TAV Exam: Proximity Sensor Using Radar and Camera Data

Scavone Alessandro - s328782

8th June 2025

Contents

1 Introduction

The goal of this project is to develop and evaluate a sensor fusion system that combines radar and camera data to implement vehicle proximity detection. By leveraging radar point clouds and visual object detection via a YOLOv8 model, the system aims to:

- Detect and track nearby vehicles in real-time
- Associate radar measurements with detected objects to estimate the vehicle speed
- Fuse radar-based velocity and camera-based velocity to improve the accuracy of the speed estimation
- Identify approaching vehicles which is critical for applications like rear collision warning and blind spot detection.

In particular this fusion-based approach seek a more robust velocity estimation by trading off between radar-based measurements with visual estimation, especially in situations where radar data is sparse or unreliable.

Proximity detection is a fundamental component in modern vehicles. It enables a vehicle to perceive and interpret its immediate surroundings, allowing it to detect nearby obstacles, other vehicles, pedestrians, or potential hazards. In the considered context, proximity detection was aimed to recognize approaching vehicles and so to perform blind-spot monitoring.

The code is available at: https://github.com/alexscavo/Proximity-Sensor-Project.git

2 Background

2.1 Truckscenes Devkit

For the project I used Truckscenes devkit, that is the world's first public dataset for autonomous trucking. The sensor suite consists of a multimodal sensor setup with 4 cameras, 6 lidars, and 6 radars. For this project I exploited only radar and camera data of the rear left side. The vehicle used for this task was a *Longhaul Truck MAN TGX 18.510*, while the two sensors were:

- Camera sensor: 4x Sekonix SF3324 RGB, sampling rate: 10 Hz, resolution 1928×1208
- Radar sensor: 6x Continental Engineering Services GmbH - ARS540CES

The camera data of the four camera sensors undergoes an undistortion process using a pinhole camera model and a Lanczos interpolation scheme with a 8×8 kernel and constant padding values. The resulting image is cropped to a 1980×943 pixel size and stored as a compressed JPEG image. For this reason as we'll see I needed to adjust the provided focal length, to take into account this image cropping.

For what concerns the dataset, it contains overall 747 scenes of 20s from Highway (70%), Rural (10%), Terminal (10%), City (5%) and different weather conditions. In particular I used the mini version of the dataset, that contained only 10 scenes.

2.2 YOLOv8

YOLOv8 (You Only Look Once, version 8) is a state-of-the-art real-time object detection and tracking model developed by Ultralytics. It offers high accuracy and speed, making it well-suited for automotive perception tasks. In this project, YOLOv8 is used to:

- Detect vehicles (cars, trucks, buses, etc.) in camera images.

- Assign class labels to each detected object.
- Track object movement across frames using ByteTrack, enabling temporal association and motion estimation.

YOLOv8 serves as the visual backbone of the system, providing semantic information (what the object is) and spatial localization (where it is) needed for associating radar data and estimating object velocities.

2.3 Motivation for Fusing Radar and Camera Data

Radar and camera data provide complementary informations:

- **Camera**: provide visual informations and can be used for object detection, but lacks of precise depth and velocity estimation
- **Sensor**: provide rich velocity measurements but has low resolution and poor semantic understanding

Another key challenge when working with radar data is the inconsistency and ambiguity of point measurements. In some cases, there are too few radar points associated with a detected vehicle, making it difficult to compute a reliable velocity estimate. This can happen due to the radar's limited resolution, occlusions, weather conditions or bad reflectivity angles.

Moreover, even when radar points fall within a bounding box, they may not actually belong to the detected vehicle. They could originate from nearby vehicles, road surface imperfections, or static background objects. As a result, using these points without validation can lead to inaccurate or misleading velocity estimations.

To mitigate these issues, the system uses a fusion strategy that dynamically adjusts the contribution of radar and camera-based estimates, based on the number of points associated to that vehicle. When radar data is scarce or potentially unreliable, greater weight is given to the camera-derived velocity, improving the robustness of the final estimation.

3 System Architecture

Dataset Loader This module initializes and accesses the TruckScenes dataset, specifically the *v1.0-mini* version. It loads samples, retrieves sensor data (camera and radar), and manages calibration and ego motion metadata necessary for synchronization and projection tasks.

YOLO-based Detection and Tracking Using the YOLOv8 model with ByteTrack, the system performs real-time object detection on camera images. It identifies objects of interest (primarily vehicles), assigns class labels, confidence scores, and maintains track IDs across frames to facilitate temporal tracking of individual vehicles. Specifically, we use ByteTrack to perform multiple object detection and maintaining track IDs from frame to frame. The model defines bounding boxes, with track IDs to track them between frames. We will use the bounding boxes to recover which radar points are related to the vehicle of interest.

Radar Point Cloud Processing Radar data is read in the form of point clouds. Each radar point contains 3D coordinates and relative velocity. The points are projected into the camera image plane, geometrically filtered, and later matched with visual bounding boxes to extract motion information. Once we have the bounding boxes from the YOLO model, we use them to detect which radar points falls into them. These radar points will be considered as points related to the vehicle. Of course there will be bad points or points that do not belong to the considered vehicle, but we'll try to address this problem during the velocity estimation.

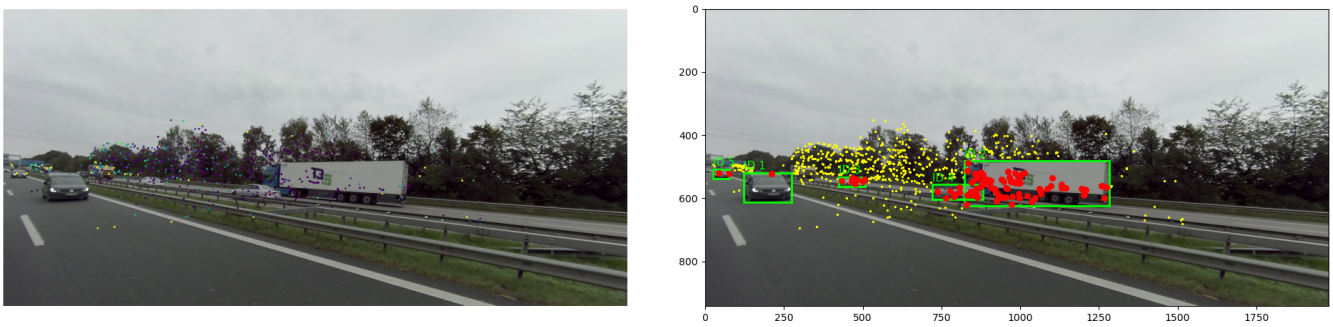

Figure 3.1: On the left side the complete radar point cloud mapped on the image. On the right side in red the points that belong to recognized vehicles and bounding boxes

Point and Bounding Box Filtering Since the radar point cloud covers the entire image, I made the assumption that a portion of that data is not useful for the specific task of proximity detection. As we can see in fig. 3.2, I consider a line above which I assume that data won't be useful to detect approaching vehicles. That line has been tuned so to not be too low on the track lane, since very high vehicles such as trucks could loose too many radar points that are our main source of velocity estimation. On the other hand, a very high filter line wouldn't have been effective, filtering very few or none points at all.

Figure 3.2: Filter line used to filter out points that are assumed not to be useful for approaching vehicle detection

In the same way, the filter also filters out bounding boxes that have at least 3 corners over the filter line, avoiding useless recognitions of other lanes.

Velocity Estimation Once we have bounding boxes and the corresponding radar points, we have all we need to estimate vehicles velocities. To obtain a robust and accurate estimation, we exploit 2 contributions: velocity computed using radar points, and velocity computed using bounding boxes. All the velocities that we'll refer to are relative velocities with respect to the ego vehicle (the truck that has the sensors on). We won't evaluate absolute velocities since they're not interesting for the analysis in our case. Anyhow, absolute velocities could be easily retrieved by adding to the ego velocity (velocity of the truck) the relative velocity of the other vehicles.

Since as we said radar points can be not strictly related to the vehicle considered, or they can simply be bad points, we average the velocity of all the points inside the bounding box. Of course the more points we have, the better will be the estimation. For this reason, if we lack radar points, we need to consider more the velocity obtained using bounding boxes. To estimate the velocity of vehicles using only camera data, I tracked detected objects across consecutive image frames. By monitoring how the size of each object's bounding box changed over time, I inferred whether the object was approaching or moving away from the camera. This change in apparent size was used to estimate motion along the viewing direction. The approach relies on basic geometric assumptions about object size and camera perspective, without requiring external sensors. Since to compute the bounding box velocity we need at least 2 frames to detect a change, the speed in that case is computed considering only radar points (if available and numerically enough). Lastly to make the boundary box speed detection more robust, I computed the velocity as the average of the velocities of the bounding box across the last 4 frames (I don't strictly require 4 frames, so I can start the computation even if I have just 2 frames, but overall I average the bounding box computation of the last 4 frames).

The main idea behind the two velocity fusion, was that I wanted to weight the radar velocity estimation with the number of points I used to compute it, reflecting the estimation reliability. When few or no radar points are available, the system relies more on the velocity estimated from the camera data. Details on the computation will be provided in next sections.

Vehicle Approaching Signal Once we store bounding box IDs and the corresponding fused velocities, we can track them across different frames to assess whether the vehicles are approaching or not. We'll see in following sections how.

4 Methodology

In this section we'll discuss more technical parts.

4.1 Filter Boundary

The filter boundary has been defined considering a slope of 0.38, and intercept of 200. Once defined those parameters, the filter is obtained as:

$$\text{geo\_filter} \;=\; v \geq \text{slope} * u + \text{intercept}$$

Where v and u are the pixel coordinates. The filter is then applied to the matrix containing the points, to filter out points above that line. Similarly I filtered the bounding boxes, looking for the 4 corners of the box and then evaluating how many of them were above that filter line, deciding whether to keep that box or not.

4.2 Radar Velocity Retrieval and Computation

The main goal is to evaluate velocities. To evaluate radar velocity we need to recover radar points. Since we want to fuse radar and camera velocities, we need them to be in the same coordinate system. For this reason I project radar points and velocities in the ego vehicle frame using the sensor calibration. Specifically I rotate each radar velocity vector into the ego frame. Then for each boundary box we save the corresponding points and relative velocities.

We compute the radar-based relative velocity by averaging all the relative velocities that we retrieved for each bounding box, therefore the more points we have the better will be the estimated value. Since the relative velocities are in m/s we convert them into km/h. If there aren't points, the radar velocity will be set to 0 and it won't be used to compute the fused velocity.

4.3 Boundary Box Velocity Computation

In order to compute boundary box velocity we need at least 2 frames. To evaluate the speed we will measure the difference between the bounding box across consecutive frames. If the bounding box is not detected anymore for more than 3 consecutive frames, it gets forgotten. To detect how much the bounding box moves in the image, I do:

1. Compute the bounding box centre
2. Compute the bounding box height in pixels (inversely proportional to the distance to the vehicle, so small box equals distant object)
3. Use the Pinhole camera model to obtain an approximated distance between the vehicle and the camera (f_x is the focal length of the camera):
$$Z = \frac{H_{\text{real}} \cdot f_x}{h_{\text{pixels}}}$$
Since YOLO model gives also a prediction of the type of vehicle, I exploited this information to create a dictionary of approximated vehicle heights $H_{\text{real}}(\text{car} \approx 1.5m, \text{truck} \approx 3.0m, \dots)$ that I give as granted, and allow me to compute Z.
4. Take the estimated Z-distances (depths) over time (in a time window of 4 frames) for a tracked vehicle and computes its forward velocity based on how that distance changes across frames.
   - 4.1 For each pair of consecutive entries compute the difference in depth $Z1 - Z0$
   - 4.2 Multiply by the sampling rate of the camera sensor (10 Hz), equivalent to dividing for the inverse of the frequency
   - 4.3 Average the velocities across the 4 frames and convert them into km/h

A negative velocity means that the rear vehicle is slowing down with respect to the ego frame, otherwise it's approaching. Since the focal length saved in the sensor calibration is referred to the original image dimension/resolution 1928×1208, I needed to convert it to match the actual image resolution on which I was working and on which radar data was projected, that was 1980×943. To address the problem I computed the new focal length as:

$$\begin{aligned} \text{scale}_x &= \frac{\text{new\_width}}{\text{orig\_width}} \\ f_x &= f_{x_{old}} \cdot \text{scale}_x \qquad \approx 660.0 \end{aligned}$$

4.4 Velocity Fusion

We want to increasingly consider more the radar velocity as the number of radar points increase for a bounding box. To do so the fused velocity is computed as:

$$v_{\text{fused}} = \alpha \cdot v_{\text{radar}} + (1 - \alpha) \cdot v_{\text{bbox}}$$

Alpha is a confidence weight for the radar contribution so that $\alpha \in [0, 1]$, computed as:

$$\alpha = \frac{N}{N + k}$$

Where the k factor is a sort of radar smoothing constant that allows us to tune how quickly the radar's influence α grows as the number of matched radar points N increases. I tried also other types of weight schedules like

sigmoids and power lows but the method described resulted to be the best trade weight increase and number of points considered (and so confidence). The considered k was 5.

If neither radar velocity and boundary box velocity are available, the system doesn't provide any velocity measurements. To provide a relatively accurate measurement, I do not output anything if the number of radar points is less than 4 and there aren't 2 consecutive frames for that boundary box, considering that condition an edge condition that doesn't allow to trust the velocity estimation.

4.5 Signaling Approaching Vehicles

A vehicle that:

- Remains visible for 3 consecutive frames
- Have positive velocities for the last 3 frames
- Have all the last 3 velocities at least $> 5\ km/h$

is considered approaching. When a vehicle is not detected for more than 3 consecutive frames, it gets discarded.

5 Results

The system has been tested on different highway scenes with diverse lighting and weather conditions. Specifically:

- Scene 2: night scene
- Scene 3: bright day scene (from which we report some results)
- Scene 4: bright day scene
- Scene 8: cloudy day scene

The system works overall very well, in particular when the approaching vehicle starts to be relatively close to the ego vehicle, when there are enough radar points and enough light. The system still struggles:

- To evaluate correctly the bounding box velocity when the approaching vehicle is still quite far from the ego one. One possible cause is that I'm estimating velocity from changes in bounding box height, and when the object is far, the bounding box appears very small and changes very slowly, making the motion hard to detect and sensitive to noise.
- To estimate well radar velocities when there are less than 3 or 4 points and when the bounding box is not tailored to the vehicle, since it can use radar points that are not relative to the vehicle and therefore harming the radar velocity measurement.
- Even if not used and particularly relevant for the task, the model often incorrectly detects vehicles on the opposite lane: sometimes it detects positive velocities (so as if the vehicle is the approaching one, even if not possible). In particular I noticed that the bounding box measurement struggles to detect correctly the velocity of those vehicles on the opposite lane, not being able enough to measure vehicles going in the opposite direction.

As we can see from fig. 5.1, we can see that as the vehicle gets closer to the ego vehicle, the measurements of radar and bounding box gets similar, providing a better result. This is confirmed also by fig. 5.2, where we see that as we retrieve more radar points, the fused velocity gets more stable. The same thing is obtained also for another vehicle during the same scene (as in fig. 5.3), that spends more time to get closer, but showing a similar behavior. In

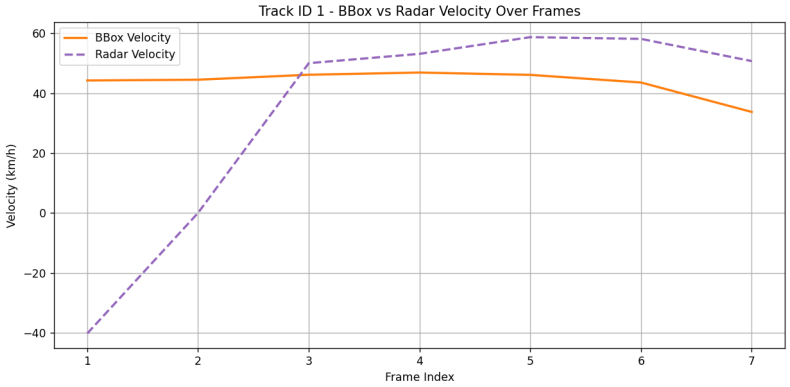

Figure 5.1: Comparison between radar velocity and bounding box velocity

this second case, we have fewer radar points until it's very close, reason for which the radar velocity is not trustable enough. We can also observe that the obtained fused velocities are overall quite reasonable, considering the the theoretical maximum velocity of the ego truck is $80km/h$, a relative speed of $40 - 50km/h$ for an overcoming vehicle leads to an approximate velocity of $120 - 130km/h$, quite common speeds on highways, as in the considered scenes. Other very good results can be observed using scene number 4.

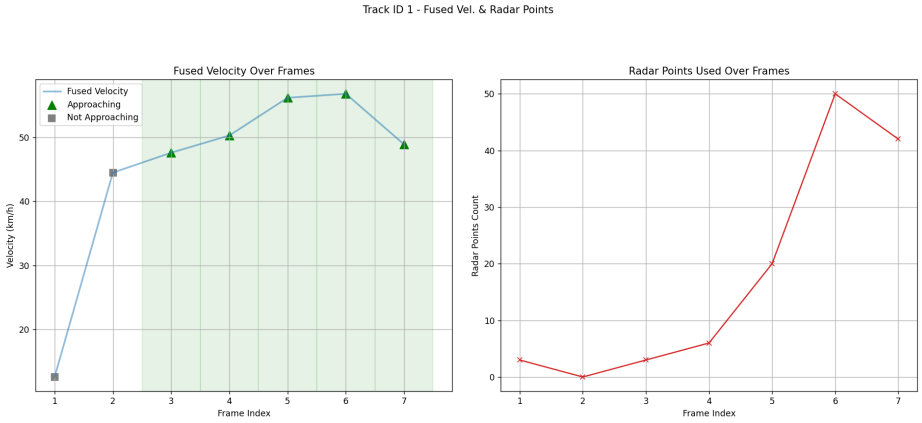

Figure 5.2: On the left, we can see the fused velocity trend, with also signed the frames in which the vehicle has been considered approaching and when not (green dots and green shadow). On the right side we see how the number of radar points increase frame by frame. The last drop is due to the fact that the vehicle is surpassing the ego one, so it's not completely visible. These two plots are for vehicle 1.

As we can see in this second case, the vehicle requires time before obtaining a stable velocity evaluation and also to be correctly recognized as approaching, due the subtle limitations described previously, considering that this second vehicle is detected for a very long time when it's still very distant.

One last limitation found is that for lighting or weather condition the YOLO model is not able to detect bounding boxes, as for scene 2 that shows a night scene. The radar points alone are not sufficient to understand where is a vehicle easily, leading to no detection. Future works could solve this major problem by analyzing how radar data is

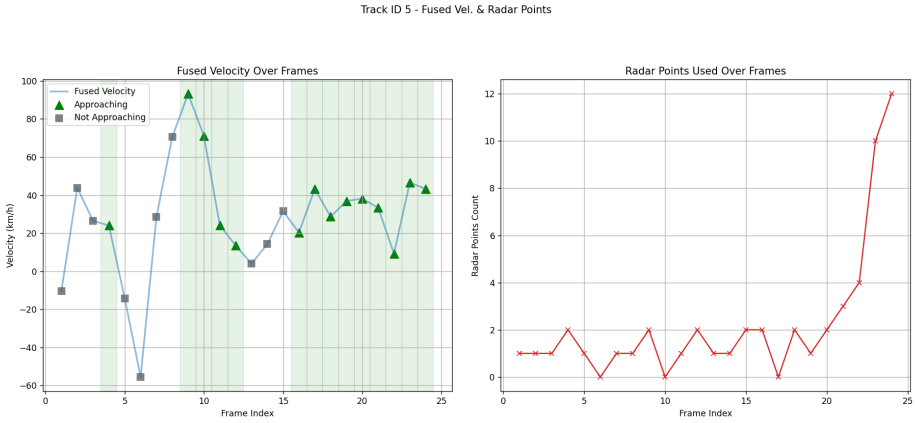

Figure 5.3: Same plots as before, but for vehicle 5. The detection starts from very far, therefore requiring more time to converge and being trustable

distributed and understanding the eventual presence of a vehicle or just noise, assessing then if necessary the velocity based only on radar points or exploiting other types of sensors.

All the hyperparameters used in the pipeline — such as the slope and position of the geometric filter line, the temporal window size for velocity estimation, the minimum number of radar points required for reliable fusion, and the weighting schedule for sensor fusion — have been assigned reasonable default values based on empirical testing. While these values have not been exhaustively optimized, they provide a good trade-off between robustness and responsiveness. Of course, further tuning or dataset-specific calibration could improve performance in edge cases, but the current configuration already yields consistent and interpretable results across different scenes.

Lastly, for what concern the approaching vehicle signaling, I simply print on the terminal a message, if the vehicle is detected as approaching. I also print a message that signals if the velocity couldn't currently be computed, condition usually matched in the first few frame iterations.

6 Conclusions

- **Sensor Fusion Improves Accuracy:** Combining radar and camera data yields more reliable velocity estimates than using either alone.
- **Radar More Reliable at Close Range:** Radar velocity estimation improves as the number of points increases when vehicles are near.
- **Camera Estimation Weak at Long Range:** Bounding box-based velocity struggles with distant vehicles due to minimal size variation.
- **Filtering Enhances Precision:** Geometric filtering of radar points and bounding boxes reduces noise and improves association accuracy.
- **Poor Night Performance:** YOLO fails in low-light conditions, causing detection loss; radar alone is insufficient for full detection.
- **Approach Detection Works When Close:** Approaching vehicles are correctly flagged when visible and close over multiple frames.

- **Parameters Are Reasonable Defaults:** Hyperparameters worked well across scenes but could benefit from further tuning.
- **Good Generalization:** The system performs well across different environments, suggesting strong potential for broader deployment.