AVOD

Aggregate View Object Detection

ONE Kitti Object Detection Dataset

TWO AVOD Algorithm

THREE AVOD code

PART 1

Kitti Object Detection Dataset

http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d

Introduction

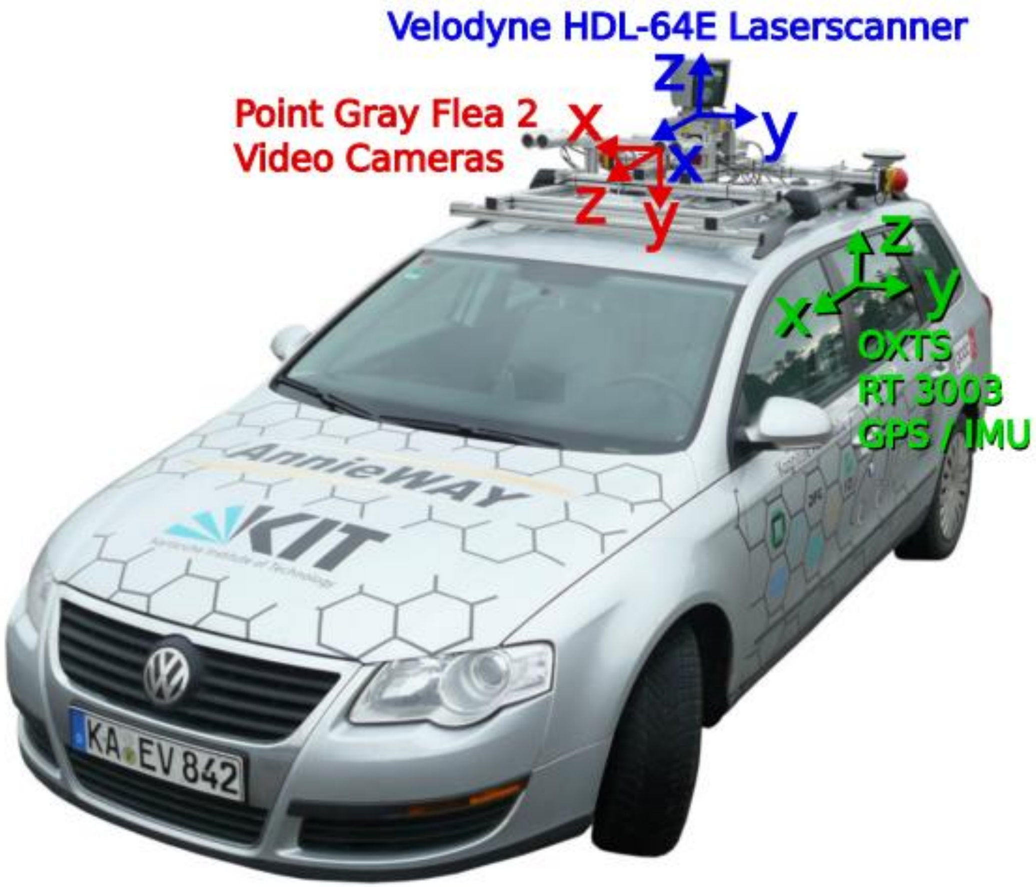


Fig. 1. **Recording Platform.** Our VW Passat station wagon is equipped with four video cameras (two color and two grayscale cameras), a rotating 3D laser scanner and a combined GPS/IMU inertial navigation system.

data collecting platform:

- 2 × PointGray Flea2 grayscale cameras
- 2 × PointGray Flea2 color cameras
- 1 × Velodyne HDL-64E rotating 3D laser scanner
- 1 × OXTS RT3003 inertial and GPS navigation sy stem
- 4 × Edmund Optics lenses

tasks of interest :

- stereo, optical flow, visual odometry, 3D object dete ction and 3D tracking.

3D Object Detection Dataset

class:

- ’Van’, ’Car’, ’Truck’, ’Pedestrian’, ’Person (sitting)’, ’Cyclist’, ’Tram’ ,’Misc’

3D bounding box overlap :

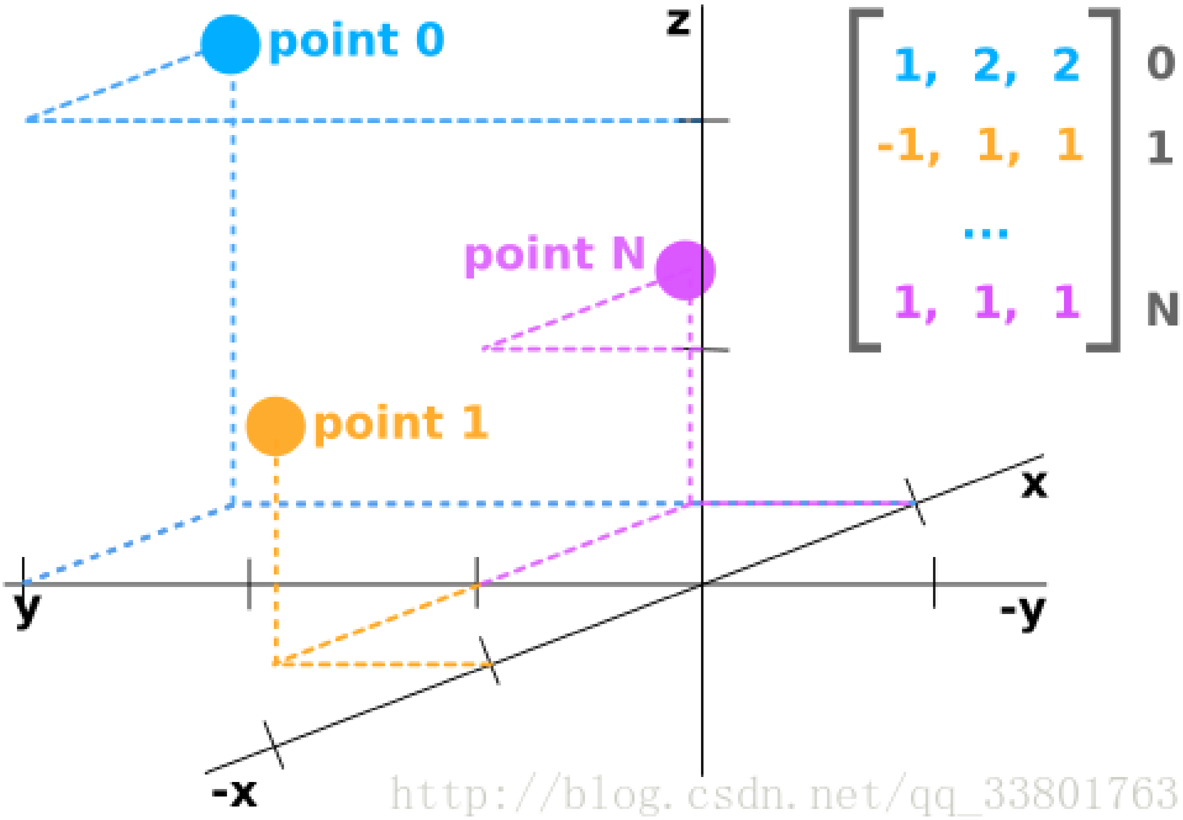
- Car:70%
- pedestrians 、cyclists:50%

Difficulty:

- **Easy:** Min. bounding box height: 40 Px, Max. occlusion level: Fully visible, Max. truncation: 15 %
- **Moderate:** Min. bounding box height: 25 Px, Max. occlusion level: Partly occluded, Max. truncation: 30 %
- **Hard:** Min. bounding box height: 25 Px, Max. occlusion level: Difficult to see, Max. truncation: 50 %

Type	truncated	occluded	alpha	bbox				dimensions			location			rotation_y	score
Truck	0.74	1	2.07	0	0	424.74	374	2.6	2.06	5.42	-3.17	1.77	5.46	1.57	
Car	0	0	-1.81	742.41	184.49	944.56	321.39	1.46	1.6	3.71	2.84	1.63	9.72	-1.54	
Van	0	0	-1.64	639.17	169.69	683.48	212.97	1.97	1.82	4.41	2.42	1.86	35.34	-1.57	
Car	0	0	-1.48	551.01	184.06	575.42	204.29	1.58	1.65	3.91	-3.85	2.52	59.59	-1.55	
DontCare	-1	-1	-10	579.35	178.15	633.56	201.11	-1	-1	-1	-1000	-1000	-1000	-10	
DontCare	-1	-1	-10	527.27	181.27	543.98	207.35	-1	-1	-1	-1000	-1000	-1000	-10	

Point Cloud

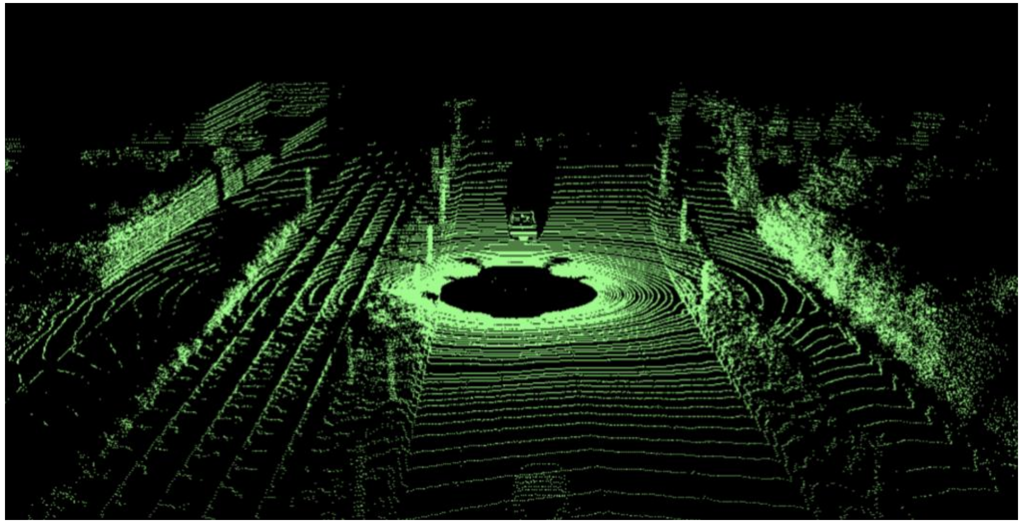


- **N*4 matrics:**

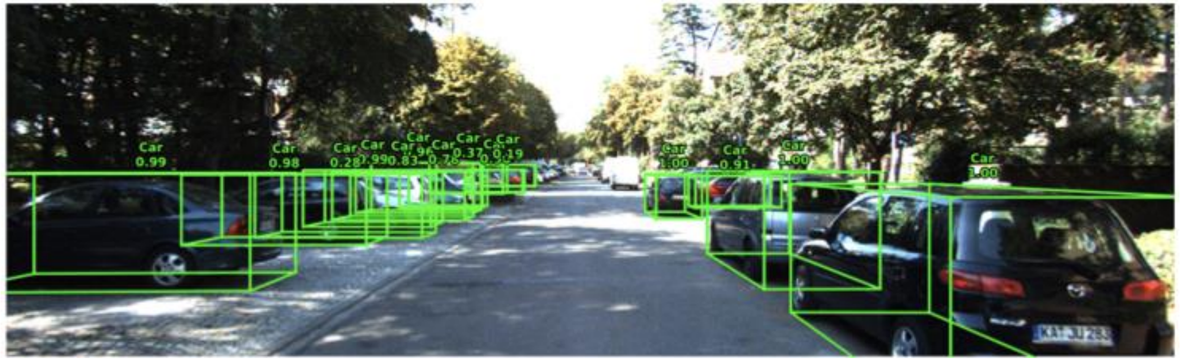
(x,y,z, reflectivity)

- **Project to image coordinate:**

x = P2 * R0_rect * Tr_velo_to_cam * y



Evaluation



$$a_o = \frac{area \quad B_p \cap B_{gt}}{area \quad B_p \cup B_{gt}}$$

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

- **Average Precision（AP）:**

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots,1\}} p_{interp}(r)$$

- **Average Orientation Similarity (AOS) :**

$$AOS = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots,1\}} \max_{\hat{r}:\hat{r} \geq r} s(\hat{r})$$

$$s(r) = \frac{1}{|D(r)|} \sum_{i \in D(r)} \frac{1 + cos\Delta_\theta^{(i)}}{2} \delta_i$$

PART 2

AVOD Algorithm

——Difficulties of 3d object detection

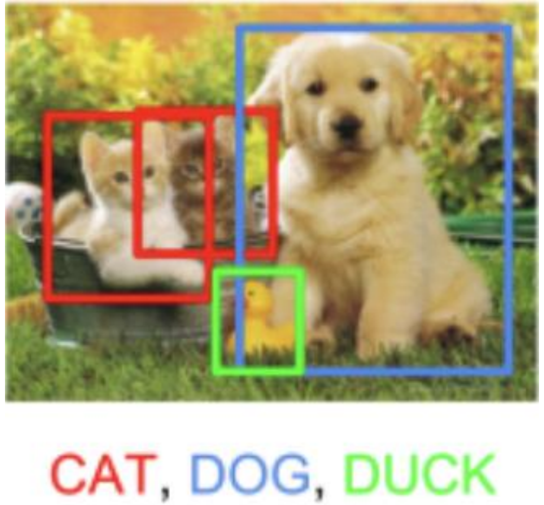
——AVOD architecture

——generating feature maps

——RPN network

——second stage detection network

Compare



1. algorithm:
Faster R- CNN, R-FCN, YOLO、SSD

2. input:
RGB images

3.output:
2D boxes、classes、confidence

1. algorithm:
3DOP，Deep3DBox，VoxelNet，MV3D

2. input:
RGB images、LIDAR

3.output:
2D boxes、3D boxes、classes、orientation、confidence

Difficulties of 3D Object Detection

- Needs to capture depth and orientation from environment
- More complex and computationally expensive processing at later detection stages
- Hard to detect smaller objects such as pedestrian and cyclists
- Methods only uses LIDAR point clouds,stereo depth maps or RGBD sensor depth maps

AVOD Architecture

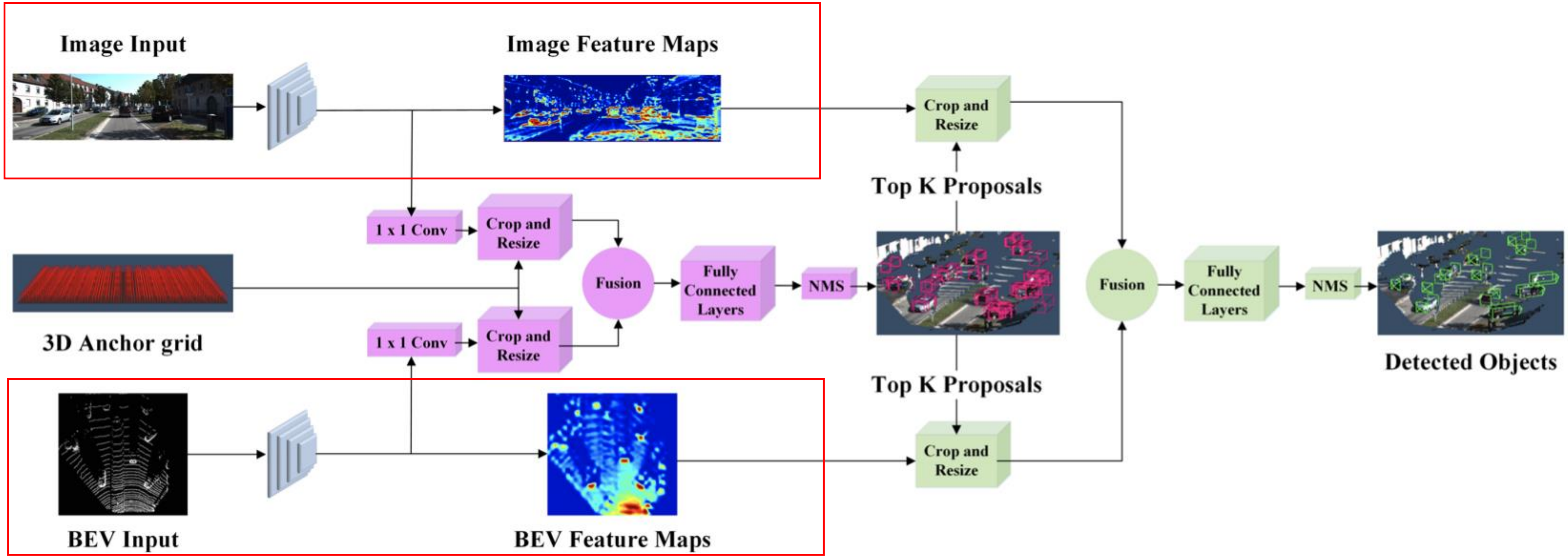
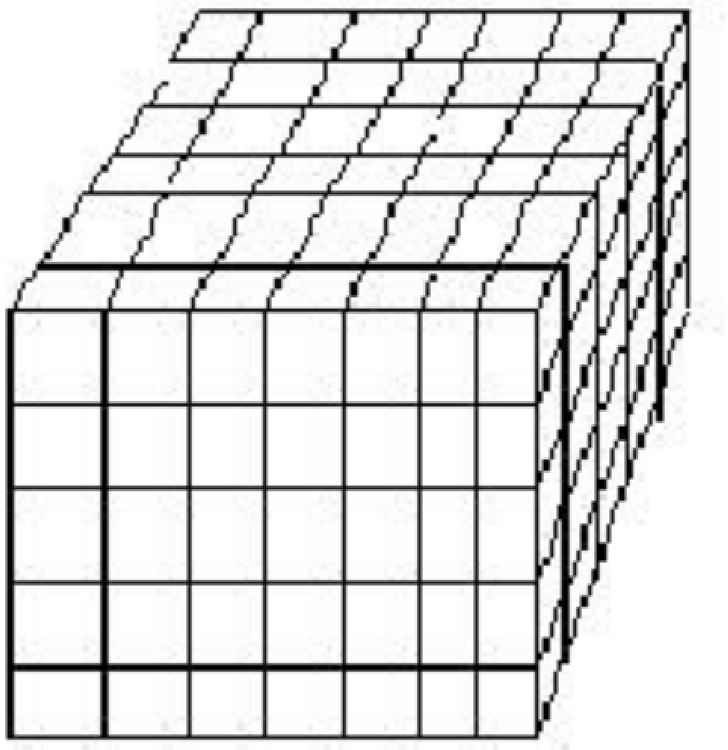


Figure 3.1: AVOD's architectural diagram. The feature extractors are shown in **blue**, the region proposal network in **pink**, and the second stage detection network in **green**.

Point Cloud Representation



reason:

- 3D grid representation requires complex and extensive comp utation for feature extraction

BEV maps:

- discretizing the point cloud with a 0.1 meters resolution
- projecting the voxels onto the xz-plane.
- the height feature is computed as the maximum height of the points in the cell
- point cloud is divided into 5 equally-sized slices

output 800*700*6:

- first 5 channels：maximum height of points in grid cell
- sixth channel: point density information,

  normalized by min(1.0, log(N +1) /log16)

  ——N is the number of points in the cell

Generating Feature Maps——Extractor

Operation	Kernel	Output
conv1	3×3	$480 \times 1590 \times 32$
maxpool	2×2	$240 \times 795 \times 32$
conv2	3×3	$240 \times 795 \times 64$
maxpool	2×2	$120 \times 397 \times 64$
conv3	3×3	$120 \times 397 \times 128$
maxpool	2×2	$60 \times 198 \times 128$
conv4	3×3	$60 \times 198 \times 256$
upsampling	NA	$240 \times 795 \times 256$
1×1 conv	1×1	$240 \times 795 \times 1$

Image branch with input of (480*1590 *3)

Operation	Kernel	Output
conv1	3×3	$700 \times 800 \times 32$
maxpool	2×2	$350 \times 400 \times 32$
conv2	3×3	$350 \times 400 \times 64$
maxpool	2×2	$175 \times 200 \times 64$
conv3	3×3	$175 \times 200 \times 128$
maxpool	2×2	$87 \times 100 \times 128$
conv4	3×3	$87 \times 100 \times 256$
upsampling	NA	$350 \times 400 \times 256$
1×1 conv	1×1	$350 \times 400 \times 1$

BEV branch with input of (700*800 *6)

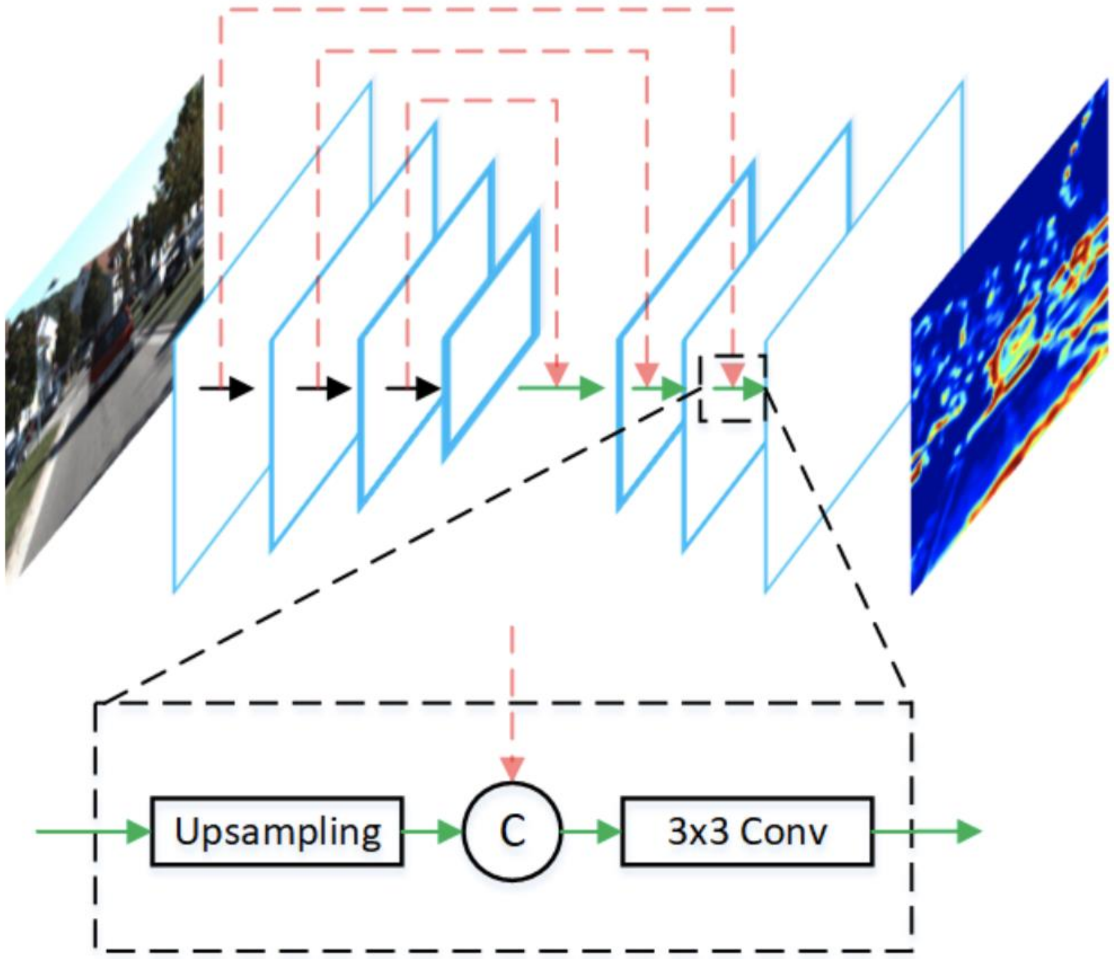
VGG16:

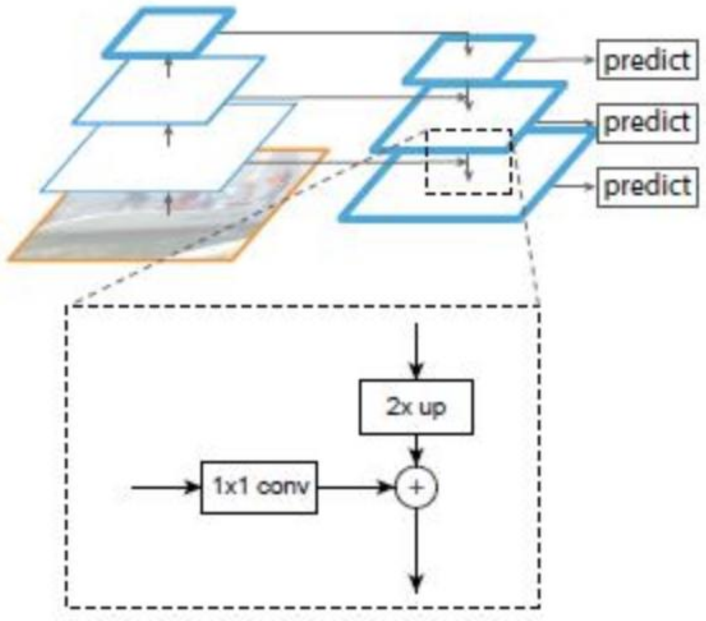
- Half filters
- Xavier weight initialize
  - $1/N_i$
  - $1/N_{i+1}$
  - $2/(N_i+N_{i+1})$
- Batch normalization
- Discard the fourth maxpooling

Upsampling

- 4*bilinear upsampling layer

Generating Feature Maps——Pyramids





advantage:

- The final feature map is of high resolution and rep resentational power
- can significantly boost the performance of the net work for detecting small objects

RPN Network

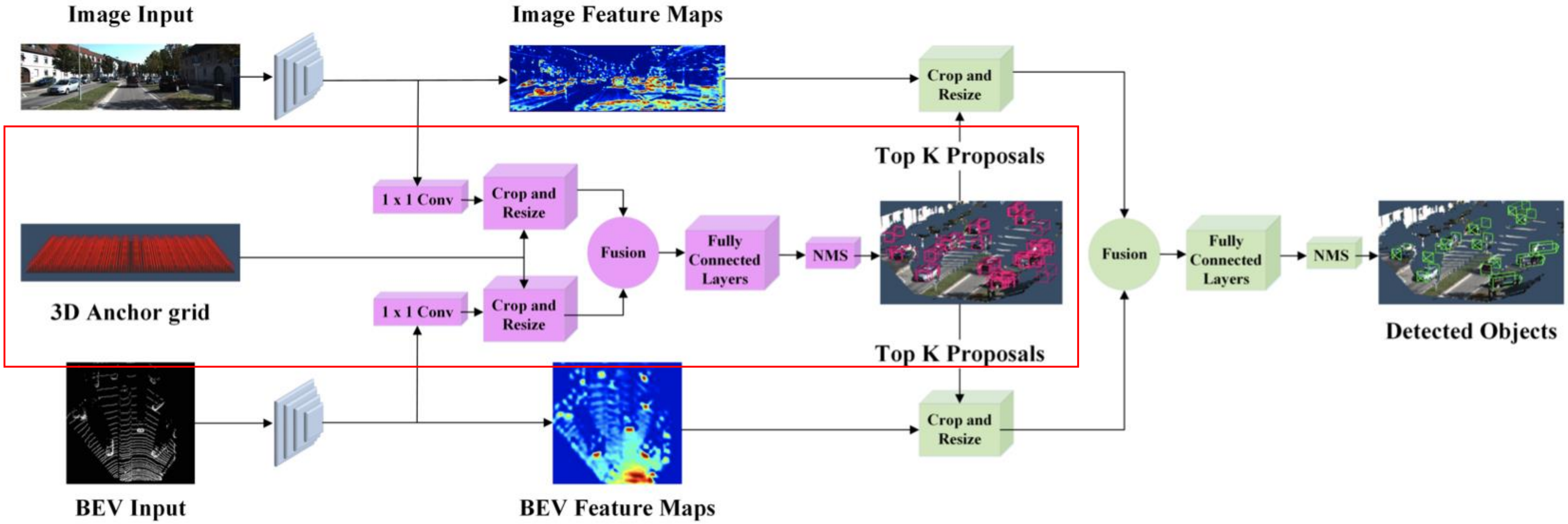
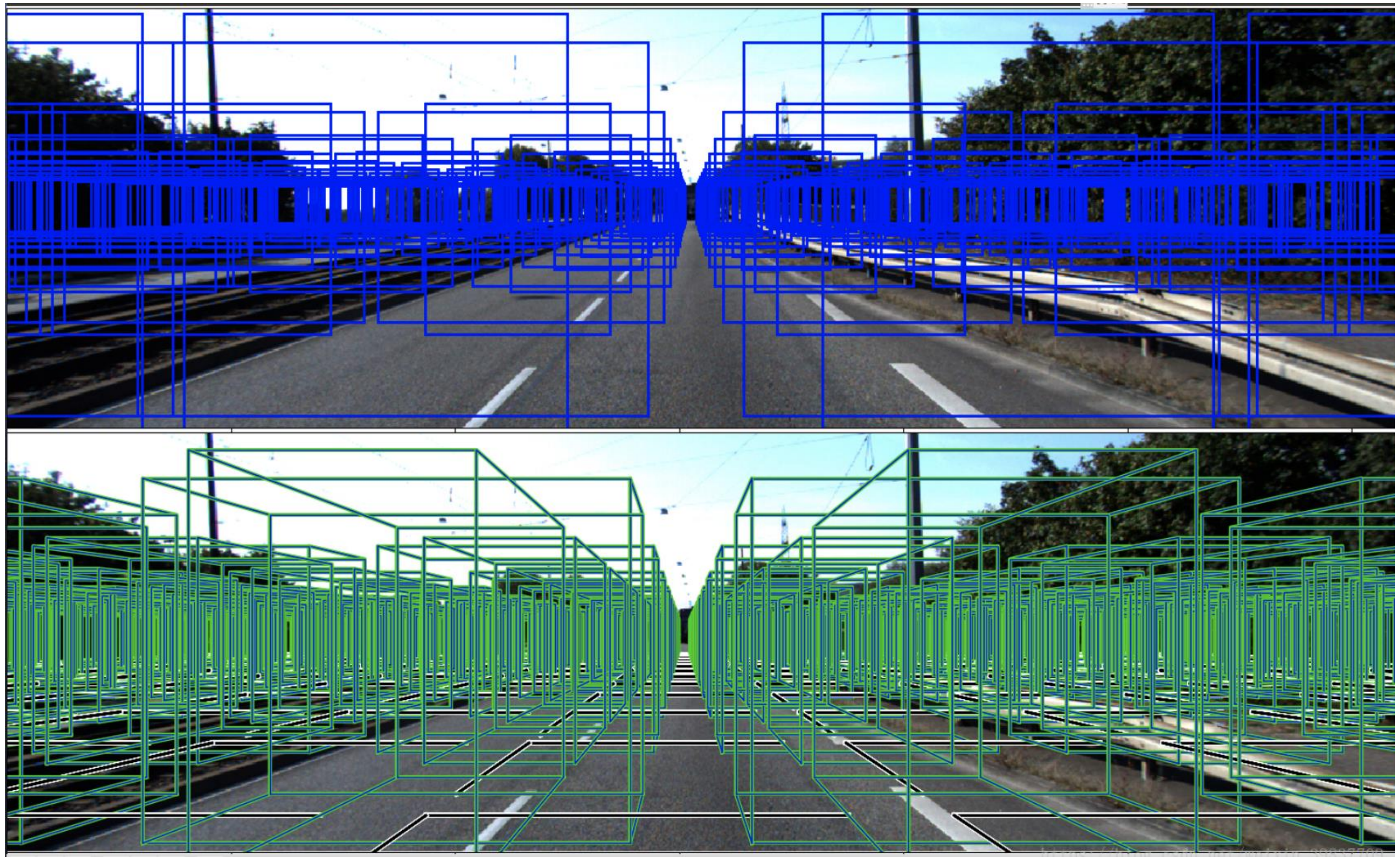
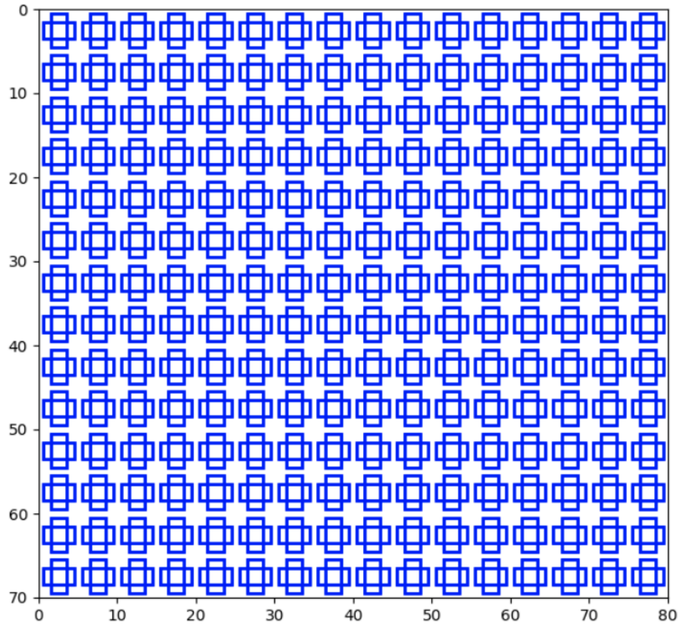
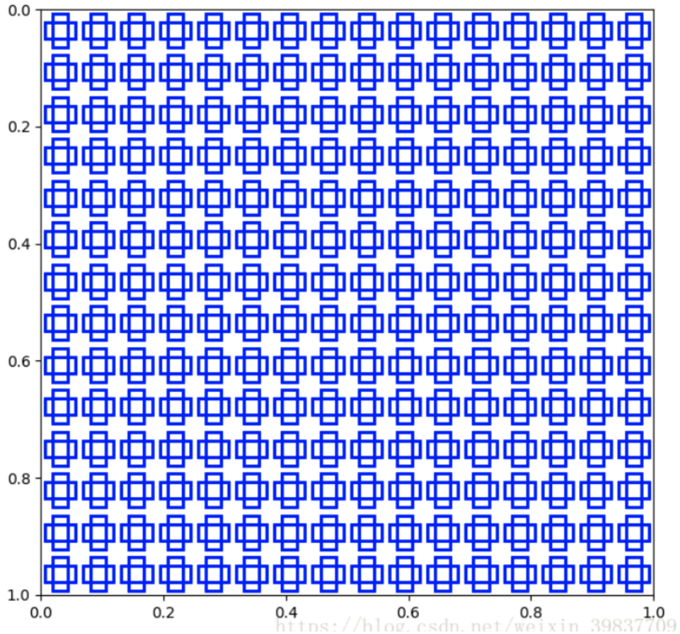


Figure 3.1: AVOD's architectural diagram. The feature extractors are shown in **blue**, the region proposal network in **pink**, and the second stage detection network in **green**.

Anchor Generation







Anchor Generation

parameter:

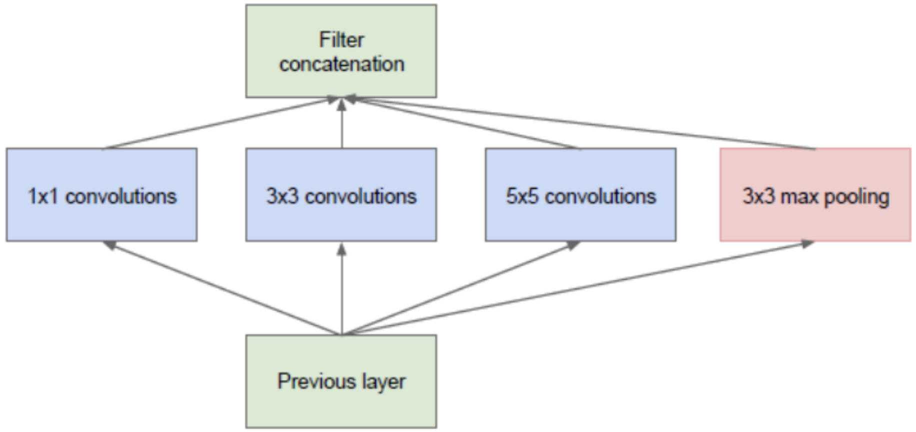
- centroid (tx,ty,tz)+ dimensions (dx,dy,dz)
- **(tx,ty)**： sampled at an interval of 0.5 meters in BEV
- **tz**： based on the sensor’s height
- **(dx,dy,dz)**： determined by clustering the training samples for each class
- **orientations**： generated with 0° and 90°

Reason for using clusters：

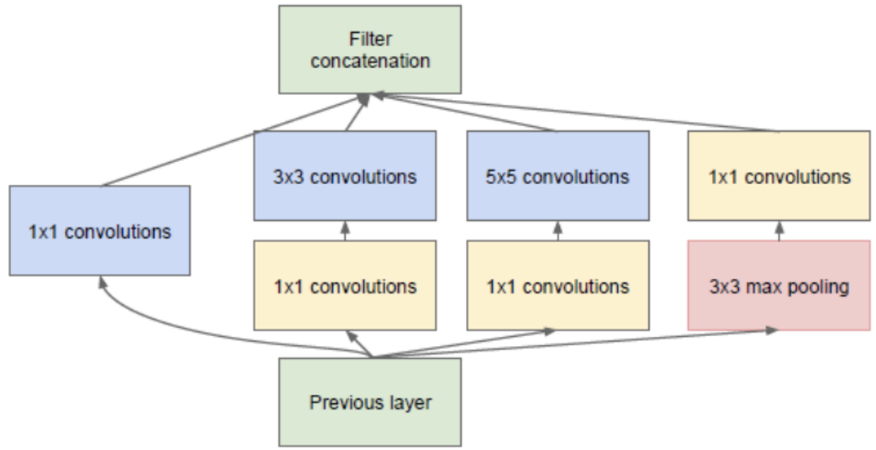
- hard to hand pick the best prior box dimensions

Result in 10~100K non-empty anchors per frame

Dimensionality Reduction Via 1*1 Convolutional Layers



(a) Inception module, naïve version



(b) Inception module with dimension reductions

作用：

- 实现跨通道的交互和信息整合
- 进行卷积核通道数的降维和升维
- 可以实现与全连接层等价的效果
- 减少内存，提高运算速度

GoogLeNet：

feature map=28×28×192

(a) 1×1×192×64+3×3×192×128+5×5×192×32

(b) 1×1×192×64+（1×1×192×96+3×3×96×128）

+（1×1×192×16+5×5×16×32）

参数大约减少到原来的三分之一。

Extracting Feature Crops Via Multiview Crop And Resize Operations:

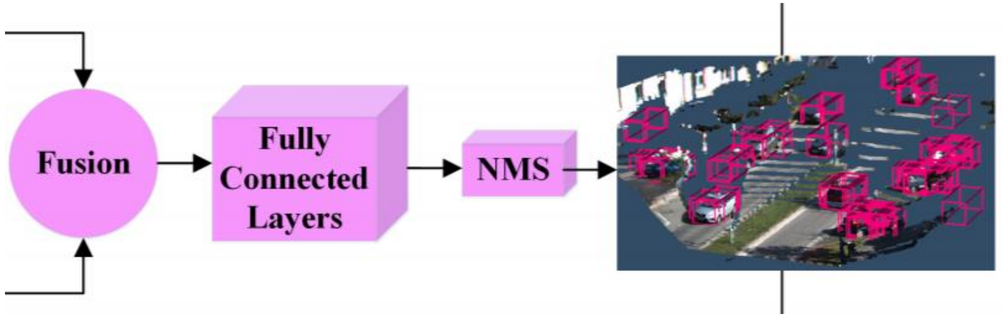

```python
# Do ROI Pooling on BEV
bev_proposal_rois = tf.image.crop_and_resize(
    bev_proposal_input,
    self._bev_anchors_norm_pl,
    tf_box_indices,
    self._proposal_roi_crop_size)
# Do ROI Pooling on image
img_proposal_rois = tf.image.crop_and_resize(
    img_proposal_input,
    self._img_anchors_norm_pl,
    tf_box_indices,
    self._proposal_roi_crop_size)
```

Given an anchor in 3D → Projecting the anchor onto the BEV and image feature maps → Obtain two regions of interest → Use the ROI to extract feature map crops from each view

↓

Bilinearly resize the feature map crops to 3×3 to obtain equal-length feature vectors

3D Proposal Generation:



$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

cross-entropy loss for classification

Smooth L1 loss for 3D box regression

background anchors are ignored=>$p_i^* = 0$

- **Fusion:** fuse the feature crops from both views via an **element-wise mean operation**.
- **Fully Connected Layers: size 256**
    - **regress axis aligned object proposal boxes:** computing $(\Delta t_x, \Delta t_y, \Delta t_z, \Delta d_x, \Delta d_y, \Delta d_z)$, the difference in centroid and dimensions between anchors and ground truth bounding boxes.
    - **output an object/background "objectness" score:** calculating the **2D IoU in BEV** between the anchors and the ground truth bounding boxes.

object/background anchors

For the car class: less than 0.3 => background anchors(negative)
greater than 0.5 => object anchors(positive)

For the pedestrian and cyclist classes: greater than 0.45
=> object anchors (positive)

- **NMS:** To remove redundant proposals; IoU threshold of 0.8 in BEV
- **Output:** 300 proposal for car class,1024 proposals for pedestrian and cyclist.

Second Stage Detection Network

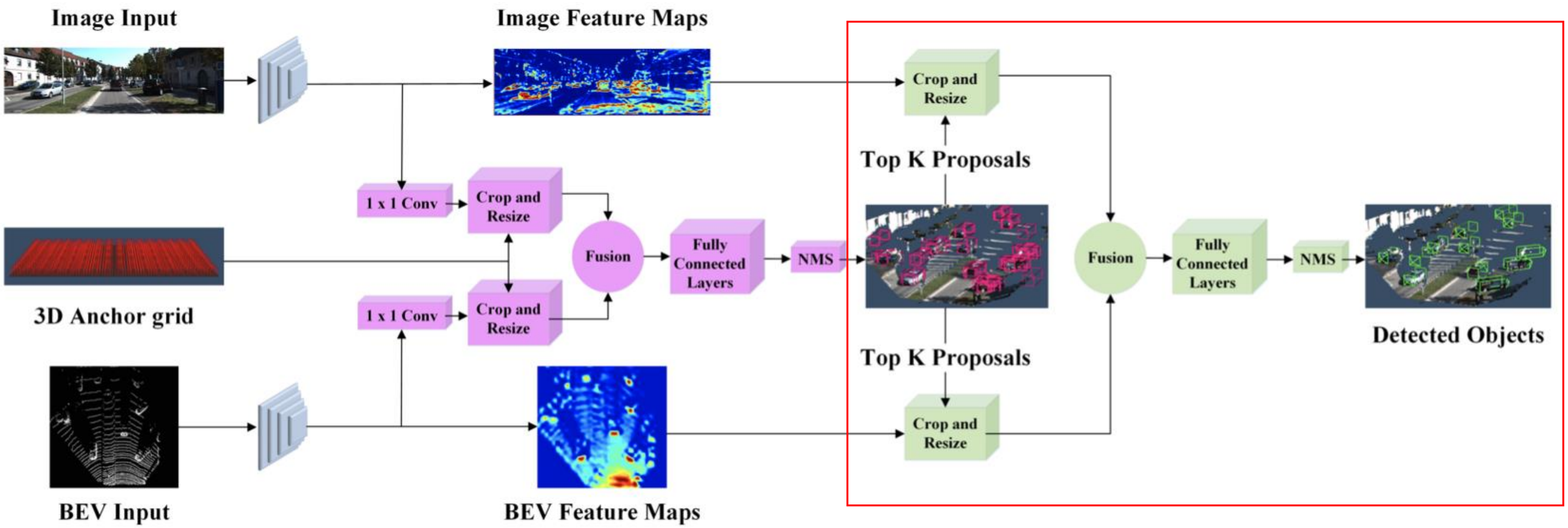


Figure 3.1: AVOD's architectural diagram. The feature extractors are shown in **blue**, the region proposal network in **pink**, and the second stage detection network in **green**.

3D Bounding Box Encoding

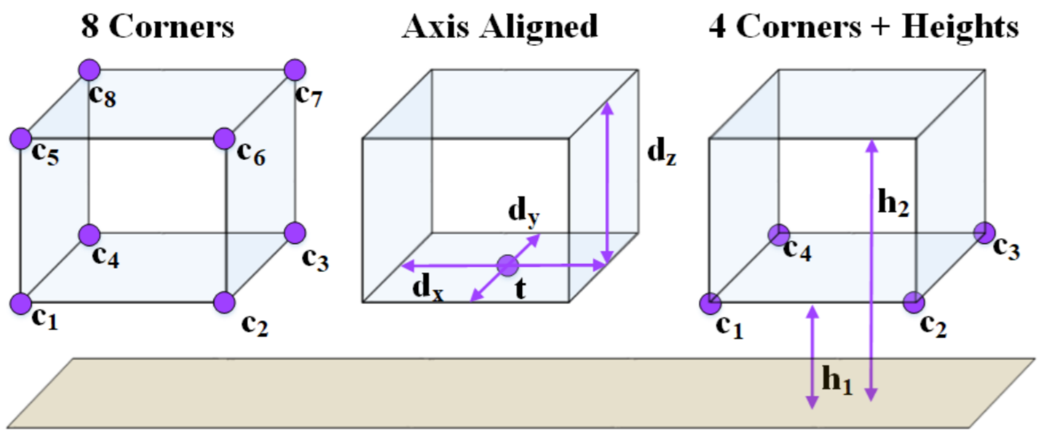


Fig. 4: A visual comparison between the 8 corner box encoding proposed in [4], the axis aligned box encoding proposed in [16], and our 4 corner encoding.

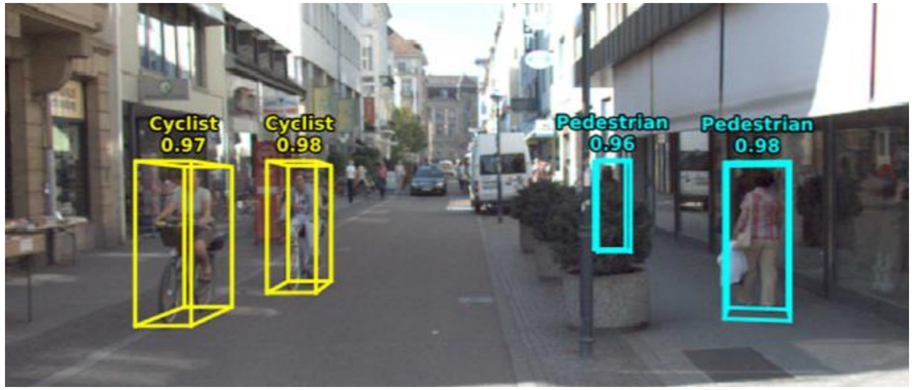
Reduces the box representation from an over parameterized 24 dimensional vecto r to a 10 dimensional one.

Regression targets:

$$(\underbrace{\Delta x_1 \ldots \Delta x_4, \Delta y_1 \ldots \Delta y_4}_{\text{correspond the closest corner of the proposals to the closest corner of the ground truth box}}, \underbrace{\Delta h_1, \Delta h_2}_{\text{from the ground plane between the proposals and the ground truth boxes}})$$

Explicit Orientation Vector Regression

MV3D relies on the extents of the estimated bounding box where the orientation vector is assumed to be in **the direction of the longer side of the box**



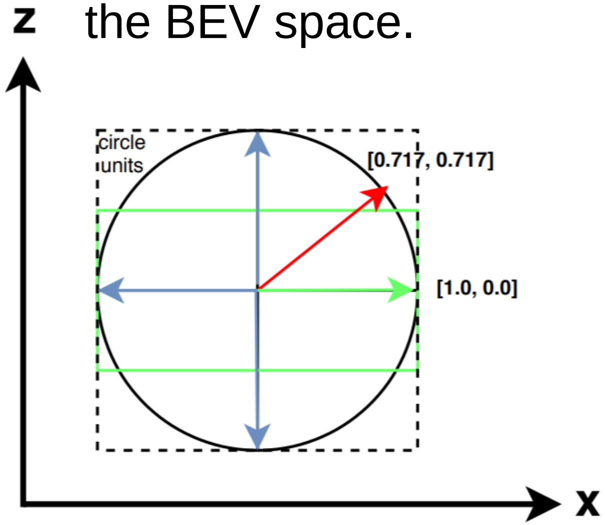
Two problems:

1. fails for detected objects like pedestrians:
2. Orientation information is lost as the corner order is not preserved in closest corner to corner matching: $\pm\pi$ radians

Solution:

Computing $$(x_{or}, y_{or}) = (\cos(\theta), \sin(\theta))$$

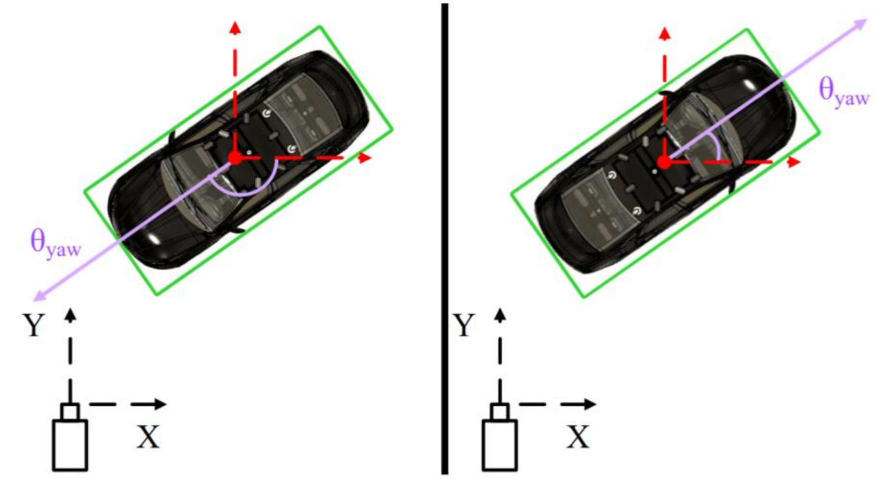
Every θ ∈ [−π, π] can be represented by a unique unit vector in the BEV space.



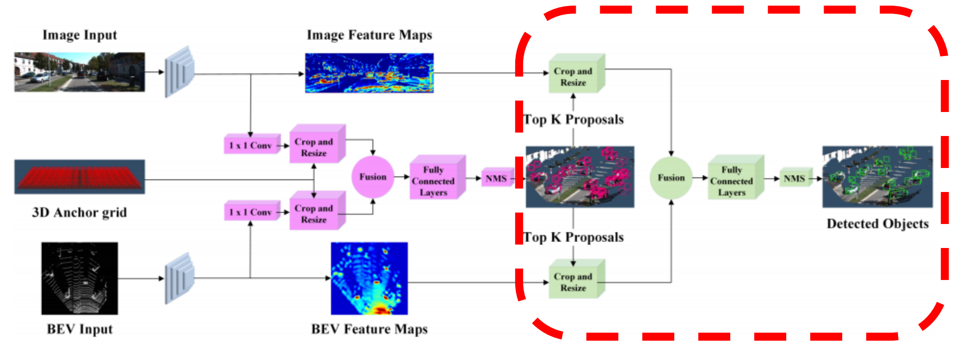
the regressed orientation vector

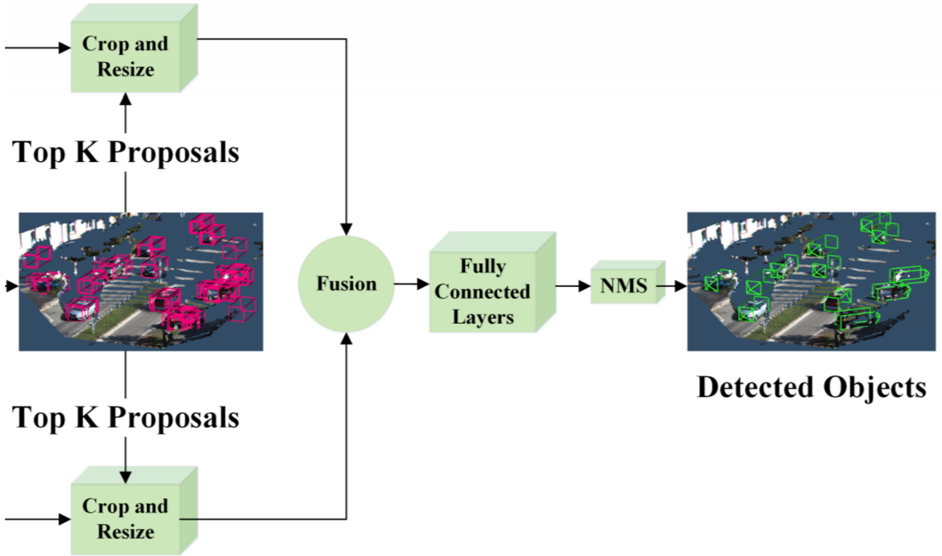
the possible headings
the closest heading

the possible headings



Generating Final Detections





Input: feature crops generated from projecting the **Top K** Proposals into the two input views.

Process:

1、Crop and Resize: resize the crops to 7x7

2、Fusion: fused with an element-wise mean operation

3、Fully Connected Layers: three fully connected layers of size 2048

4、NMS: To remove overlapping detections, set a threshold of 0.01.

Loss Function:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*) + \lambda\frac{1}{N_{ang}}\sum_i p_i^* L_{ang}(t_i,t_i^*)$$

two Smooth L1 losses for the bounding box and orientation vector regression tasks, and a cross-entropy loss for the classification task.

Notes:

Proposals 2D IoU in BEV with the ground truth boxes

For **car:** at least **0.65**

For **pedestrian/cyclist:** at least **0.55**

Result

			AP_{3D} (%)			AP_{BEV}(%)		
Method	Runtime (s)	Class	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D [4]	0.36	**Car**	71.09	62.35	55.12	86.02	76.90	68.49
VoxelNet [10]	0.23		77.47	65.11	57.73	**89.35**	79.26	77.39
F-PointNet [12]	0.17		81.20	70.39	62.19	88.70	84.00	75.33
Ours	**0.08**		73.59	65.78	58.38	86.80	**85.44**	77.73
Ours (Feature Pyramid)	0.1		**81.94**	**71.88**	**66.38**	88.53	83.79	**77.90**
VoxelNet [10]	0.23	**Ped.**	39.48	33.69	31.51	46.13	40.74	38.11
F-PointNet [12]	0.17		**51.21**	**44.89**	**40.23**	**58.09**	**50.22**	**47.20**
Ours	**0.08**		38.28	31.51	26.98	42.51	35.24	33.97
Ours (Feature Pyramid)	0.1		46.35	39.00	36.58	50.66	44.75	40.83
VoxelNet [10]	0.23	**Cyc.**	61.22	48.36	44.37	66.70	54.76	50.55
F-PointNet [12]	0.17		**71.96**	**56.77**	**50.39**	**75.38**	**61.96**	**54.68**
Ours	**0.08**		60.11	44.90	38.80	63.66	47.74	46.55
Ours (Feature Pyramid)	0.1		59.97	46.12	42.36	62.39	52.02	47.87

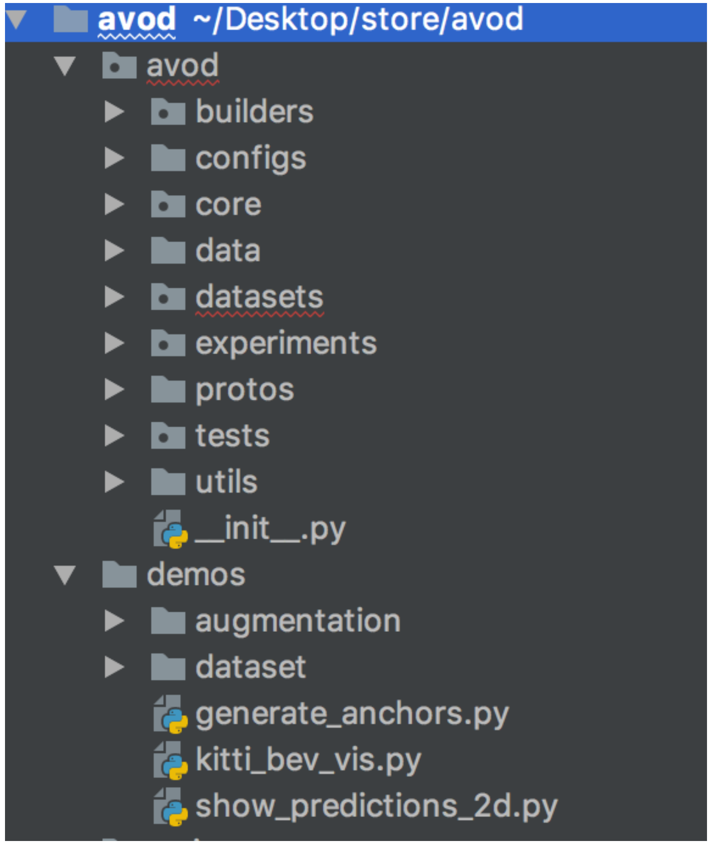
PART 3

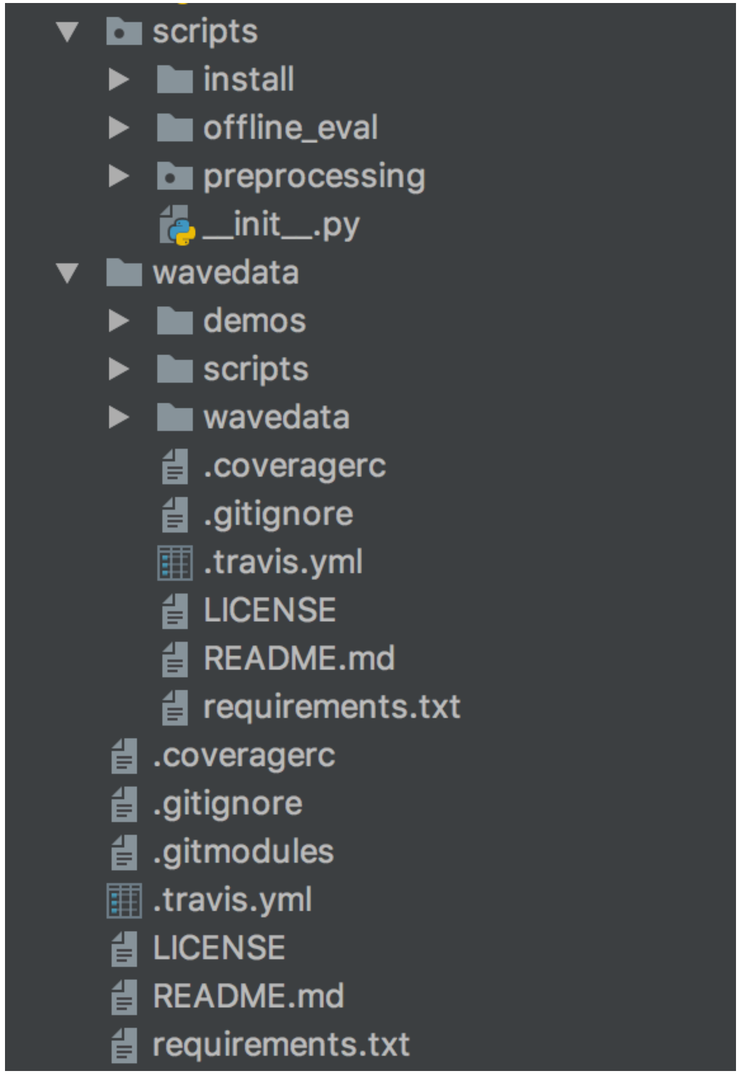
AVOD code

https://github.com/kujason/avod

Dataset and code architecture

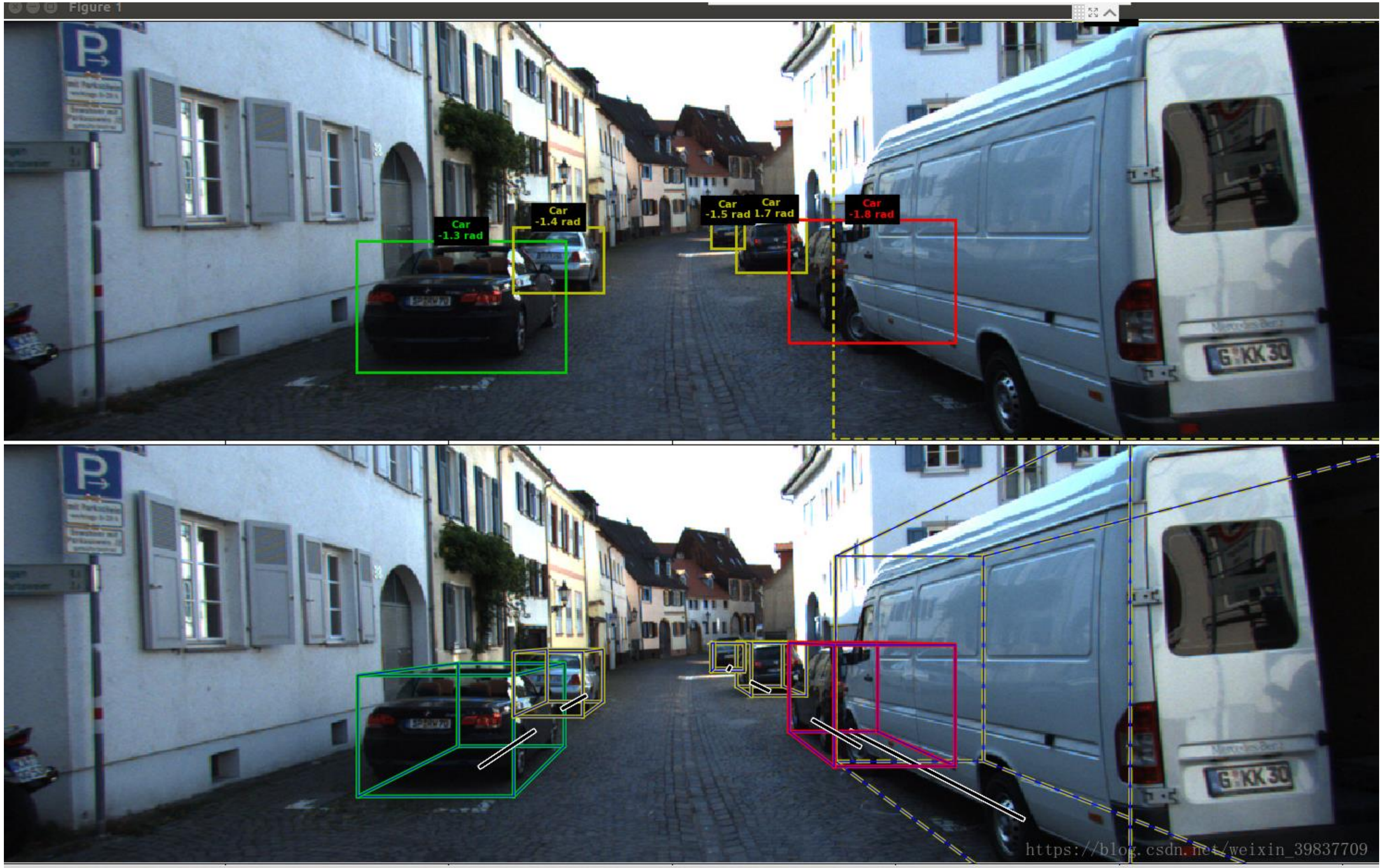
```
Kitti
    object
        testing
        training
            calib
            image_2
            label_2
            planes
            velodyne
        train.txt
        val.txt
```



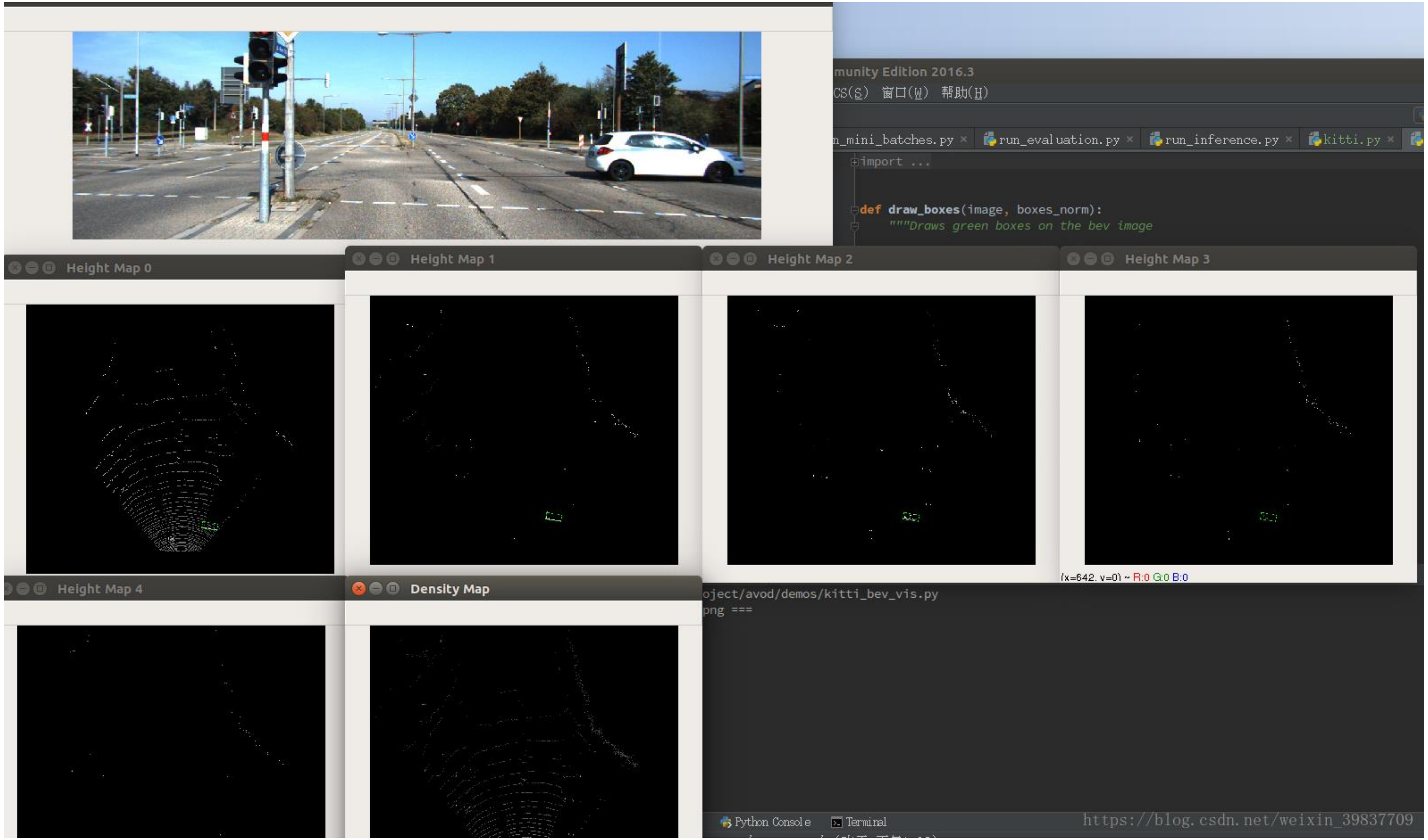


- Generate mini-batches for the RPN
- Train on the specific config
- Run Evaluator
- Run inference on the val split
- Viewing Results

kitti.py 运行效果



kitti_bev_vis.py 运行效果



END

THANK YOU