This ICCV Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Vehicle Detection With Automotive Radar Using Deep Learning on Range-Azimuth-Doppler Tensors

Bence Major* Daniel Fontijne* Amin Ansari† Ravi Teja Sukhavasi
Radhika Gowaikar† Michael Hamilton† Sean Lee† Slawek Grechnik† Sundar Subramanian†

Qualcomm AI Research* Qualcomm Technologies, Inc.†

{*bence, dfontijn, amina, radhikah, mjh, leesean, sgrzechn, sundars*}*@qti.qualcomm.com*
teja.sukhavasi@gmail.com

Abstract

Radar has been a key enabler of advanced driver assistance systems in automotive for over two decades. Being an inexpensive, all-weather and long-range sensor that simultaneously provides velocity measurements, radar is expected to be indispensable to the future of autonomous driving. Traditional radar signal processing techniques often cannot distinguish reflections from objects of interest from clutter and are generally limited to detecting peaks in the received signal. These peak detection methods effectively collapse the image-like radar signal into a sparse point cloud. In this paper, we demonstrate a deep-learning-based vehicle detection solution which operates on the image-like tensor instead of the point cloud resulted by peak detection. To the best of our knowledge, we are the first to implement such a system.

1. Introduction

As advanced driver assistance systems (ADAS) and autonomous driving systems mature, so does the range of solutions to automotive perception diversify. A variety of sensors such as LiDAR, short-range radars, long-range radars, RGB and infrared cameras, and sonars have been used for perception. The most prevalent sensor to provide detail-rich 3D information in automotive environments is the LiDAR.

Radar presents a low-cost alternative to LiDAR as a range sensor. A typical automotive radar is currently considerably cheaper than a LiDAR due to the nature of its fundamental design. In addition, radar is robust to different lighting and weather conditions (e.g., rain and fog) in comparison to LiDAR. However, owing to the specular nature of electromagnetic reflection at wavelengths employed by the radar, the resulting signal is not nearly as easy to interpret as LiDAR or camera (see *e.g.* Figure 1), so algorithm development is also more difficult.

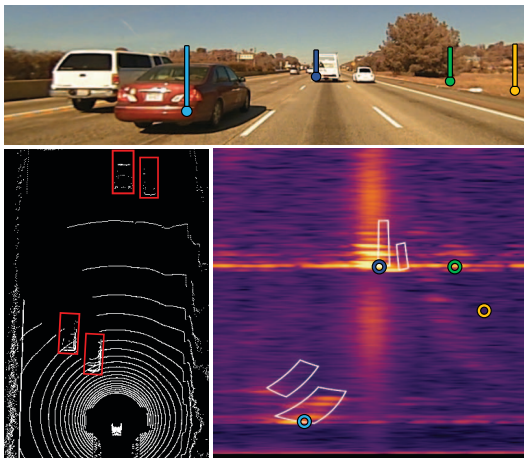

Figure 1: An example of the radar signal with the corresponding camera and LiDAR images. The radar signal resides in polar coordinate space: Vertical axis is range, horizontal axis is azimuth (angle). The Doppler (velocity) channel values for the marked points are plotted in Figure 2. Each of these points are also marked in the camera frame.

The radar data is a 3D tensor, with the first two dimensions making up range-azimuth (polar) space, and the third Doppler dimension which contains velocity information. This tensor is typically processed using a Constant False-Alarm Rate (CFAR) algorithm to get a sparse 2D point-cloud which separates the targets of interest from the surrounding clutter. This processing step removes a significant amount of information from the original signal. In this paper, we focus on performing object detection using the radar tensor for an autonomous driving platform.

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

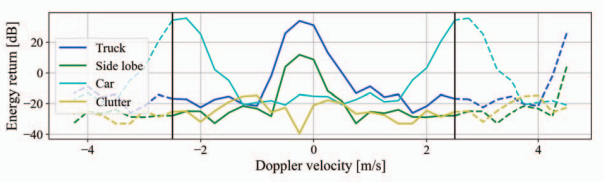

Figure 2: Example Doppler plots for the highlighted points in Figure 1. The center part between the black lines shows the original signal; the dashed lines repeat the center signal and demonstrate the aliasing nature of the Doppler dimension.

Our contributions are the following:

1. An object detection system tailored to operate on the radar tensor, providing bird's eye view detections with low latency.
2. Enhancing the detection quality by incorporating Doppler information.
3. Devising a method to solve the challenges posed by the inherent polar coordinate system of the signal.
4. Extending object detection to enable estimation of object velocity.

2. FMCW Radar Background and Signal Description

We use Frequency Modulated Continuous Wave (FMCW) radar to produce the input tensor to the deep learning model. FMCW radar uses a linear frequency modulated signal to obtain range. The received signal is mixed with the transmitted signal to obtain the *beat frequency* between the two. This beat frequency is a function of the round-trip time to the reflecting target, and therefore can be mapped directly to its range. Multiple receiver channels are used in a horizontal uniform line array (ULA) configuration. The azimuthal location of targets can be obtained by appropriate digital beamforming of these RX channels. Multiple pulses are transmitted in a train of equally spaced pulses in time. Radial motion occurring between pulses within a range resolution cell induces a phase shift over the pulses, which is used to compute the Doppler radial velocity in that cell.

The sampling rate of the ADC (analog to digital converter) determines the Nyquist limited maximum frequency that can be measured without aliasing and hence determines the maximum range. The elements of the receiver array are spaced at half wavelength, so the entire space from -90^o to $+90^o$ may be imaged without spatial aliasing. For Doppler, the Nyquist limit is dependent upon the pulse repetition rate.

For ease of calculation, beat frequency, channel, and pulse are mapped, using a three-dimensional Fast Fourier Transformation operation, into a range-azimuth-Doppler tensor. Last, we discard the phase information, and the resulting tensor is the input to the machine learning model.

Attribute	Value
Max. range	46.8m
Range resolution	0.094m
Max. Doppler	$\pm$2.5m/s
Doppler resolution	0.25m/s
Max. azimuth	$\pm\frac{\pi}{2}$
Azimuth resolution	$\approx 5^\circ$
Frame rate	125 Hz

Table 1: The description of the range-azimuth-Doppler tensor. Azimuth resolution decreases at higher off-angles. See Section 4.2 and Figure 4.

An example of the range-azimuth tensor (with the Doppler channel summed over) can be seen in Figure 1. Examples of the Doppler values at marked points are presented in Figure 2. The signal properties (*i.e.* the description of the 3D radar tensor) can be seen in Table 1. For the radar configuration parameters and their relationship to the resulting signal properties please refer to the appendix. Further details about FMCW can be found in [8, 11].

3. Prior Work

In the automotive radar space, Dickmann *et al.* [2] extract peaks from the radar tensor using traditional processing. The peaks are clustered, tracked over time and classified by a single layer neural network. Lombacker *et al.* [18] aggregate radar peaks over a 2D grid, *i.e.* a top-view of the local world. Both Random Forest and a Convolutional Neural Network (CNN) are used to detect static objects, the CNN coming out favorably. These types of works come closest to ours, but they either rely on building a map of peaks over time to aggregate details or operate on just the peaks for dynamic scenes.

Lien *et al.* [13] describe a small radar that could be embedded in consumer electronics (*e.g.* smart watch) to recognize hand gestures. They run Random Forest on hand-crafted features, citing power-concerns for picking such a solution over *e.g.* deep learning.

Zhao *et al.* [24] use a 6GHz FMCW radar to detect human pose, even through walls. The radar has both a vertical and a horizontal antenna array, resulting in range-azimuth and range-elevation radar tensor. Each tensor is processed by a separate encoder before fusion, then further processed by the pose decoder module.

Kim and Moon [9] use a 7.25 GHZ Doppler radar to record Doppler-time spectrograms of humans. They apply a CNN to perform both human detection and human activity detection. Without range, azimuth or elevation dimensions, their works does not spatially detect the human, but only detect the presence or absence of a human in the signal. Similarly, Angelov *et al.* [1] feed Doppler-time

spectrograms to 77GHz FMCW radar to various neural network architectures such as VGG and ResNet to classify between car, person and bicycle. Again, locating the object in the world is not considered.

Furukawa [4] applies deep learning to synthetic aperture radar (SAR) data. SAR data is collected by sweeping a radar, mounted on an airplane or satellite, across a scene and combining the data as if it was recorded by a radar with a very large antenna. This results in relatively high-resolution images of static targets that is not comparable to the resolution of automotive radar. Furukawa shows that a fully convolutional neural network approach works well for the automatic target recognition problem.

4. Architecture Overview

In this section, we will present how the radar perception system is composed. Each section corresponds to a consecutive block in the model: feature extraction, spatial transformation, temporal module, and detection heads. The all-encompassing solution can be seen in Figure 3.

4.1. Backbone network and radar input formats

As described in Section 2, the radar tensor is three dimensional: it has two spatial dimensions, range and azimuth, accompanied by a third, Doppler dimension, which represents the velocity of objects relative to the radar, up to a certain aliasing velocity. We propose two solutions to process the full 3D tensor.

The first approach is to remove the Doppler dimension by summing the signal power over that dimension. The input of the model is a range-azimuth tensor, hence we call this solution the Range-Azimuth (RA) model.

The second approach is to also provide range-Doppler and azimuth-Doppler tensors as input. The range-Doppler input has the azimuth dimension collapsed. Similarly, the azimuth-Doppler input has range dimension collapsed. Thus, the model has three inputs that are fused after initial processing. We call this the Range-Azimuth-Doppler (RAD) model.

Due to the properties of the radar signal, translation equivariance cannot be expected. As an example, if an object is moving directly away from the radar, it will shrink on the image due to occupying less azimuth bins. Or if the same object would move along a circle around the radar, the signal characteristics will be different because of how azimuth bins are spaced (see Section 4.2 for more details). To counteract these effects, we use CoordConv by Liu *et al*. [16] in the first layer. In practice, this means stacking two additional channels to the input which contain the pixel coordinates to enable the convolutions to be conditioned on location.

4.1.1 Range-Azimuth model

The feature extractor used for our Range-Azimuth (RA) model is motivated by the Feature Pyramid Network (FPN) architecture by Lin *et al*. [14]. It consists of multiple consecutive convolutional layers, with multiple down-sampling (*i.e*. strided convolutional) layers. The next stage is up-sampling multiple times using transposed convolutions. Skip connections are used between feature maps of matching shapes from the up-sampling and the down-sampling path. Before adding the feature maps together, an additional convolutional layer is executed for each skip-connection. The layer configuration is constructed such that a feature in the final layer has a receptive field spanning the complete input.

4.1.2 Range-Azimuth-Doppler model

The Range-Azimuth-Doppler (RAD) model operates on the three projections of the 3D radar tensor to reduce computational complexity. The projections are made by summing the power over the omitted dimension. Thus, the network has three 2D inputs: range-azimuth, azimuth-Doppler and range-Doppler.

The range-azimuth branch is exactly the same as the down-sampling part of the architecture described in the previous subsection. Additionally, there are two branches taking range-Doppler and the azimuth-Doppler tensors as input, respectively. These branches only down-sample.

The resulting feature maps are then fused as follows, also shown in Figure 3. First, each feature map is repeated along the missing dimension such that the tensors have compatible shapes. This yields three 4D feature tensors, one channel being the feature channel and the rest correspond to range-azimuth-Doppler. We then concatenate these in the channel dimension and apply 3D convolutional layers. After these convolutions, we perform max-pooling over the Doppler dimension and continue with the up-sampling layers of the range-azimuth model, as described in the RA model section.

4.2. Polar to Cartesian transformation

As discussed in Section 2, after the FFT the radar tensor is in polar space (range-azimuth). Figure 4 shows the physical center direction of the azimuth bins on a Cartesian grid. In the top right, a typical large vehicle with dimensions of two by five meters is shown for reference. From the figure it is apparent that as range increases, the distance between adjacent bins becomes larger: the angle between center of the forward bin and the center of the next bin is 3.7°, which corresponds to a distance of about three meters laterally at a distance of 47 meters, while the angle between bins increases to 11° (or 9 meters) for the most extreme bins.

Even though it is possible to get a higher resolution feature map than the original radar signal and use that as input to detection, it doesn't solve the problem that adjacent pixel locations can have vastly different distances between them depending on their range. Single-shot object detection methods, as described in Section 4.4, place a grid of prior

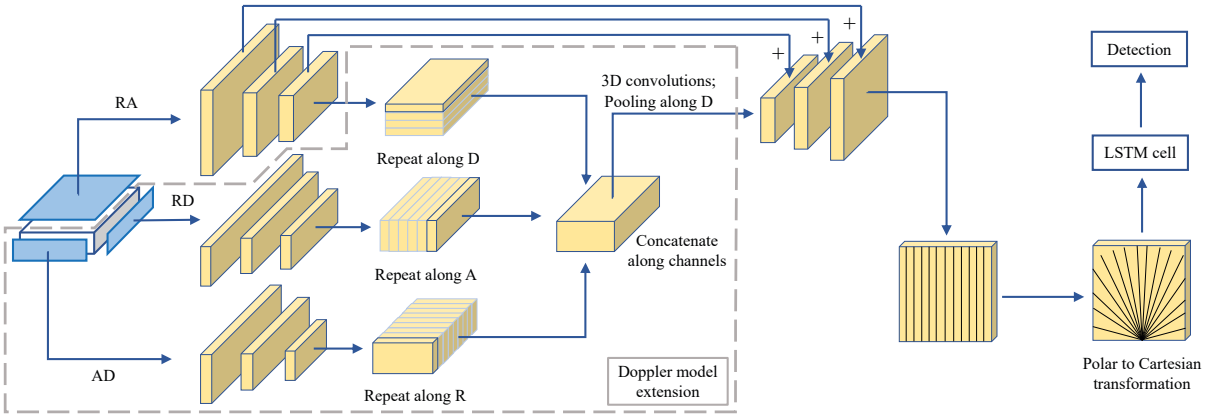

Figure 3: Conceptual diagram of our model architecture. Feature channels are not visualized in the picture. Notation: (R)ange; (A)zimuth; (D)oppler. The different 2D tensors are calculated from the original RAD tensor by summing over each of the dimensions. For the exact architecture configuration please refer to the appendix.

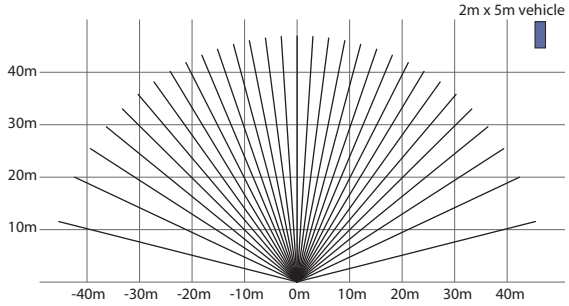

Figure 4: Azimuth bins in our radar configuration.

boxes over the input tensor. We hypothesized that using prior boxes which are distributed on such a uniform grid in *polar space* is disadvantageous. To test our hypothesis and to solve this problem, we created a baseline and three alternative approaches:

- **Polar input, polar output**. The baseline solution takes the range-azimuth radar tensor as input. The prior boxes are distributed on a uniform grid in polar space.
- **Cartesian input, Cartesian output**. The input tensor is transformed from polar space to Cartesian space using bi-linear interpolation. Approximately 23 meters of each side of the Cartesian input tensor is clipped, resulting in a square feature map. This clipping is motivated by the fact that for highway driving we do not need to detect cars laterally up to 47 meters. The distribution of the prior boxes is uniform in Cartesian space. The downside of this approach is the radar signal corresponding to close range is compressed, losing fine details where they are needed the most, while far range is expanded, wasting compute power.
- **Polar input, Cartesian output with learned transformation**. The input tensor of the neural network is in polar space, but the output boxes are on a uniform grid in Cartesian space. Thus, the neural network has to explicitly learn the polar to Cartesian transformation. The downside of this method might be that the neural network wastes capacity to learn the transformation.
- **Polar-to-Cartesian transformation on latent features**. Same as the polar input, Cartesian output solution, but after feature extraction, an explicit transformation layer transforms the latent features from polar to Cartesian space (using bi-linear interpolation). As we will show in Section 5, this method works best.

Note that for all of the described solutions, the widths and heights of the prior boxes correspond to the real width and length of the vehicles, not their span in polar space.

4.3. Temporal Processing

Due to the nature of automotive environments, exploiting the temporal aspect of the signal can provide benefits to detection quality as well as enable access to velocity information. To this end, and in order to capture the dynamics of the scene, we need to build memory into our model. As a result, we convert our network into a Recurrent Neural Network (RNN) by taking advantage of Long Short-Term Memory (LSTM) modules introduced by Hochreiter and Schmidhuber [7]. To be able to operate in a fully convolutional manner we employ a Convolutional LSTM cell proposed by Shi *et al*. [21]. In essence, compared to a more traditional LSTM cell, some of the operations are replaced with convolutions, and the cell operates on a 3D tensor.

4.4. Object detection and property estimation

As our focus is vehicle perception on highways, it is key for our solution to run at a high frame rate on embedded systems with limited compute power. We therefore employ a one-shot object detection model, namely Single Shot Detector (SSD) by Liu *et al*. [17].

In essence, the SSD algorithm operates on one or more feature maps extracted from a backbone network. The feature maps are run through additional convolutional layers that predict confidence values for each feature location to determine whether the corresponding location in the input tensor contains an object of a certain class with a size close to a pre-defined height and width. Multiple pre-defined sizes can be used for each feature location.

SSD uses regression to adapt the size and position of the pre-defined box to better match the bounding box of the actual object. During inference, non-maximum suppression is used to remove overlapping detections which are likely for the same object.

Our solution has a couple of key differences compared to the standard SSD architecture. First of all, we are only using the output of a single convolutional layer as input to SSD. Since we apply SSD on what is essentially a bird's-eye view of the road, vehicles cannot overlap as long as the feature-map has enough resolution.

Another difference is that the backbone is inspired by the RetinaNet architecture by Lin *et al*. [15] and we use Focal Loss from the same work, as it is shown to provide superior results compared to hard negative mining. If p is the confidence value for a given class, and $p_t = p$ when that class matches the ground truth label, and $p_t = 1 - p$ otherwise, then Focal Loss is defined as:

$$L_{conf} = -\alpha_t (1 - p_t)^{\gamma} log(p_t). \quad (1)$$

Here, α_t is a class-dependent weighting factor, and γ is the focusing factor.Using grid-search we decided to use $\alpha_{vehicle} = 1.0$, $\alpha_{background} = 0.9$ and $\gamma = 1.0$. We use the box regression loss L_{loc} from the original SSD paper.

Each of the detections also comes with an estimation of the vehicle's velocity (relative to the ego frame of reference). An extra head of the network regresses the velocity vector for each proposed box. The direction and the magnitude of the velocity are regressed separately. Direction is further separated into the sine and cosine of the angle. During the forward pass, the two direction components are normalized so they describe a unit vector.

These terms are L_1 losses for regressing the three components. To define the velocity loss formally, let the subscripts p and t stand for prediction and target respectively, and i indexes the i^{th} object match in the mini-batch. The variables sin_angle and cos_angle refer to the network's estimation of the sine and cosine of the angle of the velocity vector.

$$L_{vel} = \frac{1}{N} \sum_{i=1}^{N} \begin{aligned} &\|magnitude_{i,p} - magnitude_{i,t}\| + \\ &\|sin\_angle_{i,p} - sin(angle_{i,t})\| + \\ &\|cos\_angle_{i,p} - cos(angle_{i,t})\|. \end{aligned} \quad (2)$$

The training of the system works as described in Lin *et al*. [15] with the additional loss term for velocity estimation. The combined loss is:

$$L = 0.5 \times L_{conf} + 0.5 \times L_{loc} + L_{vel}. \quad (3)$$

For our experiments we observed that using loss weight of 1 for the velocity loss term works best.

5. Experiments and model analysis

5.1. Dataset

Given the lack of a public automotive radar dataset, we have collected and annotated our own. Our data collection vehicle has a radar, mounted on the front bumper, a wide-angle camera, mounted inside the vehicle, and a 32 or 64 beam LiDAR, mounted on the roof of the vehicle. In this work, the camera is used for visualization and human verification only, and the LiDAR is used for ground truth annotation only. All sensors are 3D calibrated with respect to each other.

Our annotations are based on LiDAR using a combination of automatic annotation and manual correction. The statistics of our dataset can be seen in Table 2.

The number of tracks refers to the number of unique vehicles in the dataset. Two recordings that are used in the test-set are also part of the train-set, but different parts of the recordings are used for each set. All other train- and test-set recordings were made on different days.

Dataset	Train	Test
Number of frames	106K	5200
Duration	2h56m	8m40s
Number of recordings	7	6
Number of annotations	389157	20011
Max. annotations in single frame	23	12
Number of tracks	3011	142

Table 2: Training and test set statistics.

5.2. Training details

For all experiments, stochastic gradient descent was used with a momentum of 0.9 and a weight decay of 0.0001. The weights were initialized using the method by Glorot and Bengio [6]. For non-temporal models, training consisted of 15000 iterations, with 32 images per mini-batch and an initial learning rate of 0.05, which was divided by 3 at 6 evenly spaced points throughout the process.

Temporal models were initialized using the pre-trained weights of the non-temporal equivalent. BackPropagation Through Time was used to train the LSTM, as described by Werbos [22]. Training consisted of 3000 iterations with 8 images per mini-batch and a sequence length of 10. The initial learning rate was 0.005, which was divided by 3 at 3 evenly spaced points throughout training.

The range-azimuth and range-Doppler inputs are normalized by making each range-row zero-mean and unit-variance, using statistics computed over the training-set.

All of the discussed models used the same set of prior box shapes, 8 in total. Widths: $1.9m$, $3.5m$. Lengths: $4.21m$, $6.1m$, $11m$, $18m$. As all of the input images spanned the same space, we defined the ground truth and prior boxes in meters and then mapped them to $[0, 1] \times [0, 1]$ for the loss function. The input to the SSD head was a single feature map with a size of 64×64, corresponding to a $47m \times 47m$ area, so prior boxes were spaced approximately $73cm$ from each other.

Also note that some annotated cars were not visible to the radar, and in other cases, vehicles were not visible from the LiDAR signal primarily used for annotation. These resulted in unsolvable false positives and false negatives, rendering 100% performance (*i.e.* mAP, precision, recall) unreachable.

During training, the only form of data augmentation used was horizontal mirroring with a probability of 0.5. Each experimental result is the average of at least 8 training runs. Standard deviations are also indicated. For all reported mAP scores, we used an IoU threshold of 0.5, unless noted otherwise.

5.3. Experiments

Cartesian space detections: In previous sections we already established that the original radar signal is in polar coordinate space, even though the preferred object representation is in Cartesian space. We benchmarked all four proposals listed in Section 4.2. The architecture configurations (number of layers, number of feature maps per layer, connectivity pattern of the FPN) are kept identical to the extent possible.

Table 3 lists the resulting mAP scores from each of the models. Note that the Cartesian to Cartesian model has 65% more MACs (multiply-accumulate operations) due to increased size of the input (256×256 compared to 256×64). Based on these results, we conclude that as hypothesized, the baseline polar input polar output solution has far inferior detection performance. It is also safe to conclude that applying the explicit polar to Cartesian transformation to the latent representation (instead of learning this transformation) is beneficial. Last, we attribute the lower detection performance of the Cartesian to Cartesian model to the fact that the signal in Cartesian space has properties which are more difficult to learn for a CNN, for example the azimuth side-lobes are arcs instead of lines. However, this explanation has not been verified experimentally or otherwise.

Transformation	mAP [%]
Polar to Polar	72.62 ± 1.02
Cartesian to Cartesian	83.15 ± 0.92
Polar to Cartesian (learned)	83.98 ± 0.60
Polar to Cartesian	86.15 ± 0.61

Table 3: Experimental Cartesian space detection results.

Contribution of the Doppler dimension to detection: With the Doppler dimension, the signal has characteristics which may help the detection of objects. For example, objects close to each other in physical space might be separated in the Doppler dimension. Due to this, we assessed if the Doppler dimension is indeed useful from both detection and velocity estimation points of view.

To see whether our solution with Doppler helps detection, we compare our models based on mAP scores. As can be seen from Table 4, the RAD model provides a gain over RA in the simplest case, without LSTM or joint velocity estimation training. Because the difference in mAP is relatively small compared to the standard deviation, we used bootstrap hypothesis testing suggested by Efron and Tibshirani [3] to estimate the confidence. The hypothesis is that the RAD model achieves a significantly better mAP score. Based on 10000 redraws the obtained p-value was 0.0031, which expresses high confidence that the RAD model does help with detection.

However, for the case where we also utilize LSTM and velocity estimation (see Table 5), the RAD model performs worse and the p-value shifts to 0.9657, meaning that there is a high confidence that the hypothesis is false for the case of temporal models. This inconsistency might be because of suboptimal hyper-parameters, or because of redundancy between Doppler and LSTM.

We conclude that incorporating Doppler information using our proposed RAD model helps the detection performance.

Contribution of the Doppler dimension to velocity estimation: For comparing velocity estimation performance we calculate the L_1 error between the predicted and target velocity vectors. The velocity estimation performance results can be seen in Table 5, which shows the velocity estimation performance for both RA and RAD models using LSTM. Based on these results, we cannot conclude that the RAD model has better velocity estimation capabilities. This might be due to the limited range of velocities the Doppler channel captures (see Table 1).

Effect of using a temporal model on detection performance: For determining the gain in detection

Input	LSTM	mAP [%]
RA	No	86.15 ± 0.61
RAD	No	86.75 ± 0.32
RA	Yes	87.52 ± 0.67

Table 4: Detection model results.

Input	LSTM	mAP [%]	Velocity L_1 error $[m/s]$
RA	No	86.34 ± 0.30	2.47 ± 0.06
RA	Yes	88.00 ± 0.37	1.49 ± 0.11
RAD	Yes	87.59 ± 0.56	1.50 ± 0.11

Table 5: Detection and velocity estimation model results.

performance by utilizing the temporal aspect of our signal, we compare two Range-Azimuth input models, one having an LSTM module and the other not. They have both been trained without velocity estimation. Based on the results shown in Table 4, we conclude that exploiting the temporal dependency of the scene results in superior detection.

Effect of using a temporal model on velocity estimation: Even though the main source of velocity information is the temporal aspect of the signal, it is important to mention that the location of the objects also holds prior information about their velocity. In the areas we used for data collection, vehicles in the right lanes tend to go slower than the ego vehicle, and vehicles in the left lanes tend to be faster. Furthermore, their velocity vectors are usually on the same line as the road they are following.

Even though we use random horizontal flipping for data augmentation, the neural network is able to learn these as prior information. One explanation could be that the left and ride side boundaries of the road may have different characteristics, so horizontal flipping preserves *e.g.* the fact that faster lanes are closer to the lane divider bushes. As a baseline, we used a non-temporal RA-input model to estimate the performance that can be achieved by looking at only a single snapshot of the scene.

Based on the velocity estimation results shown in Table 5, we conclude that using a temporal model results in significantly better velocity estimation.

5.4. Model analysis

Based on the conclusions drawn from the experiments, the best configuration is the range-azimuth model, with the explicit polar to Cartesian transformation, the LSTM block, and velocity estimation. We perform analysis on one instance of this specific choice. This instance has been chosen by ordering all of the trained models by mAP and selecting the one with the median score.

The precision-recall curve can be seen in Figure 6.

Attribute	L_1 Error value
Position	$36.6 \pm 22.6cm$
Velocity	$1.57 \pm 1.08m/s$
Length	$36.41 \pm 34.88cm$
Width	$14.32 \pm 14.67cm$

Table 6: The L_1 error measures for different property estimations of our solution.

Even with the lowest confidence threshold the recall of the system doesn't exceed 90%. This gives an indication about the proportion of the targets in the test set which are undetectable, likely because they are not in the radar signal (obstruction or they are in an elevated lane).

The selected model (at the optimal operating point) has 95.46% precision with 84.85% recall. The parameter count is $7.5M$ and the MAC count excluding NMS is $8.4GMACs$.

Some of the inference results can be seen in Figure 5. Note that in some cases (bottom row) the perception system misses vehicles due to them being obstructed from view by other vehicles or because they are elevated compared to the sensor. In some other cases (*e.g.* top right) the detector sees cars behind others. This can be attributed to the fact that radio waves can reflect off the road under the cars.

5.5. Comparison with LiDAR-based approaches

Even though there are no publicly available radar range-azimuth-Doppler datasets, we would still like to make an attempt at putting our results into perspective. As the most prevalent sensor in autonomous environments which enables bird's eye view detections is LiDAR, we will use that sensor for a point of comparison, even if the fundamental signal characteristics are different (see Figure 1 for an example).

On the KITTI Bird's Eye View detection benchmark initiated by Geiger *et al.* [5], state of the art solutions such as Frustum ConvNet by Qi *et al.* [20], Deep Continuous Fusion (DCF) from Liang *et al.* [12] and VoxelNet from Zhou and Tuzel [25] report 84.0%, 85.83% and 84.81% mAP respectively (on moderate difficulty vehicles). On larger scale (not publicly available) datasets, Fast and Furious (FaF) by Luo *et al.* [19], and Deep Continuous Fusion from Liang *et al.* [12] report similar numbers (83.1% and 83.89 mAP respectively).

Even though these cited results are comparable to our model's reported performance, it is not fair to directly relate them to each other for a number of key reasons. First, the cited results are all calculated with an IoU similarity threshold 0.7, while we used 0.5. In Figure 7 we visualize the mAP score of our system for different IoU matching thresholds and also include results from other works. Based on this figure we conclude that it is possible to achieve similarly high detection performance with radar just as well

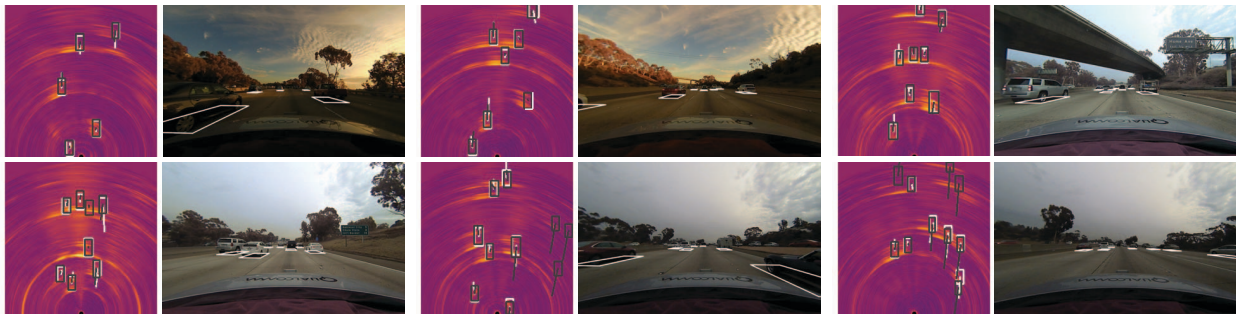

Figure 5: Examples from our test set. The radar signal has been visualized in Cartesian coordinates for easier human verification. Targets are indicated by black, predictions by white outlines. Velocity estimation targets and predictions are also visible. Best viewed digitally.

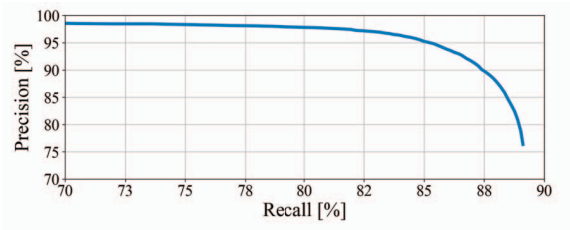

Figure 6: Precision-recall curve of our final model.

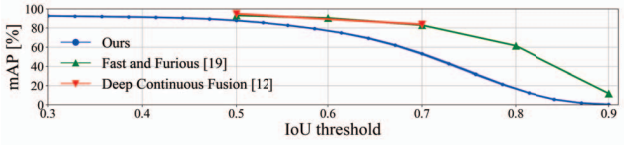

Figure 7: mAP vs IoU. Note that the methods use differing datasets for evaluation.

as with LiDAR. Given our settings and solution, however, the bounding box accuracy of our system is lower than of LiDAR-based methods. This is hinted at by the fact that the curve representing our results is shifted to the left.

We also analyzed how our system's performance degrades over distance, visualized in Figure 8. Cars are more likely to be obstructed from view the farther they are, so to examine the effect of distance only, we filtered out all obstructed vehicles for this analysis. Compared to LiDAR-based methods, the trends imply that our radar-based perception system is less sensitive to the distance of the target.

We also have to acknowledge other differences between the settings, such as that the cited datasets represent different environments than ours, or that the LiDAR sensor resides on the top of the vehicle while the radar is level with the bumper. Due of this, we advise the reader to compare the trends instead of the values.

In runtime, our method seems to be advantageous over LiDAR-based solutions. Our radar-based model runs at 95 FPS on an NVidia GeForce GTX 1080 Ti, about an

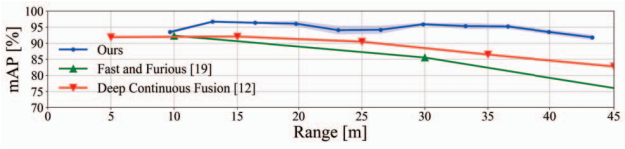

Figure 8: mAP vs range. Note that the methods use differing datasets for evaluation. We conclude that our radar-based approach's detection performance degrades more slowly with distance than the compared methods'.

order of magnitude faster than most LiDAR-based models [20, 12, 23]. However, there are also models with a significantly higher frame rate [19, 10], but the frame rate of the LiDAR sensor itself bottlenecks the latency, as modern LiDAR sensors are operating with a frame rate of about a 10Hz compared to the 125 Hz achieved with our settings.

6. Conclusion

In this paper we have presented a perception system for highway automotive environments, utilizing a radar sensor. To the best of our knowledge, we are the first ones to demonstrate a deep-learning-based object detection model that operates on the radar tensor as opposed to the point cloud resulting from peak detection. Our model includes a solution to the problem of the input being in polar coordinate system while the desired detection representation is in Cartesian space. We also proposed a novel way of handling the third, non-space-like dimension of the radar signal, the Doppler (velocity) dimension and demonstrated that it can be leveraged to increase detection performance.

Finally, we demonstrated the viability of the solution by comparing its characteristics to LiDAR-based approaches, finding that our model has a lower latency and sensitivity to target range, as well as providing similar detection performance.

References

[1] A. Angelov, A. Robertson, R. Murray-Smith, and F. Fioranelli. Practical classification of different moving targets using automotive radar and deep neural networks. *IET Radar, Sonar and Navigation*, 2018.

[2] J. Dickmann, N. Appenrodt, J. Klappstein, H.-L. Bloecher, M. Muntzinger, A. Sailer, M. Hahn, and C. Brenk. Making bertha see even more: Radar contributions. *IEEE Access*, (3):1233–1247, 2015.

[3] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[4] H. Furukawa. Deep learning for end-to-end automatic target recognition from synthetic aperture radar imagery. *Arxiv preprint, https://arxiv.org/abs/1801.08558*, 2018.

[5] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[6] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] D. H. Johnson and D. E. Dudgeon. *Array Signal Processing: Concepts and Techniques*. Simon & Schuster, Inc., 1992.

[9] Y. Kim and T. Moon. Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks. *IEEE geoscience and remote sensing letters*, pages 8–12, 2016.

[10] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *arXiv preprint arXiv:1812.05784*, 2018.

[11] J. Li and P. Stoica. *MIMO Radar Signal Processing*. Wiley-IEEE Press, Oct 2008.

[12] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018.

[13] J. Lien, N. G. M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics*, 4, 2016.

[14] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR.)*, 2017.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *The IEEE International Conference on Computer Vision (ICCV).*, 2017.

[16] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *Advances in Neural Information Processing Systems*, pages 9628–9639, 2018.

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

[18] J. Lombacker, M. Hahn, J. Dickmann, and C. Wohler. Object classification in radar using ensemble methods. *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 87–90, 2017.

[19] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[20] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[21] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, pages 802–810.

[22] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[23] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[24] M. Zhao, T. L. M. A. Alsheikh, Y. Tian, H. Zhao, A. Torralba, and D. Katabi. Through-wall human pose estimation using radio signals. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7356–7365, 2018.

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