COMPUTER RESEARCH AND MODELING
2022 VOL. 14 NO. 6 P. 1239–1253
DOI: 10.20537/2076-7633-2022-14-6-1239-1253

КиМ

MODELS IN PHYSICS AND TECHNOLOGY

UDC: 004.896

Lidar and camera data fusion in self-driving cars

M. Ahmed[1,a], M. Hegazy[1,b], A. Klimchik[2,c], R. A. Boby[3,d]

[1]Institute of Robotics and Computer Vision,
420500 Innopolis, Russia
[2]School of Computer Science,
University of Lincoln, United Kingdom
[3]Mechanical Engineering, Indian Institute of Technology Jodhpur,
Karwar, Jodhpur, Rajasthan, India, 342037

E-mail: [a] o.ahmed@innopolis.university, [b] m.hegazy@innopolis.university, [c] alexandr.klimchik@gmail.com,
[d] ribyab@gmail.com

Received 15.09.2022, after completion — 06.10.2022.
Accepted for publication 10.10.2022.

Sensor fusion is one of the important solutions for the perception problem in self-driving cars, where the main aim is to enhance the perception of the system without losing real-time performance. Therefore, it is a trade-off problem and its often observed that most models that have a high environment perception cannot perform in a real-time manner. Our article is concerned with camera and Lidar data fusion for better environment perception in self-driving cars, considering 3 main classes which are cars, cyclists and pedestrians. We fuse output from the 3D detector model that takes its input from Lidar as well as the output from the 2D detector that take its input from the camera, to give better perception output than any of them separately, ensuring that it is able to work in real-time. We addressed our problem using a 3D detector model (Complex-Yolov3) and a 2D detector model (Yolo-v3), wherein we applied the image-based fusion method that could make a fusion between Lidar and camera information with a fast and efficient late fusion technique that is discussed in detail in this article. We used the mean average precision (mAP) metric in order to evaluate our object detection model and to compare the proposed approach with them as well. At the end, we showed the results on the KITTI dataset as well as our real hardware setup, which consists of Lidar velodyne 16 and Leopard USB cameras. We used Python to develop our algorithm and then validated it on the KITTI dataset. We used ros2 along with C++ to verify the algorithm on our dataset obtained from our hardware configurations which proved that our proposed approach could give good results and work efficiently in practical situations in a real-time manner.

Keywords: autonomous vehicles, self-driving cars, sensors fusion, Lidar, camera, late fusion, point cloud, images, KITTI dataset, hardware verification

Citation: *Computer Research and Modeling*, 2022, vol. 14, no. 6, pp. 1239–1253.

The work of A. Klimchick and R. A. Boby was supported by the grant of the Russian Science Foundation No. 22-41-02006, https://rscf.ru/project/22-41-02006

Introduction

Object detection is a fundamental problem in many fields and has a huge impact on self-driving cars. Relevant reliability and safety can be achieved with the help of different sensors such as ultrasonic, cameras, radars and Lidars mounted on vehicles with redundancy resolution techniques and sensor fusion algorithms.

Classification approaches used in recent years have focused on image recognition research. To produce proposals for bounding boxes, these approaches generate object proposals such as sliding windows [Asvadi et al., 2018], edge boxes [Zitnick, Dollár, 2014], choose search [Uijlings et al., 2013], Multiscale Combinatorial Grouping (MCG) [Pont-Tuset et al., 2016], and then utilize a CNN pipeline [Girshick et al., 2015; Chen et al., 2017a] to perform recognition for the suggested object region. The high computational cost is a typical drawback. Furthermore, cameras lack information on the 3D location, orientation, and shape of objects, as well as fluctuating lighting levels, which results in inaccurate object region proposals.

Using the complementary information offered by LIDAR and cameras to obtain very precise object positions and classifications for self-driving cars is one solution. In other words, good fusion techniques can play a great role in minimizing the disadvantages of both sensors and allows autonomous vehicles to work in real time with accurate precision for object detection.

Literature Review

Many recent researches have focused on the merging of data from multiple sensors. A typical method is to merge the LIDAR point cloud data with the camera images at the pixel level, with a matching RGB color pixel for each LIDAR point within the image [Schoenberg, Nathan, Campbell, 2010]. Another approach is to take the data features of each sensor and combine them to identify and track moving objects [Cho et al., 2014; Dollár et al., 2014; Foster, Schott, Messinger, 2008; Geiger et al., 2013; Girshick et al., 2014; Girshick et al., 2015; Girshick, Fast, 2015; He et al., 2015; Ku et al., 2018; Li et al., 2015; Li, Zhang, Xia, 2016; Liang et al., 2018; Liu et al., 2016; Minemura et al., 2018; Oh, Kang, 2017]. They introduced a Multi-View 3D network (MV3D) for 3D object recognition in [Chen et al., 2017], which integrates several views of LIDAR point cloud data with images to propose and classify 3D objects. For small object classes, an enhanced deep learning model called AVOD (Aggregate View Object Detection) [Ku et al., 2018] has been presented that multimodally fuses data provided by point cloud and images to build high-resolution feature maps for the production of trustworthy 3D object suggestions. They use continuous convolutions to fuse LIDAR and image feature maps at various resolution levels for 3D object recognition in [Liang et al., 2018]. Rather than recognizing things independently from LIDAR point clouds or images, this method combines the final findings acquired by the two sensors.

Sensor fusion is one of the important solutions for the perception problem in self-driving cars and autonomous systems in general for many aspects like localization, perception, etc. We need to improve the perception of our system without losing real-time performance. At the same time, it is a trade-off problem where most of the models that have high environmental perception cannot perform in a real-time manner and vice versa for models which can work in real time will neglect important information which is necessary for enhancing the perception. Therefore, it is essential to continue tracking the performance of the developed sensor fusion technique not to lose either perception or real-time performance at the cost of the other.

Methodology

Fusion types

For fusion there are 3 types as shown in Table 1 and Figure 1. In this article we use late fusion because it depends on pure extracted features from the models associated with each sensor separately as shown in Figure 1. This gives us the advantage to precept more objects in our environment.

Table 1. Fusion Evaluation

Fusion type	Details	Pros	Cons
Late Fusion	Late Fusion uses models that process each sensor independently until the end and fusion is done at the final stage as shown in Figure 1	high recall	low precision
Early Fusion	Early fusion strengthens the power of deep learning to learn from multiple sensors modalities at the same time. With early fusion, the neural networks take raw sensor observations as input and learn the complementary strengths and weaknesses of the modalities, as shown in Figure 1	high precision	low recall
Intermediate Fusion	Intermediate fusion uses different models at different stages of the pipeline. Some of the models in a specific state use the extracted features from models of previous states as its input, as can seen in Figure 1	high precision and recall	low inference speed

Complex-Yolo Model (Lidar)

For the 3D object detector, we use the Complex-YOLOv3 model which is shown in Figures 2 and 3. Complex-Yolov3 works by prepossessing the Lidar point-cloud data and transforms this point-cloud to a bird-eye-view (bev) RGB-map [Simon et al., 2019]. This transformation is done based on density, intensity and height. For example, the blue channel represents density and the denser points are in some regions of darker blue shades than they will have on the created RGB map. The red channel is for intensity, so the higher the intensity the reflected point will have, the darker shades of red it will have on the created RGB map. Lastly, the green channel it represents the height of the point, so the higher a point, it will be represented by darker shades of green. The Complex-YOLO network takes the bev RGB map as input. It uses a simplified YOLOv3 CNN architecture extended by complex angle regression and E-RPN (Euler Region Proposal Network) to detect accurate multiclass-oriented 3D objects while still operating in real time.

Yolo-v3 Model (Images)

For the 2D objects detector, we will use the Yolo-v3 model (as we can see in Figure 4), as it gives less inference time when working with images in the KITTI dataset (15 ms) compared to the Yolo-V4 model with inference time (45 ms), we will also not resize the input because resizing the inputs gives poor results when we try to do so.

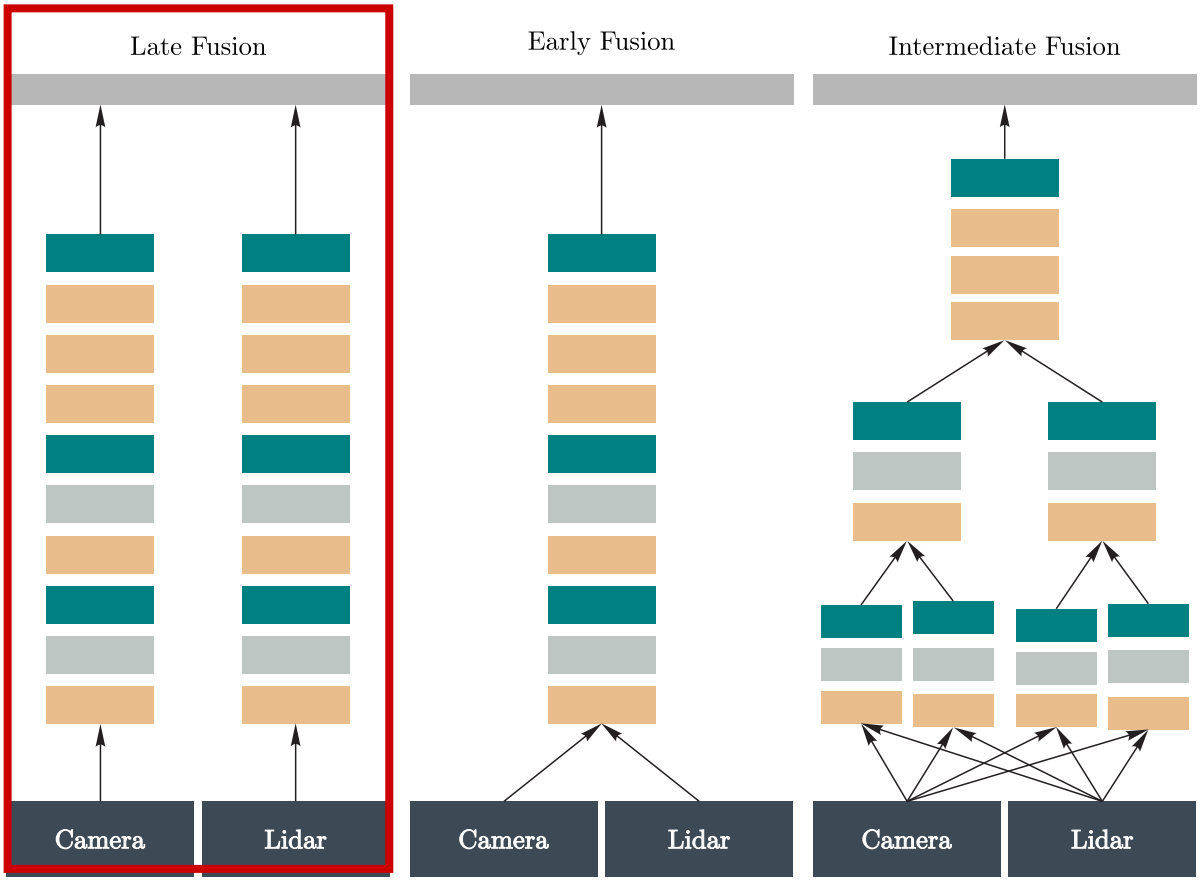


Figure 1. Types of Fusion

Image Based Fusion

The summary of the working algorithm is explained next. We use bounding boxes predicted by a 3D object detector (Complex-yolo) that are less likely to be objects and overwrite labels of those objects with that Region of Interests (ROIs) by a 2D object detector (Yolo-v3) as illustrated in Figure 5. The detected objects from the 3D object detector are then projected onto image planes, and then, if the ROIs of clusters and ROIs are overlapped by a detector, the labels of clusters are overwritten with those of ROIs by the 2D object detector. The Intersection Over Union (IoU) is used to determine whether there are overlaps between them. The advantage in image-based fusion is that we avoided the other problems of the above methods of taking fusion to higher dimension space using point cloud-based fusion. Also, we avoided problems of associations of ROI through different frames as the Kalman filter, tackling our problem in a 2D dimension wherein the use of only the current frame makes it less computationally expensive to do the fusion and getting higher perception with the possibility to work in real time.

To compute the feature based on object distance, a dedicated approach was used. We used the point cloud projected onto the image and mapped it to the bounding boxes that are output from the Yolo-V3 model.

The problem is that the Lidar data is sparse, so not every pixel in the image will have a corresponding point from Lidar. We managed to tackle this problem by using the Nearest-Neighbor technique.

The bounding box of the Yolo-V3 model is an array of 4 values $[x, y, w, h]$. We are interested in the center of the object which we will denote as $C \equiv (C_x, C_y)$, and in this case $C = \left(x + \frac{w}{2}, y + \frac{h}{2}\right)$.

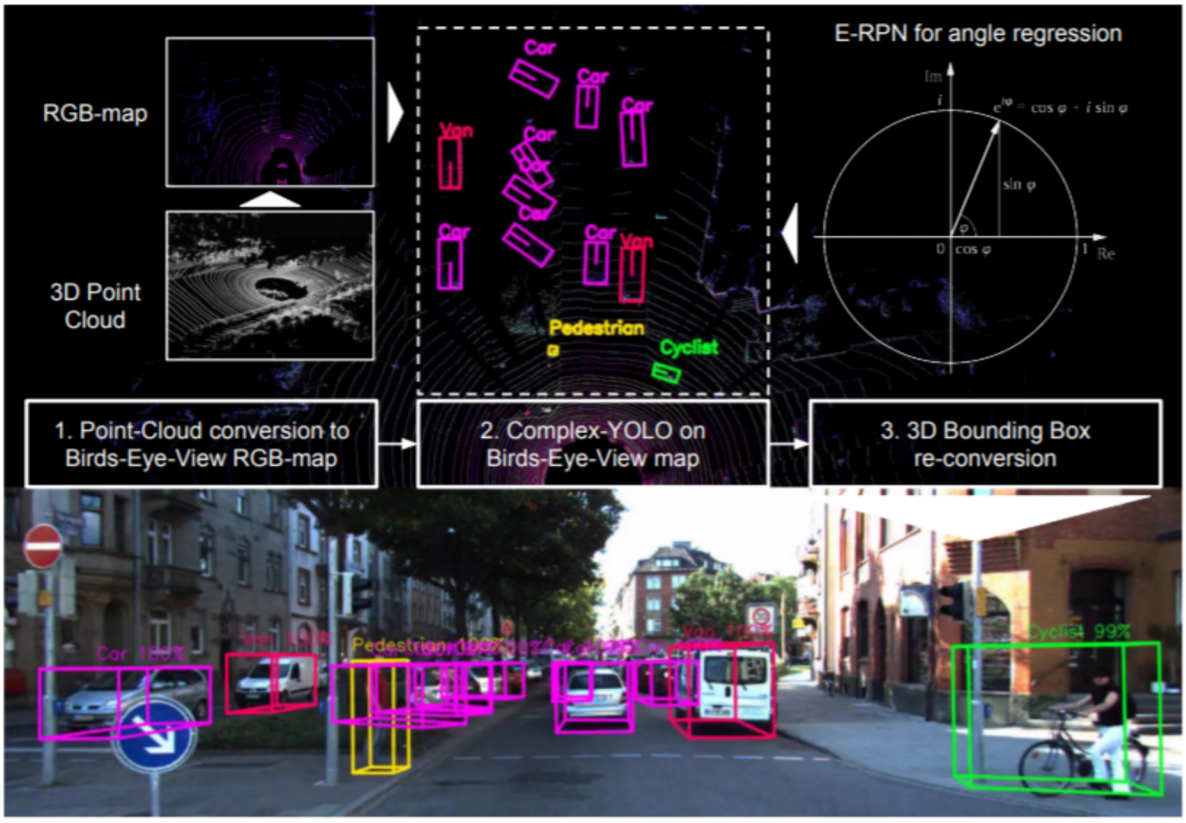


Figure 2. 3D object detector model (Complex-Yolo) for point cloud working idea and results [Simon et al., 2019]

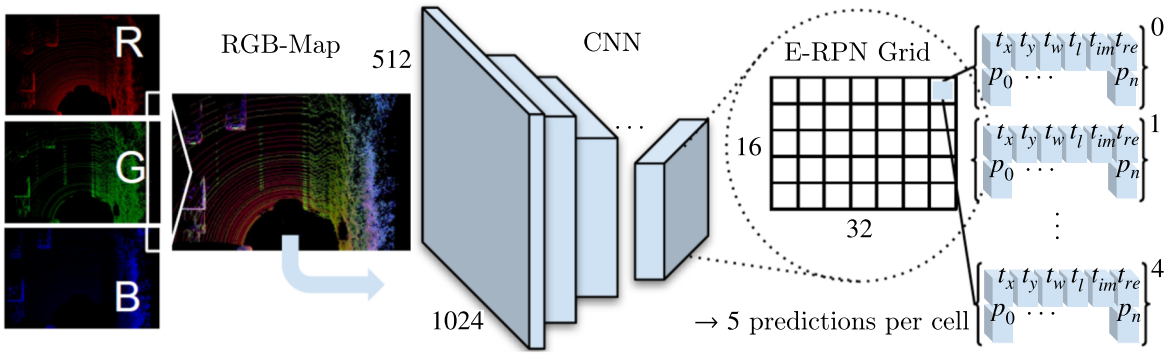


Figure 3. 3D object detector (Complex Yolo) Architecture [Simon et al., 2019]

We will then try to find the assigned Lidar projected point for C. But since Lidar is sparse, most probably we will not find a point assigned to this center pixel C, so we need to search for the nearest point to C. We will denote the projected point cloud to the image plane as a vector P, where $P = = [p_1, p_2, \ldots, p_n]$. Now we will search for the nearest element in P, which is $[p_1, p_2, \ldots, p_n]$ vector to point C, so we will try find the minimum distance from C to p_i where is the distance is given by $distance = \sqrt{(C_x - p_{xi})^2 + (C_y - p_{yi})^2}$.

Evaluation and Discussion

For evaluation of our models, we used the Average Precision metric (AP) and frames per second (FPS). For more information, one may refer to the appendix.

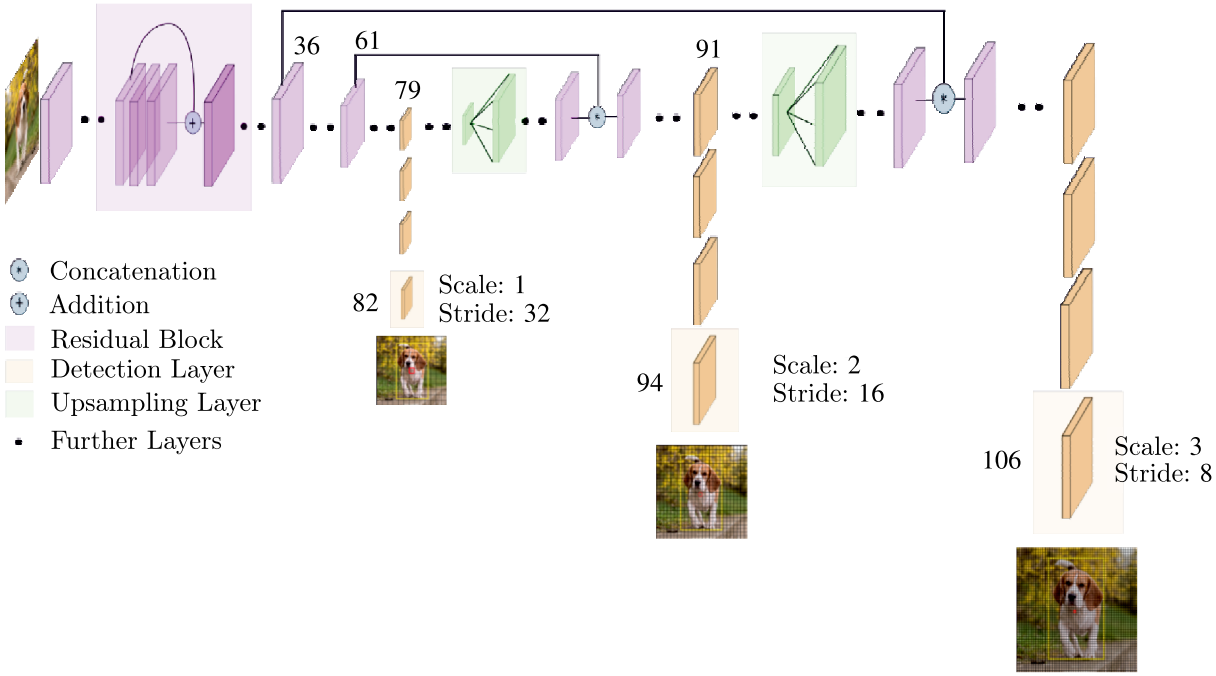


Figure 4. 2D object detector (Yolo-V3) Architecture [Redmon, Farhadi, 2017]

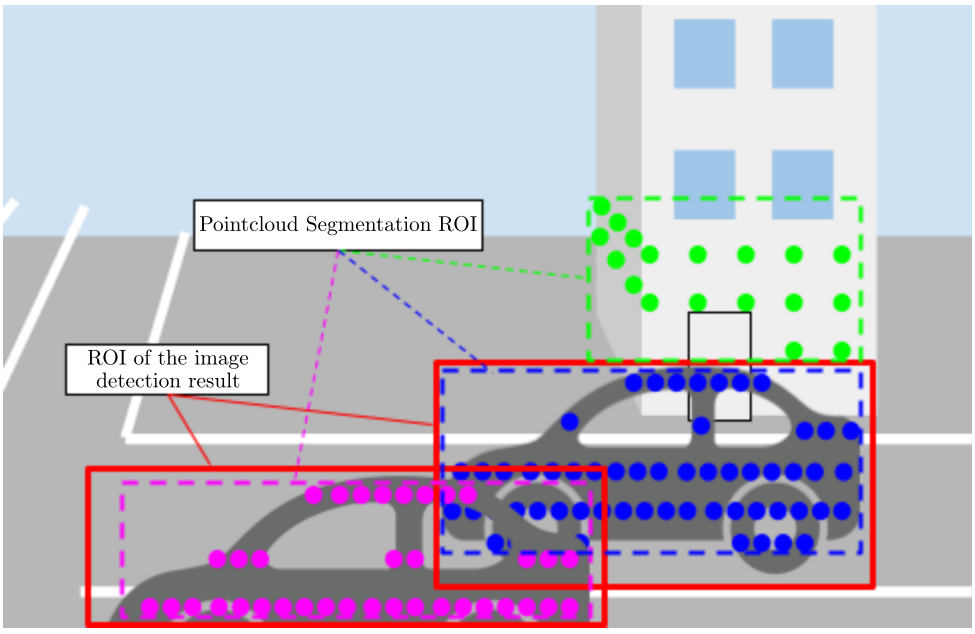


Figure 5. Image Based Fusion

Evaluation of Complex-Yolo

After 220 epochs of training for the tiny Complex-Yolo, which took 2 days, the results are obtained. As we can see in Figure 6, the dashed lines represent the precision recall curve for each class (each class with its corresponding color), while the average precision of each class and of all the classes together in addition to the frames per seconds (FPS) for the model are stated in Table 2.

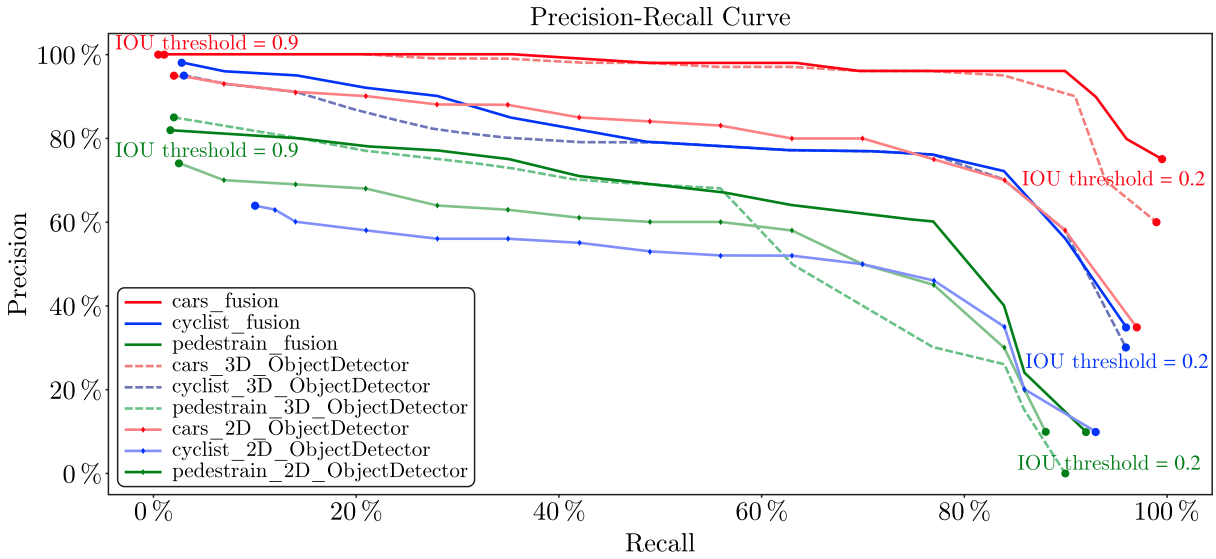


Figure 6. Comparison between different models (the dashed lines represent complex yolo {3D object detector}, the dotted lines represent yolo-v3 {2D object detector}, the solid lines represent fusion) and each color represents a different class

Table 2. Fusion Evaluation

Model/Class	Cars	Cyclist	Pedestrians	Average	FPS
Complex-Yolo	96.3 %	77.6 %	54.4 %	76 %	50
Yolo-v3	78.5 %	42 %	49 %	56.5 %	**66**
Image Based Fusion	**97.3 %**	**80 %**	**59.5 %**	**79 %**	28

Evaluation of Yolo-V3

After 2000 epochs of Yolo-V3 training, which took 4 hours, the results are obtained. In Figure 6, the dotted lines represent the precision recall curve for each class. As mentioned earlier, the average precision of each class and of all the classes together in addition to the frames per second (FPS) for the model are stated in Table 2.

Evaluation of Image Based Fusion

After we had applied the proposed image-based fusion we managed to obtain the results on different datasets which are presented in this section. The solid lines in Figure 6 are the precision recall curves after applying the fusion. The average precision of each class and then the average of all classes and the processing speed of the model are presented in Table 2.

In Figure 6 we can see that the solid red line mostly overlaps with the dashed red line. But for higher precision, the recall shifts to the right. This is logical, as the average precision of Complex-Yolo for cars class is already so high (96.25 %). But after the fusion some true positives are added over the same ground truth bounding box number, and the precision will mostly be the same as the complex-yolo already gives high precision. This proves that the fusion can give better results as the final average precision of the class cars is raised by 1 % to be 97.25 %.

On the other hand, we can see that the solid blue curve in Figure 6 results in low precision at some point of the Complex Yolo model for the cyclist class which is represented by the dashed blue line in the same figure with the help of the yolo-v3 model in which its standalone performance was not as high according to the precision recall curve (represented by the dotted blue line in Figure 6). So we can see that fusion can give better results as the final average precision of the class cyclist increased

from 77.56 % of the complex yolo and 41.96 % of yolo-v3, to be 79.85 % after fusion. This is better than both models and gives an increase of 2 % from the best result of the complex yolo model.

We can also see that the solid green line in Figure 6 overlaps with the dashed and dotted green lines in the same figure starting from one, but gives a slightly better average precision. The complex-yolo and yolo-v3 have an average precision of 54.4 % and 49 %, respectively. Fusion could give better results by raising the average precision from 54.4 % to 59.5 %. As we can see, the image and 2D range data fusion give results better than the 2 models (we can see it clearly in Figure 6) although it uses them to fuse information and give better results. This shows the power of fusion of the data in enhancing the performance of the perception task problem.

Also, the speed dropped to 28 FPS because the models are detecting sequentially, but it will be possible to add parallel threads, which will make both models work in parallel. So the FPS will rise to 50 again.

Results Visualization

The proposed approach has been used in the case of a KITTI dataset and some of the instances of classification are presented here. The KITTI dataset has 7481 frames, which are the RGB images taken from the camera and the corresponding Lidar point-cloud data. The frames have different locations and time stamps. We split those frames into training and testing frames. Figures 7 and 8 visually show the results of our models.

In Figure 7, *a* the green squares and the distance written are the predictions of yolo-v3 that work with the image data. The yellow and blue squares, as well as the distance, are the predictions of Yolo-Complex that work with the Lidar data. So when one of them fails at some point, the other can support detecting the object.

Figure 8, *a* shows the bird-eye view RGB map of the Lidar point cloud, it is the same frame as in Figure 7, *a*, the green color indicates height, the blue intensity of the reflected Lidar signal, the red is the density of points, the yellow and green squares are predictions from the complex-yolo model. In Figure 7, *a* the green squares and the distance written are the predictions of yolo-v3 that work with the image data, the yellow and blue squares and the distance are the predictions of yolo-complex that work with the Lidar data. So when one of them fails at some point, the other can support detecting the object.

Figure 8, *a* shows the birds-eye view RGB map of the Lidar point cloud, it is the same frame as in Figure 7, *a*, the green color indicates height, the blue intensity of the reflected Lidar signal, the red is the density of points, the yellow and green squares are predictions from the complex-yolo model.

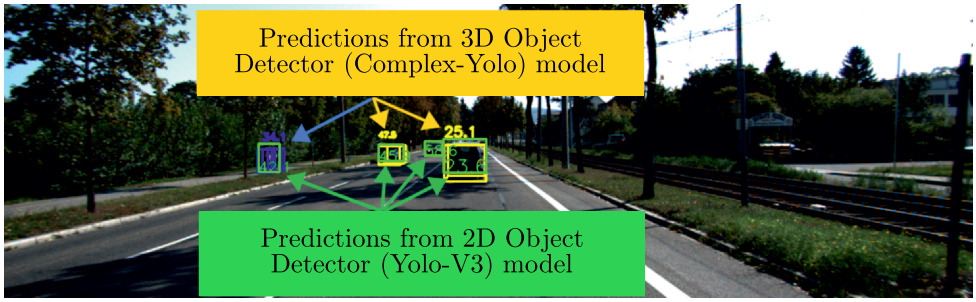
In Figure 8, *b* we can see how sometimes the yolo-complex model fails and the yolo-v3 model can support and give even comparable results.

The failure is sometimes due to the farther distance of the object such that it will be hardly detected by Lidar as the rays fired from the Lidar diverge. This makes far objects have comparatively lesser points than near objects.

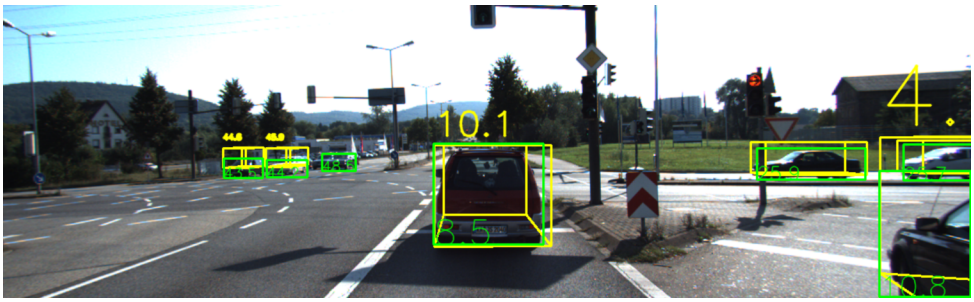
In Figure 7, *c* we can see that the distance difference between the two approaches has an error of 0 to 2 meters, depending on two factors:

1) the nearest-neighbor point of the 2D approach which gets the distance from the front part of the object;

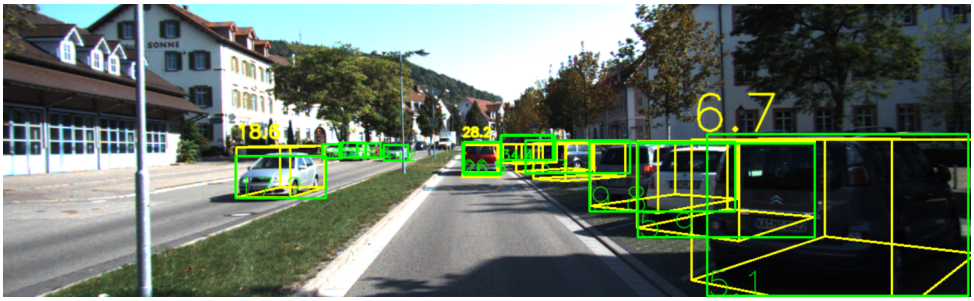
2) the 3D approach gets the distance from the center of the object, so in most cases the distance from the 2D approach will have a smaller distance.



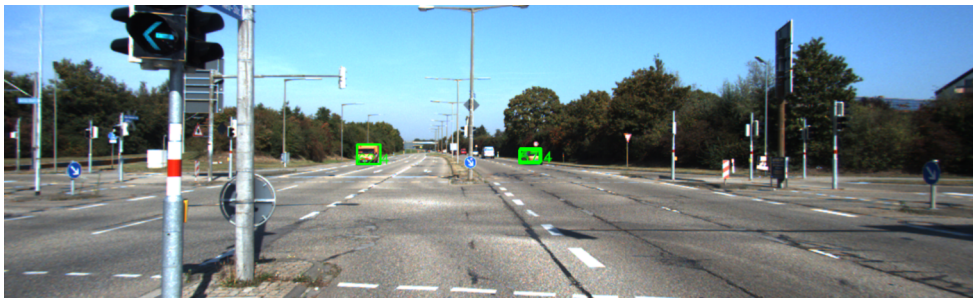
(a) Frame 1



(b) Frame 2



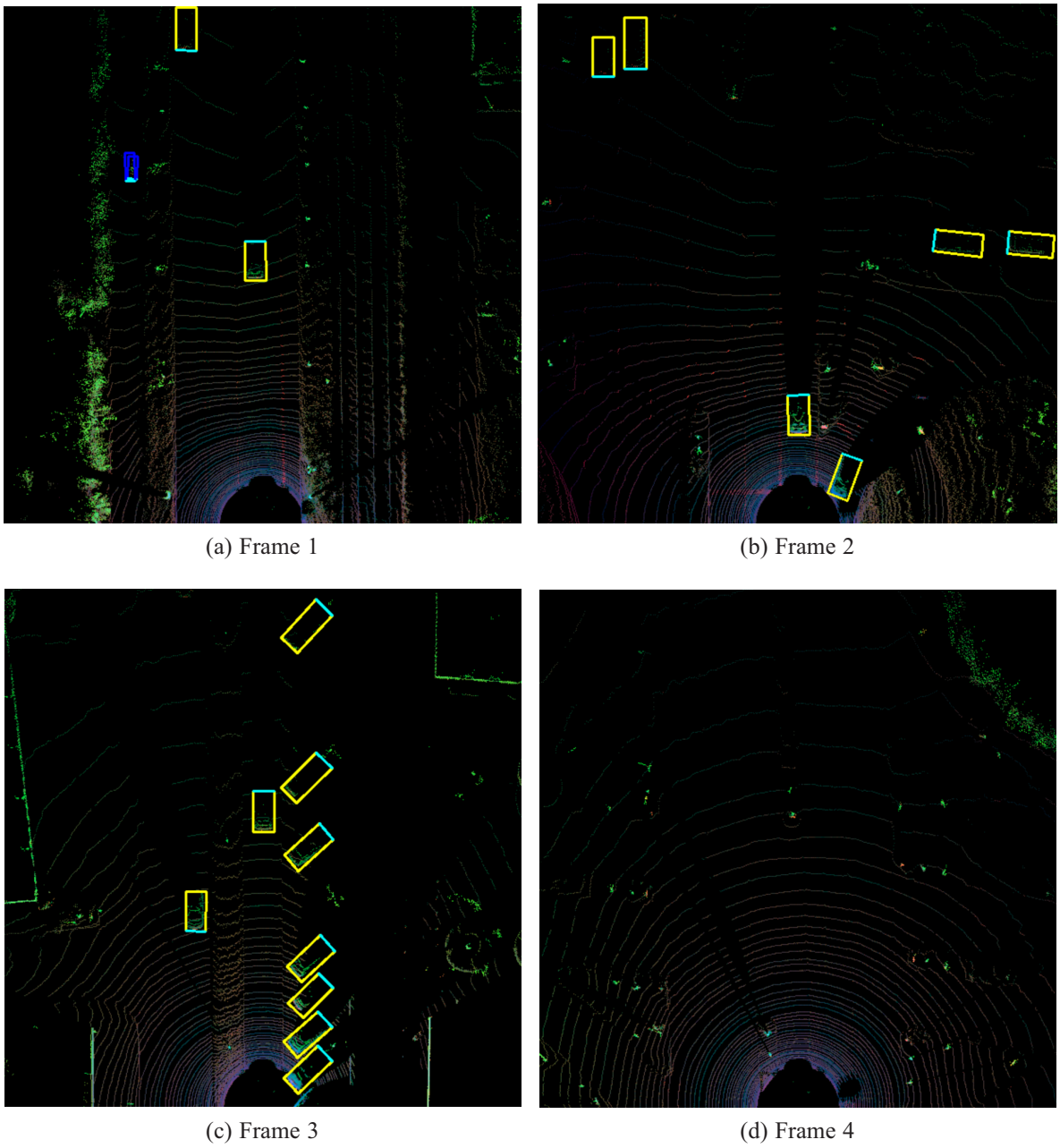
(c) Frame 3



(d) Frame 4

Figure 7. Results plotted on images taken by a camera

In Figure 7, *d* we can see that the 3D approach (Complex Yolo) totally fails, which may be due to different reasons, e. g., a far distance of the object or weather conditions, which may affect Lidar sensor reading. Only the 2D approach gives us results in this situation. It is important to use Image based fusion, since it supports and gives better results if they both detect the same object. Even if this does not happen, we may have a higher detection rate, so we do not miss an object. This helps in safe environment perception of self-driving cars.



(a) Frame 1 (b) Frame 2

(c) Frame 3 (d) Frame 4

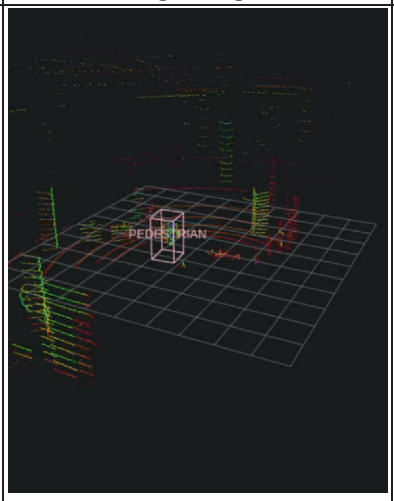
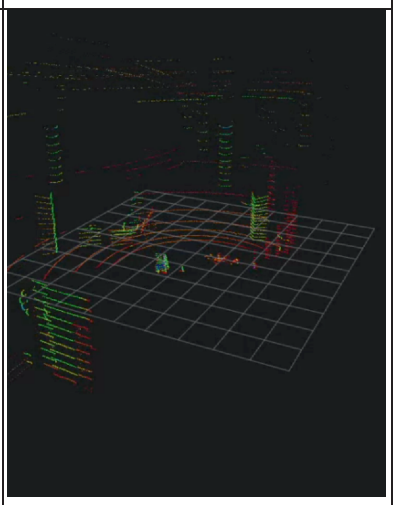
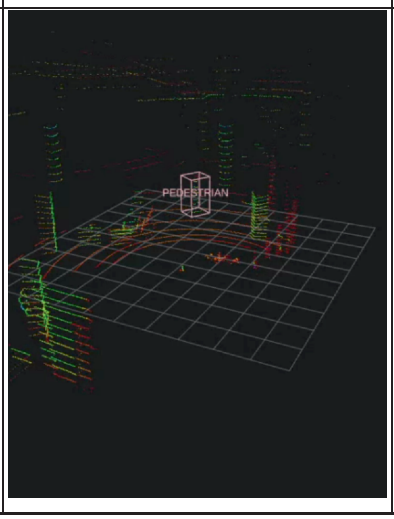
Figure 8. Results of Visualizations, bev (birds-eye view) RGB-map (a), (b), (c) and (d) are the corresponding maps for frames: 1, 2, 3 and 4 in Figure 7. The yellow boxes indicate output predictions from the 3D object detector model (Complex-yolo) the class of cars, while the blue boxes denote the class of pedestrians

The results based on an in-house real experimental setup (rather than public dataset) is discussed below. Table 3 shows a person testing the algorithm on a real hardware Lidar and camera, the fusion was performed and the person was detected correctly as a pedestrian.

Conclusion

The approaches that work with point cloud directly have a higher average precision; however, they lack real-time performance most of the time. On the other hand, approaches that transform point

Table 3. Fusion detections cases

Case	Images	Fusion output on pointcloud	Comment
True Positive			the 2D object detector and the 3D object detector models could detect the person as a pedestrian successfully, so the output of fusion was shown in Fusion output on point-cloud
False Positive			We can see that a false object was detected, but Lidar didn't detect it, and due to low confidence it was neglected by the fusion algorithm, so nothing was shown in Fusion output on point-cloud
True Positive and False Positive			We can see that this case is a combination of the previous two cases: one is true positive where the person was detected as a pedestrian, and other object is false positive, which was wrongly detected and due to low confidence it was neglected by the fusion algorithm.

cloud to bev RGB-map formats suffer from information loss, which results in lower average precision, but better real-time performance, and some models as in Complex-Yolo can give a good compromise

by giving fair results and still work in real time. Object detection from 2D images has been significantly improved over the last decade, and there exist models that can give fair results as well and work in real time for e. g., SSD (e. g. yolo-v3). Fusion between the previous two approaches is one of the best solutions to the problem in self-driving cars, and there has been significant interest in this area to enhance the autonomous cars and mobile robot systems. We passed data from Lidar to the 3D object detector model (Complex-Yolo) and evaluated it as a standalone model. Also, we did the same with a camera and passed the images to the 2D object detector model (YOLO-v3) and evaluated as a standalone model as well.

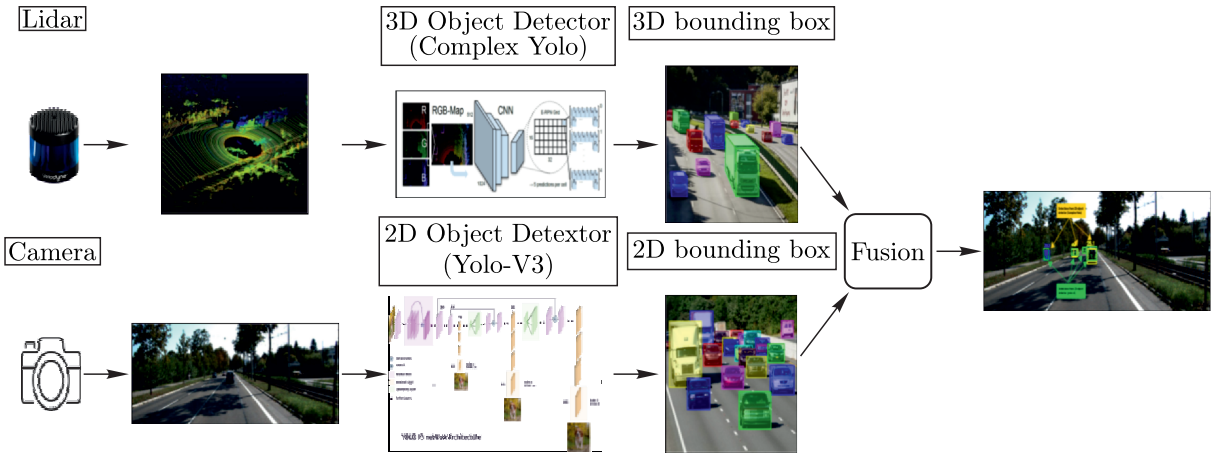


Figure 9. Image Based Fusion

In our approach (Image-Based Fusion) (see Figure 9), we could see how it combines information to obtain better results than the 2D object detection model and the 3D object detection model. This shows the power of data fusion to improve the performance of the perception task problem for self-driving cars.

However, the processing speed decreased to 28 FPS, which may be improved by parallel processing. The proposed methods have been implemented on the KITTI dataset as well as custom generated datasets. The results show that the proposed method enhances the results of individual methods that use point-cloud data or image data.

Appendix

For average precision, the calculation is performed as follows. After the final predictions are determined, the predicted bounding boxes could be measured against the ground-truth bounding boxes.

In order to calculate the mean average precision (mAP) for each class and see how the object detector is doing, we will first need to calculate the precision and recall for each class.

To do so, the number of true positives must be identified. If a predicted bounding box overlapped a ground truth bounding box by an IOU threshold (0.5), it is considered a successful detection and the predicted bounding box is a true positive. If a predicted bounding box overlapped a ground truth by less than the threshold, it is considered an unsuccessful detection, and the predicted bounding box is a false positive. Precision and recall can be calculated from true and false positives, as shown in Figure 10.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} = \\ = \frac{count(True\ Positives)}{count(all\ red\ boxes)} = \frac{2}{3},$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} = \\ = \frac{count(True\ Positives)}{count(all\ red\ boxes)} = \frac{2}{3}.$$

We need to get precision and recall at every IOU threshold and then average it for each class, and then average it again between all classes to get the map for the model.

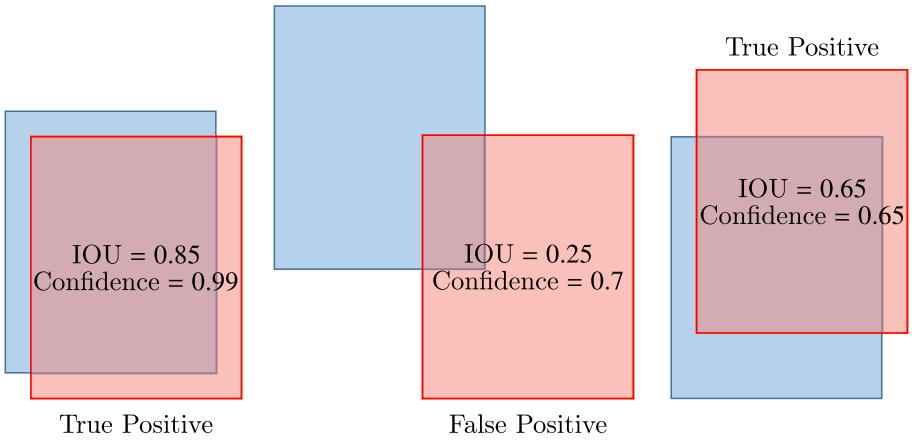
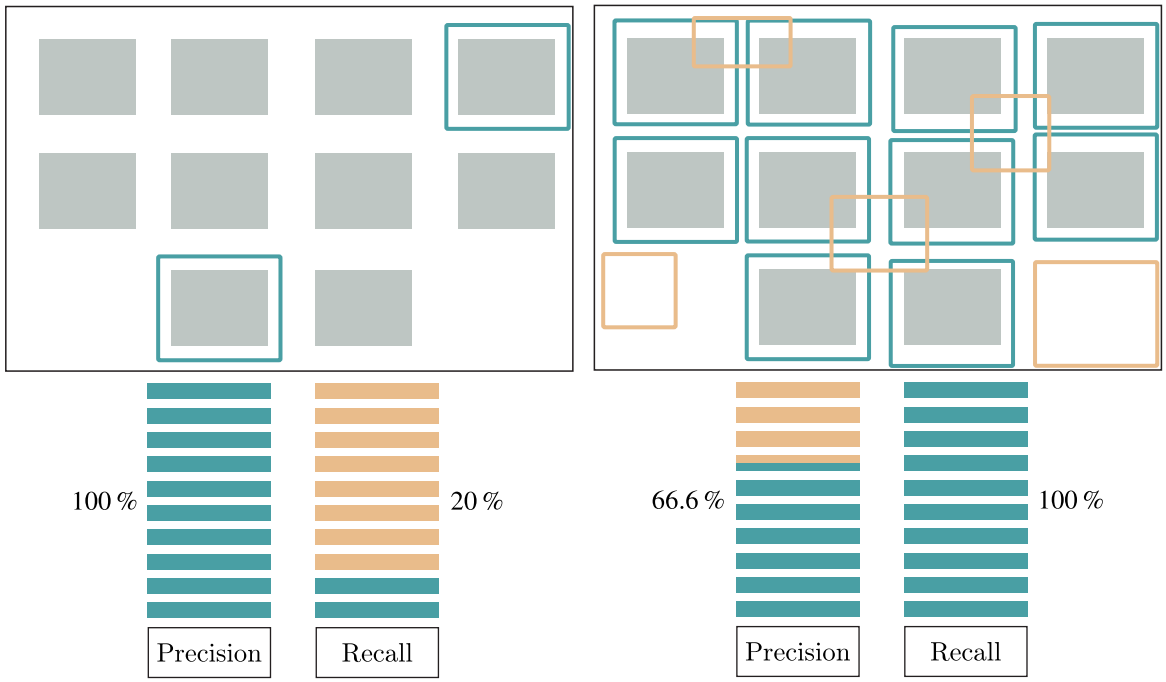


Figure 10. Precision-Recall



(a) High-precision low recall case (b) Low-precision high recall case

Figure 11. Precision & Recall cases for more clarification

When a model has high recall, but low precision, the model classifies most of the positive samples correctly, but it has many false positives (i.e., classifies many negative samples as positive). When a model has high precision, but low recall, then the model is accurate when it classifies a sample as positive, but it may classify only some of the positive samples as shown in Figure 11.

So, we need to find the threshold that gives us the best of both; the average precision is the area under the curve of the precision-recall curve.

References

Asvadi A., Garrote L., Premebida C., Peixoto P., Nunes U. Multimodal vehicle detection: fusing 3D-LIDAR and color camera data // Pattern Recognition Letters. — 2018. — Vol. 115. — P. 20–29.

Bogoslavskyi I., Stachniss C. Efficient online segmentation for sparse 3d laser scans //PFG — Journal Of Photogrammetry, Remote Sensing and Geoinformation Science. — 2017. — Vol. 85. — P. 41–52.

Chen X., Kundu K., Zhu Y., Ma H., Fidler S., Urtasun R. 3d object proposals using stereo imagery for accurate object class detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2017a. — Vol. 40. — P. 1259–1272.

Chen X., Ma H., Wan J., Li B., Xia T. Multi-View 3D Object Detection Network for Autonomous Driving // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2017. — Vol. 7.

Cho H., Seo Y., Kumar B., Rajkumar R. A multi-sensor fusion system for moving object detection and tracking in urban driving environments // 2014 IEEE International Conference on Robotics and Automation (ICRA). — 2014. — P. 1836–1843.

Dollár P., Appel R., Belongie S., Perona P. Fast feature pyramids for object detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2014. — Vol. 36. — P. 1532–1545.

Foster M., Schott J., Messinger D. Spin-image target detection algorithm applied to low density 3D point clouds // Journal of Applied Remote Sensing. — 2008. — Vol. 2. — 023539.

Geiger A., Lenz P., Stiller C., Urtasun R. Vision meets Robotics: The KITTI Dataset // International Journal of Robotics Research (IJRR). — 2013.

Girshick R., Donahue J., Darrell T., Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2014. — P. 580–587.

Girshick R., Donahue J., Darrell T., Malik J. Region-based convolutional networks for accurate object detection and segmentation // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2015. — Vol. 38. — P. 142–158.

Girshick R., Fast R. IEEE Int. Conf. Comput. Vis. // Santiago, Chile, December. — 2015b. — P. 7–13.

He K., Zhang X., Ren S., Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2015. — Vol. 37. — P. 1904–1916.

Ku J., Mozifian M., Lee J., Harakeh A., Waslander S. Joint 3D Proposal Generation and Object Detection from View Aggregation // IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — 2018. — P. 1–8.

Li K., Wang X., Xu Y., Wang J. Density enhancement-based long-range pedestrian detection using 3-D range data // IEEE Transactions On Intelligent Transportation Systems. — 2015. — Vol. 17. — P. 1368–1380.

Li B., Zhang T., Xia T. Vehicle detection from 3d Lidar using fully convolutional network // ArXiv Preprint ArXiv:1608.07916. — 2016.

Liang M., Yang B., Wang S., Urtasun R. Deep Continuous Fusion for Multi-Sensor 3D Object Detection // Proceedings of the European Conference on Computer Vision (ECCV). — 2018. — Vol. 9.

Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C., Berg A. SSD: Single Shot MultiBox Detector // Computer Vision – ECCV. — 2016. — P. 21–37.

Minemura K., Liau H., Monrroy A., Kato S. LMNet: Real-time multiclass object detection on CPU using 3D LiDAR // 2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS). — 2018. — P. 28–34.

Oh S., Kang H. Object Detection and Classification by Decision-Level Fusion for Intelligent Vehicle Systems // Sensors. — 2017. — Vol. 17.

Pont-Tuset J., Arbelaez P., Barron J., Marques F., Malik J. Multiscale combinatorial grouping for image segmentation and object proposal generation // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2016. — Vol. 39. — P. 128–140.

Qi C., Yi L., Su H., Guibas L. Pointnet++: Deep hierarchical feature learning on point sets in a metric space // ArXiv Preprint ArXiv:1706.02413 — 2017.

Qi C., Su H., Mo K., Guibas L. Pointnet: Deep learning on point sets for 3d classification and segmentation // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2017. — P. 652–660.

Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2016. — Vol. 6.

Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) — 2017. — Vol. 7.

Ren S., He K., Girshick R., Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks // Advances in Neural Information Processing Systems. — 2015. — Vol. 28. — P. 91–99.

Schoenberg J., Nathan A., Campbell M. Segmentation of dense range information in complex urban scenes // 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. — 2010. — P. 2033–2038.

Simon M., Amende K., Kraus A., Honer J., Samann T., Kaulbersch H., Milz S., Gross H. M. Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. — 2019.

Tao Y., Zhou J. Automatic apple recognition based on the fusion of color and 3D feature for robotic fruit picking // Computers and Electronics in Agriculture. — 2017. — Vol. 142. — P. 388–396.

Uijlings J., Van De Sande K., Gevers T., Smeulders A. Selective search for object recognition // International Journal of Computer Vision. — 2013. — Vol. 104. — P. 154–171.

Wang D., Posner I. Voting for voting in online point cloud object detection // Robotics: Science and Systems. — 2015. — Vol. 1. — P. 10–15.

Yin H., Yang X., He C. Spherical coordinates based methods of ground extraction and objects segmentation using 3-D LiDAR sensor // IEEE Intelligent Transportation Systems Magazine. — 2016. — Vol. 8. — P. 61–68.

Zhou Y., Tuzel O. Voxelnet: End-to-end learning for point cloud based 3d object detection // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2018. — P. 4490–4499.

Zitnick C., Dollár P. Edge boxes: Locating object proposals from edges // European Conference on Computer Vision. — 2014. — P. 391–405.