



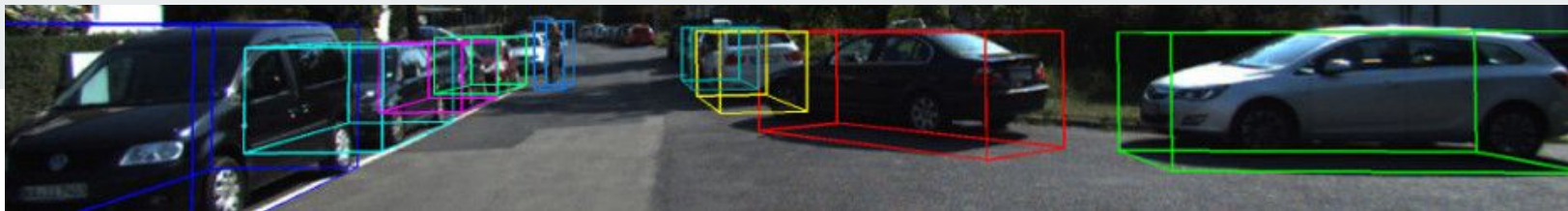
Computer Vision

3D Object Detection on Kitti Dataset



Team

Moustafa Tohamy	3851
Ziad Ezzat	4492
Moustafa Sherif Ahmed	4347



Kitti Dataset

- The 3D object detection benchmark consists of 7481 training images ,7518 test and the corresponding point clouds with a total of 80.256 labeled objects.
- precision-recall curves is computed for evaluation.
 - Train data
 - Left Colored Images
 - Right Colored Images
 - Velodyne point clouds(laser information)
 - X/Z plane (bird's eye view)
 - camera calibration matrices
 - training labels





First Paper

Frustum ConvNet: Sliding Frustums to Aggregate Local Pointwise Features for Amodal 3D Object Detection

<https://arxiv.org/pdf/1903.01864.pdf>

<https://github.com/zhixinwang/frustum-convnet>



Outline

- Related work
- Proposed Frustum-ConvNet
 - Associate point clouds with sliding frustum
 - Architecture of Frustum-ConvNet
 - Multi Resolution Frustum Feature integration variant
 - Final Refinement
- Results
- Our trail



Related Work

- Data conversion Methods
- Raw cloud data Methods



Related Work

- Data conversion Methods
 - Projecting lidar point clouds to bird eye view , then employs faster R-CNN
 - Use depth images as converted data of point clouds , estimate 3d objects based on fast R-CNN



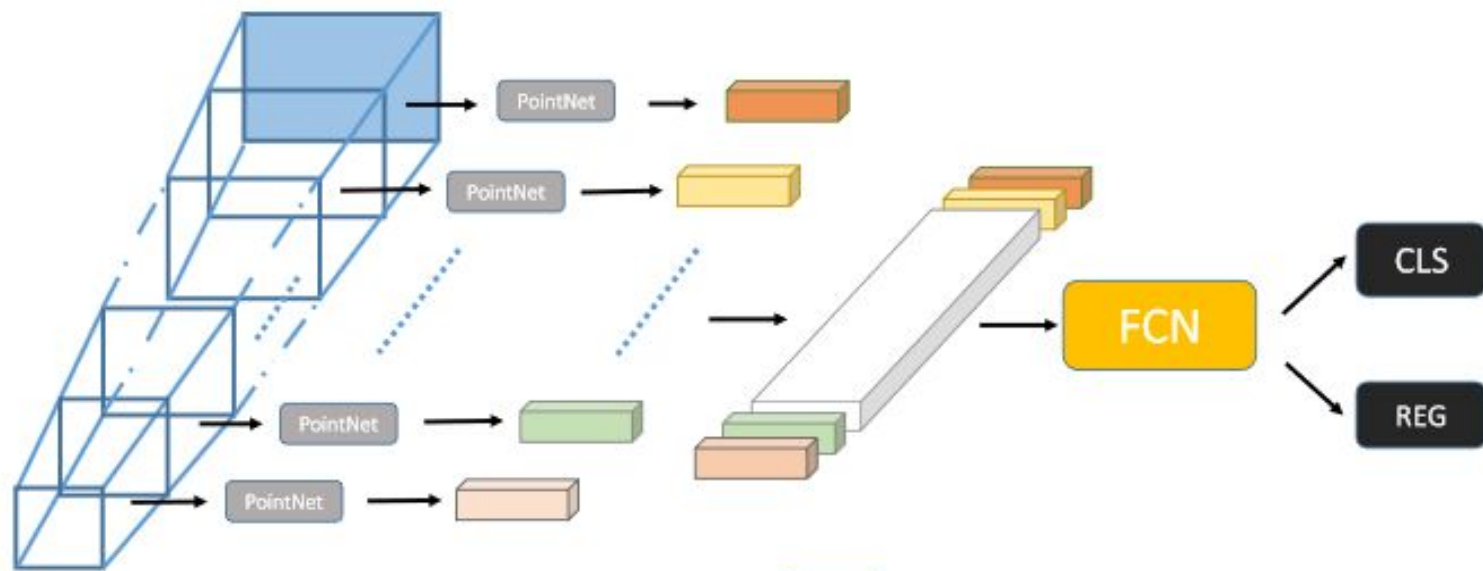
Related Work

- Raw cloud data Methods
 - Point Net :needs to start with instance segmentation so it's not an end-to-end deep learning method
 - Voxel Net:needs to know the appropriate voxel shape for each dataset



Proposed Frustum convNet

- Associate point clouds with sliding frustum
- Architecture of Frustum-ConvNet
- Multi Resolution Frustum Feature integration variant
- Final Refinement





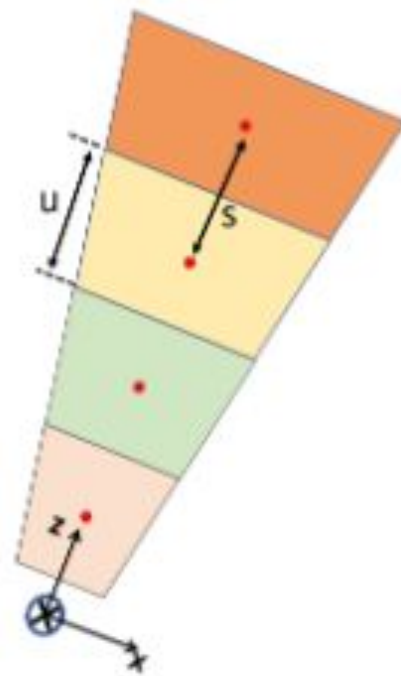
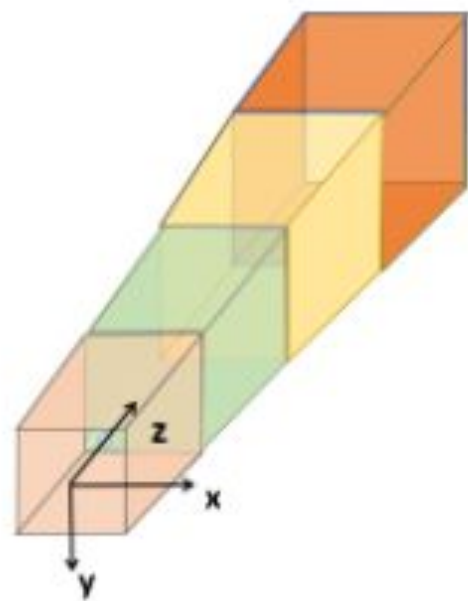
Associate point clouds with sliding frustum

- Assuming
 - RGB images, point clouds are available
 - 2D Region proposals are available
 - Optical access of the camera is perpendicular to this region



Associate point clouds with sliding frustum

- A sequence of (possibly overlapping) frustums can be obtained by sliding a pair of parallel planes along the frustum axis with an equal stride, where the pair of planes are also perpendicular to the frustum axis.





Proposed Frustum convNet

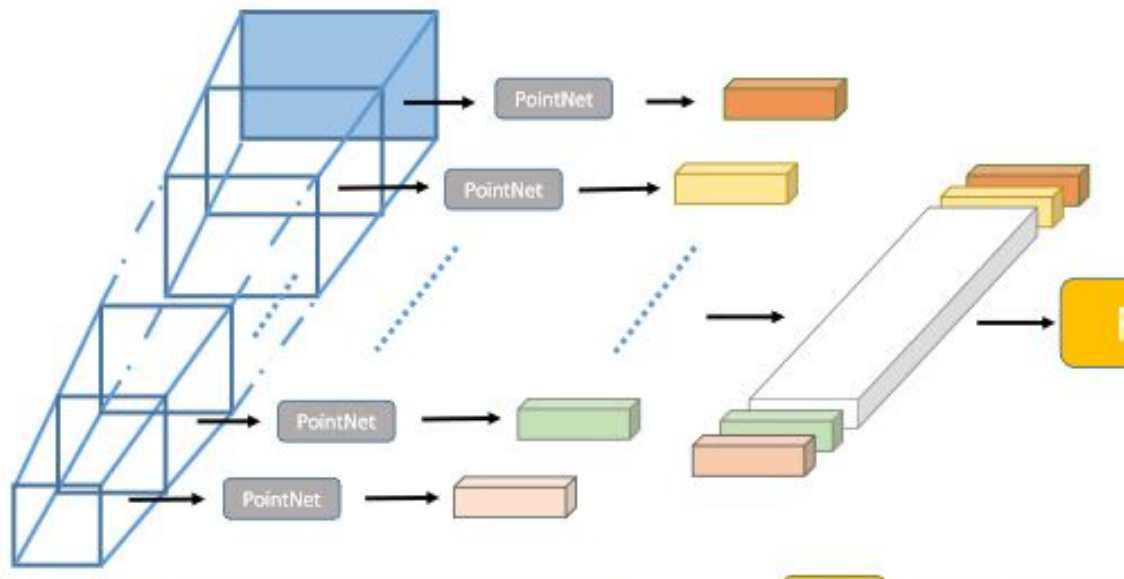
- ~~Associate point clouds with sliding frustum~~
- Architecture of Frustum-ConvNet
- Multi Resolution Frustum Feature integration variant
- Final Refinement



Architecture of F-ConvNet

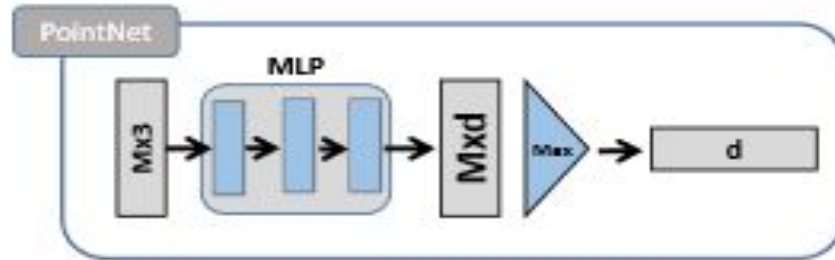
- Frustum level feature extraction
- Fully convolutional network
- Detection of object , estimate locations

Frustum level Feature Extraction



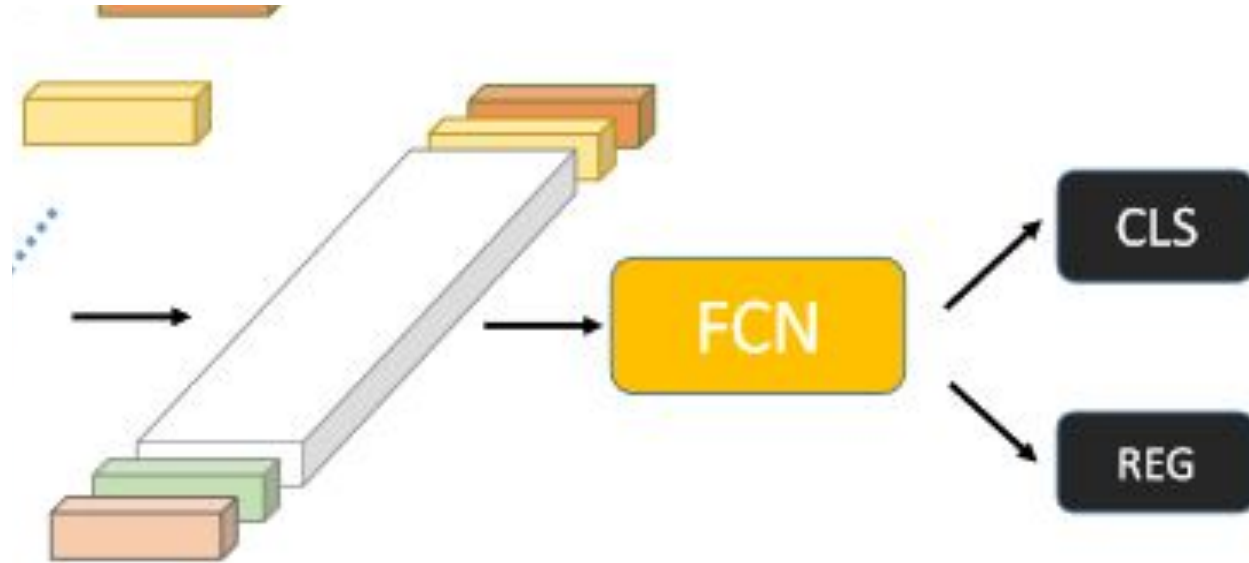
Point Net

- 3 layer of fully connected network

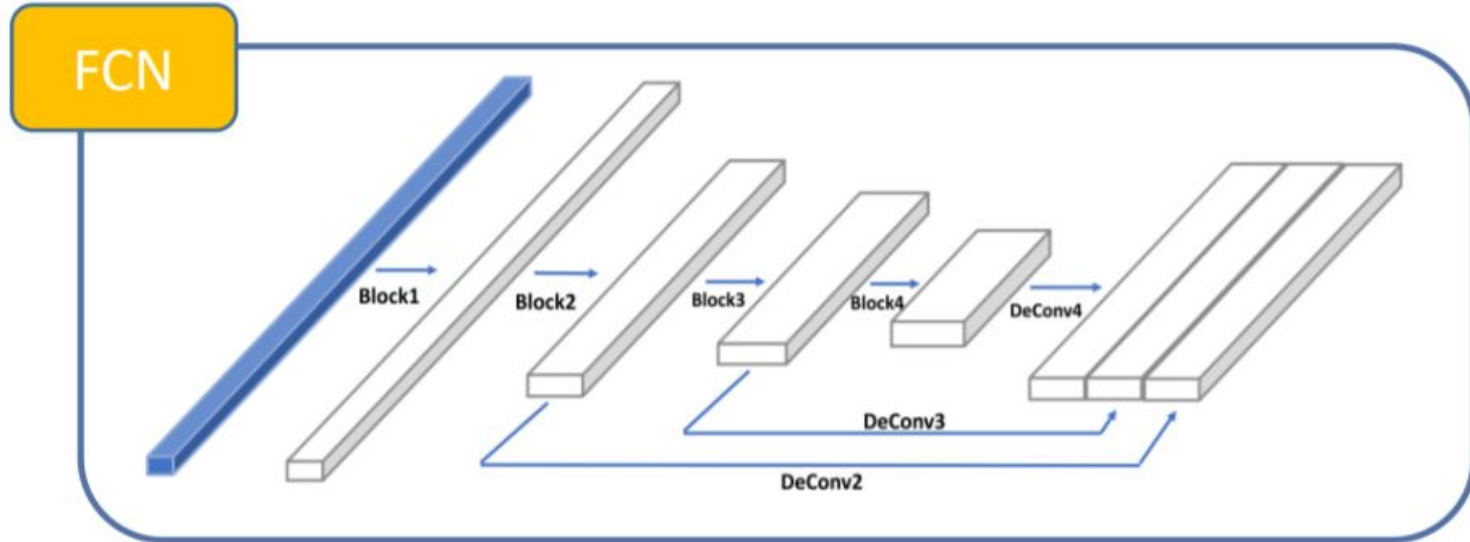


(a) PointNet

Fully Convolutional Network



Fully Convolutional Network





Classification

- For k classes , size of feature map = l
- Output shape will be $(L \times K+1)$
- Extra dimension for background



Regression

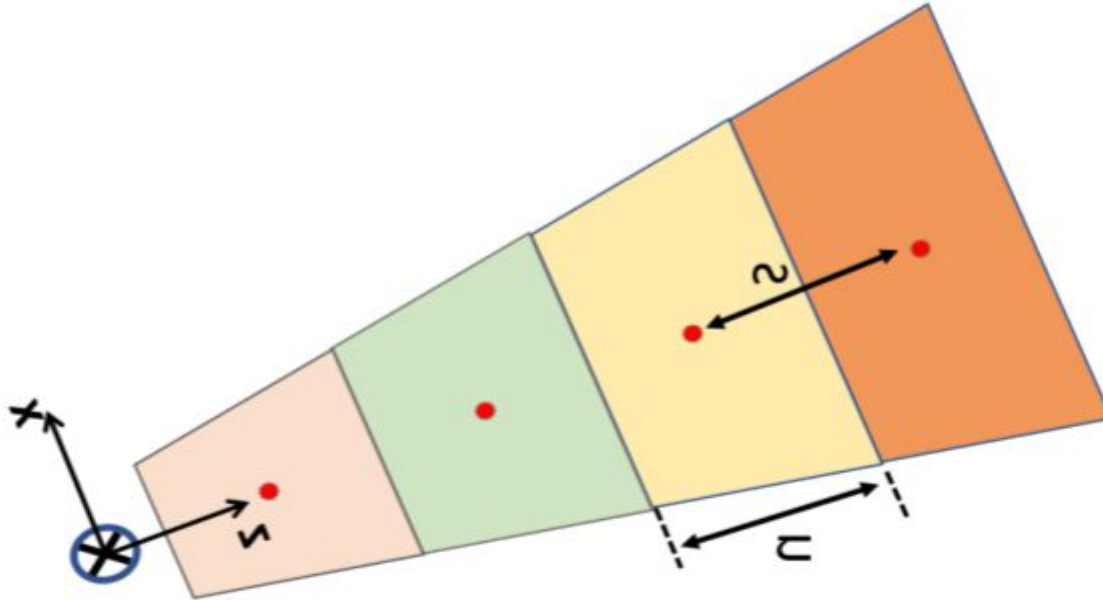
- For each box , 7 values needs to be estimated
- location of centre of the box(x,y,z)
- length ,width and height of the box (l,w,h)
- Rotation angle of the box (θ)



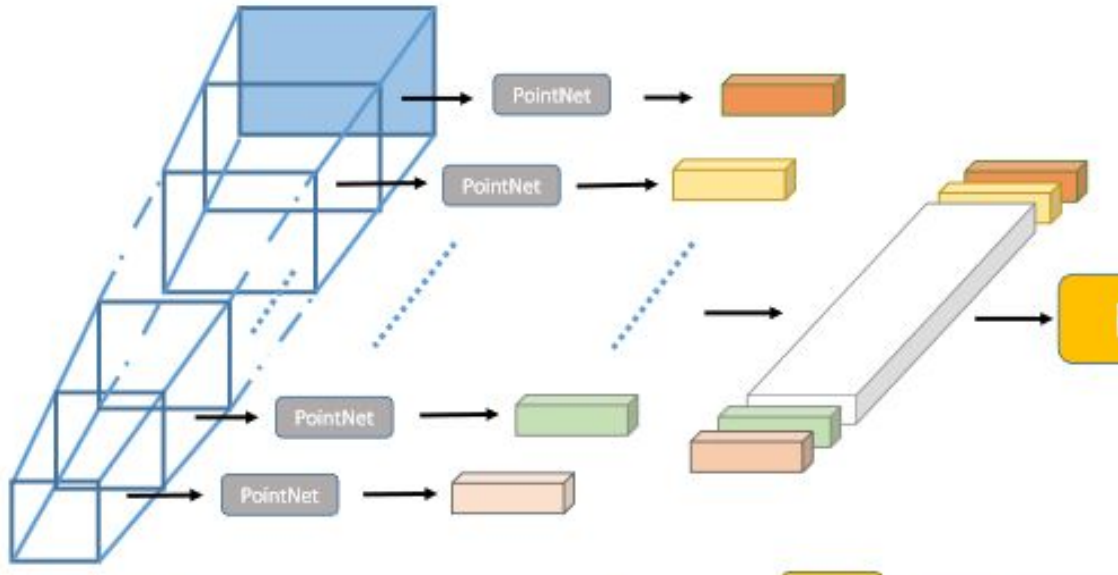
Proposed Frustum convNet

- ~~Associate point clouds with sliding frustum~~
- ~~Architecture of Frustum ConvNet~~
- Multi Resolution Frustum Feature integration variant
- Final Refinement

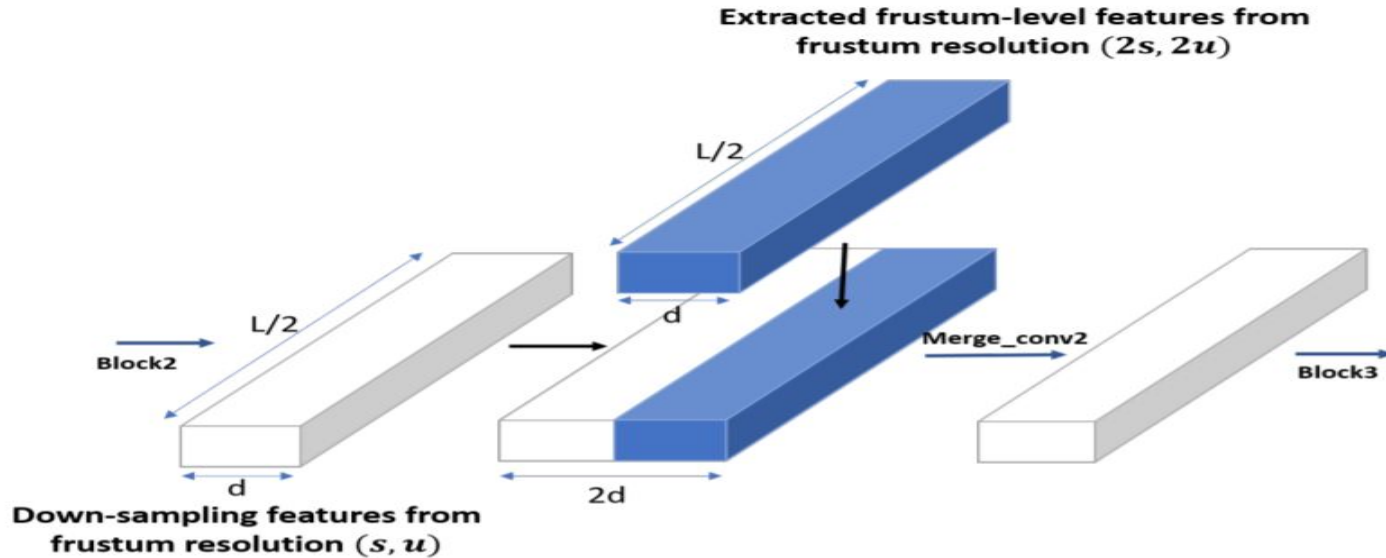
Multi Resolution Frustum feature integration



Multi Resolution Frustum feature integration



Multi Resolution Frustum feature integration





Proposed Frustum convNet

- ~~Associate point clouds with sliding frustum~~
- ~~Architecture of Frustum ConvNet~~
- ~~Multi Resolution Frustum Feature integration variant~~
- Final Refinement



Final Refinement

- 2D proposed regions is not so accurate
- Expand the detecting regions by a factor of 1.2
- Normalize the points in the regions by translation and rotation to be in the same direction of the camera centre
- Train the network again with the new regions



Proposed Frustum convNet

- ~~Associate point clouds with sliding frustum~~
- ~~Architecture of Frustum ConvNet~~
- ~~Multi Resolution Frustum Feature integration variant~~
- ~~Final Refinement~~



Results

- Ranked as
 - 31 in car category
 - 6 in pedestrian
 - 2 in cyclists

Results

Method	Cars			Pedestrians			Cyclists		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D [5]	71.09	62.35	55.12	-	-	-	-	-	-
VoxelNet [14]	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
F-PointNet [13]	81.20	70.29	62.19	51.21	44.89	40.23	71.96	56.77	50.39
AVOD-FPN [6]	81.94	71.88	66.38	50.80	42.81	40.88	64.00	52.18	46.61
SECOND [15]	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
IPOD [22]	79.75	72.57	66.33	56.92	44.68	42.39	71.40	53.46	48.34
PointPillars [16]	79.05	74.99	68.30	52.08	43.53	41.49	75.78	59.07	52.92
PointRCNN [23]	85.94	75.76	68.32	49.43	41.78	38.63	73.93	59.60	53.59
Ours	85.88	76.51	68.08	52.37	45.61	41.49	79.58	64.68	57.03

TABLE VII: 3D object detection AP (%) on KITTI test set.

Results

Method	Cars			Pedestrians			Cyclists		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D [5]	86.02	76.90	68.49	-	-	-	-	-	-
VoxelNet [14]	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
F-PointNet [13]	88.70	84.00	75.33	58.09	50.22	47.57	75.38	61.96	54.68
AVOD-FPN [6]	88.53	83.79	77.90	58.75	51.05	47.54	68.06	57.48	50.77
SECOND [15]	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
IPOD [22]	86.93	83.98	77.85	60.83	51.24	45.40	77.10	58.92	51.01
PointPillars [16]	88.35	86.10	79.83	58.66	50.23	47.19	79.14	62.25	56.00
PointRCNN [23]	89.47	85.68	79.10	55.92	47.53	44.67	81.52	66.77	60.78
Ours	89.69	83.08	74.56	58.90	50.48	46.72	82.59	68.62	60.62

TABLE VIII: 3D object localization AP (BEV) (%) on KITTI test set.




Our Trial

- Getting the data , solving it's issues with colab available space
- Applying pre processing part of the images
- We didn't manage to make the train work
- We spend almost three days on it



Our Trial

```
[ ] 1 extract_frustum_data(  
    2     os.path.join(BASE_DIR, 'image_sets/val.txt'),  
    3     'training',  
    4     os.path.join(save_dir, output_prefix + 'val.pickle'),  
    5     perturb_box2d=False, augmentX=1,  
    6     type_whitelist=type_whitelist)
```



```
----- 1  
----- 2  
----- 4  
----- 5  
----- 6  
----- 8  
----- 15  
----- 19  
----- 20  
----- 21  
----- 23  
----- 24  
----- 25  
----- 27  
----- 28  
----- 31
```



Our Trial

```
----- 92
--
[ ] 1 extract_frustum_data_rgb_detection(
    2     os.path.join(BASE_DIR, 'rgb_detections/rgb_detection_val.txt'),
    3     'training',
    4     os.path.join(save_dir, output_prefix + 'val_rgb_detection.pickle'),
    5     type_whitelist=type_whitelist)
    6
```

```
● det idx: 0/29789, data idx: 1
  det idx: 1/29789, data idx: 1
  det idx: 2/29789, data idx: 1
  det idx: 3/29789, data idx: 2
  det idx: 4/29789, data idx: 4
  det idx: 5/29789, data idx: 4
  det idx: 6/29789, data idx: 4
  det idx: 7/29789, data idx: 4
  det idx: 8/29789, data idx: 4
  det idx: 9/29789, data idx: 4
  det idx: 10/29789, data idx: 4
  det idx: 11/29789, data idx: 5
  det idx: 12/29789, data idx: 6
  det idx: 13/29789, data idx: 6
  det idx: 14/29789, data idx: 6
  det idx: 15/29789, data idx: 6
  det idx: 16/29789, data idx: 6
  det idx: 17/29789, data idx: 6
  det idx: 18/29789, data idx: 8
  det idx: 19/29789, data idx: 8
```

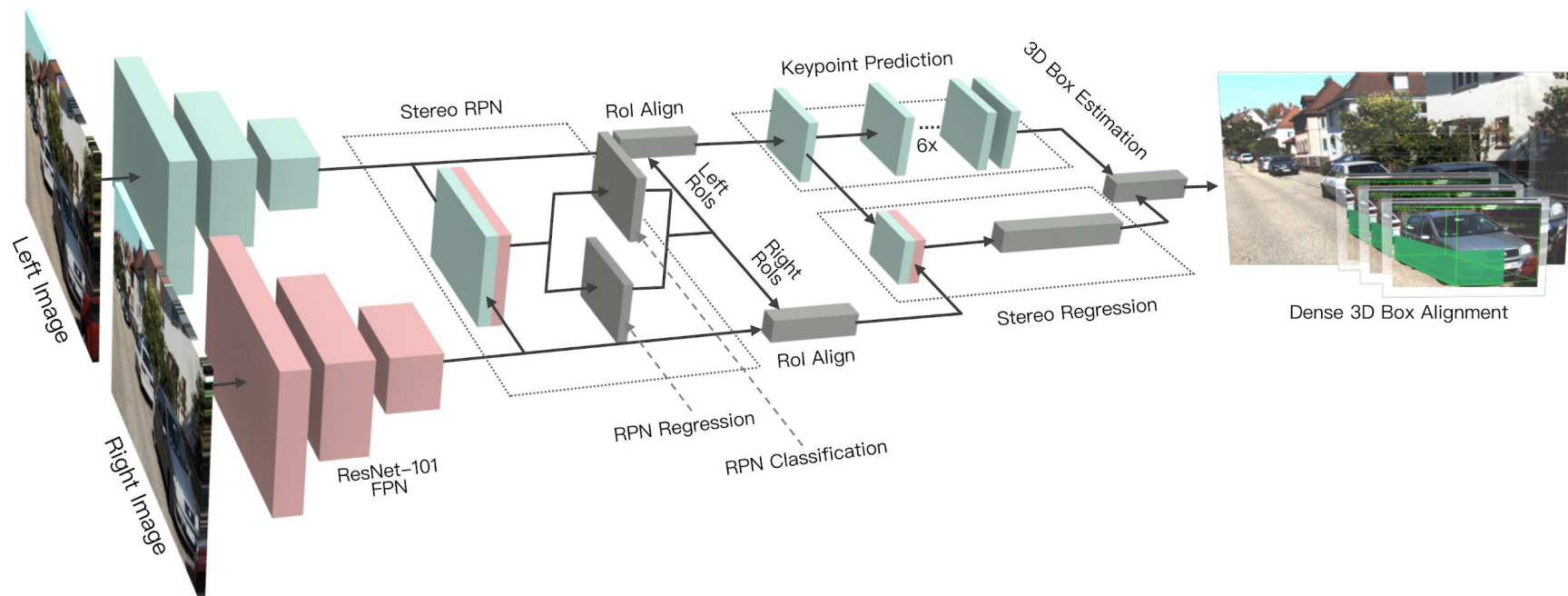


Our Trial

- Colab link

https://colab.research.google.com/drive/1By49_jrkLJmPWmO2PXgTwWQhZrNbL1H5

Paper 2 Stereo R-CNN Architecture





Outline

- Why Stereo R-CNN?
- Proposed Stereo R-CNN
 - Pre-trained Backbone Networks.
 - Stereo Region Proposal Network(RPN)
 - Stereo Regression
 - Keypoint prediction
 - 3D Box Estimation
 - Dense 3D Box Alignment
- Results



Why Stereo R-CNN?

- Lower cost than LiDAR that uses expensive sensors.
- More accurate than monocular camera.
- Stereo camera provides more precise depth information by **left-right photometric alignment**.

Left Image “input example”

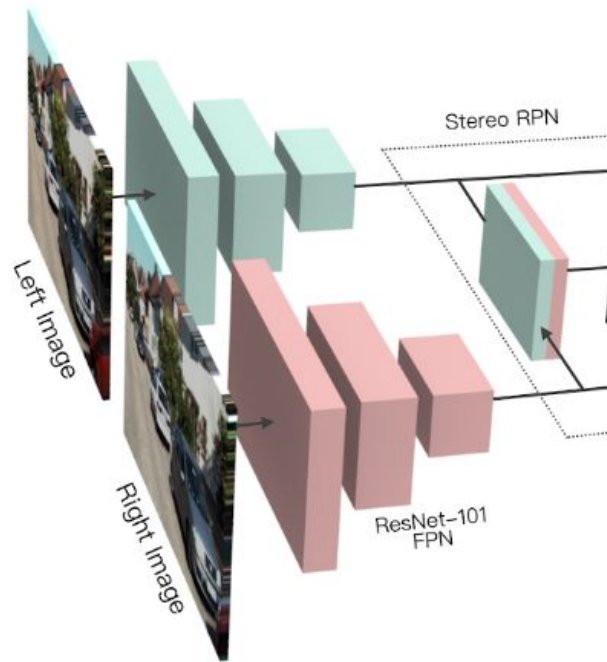


Right Image “input example”



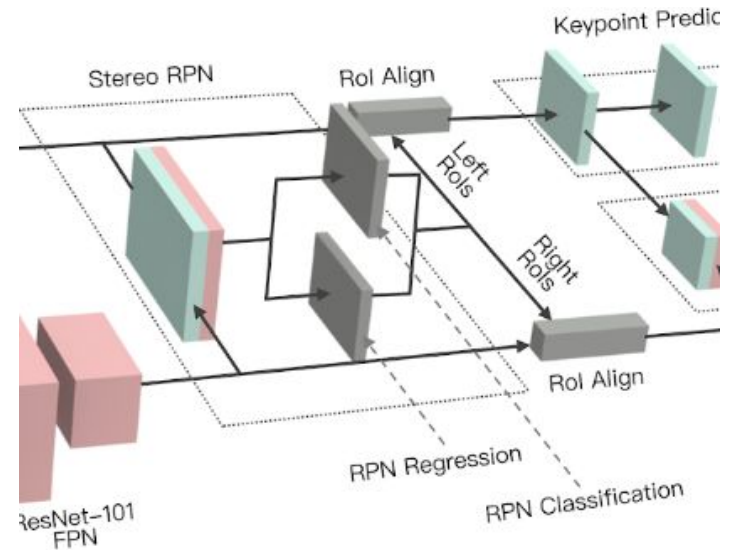
Pre-trained Backbone Networks

- For 2D features extraction in left and right stereo images.
- Weight-share ResNet-101 and FPN (Feature Pyramid Network) as our backbone network to extract consistent features.



Stereo Region Proposal Network(RPN)

- Detects left and right regions of interest (RoI).
- Two sibling fully-connected layer to classify objectness and regress box offsets for each input location.



Stereo Region Proposal Network(RPN)

- Assign the union of left and right GroundTruth boxes(referred as union GT box).
- Get Intersection-over-Union (IoU) ratio with one of union GT boxes.
- An anchor is assigned a positive if $\text{IoU} > 0.7$.
- Negative if $\text{IoU} < 0.3$



Figure 2. Different targets assignment for RPN classification and regression.

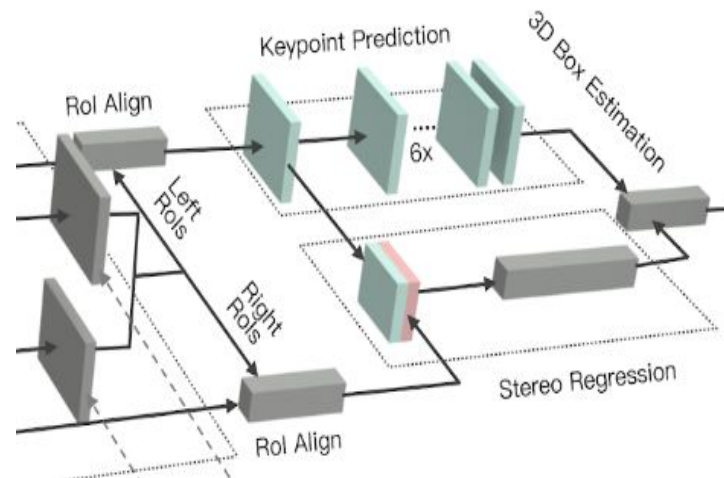


Stereo Region Proposal Network(RPN)

- Benefit from this design, the positive anchors tend to contain both left and right object regions.
- We calculate offsets of positive anchors respecting to the left and right GT boxes contained in the target union GT box, then assign offsets to the left and right regression respectively.
- Assign offsets to the left and right regression respectively.
- There are six regressing terms for the stereo regressor: $[u, w, u^*, w^*, v, h]$, where we use u, v to denote the horizontal and vertical coordinates of the 2D box center in image space, w, h for width and height of the box, and the superscript $(.)^*$ for corresponding terms in the right image using the same v and h .

Stereo Regression

- After stereo RPN, we have corresponding left-right proposal pairs ROIs.
- Apply RoI Align on the left and right feature maps respectively.
- The left and right RoI features are concatenated and fed into two sequential fully-connected layers (each followed by a ReLU layer) to extract semantic information.
- We use four sub-branches to predict object class, stereo bounding boxes, dimension, and viewpoint angle respectively.



Keypoint Prediction

- Feeding the left image only.
- Besides stereo boxes and viewpoint angle, perspective keypoint can provide more rigorous constraints to the 3D box estimation.
- Perspective keypoint: The only 3D box corner point which projected in the box middle (2D box).
- Also predict two boundary keypoints which serve as simple alternatives to instance mask for regular-shaped objects.

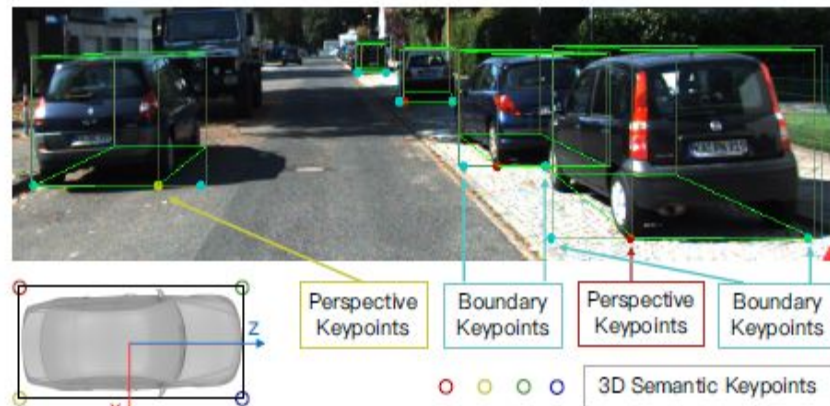


Figure 4. Illustration of 3D semantic keypoints, the 2D perspective keypoint, and boundary keypoints.

3D Box Estimation

- Given the left-right 2D boxes, perspective keypoint, and regressed dimensions, the 3D box can be solved by minimize the reprojection error of 2D boxes and the keypoint.
- States of the 3D bounding box can be represented by $x = \{x, y, z, \theta\}$, which denotes the 3D center position and horizontal orientation respectively.
- seven measurements from stereo boxes and perspective keypoints: $\{u_l, v_t, u_r, v_b, u'_l, u'_r, u_p\}$

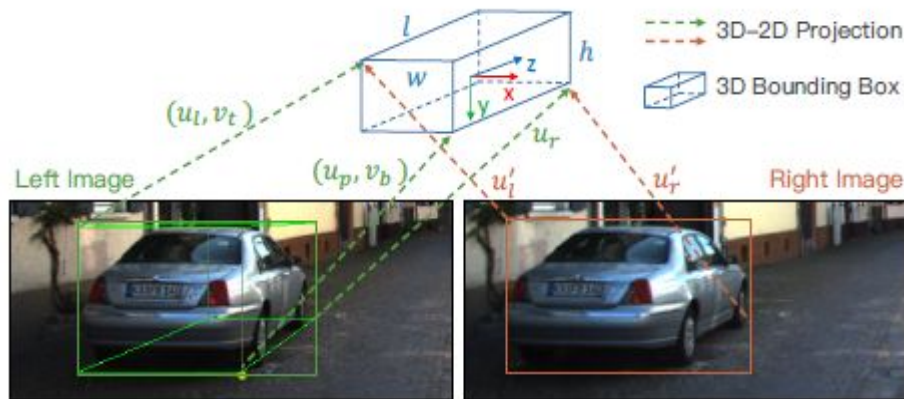
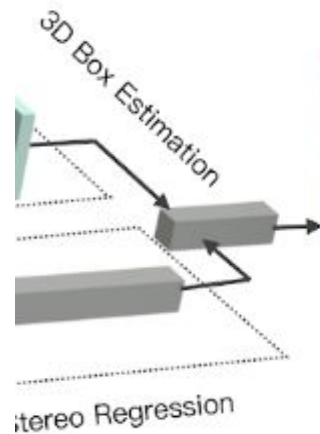


Figure 5. Sparse constraints for the 3D box estimation (Sect. 4).

Dense 3D Box Alignment

- A dense region-based photometric alignment method that ensures our 3D object localization accuracy.
- Solve the disparity of the **3D bounding box center** while using the dense object patch.



Dense 3D Box Alignment

Results

Method	AR (300 Proposals)			AP _{2d} (IoU=0.7)								
				Left			Right			Stereo		
	Left	Right	Stereo	Easy	Mode	Hard	Easy	Mode	Hard	Easy	Mode	Hard
Faster R-CNN[25]	86.08	-	-	98.57	89.01	71.54	-	-	-	-	-	-
Stereo R-CNN _{mean}	85.50	85.56	74.60	90.58	88.42	71.24	90.59	88.47	71.28	90.53	88.24	71.12
Stereo R-CNN _{concat}	86.20	86.27	75.51	98.73	88.48	71.26	98.71	88.50	71.28	98.53	88.27	71.14

Table 1. Average recall (AR) (in %) of RPN and Average precision (AP) (in %) of 2D detection, evaluated on the KITTI *validation set*. We compare two fusion methods for Stereo-RCNN with Faster R-CNN using the same backbone network, hyper-parameters, and augmentation. The Average Recall is evaluated on the *moderate set*.

Method	Sensor	AP _{bv} (IoU=0.5)			AP _{bv} (IoU=0.7)			AP _{3d} (IoU=0.5)			AP _{3d} (IoU=0.7)		
		Easy	Mode	Hard	Easy	Mode	Hard	Easy	Mode	Hard	Easy	Mode	Hard
Mono3D[3]	Mono	30.50	22.39	19.16	5.22	5.19	4.13	25.19	18.20	15.52	2.53	2.31	2.31
Deep3DBox[21]	Mono	30.02	23.77	18.83	9.99	7.71	5.30	27.04	20.55	15.88	5.85	4.10	3.84
Multi-Fusion[27]	Mono	55.02	36.73	31.27	22.03	13.63	11.60	47.88	29.48	26.44	10.53	5.69	5.39
VeloFCN[16]	LiDAR	79.68	63.82	62.80	40.14	32.08	30.47	67.92	57.57	52.56	15.20	13.66	15.98
Multi-Fusion[27]	Stereo	-	53.56	-	-	19.54	-	-	47.42	-	-	9.80	-
3DOP[4]	Stereo	55.04	41.25	34.55	12.63	9.49	7.59	46.04	34.63	30.09	6.55	5.07	4.10
Ours	Stereo	87.13	74.11	58.93	68.50	48.30	41.47	85.84	66.28	57.24	54.11	36.69	31.07

Table 2. Average precision of bird's eye view (AP_{bv}) and 3D boxes (AP_{3d}) comparison, evaluated on the KITTI *validation set*.



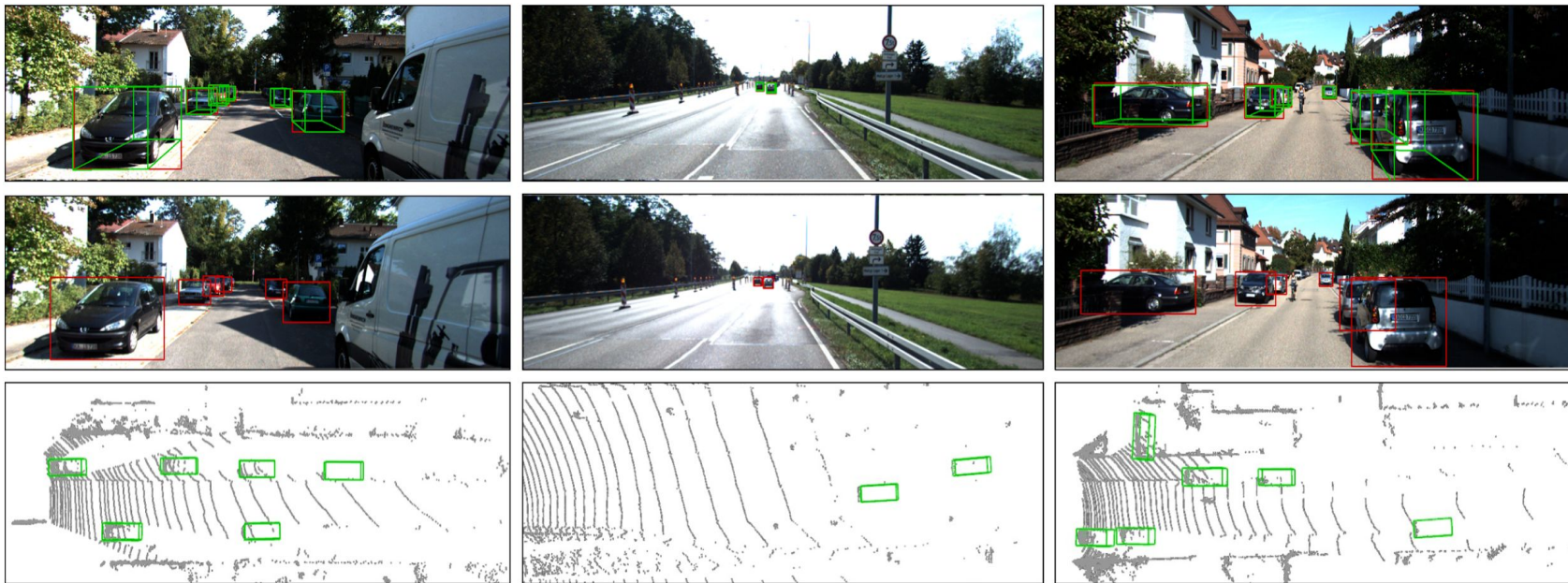
Results

Benefits of the Dense Alignment.

Config	Set	$AP_{bv}^{0.5}$	$AP_{bv}^{0.7}$	$AP_{3d}^{0.5}$	$AP_{3d}^{0.7}$
w/o Alignment	Easy	45.59	16.87	41.88	11.37
	Mode	33.82	10.40	27.99	7.75
	Hard	28.96	10.03	22.80	5.74
w/ Alignment w/o 3D rectify	Easy	86.15	66.93	83.05	48.95
	Mode	73.54	47.35	65.45	32.00
	Hard	58.66	36.29	56.50	30.12
w/ Alignment w/ 3D rectify	Easy	87.13	68.50	85.84	54.11
	Mode	74.11	48.30	66.28	36.69
	Hard	58.93	41.47	57.24	31.07

Table 6. Improvements of using our dense alignment and 3D box rectify, evaluated on KITTI *validation set*.

Results



Qualitative results. From top to bottom: detections on left image, right image, and bird's eye view image.



Download Dataset from kitti benchmark urls

```
!wget https://s3.eu-central-1.amazonaws.com/avg-kitti/data_object_image_2.zip
!wget https://s3.eu-central-1.amazonaws.com/avg-kitti/data_object_image_3.zip
!wget https://s3.eu-central-1.amazonaws.com/avg-kitti/data_object_label_2.zip
!wget https://s3.eu-central-1.amazonaws.com/avg-kitti/data_object_calib.zip
```

```
!unzip -qq /content/data_object_image_2.zip
!rm /content/data_object_image_2.zip
!unzip -qq /content/data_object_image_3.zip
!rm /content/data_object_image_3.zip
!unzip -qq /content/data_object_calib.zip
!rm /content/data_object_calib.zip
!unzip -qq /content/data_object_label_2.zip
!rm /content/data_object_label_2.zip
```



Downloaded Paper 2 Code from Github repo

```
!wget https://github.com/HKUST-Aerial-Robotics/Stereo-RCNN/archive/1.0.zip
```

```
!unzip -qq /content/1.0.zip
```

```
!pip install -r /content/Stereo-RCNN-1.0/requirements.txt
```

```
# !cd /content/Stereo-RCNN-1.0
```

```
%cd /content/Stereo-RCNN-1.0/lib
```

```
!python setup.py build develop
```

```
!cd ..
```



Preparing data and downloading models

```
%cd /content/Stereo-RCNN-1.0
```

```
!mkdir models_stereo
```

```
%cd models_stereo
```

```
my_file_id = "1rIS43NzTvjRMX9m3UZIG5EvGFzXOVZWX"
```

```
!gdown https://drive.google.com/uc?id={my\_file\_id}
```

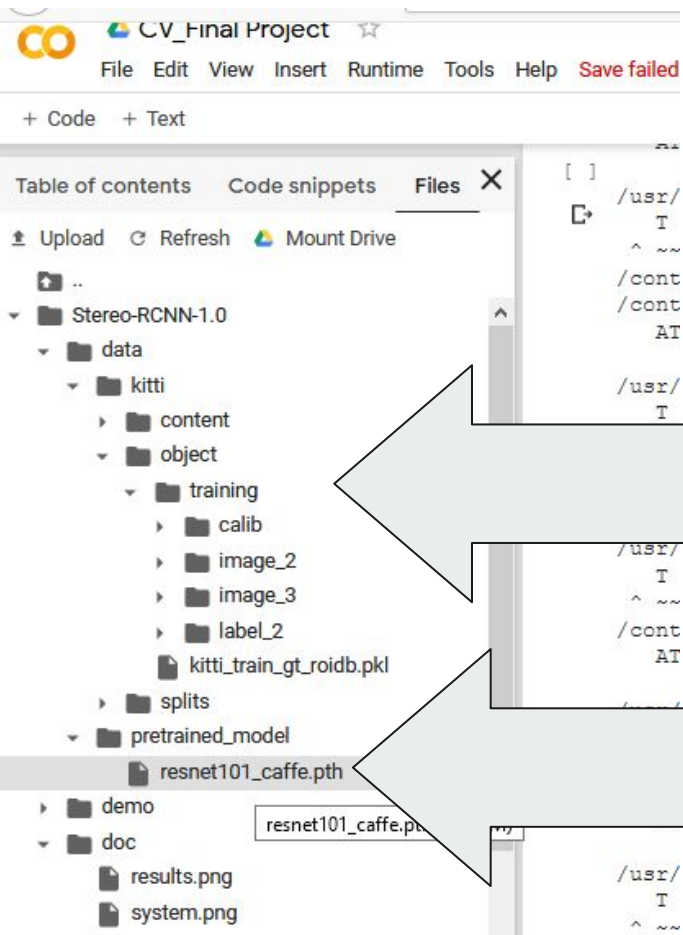
```
%cd /content/Stereo-RCNN-1.0
```

```
!python demo.py
```

Edited some code to solve pytorch compatibility issues before running demo.py and train.sh

```
%cd /content/Stereo-RCNN -1.0/data/kitti
# !rm /content
!mkdir object
%cd /content/Stereo-RCNN -1.0/data/kitti/object
!ln -s /content/training
%cd ../../
%cd /content/Stereo-RCNN -1.0/data
!mkdir pretrained_model
%cd pretrained_model
#
https://drive.google.com/file/d/1\_t8TtUevtMdnvZ2SoD7Ut8sSladyCKTt/view?usp=sharing

my_file_id = "1_t8TtUevtMdnvZ2SoD7Ut8sSladyCKTt"
!gdown https://drive.google.com/uc?id={my\_file\_id}
%cd /content/Stereo-RCNN -1.0
!sh train.sh
%cd /content/Stereo-RCNN -1.0
!sh test.sh
```



Training Data linked into
kitti/data/object folder

Pretrained Resnet 101

Models

3D box alignment and
regressors

Region Of Interest Layers

Region Proposal Network

Resnet and stereo RCNN
“For 2D stereo Detection”

- └─ Kitti.py
- └─ faster_rcnn.egg-info
- └─ model
 - └─ csrc
 - └─ dense_align
 - └─ __init__.py
 - └─ box_3d.py
 - └─ dense_align.py
 - └─ roi_layers
 - └─ __init__.py
 - └─ nms.py
 - └─ roi_align.py
 - └─ roi_pool.py
 - └─ rpn
 - └─ __init__.py
 - └─ anchor_target_layer.py
 - └─ bbox_transform.py
 - └─ generate_anchors.py
 - └─ proposal_layer.py
 - └─ proposal_target_layer.py
 - └─ stereo_rpn.py
 - └─ stereo_rcnn
 - └─ __init__.py
 - └─ resnet.py
 - └─ stereo_rcnn.py
 - └─ utils

3D Detection and Localization

