Camera only 3D object detection

Yukang Yang
yukang@stanford.edu

Abstract

While Lidar provides a more accurate 3D point cloud for perception tasks, camera only solution is more accessible due to lower cost. This project explores to replace Lidar with a pair of rectified cameras, produce 3D point cloud based on disparity, and apply the camera based point cloud to downstream task such as 3D object detection.

Code location: commit 1, commit 2

1. Introduction

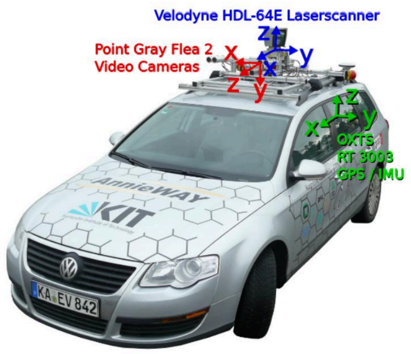
Autonomous driving starts with understanding the surrounding environment. Data of the environment are commonly collected from sources such as camera, lidar, radar, map, GPS, etc. Lidar and radar are less accessible due to their cost. Map and GPS are more of supplemental data. Camera is the only accessible way to receive environment data. So camera only environment understanding is an interesting problem to solve. 3D object detection is a common task for understanding the environment.

2. Problem statement

The camera only 3D object detection detects the 3D bounding box of objects and object category from camera image(s). When a sequence of images are provided, the task may also ask for velocity of the object. The image(s) can come from single camera or multiple camera. The images may or may not overlap. Camera parameters are provided.

To simplify the problem, this work only detects the 3D bounding box. Object label and object velocity are not target of this work. When only one image is provided or when images don't have overlap, 3D reconstruction requires prior knowledge about size of the object. Learning the prior knowledge requires a large dataset. Given the limited data and timeline, this work focuses on cases when stereo information is available. In other words, multiple images have to be provided and the object in interest has to be fully visible in both images.

The waymo dataset was explored initially but unfortunately, there is almost no overlap between camera views in the waymo dataset. Objects are only fully visible in one image. Another difficult part of the waymo dataset is that objects can be split into 2 cameras. So to reconstruct the 3D bounding box of objects, object point clouds from all images need to be consolidated into a combined point clouds. Due to the complexity above, this work runs on KITTI dataset.



The KITTI dataset collects images from a pair of rectified cameras. Images from 2 cameras has a large overlap so stereo information can be extracted. The dataset contains around 7400 image pairs in training set. Object 3D bounding box is annotated for every image pair. Camera projection matrix is provided. Images are of size (1227, 370). Baseline between 2 cameras is roughly 54cm.

2.1. Prior work

Object detection has been studied extensively in 2D images. CNN based approach such as YOLO [7], SSD [4] and transformer based approach such as DETR [1] have been proposed.

In 3D objective detection, there are several tracks:

- start with 2D object detection in image plane, and then transform 2D bounding box into 3d with other knowledge (depth or other intermediate representation) from the image [8].
- convert image into bird eye view (BEV) or point cloud with depth estimation and then find 3D bounding box from the bird eye view [6].

- extend DETR to 3D by building a 3D positional encoding with depth for the 2D image [5].

2.2. Evaluation

Average precision AP is commonly used for evaluating object detection task [2].

There are multiple ways to define match between 3D bounding box:

- Intersection over union (IoU): For 3D IoU, first option is to compute the ratio between intersection volume and union volume. Second option is to convert the 3D box into 2D polygon in image plane and do 2D IoU. Third option is to project 3D box to the 2D ground plane and do 2D IoU.
- instead of IoU, compute the distance between the center of the box. Similar to IoU, the box can be 3d box, image plane 2d polygon, or ground plane 2d box.

After match method is determined, AP is calculated with following process:

- Given a match threshold, find matching pairs between predicted box and ground truth box. Matched predicted box is true positive (TP). Non matched predicted box is false positive (FP). Non matched ground truth box is false negative (FN). precision=TP/(TP+FP). recall=TP(TP+FN).
- Enumerate different match thresholds to plot the precision recall curve (PR curve).
- compute AP as the area under the PR curve.

3. Technical approach

The 3D object detection will be broken down into 2 steps: first get 3D point cloud, then convert 3D point cloud into 3D bounding box.

3.1. Camera parameter extraction

KITTI dataset provides projection matrix of both left and right cameras. Intrinsic and extrinsic parameters are not provided but can be derived with a closed form formula from lecture. The formula in lecture contains a non-unique ρ which represents the scale of projection matrix. Assume the provided projection matrix is already calibrated to the right scale, ρ can be assumed to be 1. This can also be partially verified by checking whether the norm of a_3 is 1, where a_3 is the first 3 numbers in the 3rd row of projection matrix. The intrinsic provides information about focal length, skewness, and aspect ratio of image pixel. The first number in the first row of intrinsic matrix is focal length on x axis.

3.2. Sanity check the camera setup

After intrinsic and extrinsic are known, the camera setup can be verified. The check examines how well cameras are rectified and how accurate is the provided baseline length.

The baseline and rectification can both be verified by examining the relative pose between cameras. Let left camera's projection matrix be M_1, right camera's projection matrix be M_2.

$M_1 = K_1[R_1|T_1]$
$M_2 = K_2[R_2|T_2]$
$R_{12} = R_2 R_1^T$
$T_{12} = T_2 - R_{12}T_1$

T_{12} is the relative translation from camera 1 to camera 2.

R_{12} is the relative rotation after relative translation from camera 1 to camera 2.

For a properly rectified camera setup. T_{12} should be 0 in y and z axis. The x axis of T_{12} should be close to provided baseline length.

Besides the relative pose check, the rectified setup can also be verified by comparing the epipolor line of matching points in left and right images. For a correct setup, the 2 epipolor line should be horizontal and have the same vertical coordinate.

The epipolar line can be compared through follow process:

- pick a corner of object 3D bounding box
- project the 3D corner to image: $p = MP$
- compare vertical difference between matching points
- compute epipolar line:

  $E = [T_{12}]_x R_{12}$
  $F = K_2^{-T} E K_1^{-1}$
  $l = Fp'$
  $l' = F^T p$
- compute the angle of between epipolar lines and x axis.

3.3. Disparity computation

Get disparity d by stereo block matching. Images are converted to grayscale for matching. Grayscale image is created with following formula:

$Y = 0.2125R + 0.7154G + 0.0721B$

In rectified image pairs, matching point always has the same vertical coordinate as the original point. The block matching algorithm matches block on the same vertical coordinate between left image and right image. Disparity d is computed as the horizontal pixel distance between matching blocks. This work examines 3 ways to match blocks: sum square difference (SSD), sum absolute difference (SAD),

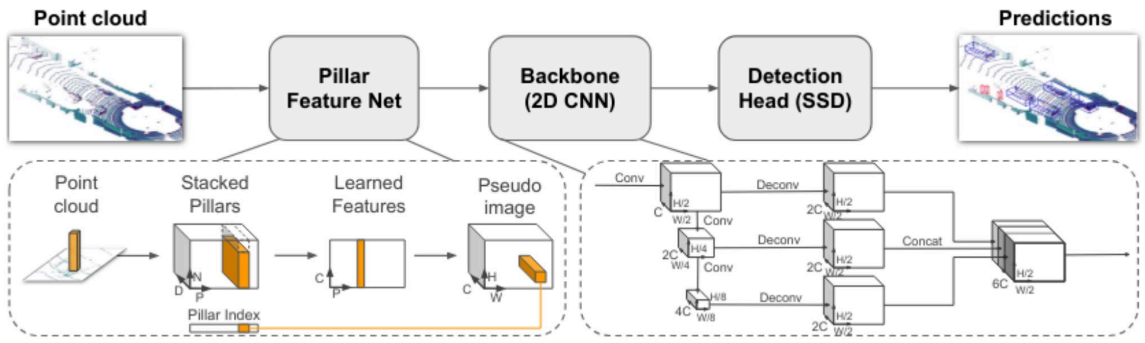


Figure 1. PointPillar architecture from PointPillar paper [3]

and normalized cross correlation (NCC). Suppose x is the block from left image, y is the block from right image.

$SSD = \Sigma_{ij}(x_{ij} - y_{ij})^2$

$SAD = \Sigma_{ij}|x_{ij} - y_{ij}|$

$NCC = \frac{(x-\bar{x})\cdot(y-\bar{y})}{|x-\bar{x}||y-\bar{y}|}$

The naive implementation of block matching algorithm has complexity of $O(w * h * d * bk^2)$, where w and h are image shape, d is max disparity to search, bk is block size. The algorithm can be optimized through dynamic programming. When block moves horizontally or vertically, only last row or column is new to computation. Other rows or columns can be memorized from earlier computation. So the complexity can be reduced to $O(w * h * d * bk)$.

Disparity is computed just for left image. Depth and point cloud below will both be computed from left image.

3.4. Depth computation

Depth z can be computed as:

$z = \frac{b*f}{d}$

where b is the baseline length (distance between 2 camera centers) in meter, f is the x axis focal length in pixel, d is the disparity in pixel.

3.5. Point cloud

Once depth is known, each image coordinate can be converted into a unique homogeneous coordinate (uz, vz, z), where u is the x coordinate in image, v is the y coordinate in image, z is the depth in world.

Depth map is first converted to 3d point cloud P_c in homogeneous camera coordinate and then converted to 3d point cloud P_w in homogeneous world coordinate

$$P_c = \begin{pmatrix} K_1 & 0 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} uz \\ vz \\ z \\ 1 \end{pmatrix}$$

$$P_w = \begin{pmatrix} R_1^T & -R_1^T T_1 \\ 0 & 1 \end{pmatrix} P_c$$

3.6. 3D bounding box

PointPillar [3] is a prior work to do 3D object detection from lidar point cloud. The project reuses the model architecture of PointPillar but replaces the lidar point cloud with camera point cloud. Lidar point cloud consists of point 3D coordinate and point reflective signal. For camera based point cloud, greyscale will be used as the reflective signal.

PointPillar processes the point cloud in follow steps:

- Convert 3D point cloud to pseudo-image
- Process the pseudo-image with a 2D object detection network and output prediction for each anchor
- Run non max suppression to drop duplicated detections

3.6.1 3D point cloud to pseudo-image

The x-y plane is divided into grids that are called pillar. For each pillar, keep only a specified number of points. If there are more points then needed, random sample up to expected number of points. If there are not enough points, fake points are padded. For the specified number of points, append their coordinate and distance from the pillar center to the pillar. The information in each pillar is then converted to a embedding with non-linear transformation. The output has shape (x, y, d), which is similar to a image expect that it has more channels than image. The output is therefore called pseudo-image.

3.6.2 Pseudo-image to anchor prediction

The pseudo-image is then passed to a 2D object detection pipeline. A difference from 2D detection is that the anchor is now 3D. Since the pillar compresses all depths in the same x-y grid, the 3D anchor is also just one layer in z-axis. For each anchor, the pipeline predicts the likelihood of each object category and if there is object, the offset from the anchor , the size of the object, and the orientation of the object.

The anchor predictions will be deduplicated with non max suppression, which only keeps the most confident detection and removes less confident overlapping detections.

Ground truth is attached to each anchor based on intersection over union ratio.

3.6.3 Data augmentation

Besides traditional augmentation strategy such as flipping image horizontally, random rotation, scaling, translation, and randomly adding noise to ground truth. A special augmentation in PointPillar [3] and its predecessor SECOND [9] is to artificially add objects from other images to the image. SECOND [9] claims that without the artifical objects, there are too few objects in the image, which would lead to slow convergence and affect final performance. When artificial objects are added, they are checked with existing objects to avoid overlapping objects.

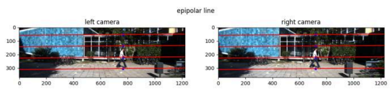
4. result

4.1. Camera parameters

The intrinsic matrix shows the camera has square pixel, zero skew, and a focal length of 707 pixels.

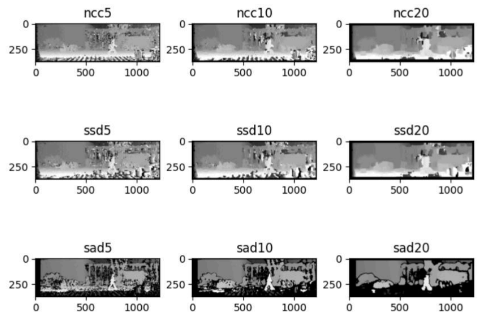
4.2. Rectification verification

Based on computed relative pose, 2 cameras has no relative rotation. But there is slight translation in y and z: +4mm translation in y and -1mm translation in z. The derived baseline length is 53.57cm, slightly less than the provided 54cm. The angle between epipolar line and x axis is around $0.45°$. The vertical difference between matching points is 0.4 pixel. The setup is not fully rectified due to the small relative translation in y and z but the error is negligible because matching epipolar lines are mostly horizontal and matching point vertical difference is less than 1 pixel.



4.3. Disparity and depth map

SSD, SAD, NCC are all tried with half block size of 5 pixels, 10 pixels, and 20 pixels. SSD and NCC are naively implemented. SAD uses opencv implementation.



In figure 2 below, white means large disparity, black means small disparity or missing disparity. Small block size is more detailed but more noisy. Large block size is less detailed and less noisy. The pedestrian disparity is correctly derived but ground disparity is wrong or missing due to homogeneous region and repetitive patterns.

Depth map is just an inverse of disparity map. White turns black and black turns white. So depth plot is skipped here.

4.4. 3D point cloud

A corner of the object bounding box is picked for one off comparison. The ground truth 3D coordinate is (1.84, 1.47, 8.41). The computed 3D coordinate from disparity is (1.83150045, 1.46007197, 8.41260183). The error is around 1cm. At least for this corner, the 3D construction is correct.

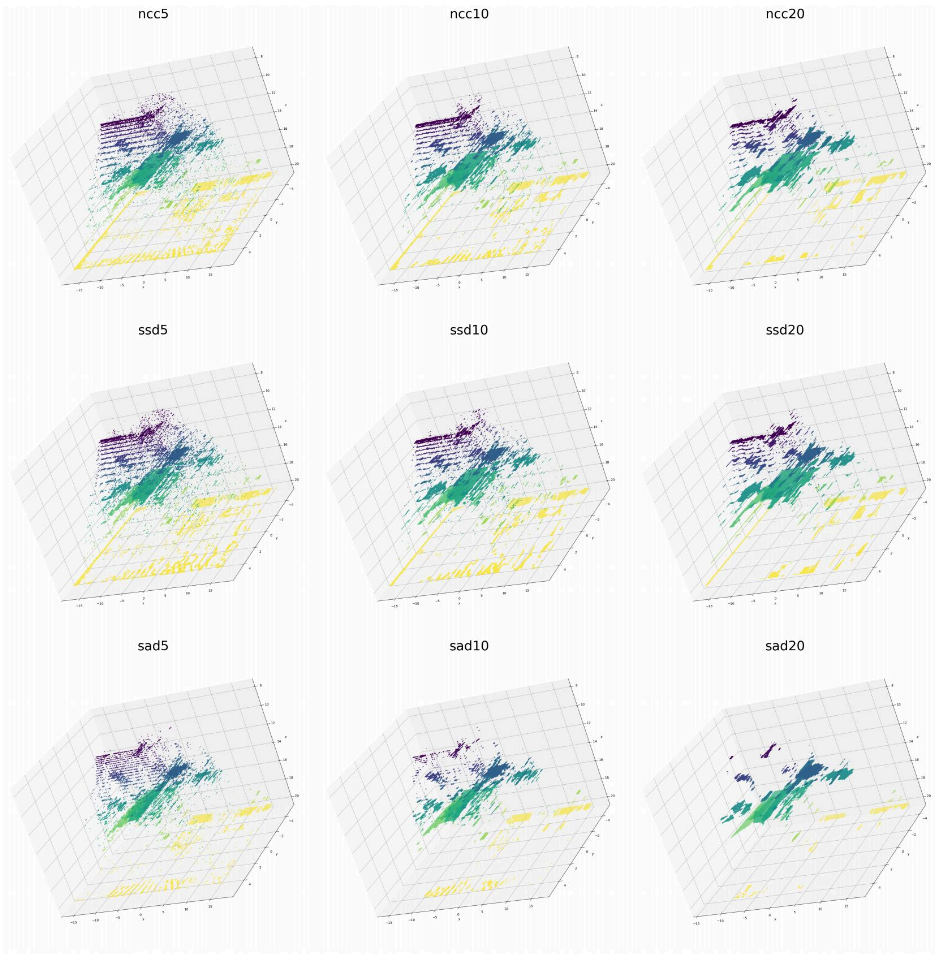
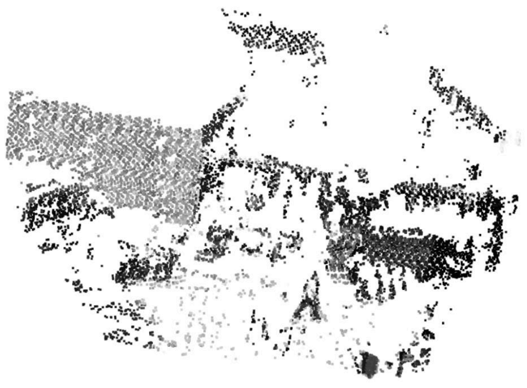
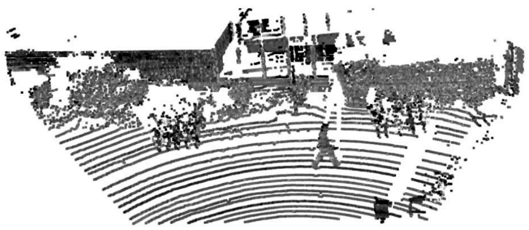


Figure 2. Point cloud


(a) Camera point cloud


(b) Lidar point cloud

The point cloud shows the similar result as disparity graph. Pedestrian 3D coordinates are successfully constructed. Ground before the pedestrian largely failed in the 3D construction.

Camera point cloud contains 10 times more points than the Lidar point cloud. To avoid out of memory error, the camera point cloud is downsampled 10 times.

4.5. 3D detection

Separate PointPillar models are trained with lidar point cloud and camera point cloud. Camera point cloud is created with sum absolute difference (SAD) with half block size of 5, 10, 20. The models are trained with 10 epochs for each point cloud.

Object category	Lidar	SAD5	SAD10	SAD20
Pedestrian	0.37	0.13	0.14	0.11
Cyclist	0.53	0.06	0.05	0.04
Car	0.67	0.24	0.23	0.14
All	0.52	0.14	0.14	0.10

Table 1. Average precision of each method

The low performance of camera point cloud is partially due to the downsampling and also partially due to missing disparity on the objects. Downsampling and missing disparity lead to much less points on the objects.

Object category	Lidar	SAD5	SAD10	SAD 20
Pedestrian	92	8	6	0
Car	88	39	41	24
Cyclist	79	8	5	0

Table 2. Median number of points on object

5. Conclusion

While camera only solution reduces the hardware cost, an additional step to predict the depth is introduced. Naive method such as block matching isn't reliable enough to predict depth on key objects. A future work would be to use deep learning based depth predictor which can hopefully close the gap between camera and lidar.

6. Scope

This project is related to epipolar geometry, stereo system, representation, and representation learning from the course.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. 1

[2] Wei-Chih Hung, Henrik Kretzschmar, Vincent Casser, Jyh-Jing Hwang, and Dragomir Anguelov. Let-3d-ap: Longitudinal error tolerant 3d average precision for camera-only 3d detection, 2022. 2

[3] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds, 2019. 3, 4

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, page 21–37. Springer International Publishing, 2016. 1

[5] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection, 2022. 2

[6] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d, 2020. 1

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 1

[8] Andrea Simonelli, Samuel Rota Rota Bulò, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection, 2019. 1

[9] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 4