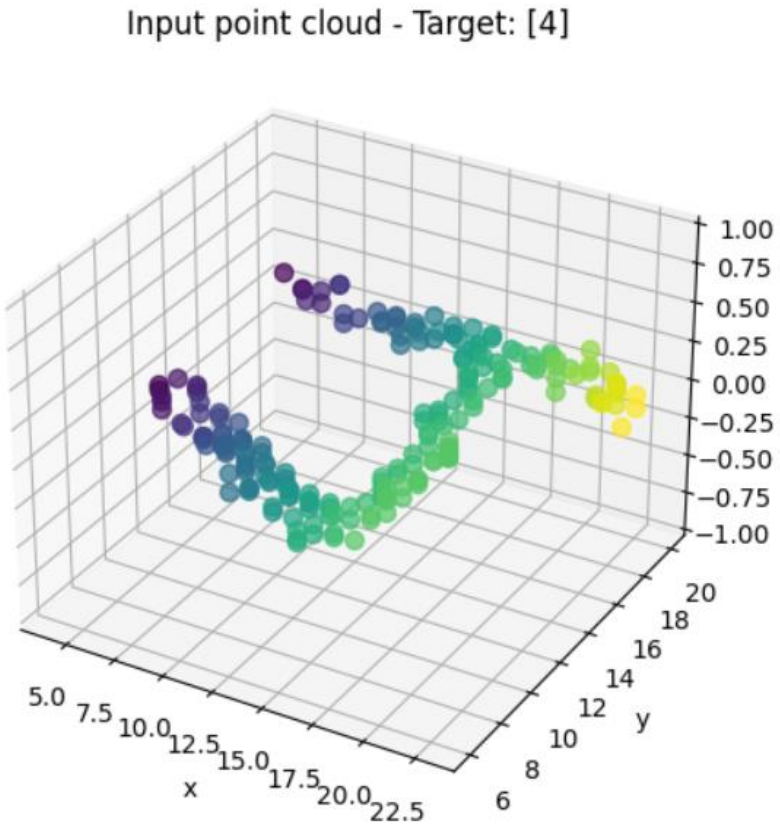


Reference: X. Pan, et al., 3D object detection with pointformer, CVPR 2021, https://arxiv.org/abs/2012.11409

Point cloud classification using PointNet

Reference: https://datascienceub.medium.com/pointnet-implementation-explained-visually-c7e300139698





Thank you!

Dr TIAN Jing
Email: tianjing@nus.edu.sg