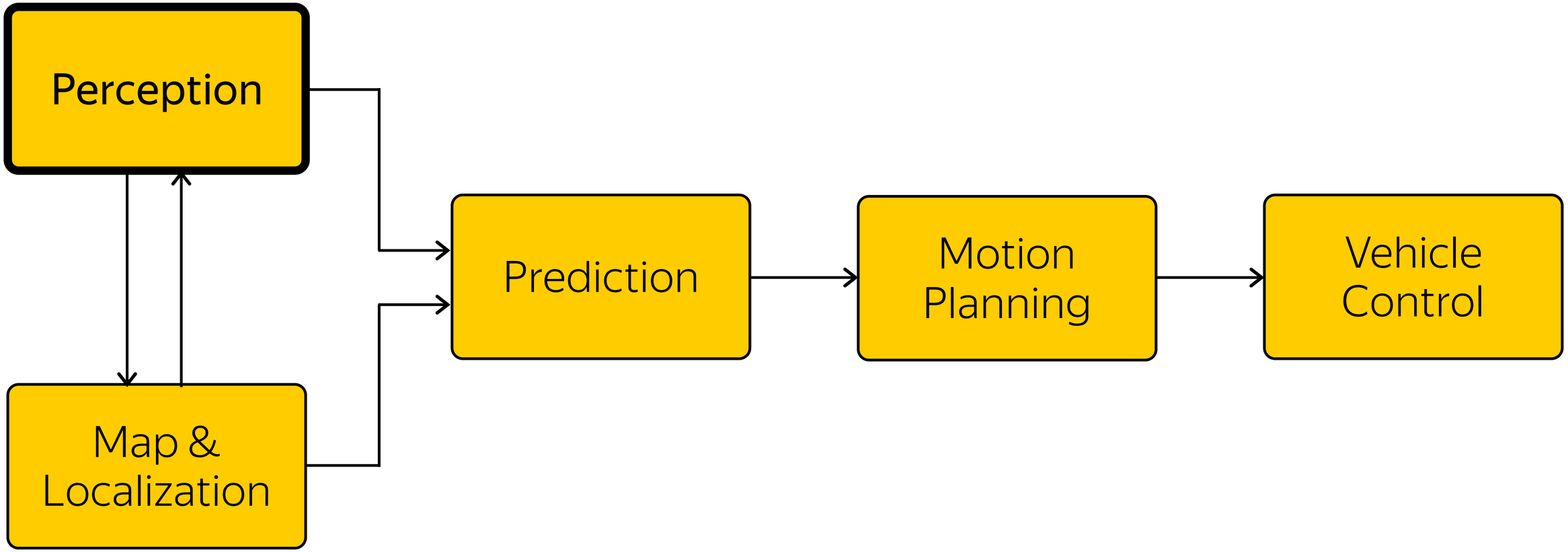
Yandex

Yandex

Self-driving Car Perception

High-level view



Dynamic objects

- Car/truck/bus/...
- Pedestrians
- Bicycles
- Rider (Bicyclist,Motorcyclist)
- Animals
- Airplanes (?)
- Arbitrary movable objects (?)

Dynamic objects

- Car/truck/bus/…
- Pedestrians
- Bicycles
- Rider (Bicyclist,Motorcyclist)
- Animals
- Airplanes (?)
- Arbitrary movable objects (?)

Frequent 3D objects – specific detectors

Rare cases – general detector

More dynamic objects

Objects with temporal state

- Traffic light current signal
- Cars stop lights / turn lights

Static objects/stuff

- Road boundaries
- Lane markings
- Curb
- Buildings
- Road signs
- Parked cars
- Road temporal constructions

Static objects/stuff

- Road boundaries
- Lane markings
- Curb
- Buildings
- Road signs
- Parked cars
- Road temporal constructions

Changes are rare -> retrieve from HD map

Frequent changes – on-the-board detector

Sensors for Perception

Camera

Cheep sensor -> attractive for deploy

The only option for detection of color-coded information (i.e traffic light state, car stop signal)

Sensitive to ego car motion, weather, dirt

30 fps

Radar

Of-the-shelf car detections

Returns tangential component of speed

False positives on any metallic stuff

Multiple detections on one object

13 fps

Lidar

Expensive but effective sensor

Produces sparse point cloud

Closer an object to the device, denser point cloud of the object

10 fps

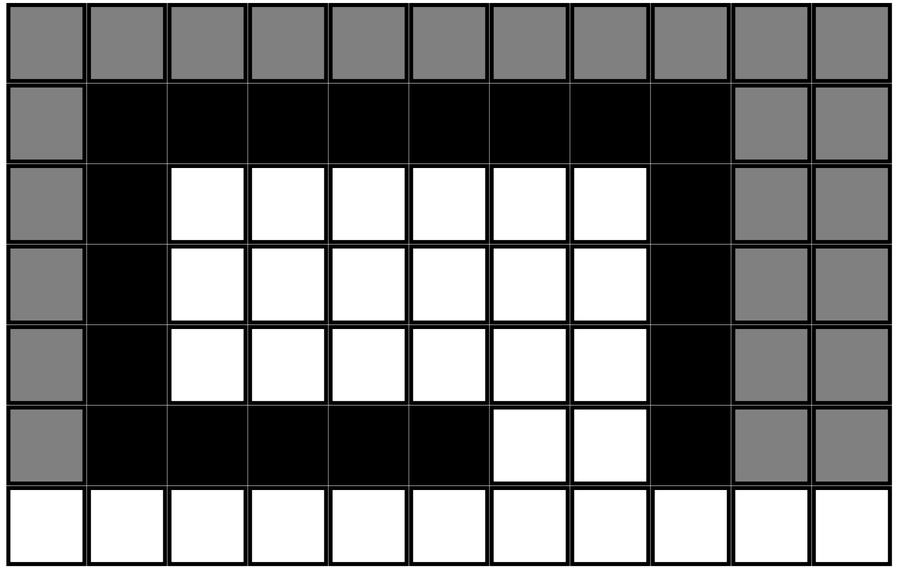
Static Obstacles Detection

Occupancy grid

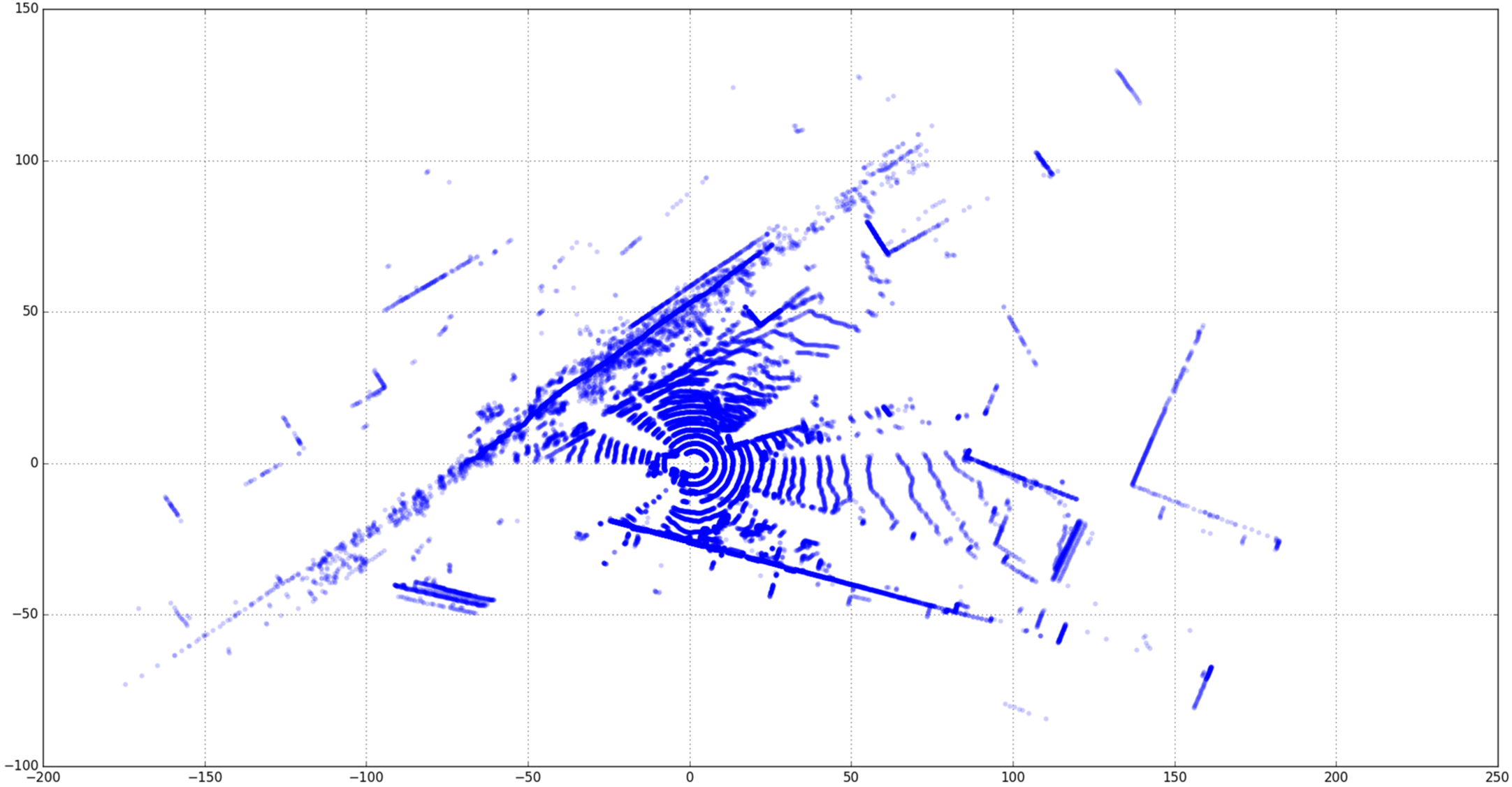
Occupancy grid – a way of free space representation

Environment map is represented as array of cells of predefined size.

Every cell holds a probability value that cell is occupied



Lidar bird-eye view

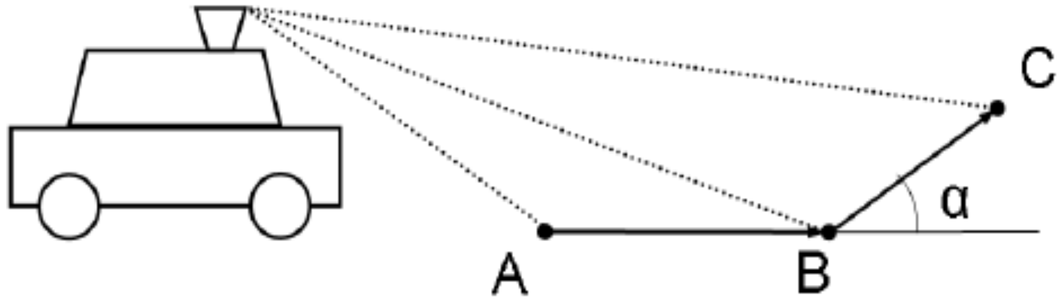


VSCAN

Slice points – interval [0, 2m]

Build 3D grid in spherical coordinates

Analysis of grid vertical slice



If A, B, C are ground readings

$\alpha \sim 0$

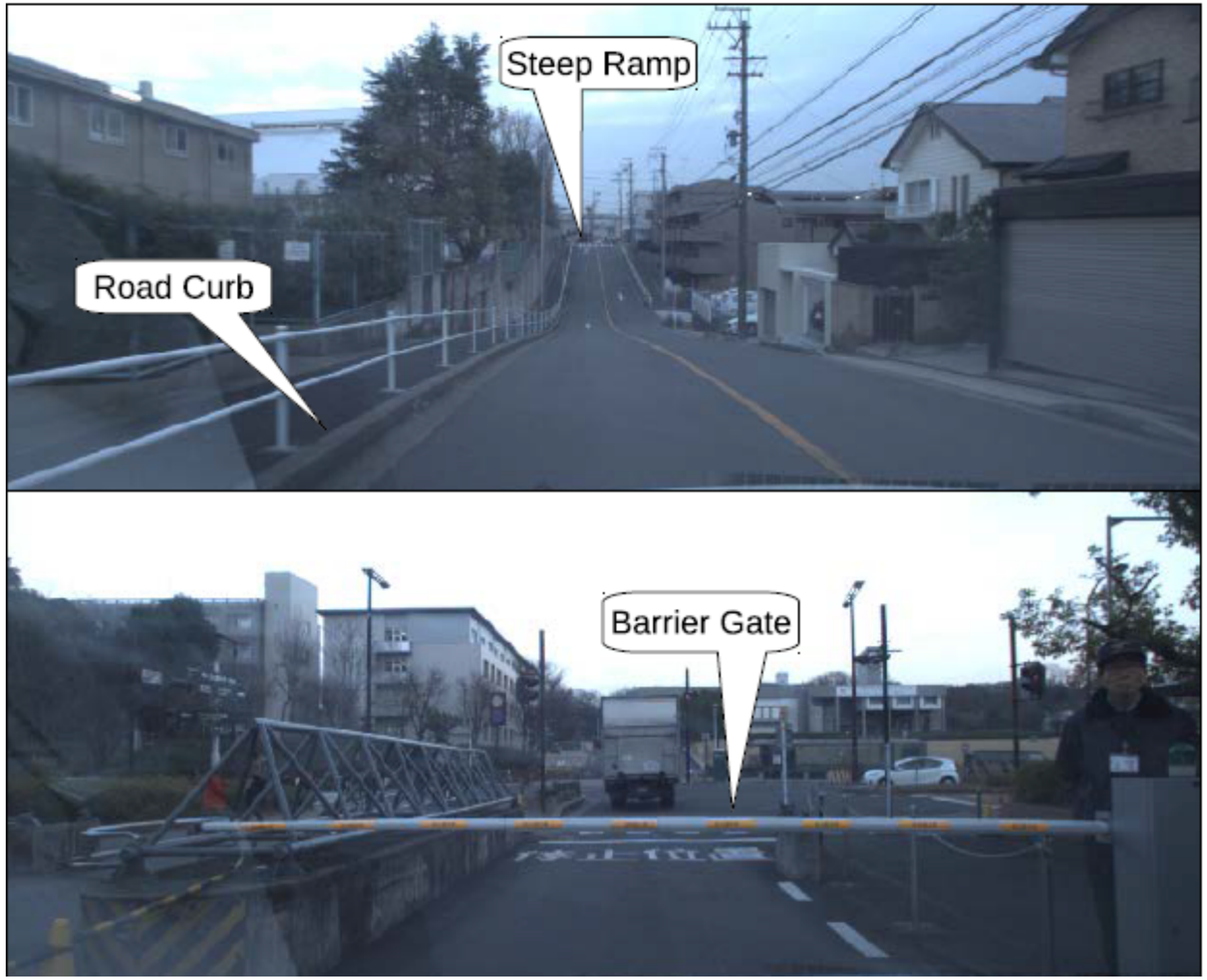
$\Rightarrow \cos \alpha \sim 1$

[A.Petrovskaya, S.Thrun “Model based vehicle tracking for autonomous driving in urban environment”, RSS 2008]

Robust VSCAN

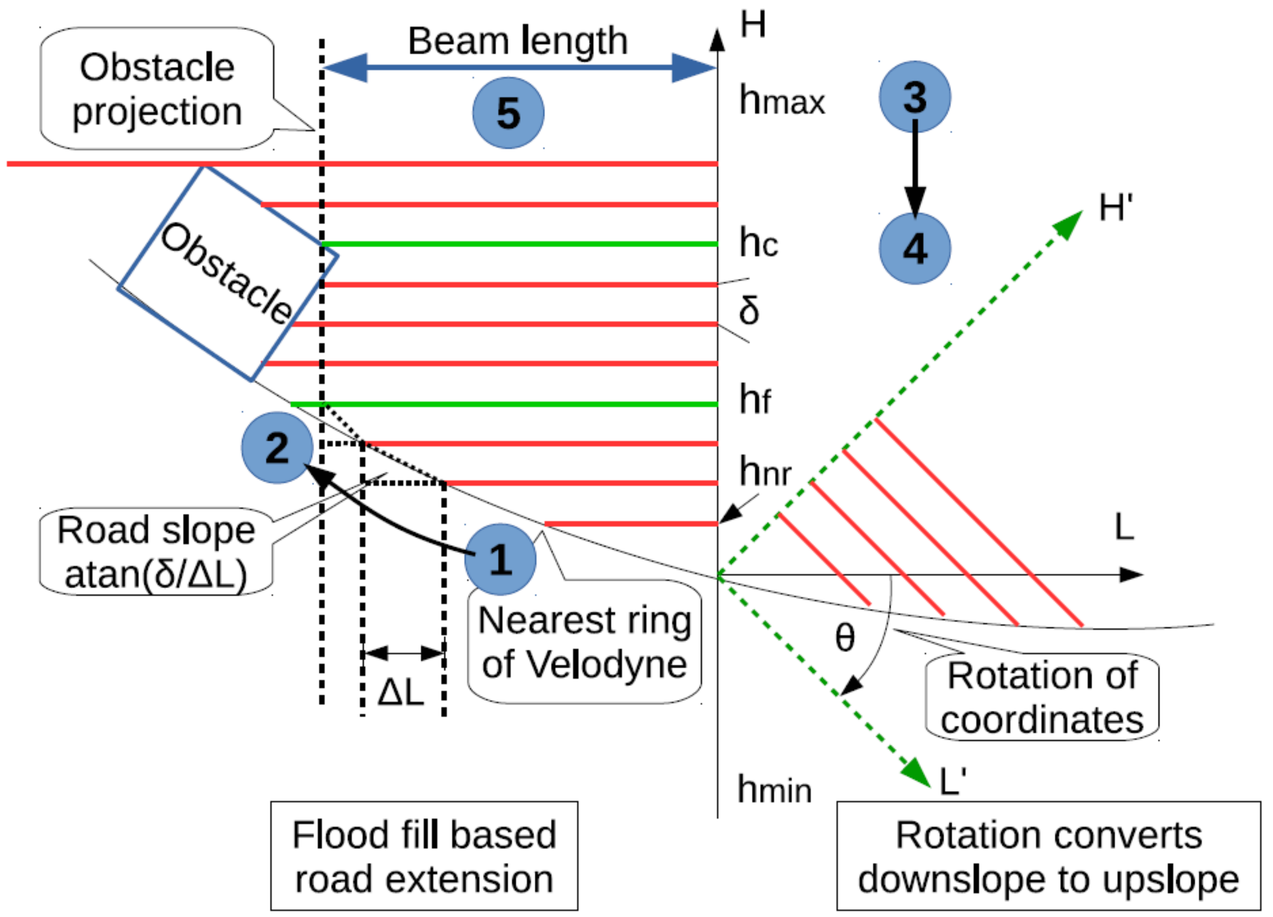
Two corner cases of VSCAN:

- Steep ramp of road
- Overhung obstacle



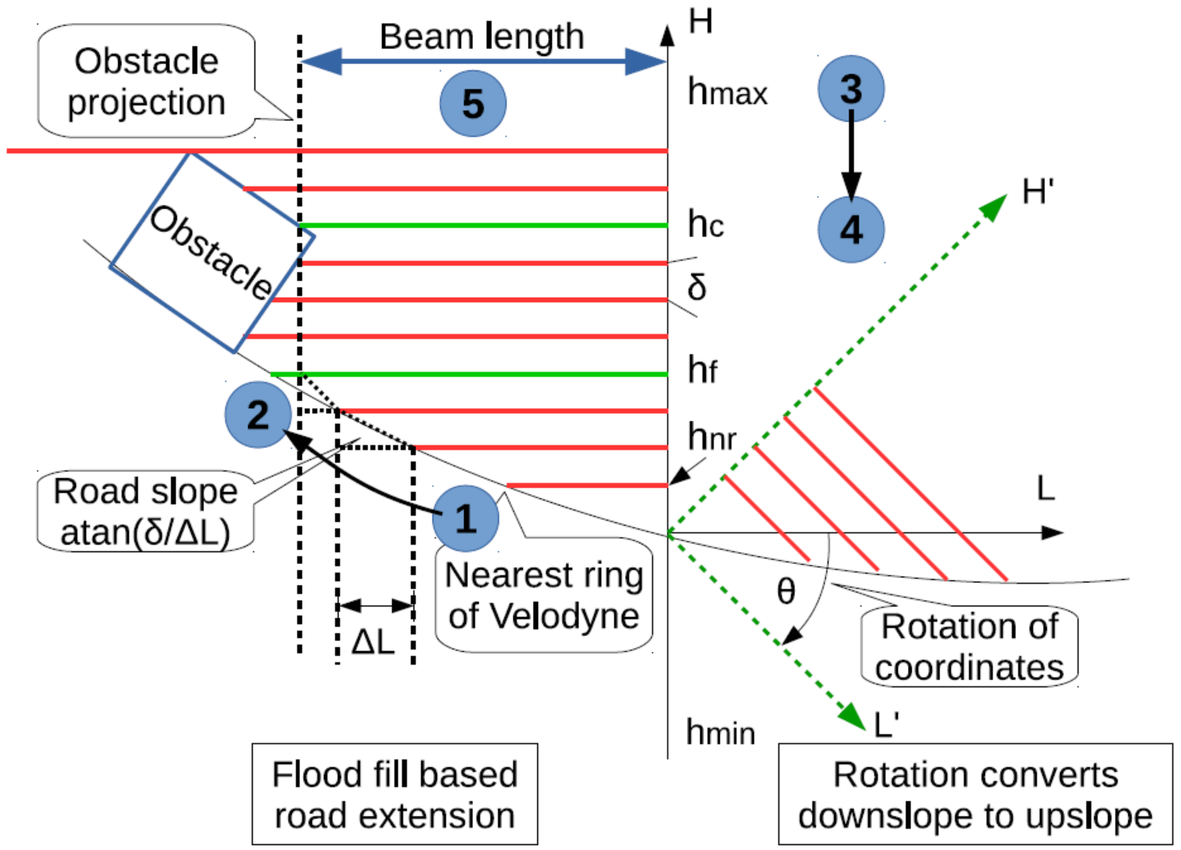
[M.He et al. “Robust Virtual Scan for Obstacle Detection in Urban Environments”, IV 2016]

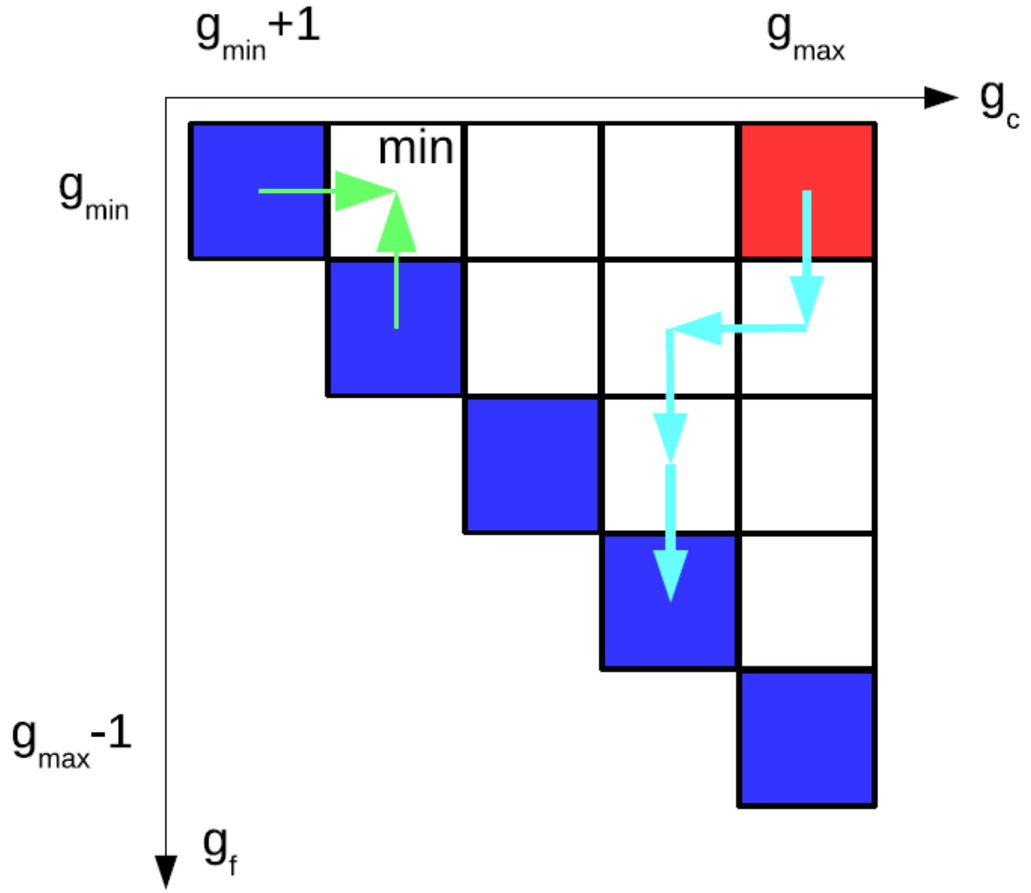
Robust VSCAN



[M.He et al. “Robust Virtual Scan for Obstacle Detection in Urban Environments”, IV 2016]

Robust VSCAN





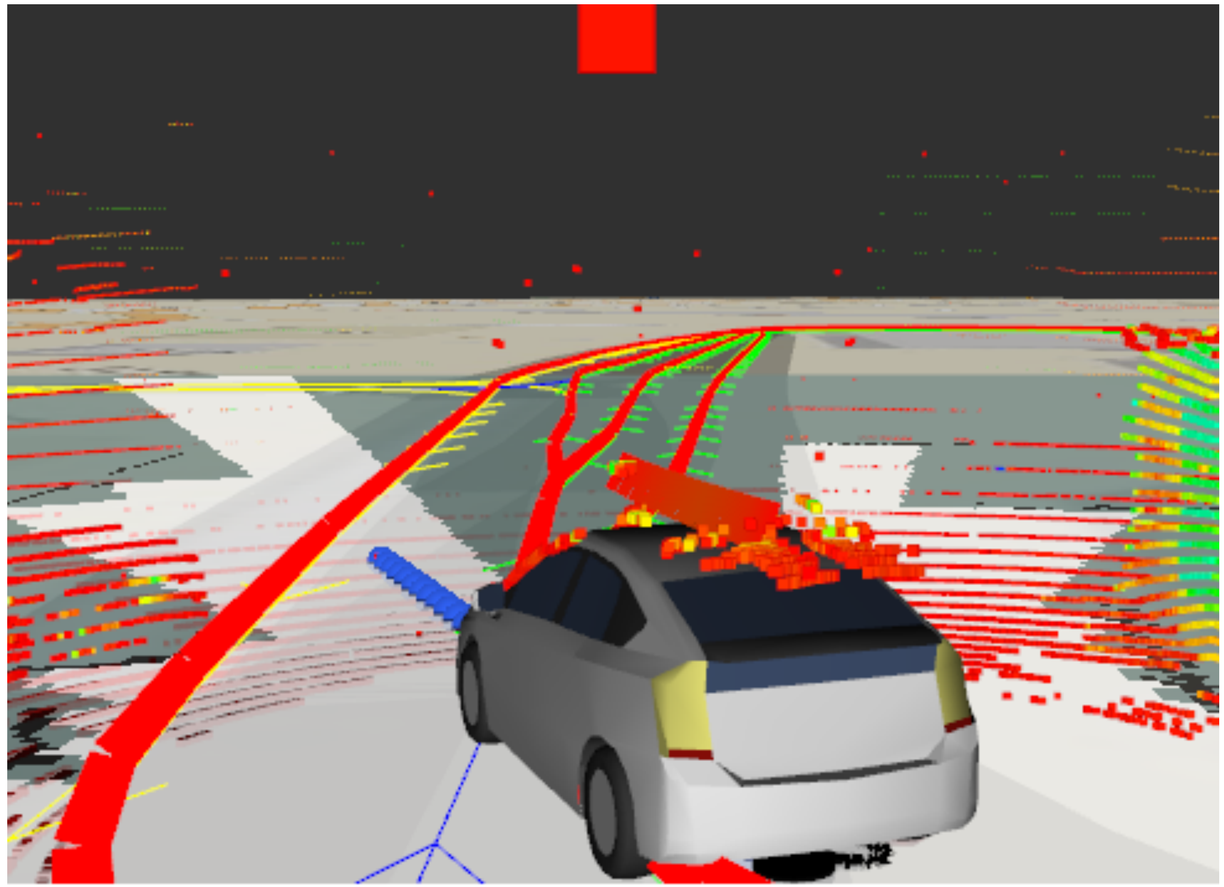
[M.He et al. “Robust Virtual Scan for Obstacle Detection in Urban Environments”, IV 2016]

Lidar points filtering

Lidar can be sensitive to rain and snow

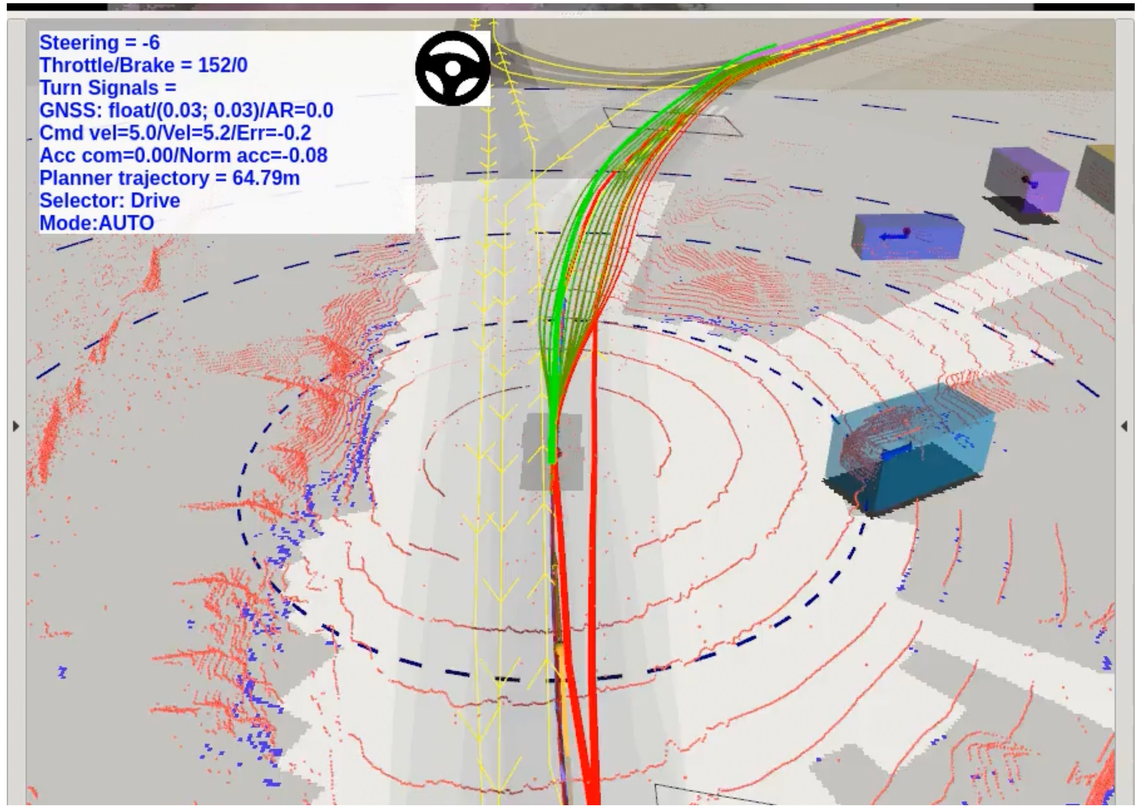
Voxel filter:

- Split points in voxels in spherical coordinates
- Voxels with few points contains outliers



Lidar points filtering

https://yadi.sk/i/0Sj-IKDI3UVKfZ



PointNet

Vscan и Voxel недостаточно хороши

Хочется использовать нейросети

- Unordered. Unlike pixel arrays in images or voxel arrays in volumetric grids, point cloud is a set of points without specific order. In other words, a network that consumes N 3D point sets needs to be invariant to $N!$ permutations of the input set in data feeding order.
- Interaction among points. The points are from a space with a distance metric. It means that points are not isolated, and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the combinatorial interactions among local structures.
- Invariance under transformations. As a geometric object, 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 global point cloud category nor the segmentation of the points.

PointNet

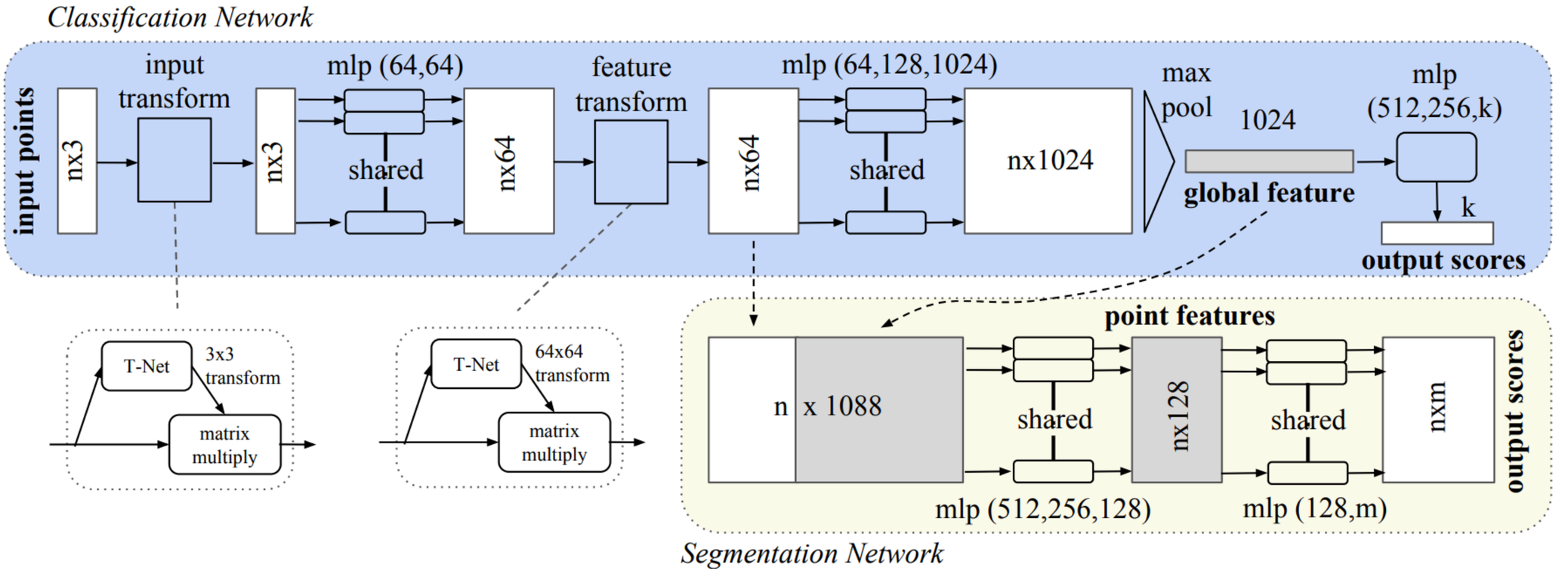


Figure 2. **PointNet Architecture.** The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling. The output is classification scores for k classes. The segmentation network is an extension to the classification net. It concatenates global and local features and outputs per point scores. "mlp" stands for multi-layer perceptron, numbers in bracket are layer sizes. Batchnorm is used for all layers with ReLU. Dropout layers are used for the last mlp in classification net.

PointNet

https://github.com/charlesq34/pointnet/

```
with tf.variable_scope('transform_net1') as sc:
    transform = input_transform_net(point_cloud, is_training, bn_decay, K=3)
point_cloud_transformed = tf.matmul(point_cloud, transform)
input_image = tf.expand_dims(point_cloud_transformed, -1)

net = tf_util.conv2d(input_image, 64, [1,3],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv1', bn_decay=bn_decay)
net = tf_util.conv2d(net, 64, [1,1],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv2', bn_decay=bn_decay)

with tf.variable_scope('transform_net2') as sc:
    transform = feature_transform_net(net, is_training, bn_decay, K=64)
end_points['transform'] = transform
net_transformed = tf.matmul(tf.squeeze(net, axis=[2]), transform)
net_transformed = tf.expand_dims(net_transformed, [2])
```

```
net = tf_util.conv2d(net_transformed, 64, [1,1],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv3', bn_decay=bn_decay)
net = tf_util.conv2d(net, 128, [1,1],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv4', bn_decay=bn_decay)
net = tf_util.conv2d(net, 1024, [1,1],
                     padding='VALID', stride=[1,1],
                     bn=True, is_training=is_training,
                     scope='conv5', bn_decay=bn_decay)

# Symmetric function: max pooling
net = tf_util.max_pool2d(net, [num_point,1],
                         padding='VALID', scope='maxpool')
```

PointNet

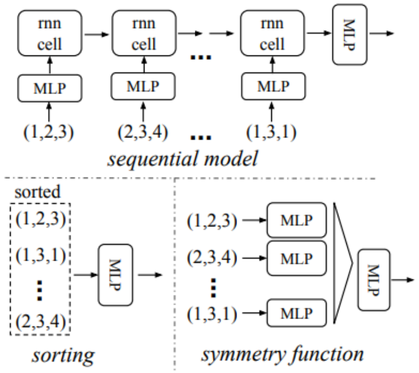
	mean	aero	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
# shapes		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
Wu [27]	-	63.2	-	-	-	73.5	-	-	-	74.4	-	-	-	-	-	-	74.8
Yi [29]	81.4	81.0	78.4	77.7	**75.7**	87.6	61.9	**92.0**	85.4	**82.5**	**95.7**	**70.6**	91.9	**85.9**	53.1	69.8	75.3
3DCNN	79.4	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
Ours	**83.7**	**83.4**	**78.7**	**82.5**	74.9	**89.6**	**73.0**	91.5	**85.9**	80.8	95.3	65.2	**93.0**	81.2	**57.9**	**72.8**	**80.6**

Table 2. **Segmentation results on ShapeNet part dataset.** Metric is mIoU(%) on points. We compare with two traditional methods [27] and [29] and a 3D fully convolutional network baseline proposed by us. Our PointNet method achieved the state-of-the-art in mIoU.

PointNet

	#params	FLOPs/sample
PointNet (vanilla)	0.8M	148M
PointNet	3.5M	440M
Subvolume [18]	16.6M	3633M
MVCNN [23]	60.0M	62057M

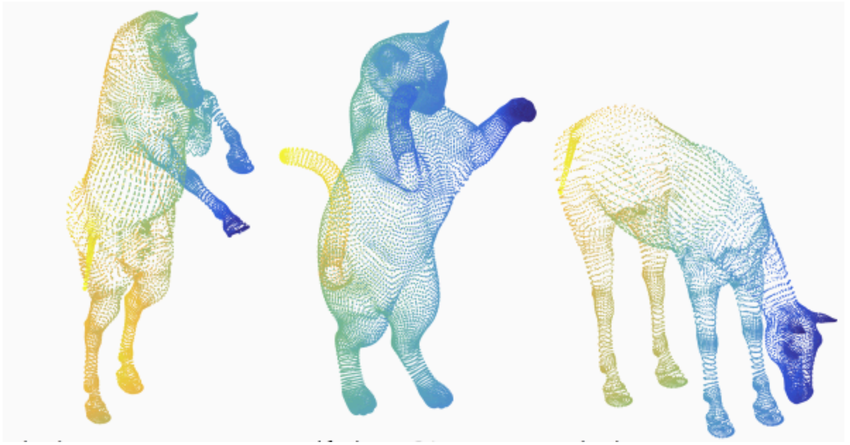
Table 6. **Time and space complexity of deep architectures for 3D data classification.** PointNet (vanilla) is the classification PointNet without input and feature transformations. FLOP stands for floating-point operation. The "M" stands for million. Subvolume and MVCNN used pooling on input data from multiple rotations or views, without which they have much inferior performance.



	accuracy
MLP (unsorted input)	24.2
MLP (sorted input)	45.0
LSTM	78.5
Attention sum	83.0
Average pooling	83.8
Max pooling	**87.1**

Figure 5. **Three approaches to achieve order invariance.** Multi-layer perceptron (MLP) applied on points consists of 5 hidden layers with neuron sizes 64,64,64,128,1024, all points share a single copy of MLP. The MLP close to the output consists of two layers with sizes 512,256.

PointNet++



PointNet++

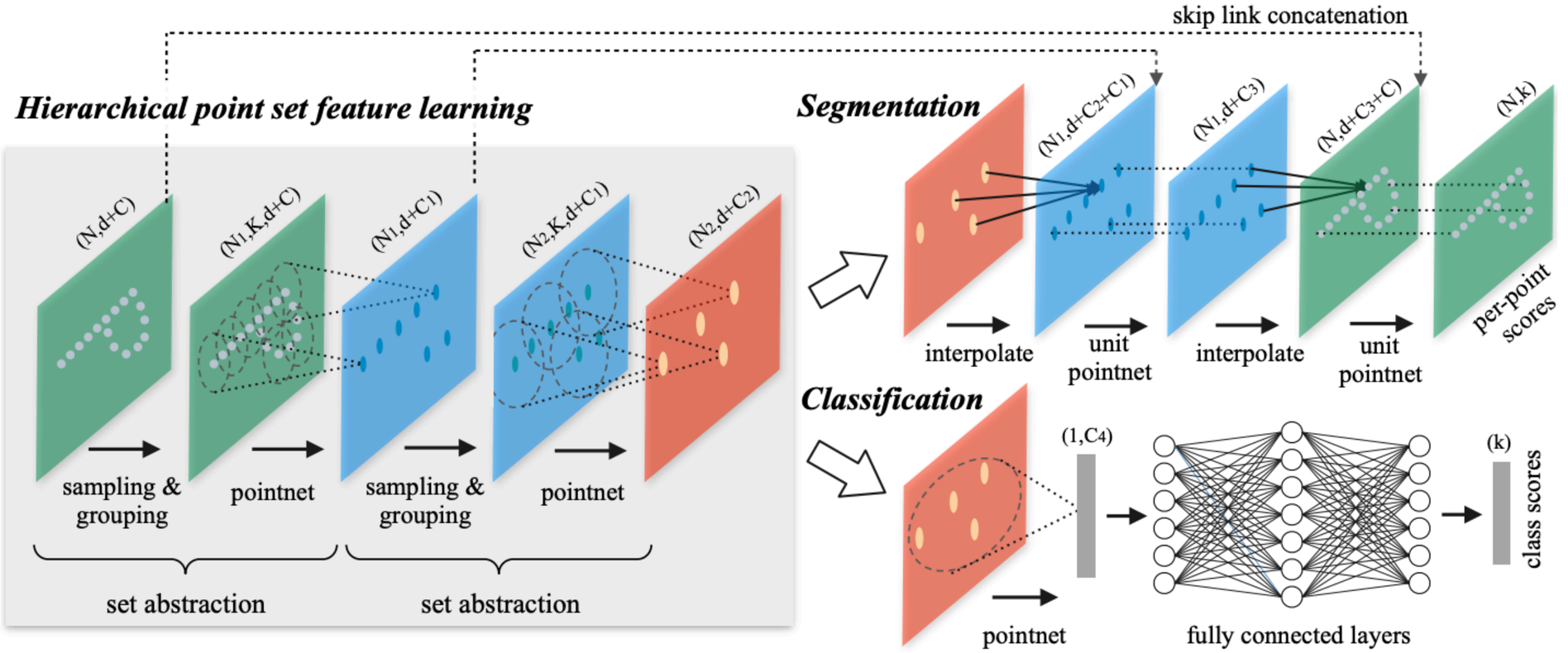


Figure 2: Illustration of our hierarchical feature learning architecture and its application for set segmentation and classification using points in 2D Euclidean space as an example. Single scale point grouping is visualized here. For details on density adaptive grouping, see Fig. 3

VoxelNet

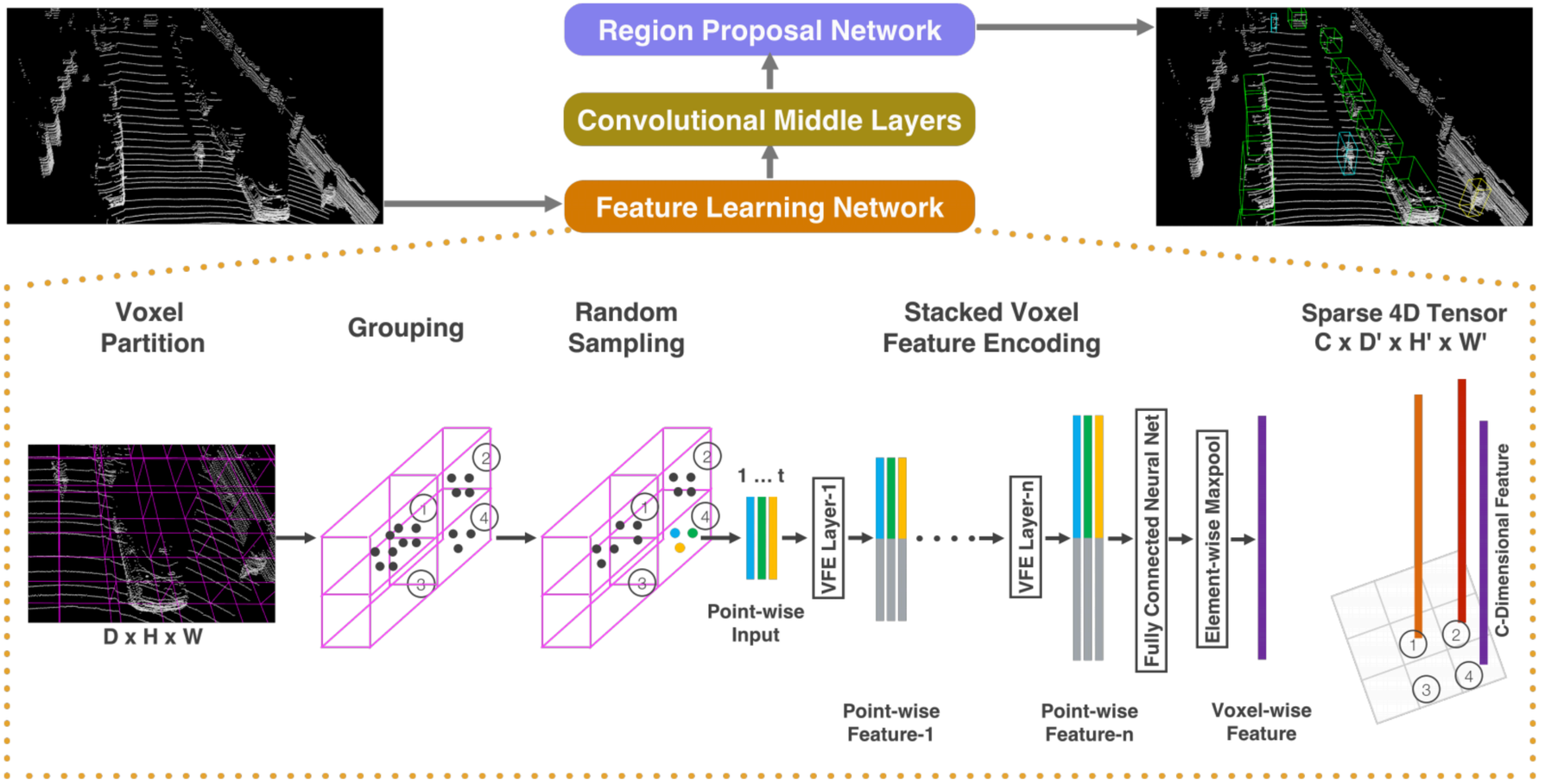


Figure 2. VoxelNet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information. The space is represented as a sparse 4D tensor. The convolutional middle layers processes the 4D tensor to aggregate spatial context. Finally, a RPN generates the 3D detection.

PointPillars

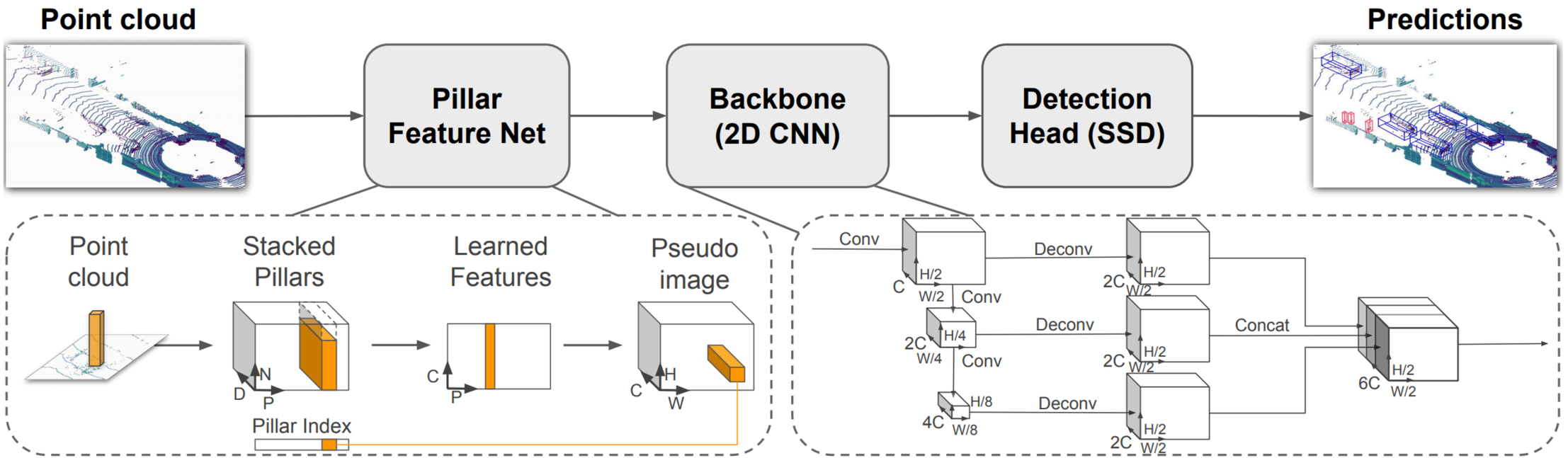


Figure 2. Network overview. The main components of the network are a Pillar Feature Network, Backbone, and SSD Detection Head. See Section 2 for more details. The raw point cloud is converted to a stacked pillar tensor and pillar index tensor. The encoder uses the stacked pillars to learn a set of features 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: here we show the backbone dimensions for the car network.

PointPillars

Method	Modality	Speed (Hz)	mAP	Car			Pedestrian			Cyclist		
			Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [2]	Lidar & Img.	2.8	N/A	86.02	76.90	68.49	N/A	N/A	N/A	N/A	N/A	N/A
Cont-Fuse [15]	Lidar & Img.	16.7	N/A	88.81	85.83	77.33	N/A	N/A	N/A	N/A	N/A	N/A
Roarnet [25]	Lidar & Img.	10	N/A	88.20	79.41	70.02	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN [11]	Lidar & Img.	10	64.11	88.53	83.79	77.90	**58.75**	**51.05**	**47.54**	68.09	57.48	50.77
F-PointNet [21]	Lidar & Img.	5.9	65.39	88.70	84.00	75.33	58.09	50.22	47.20	75.38	61.96	54.68
HDNET [29]	Lidar & Map	20	N/A	89.14	**86.57**	78.32	N/A	N/A	N/A	N/A	N/A	N/A
PIXOR++ [29]	Lidar	35	N/A	**89.38**	83.70	77.97	N/A	N/A	N/A	N/A	N/A	N/A
VoxelNet [31]	Lidar	4.4	58.25	89.35	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
SECOND [28]	Lidar	20	60.56	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
PointPillars	Lidar	**62**	**66.19**	88.35	86.10	**79.83**	58.66	50.23	47.19	**79.14**	**62.25**	**56.00**

Table 1. Results on the KITTI test BEV detection benchmark.

Lidar limitations for static

Lidar produces sparse point cloud

Few information for far points

Fuse with image:

- Semantic segmentation for points labeling
- Depth completion