Radar and Camera Early Fusion for Vehicle Detection in Advanced Driver Assistance Systems

Teck-Yian Lim
University of Illinois at Urbana-Champaign
Urbana, Illinois
tlim11@illinois.edu

Amin Ansari
Qualcomm Technologies Inc.
San Diego, California
amina@qti.qualcomm.com

Bence Major
Qualcomm AI Research*
San Diego, California

Daniel Fontijne
Qualcomm AI Research
San Diego, California

Michael Hamilton
Qualcomm Technologies Inc.
San Diego, California

Radhika Gowaikar
Qualcomm Technologies Inc.
San Diego, California

Sundar Subramanian
Qualcomm Technologies Inc.
San Diego, California

Abstract

Perception module is at the heart of modern Advanced Driver Assistance Systems (ADAS). To improve the quality and robustness of this module, especially in the presence of environmental noises such as varying lighting and weather conditions, fusion of sensors (mainly camera and LiDAR) has been the center of attention in the recent studies. In this paper, we focus on a relatively unexplored area which addresses the early fusion of camera and radar sensors. We feed a minimally processed radar signal to our deep learning architecture along with its corresponding camera frame to enhance the accuracy and robustness of our perception module. Our evaluation, performed on real world data, suggests that the complementary nature of radar and camera signals can be leveraged to reduce the lateral error by 15% when applied to object detection.

1 Introduction

ADAS and autonomous driving have become one of the main forces behind research in the area of deep learning in the last few years. Object detection is a key challenge in the design of a robust perception system for these systems. The camera has established itself as the main sensor for building the perception module. In recent years, there is an increased emphasis on diversifying the set of sensors in order to improve the robustness to a range of operating conditions. A variety of sensors such as LiDAR, short-range radars, long-range radars, infrared cameras, and sonars have been used to enhance the quality of the perception module output.

In our work, we focus on the fusion of camera and radar sensors. Radar presents a low-cost alternative to LiDAR as a range determining sensor. A typical automotive radar is currently considerably cheaper than a LiDAR due to the nature of its fundamental design. Besides costs, radar is robust to different lighting and weather conditions (e.g., rain and fog) and is capable of providing instantaneous measurement of velocity, providing the opportunity for improved system reaction times.

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

With multiple sensors on a vehicle, sensor fusion is a natural next step for ADAS systems as it can improve the accuracy and especially robustness of object detection in a relatively noisy environment. Traditionally, approaches such as Extended Kalman Filter (EKF) [8] are used to combine the detections of different perception modules. More recently, deep learning has also been employed for sensor fusion across camera and LiDAR [3, 15, 5, 13, 17].

The fusion of data across different sensors can occur at a late stage, e.g.,the objects/vehicles are detected by the camera and LiDAR/Radar independently, and the detected object properties (like object bounding boxes) are combined at a later stage. Typically such a fusion technique is of lower complexity than an early fusion method where the sensor measurements from multiple modalities are jointly processed to generate object properties. Traditionally, early fusion allows low-level fusion of the features, which results in better detection accuracy. For instance, in the case of early fusion, side mirrors of a vehicle can be detected by one sensor while the front bumper might be detected by another sensor. The trade-offs between early fusion and late fusion have been studied recently by researchers [12, 27].

Radar data, in the context of autonomous driving and ADAS, has been used to improve the accuracy of sensor fusion and/or the perception module. However, radar data is typically processed using a CFAR algorithm to convert the raw data into a point-cloud which separates the targets of interest from the surrounding clutter. Converting raw 4D radar tensor (comprised of a dense 2D Euclidean space, Doppler, and time) into a sparse 2D point cloud removes a significant amount of information in the signal. In contrast, we rely on the raw radar data to minimize the artefacts that are introduced by post-processing of the signal as well as minimizing the abstraction of radar output.

In this paper, we make the following contributions:

1. A new approach to consuming radar data, performing both detection and classification
2. A new deep learning architecture which allows fusing of radar signals and camera images to produce object bounding boxes jointly.
3. An infrastructure and methodology to develop and evaluate the performance of our sensor fusion system on real world data.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. Section 3 describes our solution for the fusion of camera and radar sensors. Section 4 highlights the training setup and dataset used for our evaluation. Section 5 discusses our experimental studies and corresponding results. Finally, Section 6 presents our conclusions.

2 Related Work

Image Object Detection Image classification and object detection have been two challenging tasks in the computer vision community for years. In the last few years, there has been a breakthrough in the performance of image classification tasks. Deep learning-based neural networks have become the de facto solution to tackle these challenges. Object detection builds on top of image classification by addressing two more issues - the list of objects in a given image and the location of objects within this image. There have been mainly two categories of solutions being developed in this domain: single-stage and two-stage detectors. Single-stage detectors, such as Single-Shot Multibox Detector (SSD) [16] or YOLO [20], mainly focus on the inference time per frame. Since speed is of top concern, these networks internally rely on a single pass to identify the prior boxes. Two-stage detectors, on the other hand, are usually slower and achieve better accuracy results. Networks such as RCNN[7], Fast-RCNN[6], Faster-RCNN[22], R-FCN [4], and Mask-RCNN[9] are some of the examples. These networks rely on a Region Proposal Network (RPN) to identify the Regions of Interest (RoI) which subsequently get passed to the second stage which performs the final classification and box coordinate adjustments.

Radar Deep Learning for Autonomous Driving Due to the uniqueness of radars, there is an understandable lack of literature and datasets in this regard. Furthermore, traditional radar literature often refer to 'detection' as the task of returning a spatial *point*, in contrast to the computer vision community where detection refers to the task of returning a *region* an object ocupies. In the context of autonomous driving, published approaches [24] [19] made use of sparse point clouds from commercial automotive radar systems. In these commercial radar systems, these points are generated by processing the raw radar signals with the Constant False Alarm Rate (CFAR) [23] algorithm. The down side of this approach is that contextual information of radar returns are lost, and only the range, azimuth and doppler information

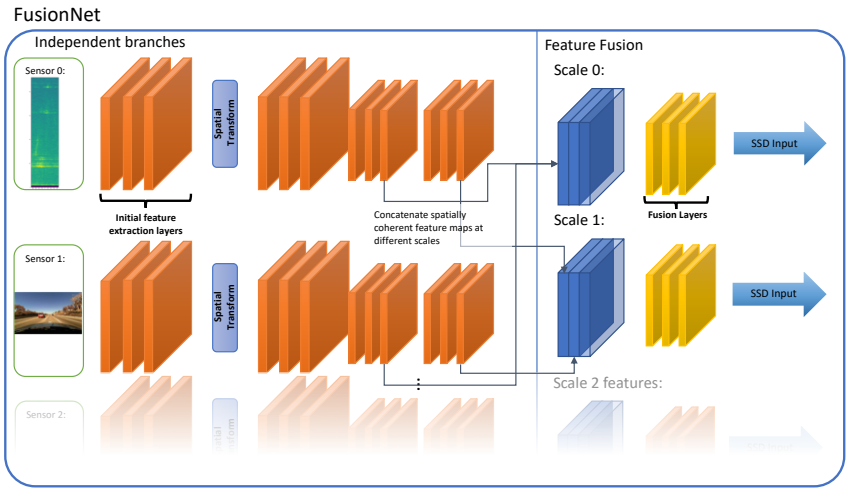


Figure 1: Highlevel overview of the FusionNet architecture. FusionNet consists of independent branches for each sensor. Each branch outputs spatially aligned feature maps. These feature maps are concatenated and fed to additional layers, before being used as input feature maps for the SSD[16] regression and classification heads.

are retained. This limits the ability of performing higher level classification of the radar signals. Outside of autonomous driving, deep neural nets have been applied to perform activity classification using micro-Doppler [25] [2], these approaches, however, do not tackle the problem of spatial localization.

Sensor Fusion using Deep Learning In recent work, a few authors have focused on the fusion of camera and LiDAR, which is different from our objective. Furthermore, the sparsity of point clouds, typically ≤ 64 points, returned by automotive radar systems limits the use of LiDAR approaches with radar point clouds. Multi-View 3D (MV3D) [3] applies feature extraction to 3 frames separately: LiDAR bird's eye view, LiDAR front view, and camera front view. Then, LiDAR bird's eye view feature map is being used to generate 3D bounding box proposals to guide the final fusion across the other two frames. In [15], authors rely on LiDAR and camera to improve the accuracy of object detection. LiDAR bird's eye view is used as the guide for fusing the camera features across multiple resolutions with the LiDAR features. The continuous fusion layers take into account the occlusion happening in the camera frame and enable fusion throughout the network. PointFusion [26] uses an earlier work (i.e., PointNet [18]) to process the LiDAR point cloud directly without the need to map it into 2D planes. Usage of 3D anchor boxes for the fusion adds to the complexity while making the architecture more generalizable.

3 FusionNet

We present FusionNet (Figure 1), our proposed architecture that fuses feature maps from multiple sensors for object detection. Our network design is inspired by SSD[16], whereby a feature extractor network generates feature maps, potentially at multiple scales, followed by detection heads. However, instead of using one feature extraction network for camera images, our network combines input from different sources observing the same physical scene. The goal of FusionNet is to extract and combine features extracted from different sensors observing the same space, from a potentially different perspective, and with their relative positions known. Each feature extraction branch incorporates a spatial transformation such that the output feature maps from each branch are spatially aligned with the other branches.

High-Level Architecture We implemented two branches in FusionNet, namely the Radar branch, that processes the range-azimuth image from the radar, and the Camera branch that processes the images captured by a forward-facing camera. After the independent feature extractor branches, these features are then passed through the fusion layer(s). In order to ensure that the network learns meaningful representations from different signal sources, we employed a unique training strategy of partially freezing the network and fine-tuning.

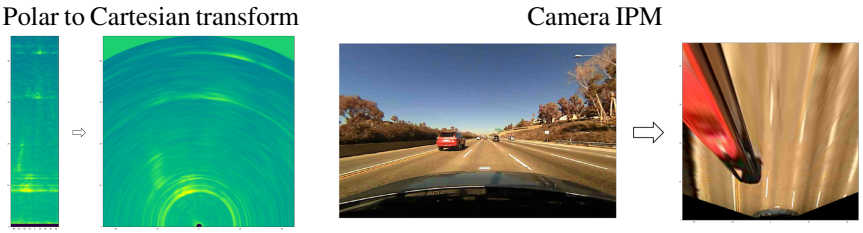


Figure 2: Visualization of the Radar and Camera's spatial transforms. The radar's spatial transform is the polar to cartesian transform. After the 3D-FFT processing of the raw radar signals, we obtain the polar range-azimuth polar 'image'. After this transform, the horizontal axis is now $-23.4m$ to $23.4m$, the vertical axis remains unchanged (0 to $46.8m$). The camera's transform, the Inverse Projection Mapping (IPM), is obtained from the known calibration between radar and the camera. We assume a planar road scene and compute a homography transform to project the camera image to the radar plane.

Radar Branch In contrast with other literature that utilizes automotive radar, the input to our network is not a point cloud. Instead, our radar branch takes a dense 2D range-azimuth "image", allowing us to employ feature pyramid network structures popular in image object detection networks. Since the goal is to predict bounding boxes in Cartesian coordinates, a mapping layer (as illustrated in Figure 2) is added to the intermediate feature maps. Empirically, we found that placing the spatial transformation early in the intermediate feature layers gave the best performance. After this transformation, more convolutional layers were added before concatenation with other branches.

Camera Branch In order to transform the camera image into the Cartesian space, we devised an Inverse Projection Mapping, which is a homography transformation of the camera image. To compute this projection mapping, we first assume that the camera is imaging a planar scene (i.e. the radar plane, which is approximately parallel to the road plane). Next, we utilized the intrinsic and extrinsic calibration information to project a set of points in the Cartesian radar plane to image coordinates. A planar homography transformation can then be found by using the standard 4-point algorithm. In the event that calibration is not available, it is also possible to manually assign multiple tie points, finally solving for the best homography using a least squares method.

The structure of the camera branch is very similar to the radar branch. However, instead of performing the coordinate transformation in the feature maps, we discovered empirically that the network performed the best when this transformation is applied directly to the camera image instead of the feature maps. After the homography transformation, the input image to the network is a 3-channel 256×256 color image. The image coordinates should now match the Cartesian radar image coordinates if the planar assumption is correct and the camera does not move with respect to the radar.

Fusion Layers The output from the independent feature extractor branches is only dependent on data from a single sensor. In order for the network to predict using input from multiple sensors, we utilized additional fusion layers to combine features across both branches. We designed these two branches such that the resolutions of their output feature maps match. Therefore, we can simply concatenate these output features from the radar branch and the camera branch to form a unified feature map with twice the number of channels. Next, we applied a dropout of $p = 0.5$ during training to guide the network to combine partial features from both branches. Finally, we applied a 1×1 convolution to reduce the channel count back to original single sensor channel count.

Detection Outputs For object detection, we applied SSD heads on the fusion feature maps. We chose anchor boxes to match the distribution of the ground truth boxes from our training set. We use k-means clustering (similar to [20, 21]) to construct a set of anchor boxes that are more suitable for our vehicle detection network. It should be clear that since we are mainly focusing on vehicles, there are only a handful classes of vehicles that are usually on the road (e.g., mid-size sedan, truck). In particular, there is a low variation on the width of these vehicles given the limits that are imposed by the US Department of Transportation on the lane widths.

On the Strength of the Planar Assumption It may seem that a planar road is a very strong assumption, but this is not the case. US interstate highways have a maximum grade of 6%[1]; considering

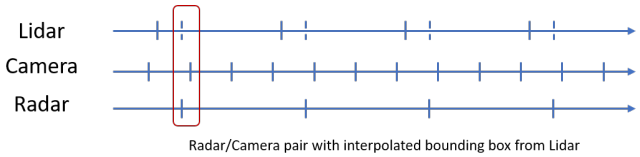


Figure 3: Temporal synchronization between different sensors. Radar frames are used as time-stamp reference, LiDAR boxes are interpolated and the nearest camera frame is selected.

an immediate transition from 0% to 6% grade results in an error of $0.08m$ at the maximum range of our radar, which is lower than the range resolution of our radar system (Sec. 4.1). Moreover, road grades change gradually and we should not see significant and sustained grade changes on most roads.

On the other hand, a greater source of error are unavoidable mechanical vibrations. We cannot expect the any sensor's mounting to be perfectly rigid on a moving platform. For the camera, this error manifests as a wobbling transformed image, most visible at the top of the transformed image. For the radar, this will translates to taking a tilted slice of the scene. No explicit handling or data cleaning was performed to exclude these distortions, and we expect the network to learn how to deal with such errors during fusion.

4 Experimental Setup

To the best of our knowledge, no openly available dataset suits our needs. Thus, we collected a novel dataset that consists of raw radar measurements, wide-FOV forward-facing camera and LiDAR measurements. In this paper, we have used more than 150K synchronized pairs of radar and camera frames, of these frames, 5000 were set aside as the test set. The dataset comprises of day time highway driving scenes from the greater San Diego area in southern California. More challenging scenes from suburban, city driving and adverse weather were not yet included in this work.

4.1 Dataset

While there might be datasets that have radar or LiDAR data, none of the publicly available datasets has the low-level radar signals that we utilized for our application. As a result, we built a data collection platform, performed cross-sensor calibration and finally utilized an automatic annotation set up to generate the training data.

Radar Data: Our radar system is a 77Ghz FMCW radar with a lateral array of 2 transmitting antennas and 16 uniformly spaced receive antennas at $1/2$ wavelength apart, providing us with a virtual array of 32 antennas, allowing resolution of $\approx 5^\circ$, $\pm 90^\circ$ on each side. Chirp, sampling and processing intervals were selected to provide range resolution of $0.094m$, max range of $46.8m$ and Doppler resolution of $0.25m/s$ for $\pm 2.5m/s$ at a frame rate of 125Hz. In constrast with publicly available datasets, our radar data is minimally processed. Only 3D-FFT was performed, resulting in a range-azimuth-Dopper data-cube for each radar frame. Further details on FMCW radar signal processing can be found in [11, 14]

Camera Data: The camera is a 1 megapixel, 150°FOV camera mounted on the windscreen inside the car, tilted slightly downwards, capturing 1280×800 RGB frames at 30fps.

LiDAR: The LiDAR is a Velodyne 32 line LiDAR mounted on the roof of the car, running (i.e. completes a full rotation) at 10Hz. While our network does not consume LiDAR as an input, we utilize detections from the LiDAR as our annotation. It is important to note that when the vehicle is moving at highway speeds, points from objects at the boundary of a 'frame' might have moved significantly. This discontinuity is taken care of in our LiDAR processing pipeline and is beyond the scope of this paper.

Temporal Synchronization Between Sensors: Note that the sensors listed above run at different sampling rates and no external synchronization mechanism is used during data collection. Instead, individual timestamps were recorded for each frame of data. While the radar is capable of generating frames at a very high rate, we extracted one frame of data every 100ms, so as to match the sampling rate of the LiDAR. Figure 3 illustrates our approach to generate temporally synchronized pairs for the dataset. The closest frame from the camera is selected, and an interpolated bounding box is obtained from the LiDAR detections.

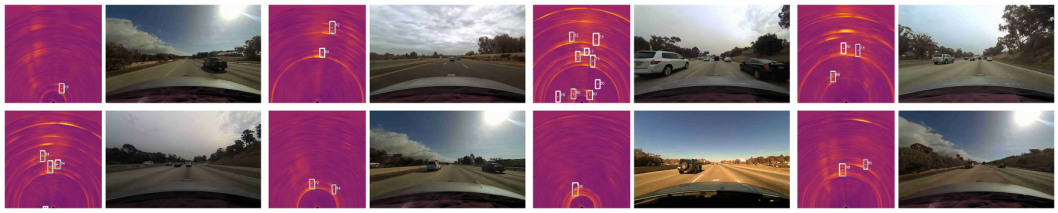


Figure 4: Examples of scenes where the network performs well. White boxes are network outputs, black boxes are automatic annotations from the LiDAR, the numbers beside the white boxes are the confidence score assigned to the detection. The network performs very similarly to the LiDAR, showing that the capabilities of the network to replicate the performance of a LiDAR using radar and camera inputs.

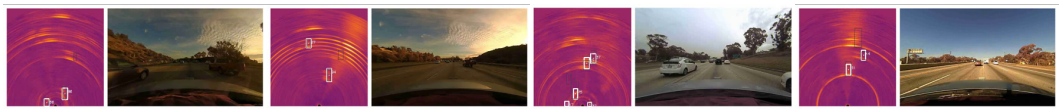


Figure 5: A sampling of missed detections. As the LiDAR has a higher vantage point than both the radar and camera, we see that missed detections tend to be vehicles that are occluded in the camera view, and also having obstructed radar returns.

Automatic Annotation and Dataset Limitations: We generated ground truth by applying object detection routines on LiDAR input. While this is a quick way of generating training data for radar and camera input, there are some obvious deficiencies with this approach.

In an effort to provide a cleaner dataset for training, we performed multiple passes of manual inspections of the annotations and implemented improvements to the automatic annotation algorithms that is not possible for online, fully automated methods. Parameters are tuned for different subsets, and non-causal processing and temporal tracking is also utilized.

However, being automatically annotated, the training and test set contains some inherent errors. Distant objects are often missed by the LiDAR detection algorithm. In addition, as the dataset consists only of California highway driving, the majority of objects in the scene are cars and smaller objects (e.g., motorcycles, bicycles, and pedestrians) are limited at best.

4.2 Training Setup

Our models were trained on an NVidia GTX 1080 Ti, with mini-batches of size 16. While we used a wide range of learning rates and optimizers in our experiments, our best model, with 2-stage training, was trained using Stochastic Gradient Descent (SGD) with momentum 0.9 and learning rate 0.0005 for 30 epochs on the camera only network, followed by another 30 epochs on the fusion network. This is comparable to a 30 epoch end to end training, as the camera branch is no longer updated during the second stage.

5 Evaluation

To evaluate the performance of our network, we compared the mAP score, the x (across) and y (along) position and size estimates of the bounding boxes and also the number of matched bounding boxes (for relative comparison). Figures 4, 5 and 6 shows a sampling of good detections, missed detections, and instances where the learnt detection network outperforms the annotations. We observe that the camera sensor is capable of better position and size estimates, while the radar branch provides a higher mAP. Our proposed FusionNet is able to succesfully combine the advantage of each individual sensor, to achieve performance that exceeds both of the individual sensors (Table 1).

5.1 Accuracy and mAP

We compare the performance of our network trained using SGD and Adam optimizers in Table 1. Similar to most published approaches to object detection, we used the IoU threshold of 0.5 for mAP

Figure 6: We also note that the annotations are not perfect, this panel shows some examples of such cases. This tends to occur in highly cluttered scenes and also on distant objects. In these instances, we see that the network is detecting vehicles clearly visible in the camera frame and also having visible radar returns, but annotation boxes are missing.

computation and we only considered boxes that meet this IoU threshold when counting true positives, false positives, etc. For the computation of distance and size metrics, we only considered true positives. It can be seen that our network outperforms single branched networks in all performance metrics. Specifically, in regards to Position metrics, we can see that the FusionNet makes a considerable improvement over the Radar only network by taking into account the visual features of the scene.

5.2 Freezing Weights

Due to the significant differences in signal characteristics, we discovered that different branches train differently, and the network has a tendency to learn to ignore one branch if the hyper-parameters used for training is more suitable for other branches. In order to overcome this difficulty, trained single branched network with the optimal training hyper-parameters, and load these weights into the corresponding branches to train the fusion network. The weights of the pre-trained feature extractors were frozen during the training of the fusion network. Table 2 shows a comparison of different variations of this weights freezing strategy. As can be seen, training the camera branch in advance and training the entire network with the gradient propagation disabled through the camera branch worked the best. This is due to the fact that the training of the camera branch, given the variety and complexity of the features, is naturally slower than the radar branch. As a result, the end-to-end training (without the freezing of the weights) is more inclined toward getting the radar branch to its optimal state as it can minimize the loss function faster.

Table 1: mAP scores, position and size L1 error, FusionNet combines the advantage of individual sensors

	Fusion (SGD)	Fusion (ADAM)	Camera Only	Radar Only
mAP	**73.5%**	71.7%	64.65%	73.45%
Position x	**0.145m**	0.152m	0.156m	0.170m
Position y	**0.331m**	0.344m	0.386m	0.390m
Size x	0.268m	0.261m	**0.254m**	0.280m
Size y	**0.597m**	0.593m	0.627m	0.639m
Matches	**8695**	8597	7805	8549

Table 2: Comparison of different training strategies using freezing of the pre-trained weights for the camera and/or radar branches.

	End-to-end	Freeze Camera	Freeze Radar	Freeze Both
mAP	72.0%	**73.5%**	73.0%	72.5%
Position x	0.163m	**0.145m**	0.172m	0.150m
Position y	0.349m	**0.331m**	0.369m	0.337m
Size x	**0.261m**	0.268m	0.275m	0.259m
Size y	0.636m	**0.597m**	0.616m	0.604m
Matches	8680	**8695**	8512	8509

Table 3: Sensitivity of fusion network to inputs from different sensors. Without retraining, the network was evaluated with the camera image or the radar range-azimuth image was set to zero (Camera 0 and Radar 0 respectively), and adding Gaussian noise of mean 0 and $\sigma = 0.1$ to the normalized input camera image and radar frame.

	Fusion	Camera 0	Radar 0	Camera + Noise	Radar + Noise
mAP	73.5%	55.0%	19.4%	61.2%	71.9%
Position x	0.1458m	0.1883m	0.1816m	0.1667m	0.1524m
Position y	0.3315m	0.4297m	0.3602m	0.3847m	0.3360m
Size x	0.2688m	0.3042m	0.3230m	0.4126m	0.2686m
Size y	0.5975m	0.7829m	0.5653m	0.7022m	0.5853m
Matches	8695	8259	3051	8004	8554

5.3 Contribution of Branches and Ablation Studies

To further determine the contribution of each of the branches and the value of early fusion, we conducted a sensitivity study by adding noise to individual branches and also by zeroing out the inputs on the branches. If the network does indeed utilizes evidence from individual branches, we should expect to see significant accuracy drop when one channel has no input or has significant noise added. The results, as seen in Table 3, aligns with our expectation as a significant drop in mAP is observed when either branch is corrupted.

An interesting observation is that the mAP score drops much more significantly when the radar input is removed as compared to removing the camera input. We hypothesize that this is due to the following reasons:

- Visually occluded vehicles may still give a radar return. Thus the network might be able to learn to detect some cases of vehicles without visual information.
- The highway scene is particularly easy for radar to work with, as there is limited clutter that could provide strong returns (e.g. fire hydrants, mailboxes, etc.).
- Distant objects are no longer visible in the camera after inverse projection, thus prediction at longer ranges are mostly reliant on radar returns (this, however, is constrained by our automatic annotation pipeline).

5.4 Inference Time

As a component of an ADAS system, latency is a key design evaluation criteria for the perception module. In our scenario, this latency is mainly impacted by the inference time per frame. For our best performing network in terms of accuracy, inference (including non-maximum suppression) runs at more than 40 fps (i.e., less than 25ms per frame) on the setup described in Section 4.2.

6 Conclusion

We introduced a novel network capable of ingesting and combining inputs from radar and camera sensors with vastly different signal characteristics. In addition, we introduced training strategies to overcome potential differences in convergence rates for feature extraction branches from different sources. We have also demonstrated that our proposed network is able to combine signals effectively from these complementary sources to produce better results than either of the sensors alone.

Even though our evaluation was done on an automatically annotated dataset, we were able to show quantitative improvements in most error metrics. Higher quality annotation and calibration will potentially improve these results even further. Lastly, the dataset can also be expanded by incorporating a greater variety of frames and scenarios such as urban/city driving, pedestrians, road-side clutter and adverse weather conditions.

References

[1] AASHTO. *A Policy on Geometric Design of Highways and Streets*. Washington, DC: American Association of State Highway and Transportation Officials, 2001.

[2] S. Abdulatif, Q. Wei, F. Aziz, B. Kleiner, and U. Schneider. Micro-doppler based human-robot classification using ensemble and deep learning approaches. *CoRR*, abs/1711.09177, 2017.

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. *CoRR*, abs/1611.07759, 2016.

[4] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[5] X. Du, M. H. A. Jr., S. Karaman, and D. Rus. A general pipeline for 3d detection of vehicles. *CoRR*, abs/1803.00387, 2018.

[6] R. Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[8] S. Haykin. *Kalman filtering and neural networks*, volume 47. John Wiley & Sons, 2004.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] D. H. Johnson and D. E. Dudgeon. *Array signal processing: concepts and techniques*. PTR Prentice Hall Englewood Cliffs, 1993.

[12] J. Kim, J. Koh, Y. Kim, J. Choi, Y. Hwang, and J. W. Choi. Robust deep multi-modal learning based on gated information fusion network. *arXiv preprint arXiv:1807.06233*, 2018.

[13] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. *CoRR*, abs/1712.02294, 2017.

[14] J. Li and P. Stoica. *MIMO radar signal processing*, volume 7. Wiley Online Library, 2009.

[15] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[17] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, abs/1711.08488, 2017.

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[20] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

[21] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[23] M. A. Richards, J. Scheer, W. A. Holm, and W. L. Melvin. *Principles of modern radar*. Citeseer, 2010.

[24] O. Schumann, M. Hahn, J. Dickmann, and C. Wöhler. Semantic segmentation on radar point clouds. *2018 21st International Conference on Information Fusion (FUSION)*, pages 2179–2186, 2018.

[25] m. s. seyfioğlu, s. z. gürbüz, a. m. özbayoğlu, and m. yüksel. deep learning of micro-doppler features for aided and unaided gait recognition. In *radar conference (radarconf), 2017 ieee*, pages 1125–1130. ieee, 2017.

[26] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. *CoRR*, abs/1711.10871, 2017.

[27] R. Zhang, S. A. Candra, K. Vetter, and A. Zakhor. Sensor fusion for semantic segmentation of urban scenes. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1850–1857. IEEE, 2015.

A Appendix

A.1 Details on FusionNet Branches

In this section, we provide a specific implementation of FusionNet, combining the range-azimuth measurements of a scene from the radar and the camera image of the scene. Finally, we cover potential pitfalls when training a network that uses signal sources with significantly different characteristics and the training strategy we utilize to ensure that the network learns useful representations from both signal sources. Our feature extraction network is a variant on ResNet [10] with a modified basic block, including both dropout and batch normalization (Figure 7).

In the radar branch, Figure 8a, we used the output at the end of Block 2 and Block 3 as input feature maps for the fusion layers. These feature maps are of sizes 64×64 and 32×32 respectively. We chose these feature map sizes based on the expected sizes of target objects and our maximum radar range. Performing a k-means clustering on the ground truth bounding boxes from our training set shows that the variation in the size of the bounding boxes is more limited compared to the typical ImageNet object detection. This small variation in sizes of target objects is because we are focused on detecting objects in real-world metric space, in contrast to the arbitrary scaled image space of photographs in ImageNet. This relative low variance in the shape and size of the bounding boxes enables us to reduce the complexity of our object detection backend drastically.

In the camera branch, Figure 8b, we applied spatial transform, i.e. the Inverse Projection Mapping introduced in Section 3, directly to the input camera image. We emperically determined that this results in a network with the best performance. Similarly, we output 2 feature maps of the same size and channel count at the radar branch.

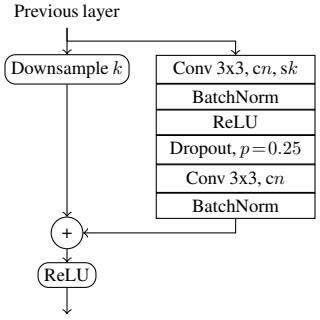
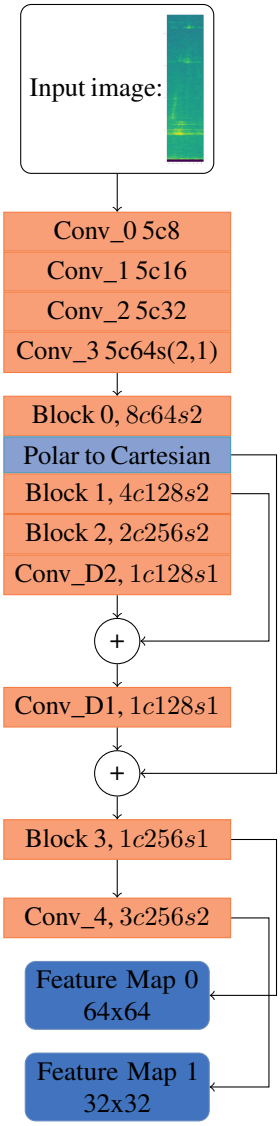
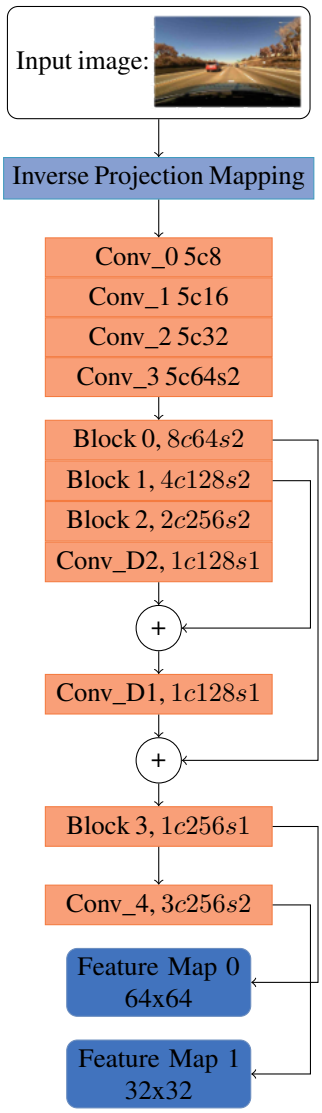


Figure 7: **Dropout Residual Block** A repeated unit used in our network. This block is repeated b times in each macro-block, denoted *Block #* in Figures 8a and 8b, with n number of channels in the convolutional layers. In this figure, k denotes the downsampling factor which is equal to two in the first instance and one in the subsequent repetitions within the macro-block.



(a) **Radar branch**: The input to the radar branch is a range-azimuth 'image' in polar coordinates. The horizontal axis is the azimuth, from -90° to 90°, the vertical axis is range from 0 to $46.8m$. The intensity of each pixel is the strength of radar returns from the corresponding spatial location. The network consists of convolutional layers, a linear mapping layer (for polar to cartesian transform), followed by additional convolutional layers. Each block consists of n number of Residual Dropout Blocks (Figure 7).



(b) **Camera Branch**: It was found emperically that the camera branch performed the best when spatial transform is performed before feeding into a deep neural network.

Figure 8: The network structure is color coded to match component blocks in Figure 1. Each of these branches outputs spatially aligned feature maps at 2 scales. Convolutional layers denoted *Conv_D#* do not have BatchNorm and any activation function, furthermore, these layers were upsampled using bilinear upsampling so that the size of the tensors matches that of the layer it is summing with. Other convolutional layers are specified in the shorthand $kcpsm$, for k kernel size, p output channels, m strides, and have BatchNorm and ReLU appended after them.