RWU Hochschule Ravensburg-Weingarten University of Applied Sciences

Faculty for Electrical Engineering and Computer Science

Task Report

Multi Sensor Fusion - Estimate the Distance

Guided by:

Mr. Felix Berens

*S*ubmitted by:

Akshay Chobe-36316

Master-Mechatronics

10 May 2024

Introduction

In this report, a multi-sensor fusion method for vehicle distance estimation is proposed. This task is crucial for applications like autonomous vehicles, where accurate distance perception is essential for safe navigation. The proposed algorithm utilized YOLO v8 for vehicle detection, and LiDAR and RADAR point clouds are employed to gather 3D environmental data. 3D points are projected onto the 2D image plane using camera parameters from calibration data. Distances to vehicles are calculated separately for LiDAR and RADAR. Subsequently, these distances are combined using weighted average sensor fusion method to enhance accuracy. Finally accuracy evaluation is conducted using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics against ground truth 3D bounding box distances.

Step 1: Object Detection with YOLOv8

The script utilizes a pre-trained YOLOv8x model which is then specifically configured to recognize vehicles commonly found on roads [1]. To achieve this, specific classes like 2:car, 5:bus and 7:truck from YOLOv8 model were chosen, so that unnecessary detections like traffic light and stop signs will not be detected. This helps in more robust and focused detections according to the project requirements.

First of all the 100 images were manually separated out of the original dataset and YOLOv8 model performs object detection on each of this image. The model's output predictions were filtered to a confidence threshold of 35% to minimize the false positives in detection, indicating a 35% certainty level required for an object to be considered a valid detection which avoids processing detections with lower confidence. These detections include bounding boxes around the identified vehicles. The detected bounding box labels were stored in a specific folder to work on later.

Figure 1: Image 000042.jpg with vehicle detection

Step 2: Project Point Cloud

This step transforms 3D points from the LiDAR and RADAR sensors (represented as a point cloud) onto the corresponding 2D image plane. This allows for visualization LiDAR

and RADAR point clouds in relation to the image coordinate system.

- **Camera Calibration Data Integration:** Our algorithm begins by extracting relevant data based on a specific image. This data includes calibration information for the camera and sensors, 3D point cloud data from LiDAR and RADAR text files and also the corresponding image for accurately transforming 3D points from the LiDAR and RADAR sensors into the 2D image plane.

- **Point Transformation and Projection onto Image Plane:** Next, the script iterates through images and for each image, it utilizes the camera calibration data to transform 3D points from the LiDAR and RADAR point cloud data into the camera's coordinate system. This transformation considers the camera's position, orientation, and intrinsic properties and calculates the corresponding 2D pixel coordinates within the image for each 3D point projection.

Figure 2: Image 000131.jpg with LiDAR point Projections

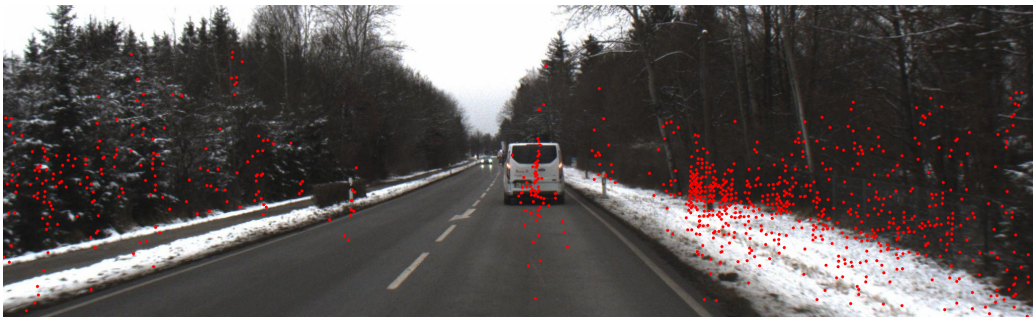

Figure 3: Image 000131.jpg with RADAR point Projections

Step 3: Filter points in Bounding boxes

Once the 3D LiDAR and RADAR points are transformed according to the 2D image plane, this step involves filtering the points that lie within the detected bounding boxes and visualising them.

- **Potential Bounding Box Filtering:** The script then focuses on the most relevant areas of the scene. The algorithm computes the boundaries of the bounding box and

filters the projected points, keeping only those that fall within the bounding box of the detected vehicle by using the labels of the bounding box stored already during YOLO vehicle detection. This ensures the visualization focuses on the specific object of interest.

- **Visualization:** Finally, the script visualizes the results by drawing the detected bounding boxes on the image and overlaying colored circles to represent the filtered LiDAR (green) and RADAR (red) points within each box. This visualization combines information about object presence and location, providing a richer understanding of the scene.

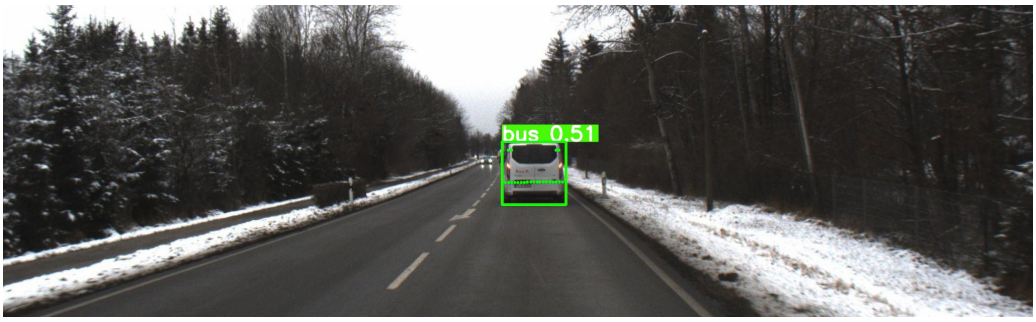

Figure 4: Image 000131.jpg with Filtered LiDAR points projection

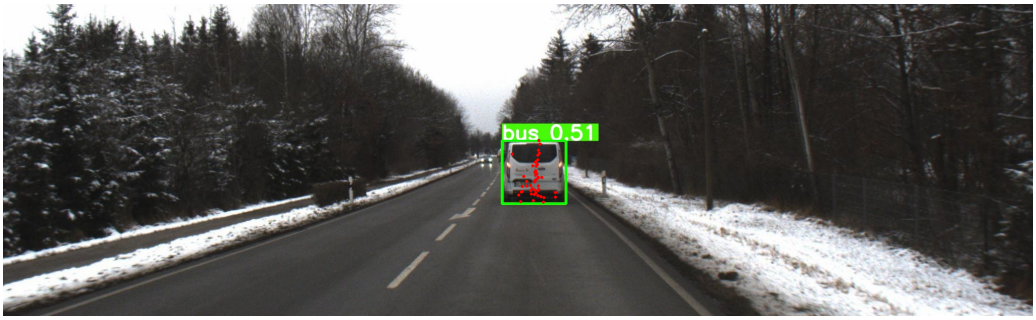

Figure 5: Image 000131.jpg with Filtered RADAR points projection

Step 4: Estimate the Distance

This step focuses on estimating the distance of the detected vehicles within each bounding box using the LiDAR and RADAR data. Filtering the points ensures the distance estimation is based on points relevant to the specific vehicle. While filtering the projected points(pixel coordinates u,v) in the bounding box, their original coordinates (represented in x,y,z format) are stored in a different list. This list will then be used to extract the x coordinate information of all projected points, which represents the depth information.

However, simply calculating the average distance (mean) might be inaccurate. Bounding boxes can contain outliers from the background, skewing the mean. For example, consider distances like 23m, 25m and 55m (outlier). The mean (around 34m) wouldn't accurately represent the distance.

- **Outliers:** The bounding box might contain some points from the background environment besides the actual vehicle. These outliers can significantly skew the mean distance, leading to inaccurate results.

- **Median:** To address outliers, our algorithm calculates the median distance for both LiDAR and RADAR points within each bounding box. The median, representing the "middle" value, is less susceptible to outliers. In our example, the median would be 25m, providing a more accurate estimate.

The script then calculates the median distances for both LiDAR and RADAR points within each bounding box and stores them in separate lists which will be used in the accuracy estimation at the end.

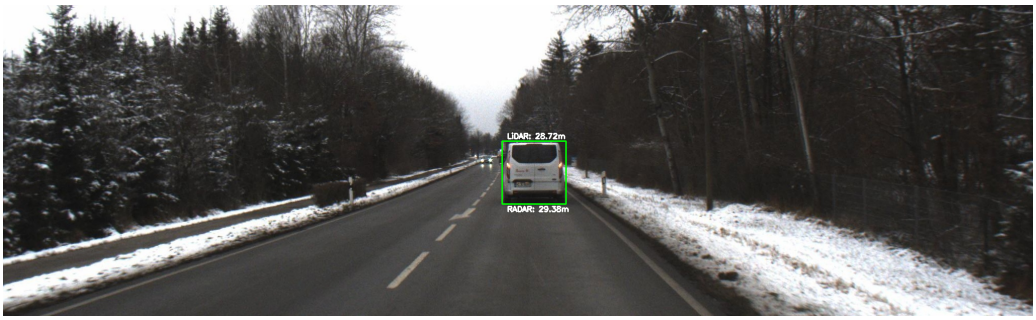

Figure 6: Image 000131.jpg with LiDAR and RADAR Distances

Step 5: Weighted Average Sensor Fusion

This step refines distance estimates by fusing LiDAR and RADAR data using a weighted average method. LiDAR, known for its high accuracy in distance measurement due to shorter wavelength laser pulses and wide field of view (FOV), receives a higher weight (80%) compared to RADAR (20%) weight which is mostly used for velocity estimation, have shorter FOV but less affected by weather conditions [2]. Our project focuses on distance estimation, so accordingly weights are distributed.

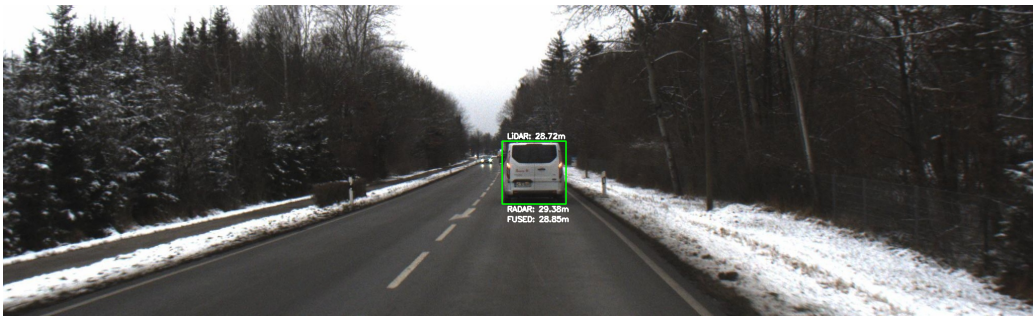

Figure 7: Image 000131.jpg with Weighted Average Sensor Fusion Distance

The code calculates a weighted average sensor fusion distance for each bounding box using the formula from [3]:

$$\textbf{Fusion distance} = \textbf{Lidar weight} * |\textbf{Lidar distance}| + \textbf{Radar weight} * |\textbf{Radar distance}|$$

This approach leverages the strengths of both sensors while potentially compensating weaknesses of individual sensors, ultimately aiming for a more robust and reliable distance estimation.

Step 6: Accuracy Evaluation

This final step assesses the accuracy of the distance estimates using two common metrics: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Lower values in both cases indicate better accuracy.To calculate these metrics, the script first extracts the depth information from groundtruth centre3D information of the vehicles and compares them with the final weighted average distances obtained in the previous step. It's important to note that there are discrepancies in the number of vehicles detected between the ground truth and YOLO predictions. To address this for calculating MAE and RMSE (which require equal-length data for comparison), the script either truncates or pads the mismatched values.

The evaluation resulted in a Mean Absolute Error (MAE) of 5.49 meters and a Root Mean Square Error (RMSE) of 6.67 meters. These values suggest that, on average, the distance estimates were generally within 5.49 meters of the actual distances. The slightly higher RMSE indicates there might have been a few instances with larger errors. Overall, these metrics suggest good accuracy in the final distance estimates.

Conclusion

In conclusion, this report presented a multi-sensor fusion approach for estimating vehicle distances in images. The system leverages YOLOv8 for object detection, LiDAR and RADAR point clouds for 3D information, and a weighted average of LiDAR and RADAR distances for robust distance estimation. The final step evaluates the accuracy using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), achieving a MAE of 5.49 meters and RMSE of 6.67 meters, indicating good overall accuracy. To mitigate these errors, in future instead of filtering projected points in specified bounding box, a segmented mask of vehicle can be considered for estimating the actual vehicle boundary. This ensures that there are no points from environment that are unnecessarily filtered as points on vehicle.

References

1. Jocher, G., Chaurasia, A., Qiu, J. "Ultralytics YOLOv8" [Ultralytics YOLO (Version 8.2.9)], *Computer Software*, https://github.com/ultralytics/ultralytics.

2. Daniel G., Miao W., Michael S.r, Tinosch G. "Radar/Lidar Sensor Fusion for Car-Following on Highways" *Research Article*, https://shorturl.at/fhAUW

3. Peng Lu1, Fengzhi Dai. "A Study on Multi-sensor Data Fusion Algorithm" *Research Article*, https://rb.gy/ymurcg.