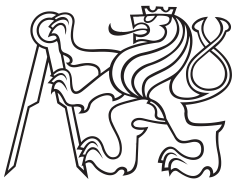
Bachelor's Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Speed Bump Detection in Automotive Scenarios

Jindřich Macek

Supervisor: Ing. Antonín Novák, Ph.D.
Study program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science
May 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Macek Jindřich** Personal ID number: **499157**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Speed Bump Detection in Automotive Scenarios

Bachelor’s thesis title in Czech:

Detekce zpomalovacích prahů v automobilových scénářích

Guidelines:

The aim of this work is to design, develop, and test methods for the detection of speed bumps in outdoor environments considering applications in the automotive industry. The envisioned solution should combine measurements from a LiDAR sensor and a camera to increase the overall earliness and reliability of detection.
1) Learn about the ROS environment, the characteristics of Livox LiDAR data, and methods for speed bump detection.
2) Propose detection algorithms using LiDAR point cloud and camera image.
3) Analyze and propose methods to combine the outputs of each detector considering sensor parameters and environmental conditions in automotive scenarios.
4) Evaluate the proposed methods in terms of quality, reliability, and earliness on real driving records under different ambient conditions and compare them with related methods in the literature.

Bibliography / sources:

[1] Zimmer, Walter, et al. "A survey of robust 3d object detection methods in point clouds." arXiv preprint arXiv:2204.00106 (2022).
[2] Yun, HS., Kim, TH. & Park, TH. Speed-Bump Detection for Autonomous Vehicles by Lidar and Camera. J. Electr. Eng. Technol. 14, 2155–2162 (2019).
[3] Varma, V. S. K. P., et al. "Real time detection of speed hump/bump and distance estimation with deep learning using GPU and ZED stereo camera." Procedia computer science 143 (2018): 988-997.
[4] Zou, Zhengxia, et al. "Object detection in 20 years: A survey." arXiv preprint arXiv:1905.05055 (2019).

Name and workplace of bachelor’s thesis supervisor:

Ing. Antonín Novák, Ph.D. Department of Control Engineering FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **01.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Antonín Novák, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

Acknowledgements

I would like to sincerely thank my supervisor Ing. Antonín Novák, Ph.D., for his time, guidance and willingness. Furthermore, I would also like to thank all my colleagues with whom I could cooperate for their advice and help. Last but not least, I would also like to thank my family and friends for their endless support during my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, May 25th 2023

.....
Author's signature

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 25. května 2023

.....
Podpis autora práce

Abstract

Modern vehicles often use various electronic devices to detect objects and observe the surroundings around the car. Presently, a widely used device employed for such purposes is the camera, which provides visual assistance to the driver and can also serve as a tool for object detection by the vehicle itself. In addition, LiDAR technology, which employs laser light to measure distances and reflectivity in 3D space, is also utilized.

This thesis aims to propose, implement and evaluate an algorithm to detect speed bumps on roads under various weather conditions. Deploying such functionality to the vehicle could enhance driver safety and comfort when crossing over speed bumps. The proposed solution combines the utilization of LiDAR and a camera, allowing the system to detect speed bumps earlier and with greater accuracy. The first method employed in this algorithm fits a plane to the collected points from LiDAR and subsequently determines the detection based on the distances of the points from the computed plane. The second method uses a convolutional neural network trained on custom data containing classes describing a speed bump and its respective traffic signs. This neural network identifies and classifies speed bumps within the captured images from a camera.

The algorithm was tested and tuned on recordings from the test vehicle. Its functionality was demonstrated by the automatic deceleration of the test vehicle when approaching a speed bump with activated cruise control. The vehicle can slow to 18 kilometers per hour in time and thus ensure a comfortable crossing over a speed bump.

Keywords: object detection, RANSAC, LiDAR, camera, speed bump, point cloud, convolutional neural network, YOLOv8

Abstrakt

Moderní vozidla často využívají různá elektronická zařízení k detekci objektů a pozorování okolí kolem vozu. V současnosti je pro tyto účely hojně používaným zařízením kamera, která poskytuje vizuální asistenci řidiči a může sloužit i jako nástroj pro detekci objektů samotným vozidlem. Kromě ní se používá také technologie LiDAR, využívající laserové světlo k měření vzdáleností a odrazivosti ve 3D prostoru.

Cílem této práce je navrhnout, implementovat a vyhodnotit algoritmus pro detekci zpomalovacích prahů na silnicích za různých povětrnostních podmínek. Integrování takové funkce do vozidla by umožnilo zvýšení bezpečnosti a pohodlí řidiče při přejíždění zpomalovacích prahů. Navržené řešení kombinuje použití LiDAR a kamery, zajišťující dřívější a přesnější detekci zpomalovacích prahů. První metoda použitá v tomto algoritmu prokládá body nasbírané pomocí LiDAR rovinou a následně rozhoduje detekci na základě vzdáleností bodů od vypočítané roviny. Druhá metoda využívá konvoluční neuronovou síť natrénovánou na vlastních datech obsahujících třídy popisující zpomalovací práh a jeho příslušné dopravní značení. Tato neuronová síť identifikuje a klasifikuje zpomalovací prahy v pořízených snímcích z kamery.

Algoritmus byl testován a laděn na záznamech pořízených z testovacího vozidla. Jeho funkčnost byla demonstrována automatickým zpomalováním testovacího vozu s aktivovaným tempomatem při přiblížení se k retardéru. Vozidlo dokáže včasně zpomalit na rychlost 18 kilometrů za hodinu a zajistit tak pohodlné přejetí přes zpomalovací práh.

Klíčová slova: detekce objektů, RANSAC, LiDAR, kamera, zpomalovací práh, point cloud, konvoluční neuronová síť, YOLOv8

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

In recent years, there has been a vast development of electronic devices in vehicles. Modern cars are equipped with sensors, cameras and radars that help monitor the area around the vehicle and provide the driver with much information.

One type of sensor is LiDAR (Light Distance and Ranging) which uses a laser beam to calculate the distances of objects in 3D space [1]. LiDAR data are collected in the point cloud, containing 3-dimensional points and other attributes such as reflectance. Furthermore, LiDAR provides precise length perception, allowing vehicles to accurately measure distances to objects and obstacles. This enhanced perception capability can improve the safety and reliability of vehicles in traffic.

The thesis discusses the topic of speed bump detection using a combination of LiDAR and a camera. The motivation came from the automotive industry, which would like to install LiDAR technology more frequently and combine its functionality with other sensors.

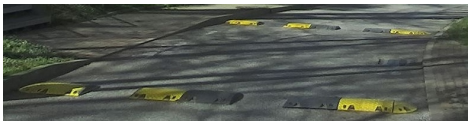
Figure 1.1: The test vehicle approaching a speed bump.

The addition of a new sensor may help with the elimination of the deficiency of current detection systems. Unlike the camera, LiDAR can operate effectively in low light conditions, including fog, cloudy weather, and even a night. On the other hand, in certain weather conditions, such as rain, LiDAR may encounter difficulties due to poor reflection of the laser beam from the road. A more robust and accurate understanding of the surrounding environment can be achieved by fusing the data from multiple sensors, enabling advanced features such as adaptive cruise control, lane-keeping assist and other detection systems.

1.2 Project objective

This thesis aims to design an algorithm that will run on the computer inside the vehicle and detect speed bumps on roads under various ambient conditions. Successful speed bump detection could be used to adjust the chassis or decelerate when driving with activated cruise control. Besides, information about speed bumps could be used to improve maps or navigation.

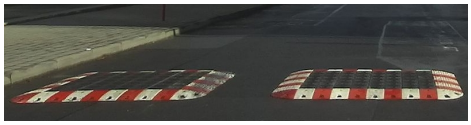
The solution of the thesis assignment has to reckon with the enormous differences between the appearance of various speed bumps. Some of them are colorfully highlighted or even contain drawn arrows. A few examples of speed bumps with drawn arrows are shown in Figures [1.2c, 1.2e, 1.2f, 1.2h, 1.2i]. Furthermore, some speed bumps are across the entire road, while others cover only some parts. A few examples of split speed bumps are in Figures [1.2a, 1.2c, 1.2d, 1.2f]. Some are made of different materials, such as interlocking pavement, as in Figures [1.2b 1.2g, 1.2i, 1.2j]. Moreover, some are barely visible even to humans, such as speed bumps shown in Figures [1.2b, 1.2j]. Some types of speed bumps, examples are shown in Figures [1.2a, 1.2k], are more easily visually damaged by crossing vehicles. All these mentioned types are shown in examples collected around Prague 6 in Figure 1.2.



(a) : Type 1.

(b) : Type 2.

(c) : Type 3.



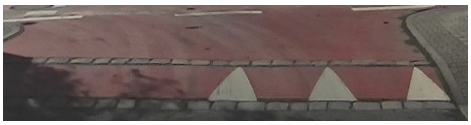
(d) : Type 4.

Figure 1.2: Different types of speed bumps.

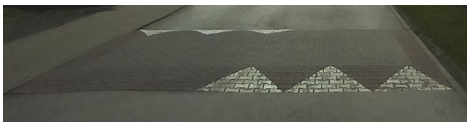
(e) : Type 5.

(f) : Type 6.

(g) : Type 7.



(h) : Type 8.



(i) : Type 9.

(j) : Type 10.

(k) : Type 11.

Figure 1.2: Different types of speed bumps.

The test vehicle is an upgraded Porsche Cayenne equipped with ZED 2 stereo camera, two Livox Horizons LiDAR front and rear and a powerful computer in the car's trunk.

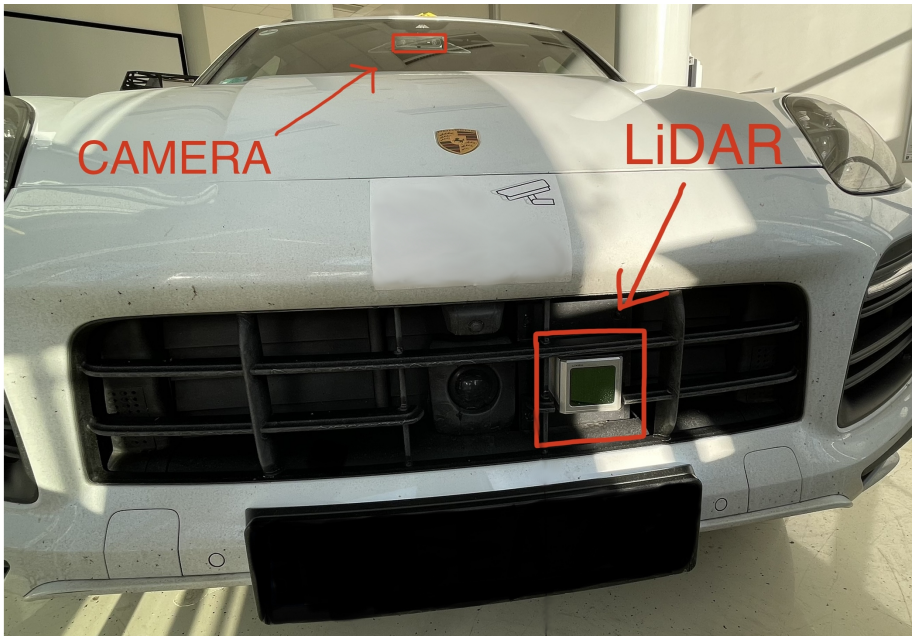


Figure 1.3: Livox Horizon LiDAR and ZED 2 stereo camera in the front of the test vehicle.

Figure 1.4: Computer in the trunk of the test vehicle.

The algorithm is deployed inside ROS1 (Robot Operating System) environment [2] and combines two distinct approaches. The first evaluates multiple consecutive scans acquired by LiDAR. The evaluation is based on a computation of a plane from the points in front of the vehicle and applying a robust fit of a computed plane to the points with subsequent calculation of their distance from it. The fitting itself is done using the RANSAC algorithm [3] implemented in PCL [4]. The second method uses a pre-trained convolutional neural network (CNN) to recognize prepared classes from the camera's image. Both approaches are independent of each other, and the provided combination secures functionality even when one of them malfunctions.

1.2.1 Outline

Other works dealing with detecting objects and speed bumps in the surroundings are presented and described in Chapter 2. The algorithm solution with all its components and improvements is explained more deeply in Chapter 3. An evaluation of the proposed solution can be found in Chapter 4. Furthermore, the summary of the thesis results can be found in Chapter 5.

Chapter 2

Related work

2.1 Object detection in images

Object detection is one of the most advanced computer vision tasks, recently achieving enormous popularity. The newly arriving object detection techniques are rapidly improving, leaving the traditional ones far behind.

One of the classic detection methods combines Haar-like features or Histogram of Oriented Gradients (HOG) [5] with classifiers like Support Vector Machine (SVM) [6]. The Haar-like features form by comparison of the sum of pixel intensities within adjoining rectangular regions in an image. The HOG feature descriptor creates a histogram representation of the image by extracting regional image gradient orientations and magnitudes. Then the SVM classifier is trained on these data to classify objects. In some cases, such as pedestrian detection, this combination achieves excellent results, but detecting more complex objects could be difficult based on the dependence of the features.

Another widely adopted object detection method utilizes a convolutional neural network (CNN). This technique can be divided into two groups, the two-stage detectors and the single-stage detectors. The development made vast progress after introducing the Regions with CNN (RCNN) [7] in 2014. It is one of the most famous two-stage detectors, which generates the Regions of Interest (ROI) by selective search [8] and then applies object classification for each ROI. The RCNN has significant drawbacks by the redundant computing of features, leading to an extremely slow detection speed. In 2015, R. Girshick, the author of the RCNN, introduced the Fast RCNN detector [9], integrating the advantages of RCNN and Spatial Pyramid Pooling Networks (SPPN) [10], which avoids the redundant computation of the convolutional features. The Fast RCNN increased the mean Average Precision (mAP) from 58.5% to 70.0% on VOC2007 [11] (PASCAL Visual Object Classes Challenge) dataset in comparison to RCNN, while the detection speed was 200 times faster. In the same year, the Faster RCNN [12], as the first near-realtime deep learning detector, was introduced. It presents the Region Proposal Network (RPN), enabling nearly cost-free region proposals. The Faster RCNN still contains some computation redundancy, and its improvements, such as Region-based Fully Convolutional Networks (RFCN) [13], have been later proposed.

Although the two-stage detectors made vast progress compared to their first proposals, the recently introduced single-stage detectors outperform them in real-time object detection.

The first single-stage detector, introduced by R. Joseph in 2015, was the You Only Look Once (YOLO) detector [14]. It was extremely fast compared to the two-stage detectors, and its enhanced version had only a slightly worse mAP result. As the concept of a single-stage detector indicates, it uses only a single neural network that predicts each bounding box based on the features from the entire image. Moreover, it simultaneously predicts all bounding boxes across all classes with their computed probabilities. The model's training is done on a loss function directly corresponding to detection performance. Despite the excellent results in detection speed, it had some drawbacks in localization accuracy compared with two-stage detectors. Further proposed versions of the YOLO detector pay more attention to solving this issue and push the state-of-the-art in real-time object detection.

2.2 Methods for speed bump detection using LiDAR

One of the generally used sensors in the computer vision industry, especially in robotics, is LiDAR. It scans a 3D space as a point cloud containing points with X, Y, and Z coordinates and additional attributes. Further process of the points representing the 3D space can lead to a robust object detector that detects objects such as cars, people and other obstacles. This section discusses the methods which can be applied to a point cloud to achieve a usable speed bump detector.

One of the possible methods to detect objects from point clouds is model fitting. It takes a set of points and tries to fit into them a defined mathematical model such as a line, plane, cylinder or sphere. In the case of object detection, the combination of planes and the SAmple Consensus (SAC) methods like Random Sample Consensus (RANSAC) algorithm [3] are usually chosen. The RANSAC algorithm is an iterative method for estimating the parameters of a mathematical model from provided points without the influence of outliers, first published in 1981 by M. Fischler and R. Bolles. However, it is a non-deterministic algorithm that produces a reasonable result only with a certain probability that can be increased with more iterations. This algorithm can be used for speed bump detection by fitting a plane representing a road to the obtained points and evaluating the outlier points considering properties such as a change in road pitch.

Another method that can be used for speed bump detection is the combination of the scan unfolding technique and the vision object detection architecture. It is used in publication [15] presenting an experimental study on projection-based semantic segmentation of LiDAR point clouds. The scan unfolding method projects the LiDAR scan into an image with reduced mutual point occlusions, minimizing the loss of information. Since the LiDAR scan

includes the point coordinates, the transformation to an image is correctly computed related to the view angle. Afterward, an object detection technique is applied to the image, resulting in a possible detection.

Another technique enabling the detection of objects from LiDAR scans is the direct processing of point clouds using convolutional neural networks. Methods using this technique must deal with the difficulty of the high computational cost of 3D CNNs since the computational complexity of 3D CNN grows cubically with the voxel resolution. Therefore, using a 2D detector with a bird's eye view (BEV) or front-view representations for object detection is significantly faster. Publication [16] takes advantage of this approach. Before downsampling the 3D LiDAR scan to 2D image data, it uses the sparse convolutional network, which is extremely fast because the convolution is computed only for nonzero elements, to extract information from the Z-axis of the 3D scan and then applies 2D CNN. This solution outperformed other state-of-art approaches since it can run in real-time and has accuracy almost like processing the original point cloud.

2.3 Speed bump detection as an automotive application

There are existing research studies proposing solutions for the speed bump detection task. Some use a stereo camera, others a combination of LiDAR and a camera.

Study [17] uses LiDAR and a camera and proposes the following detection method. First, the camera image is extracted into candidate areas through binarisation. At the same time, noise is removed using Gaussian and median filters. Then, the candidates are verified with a fusion of LiDAR and the camera. This verification is based on a histogram of oriented gradient (HOG) feature and support vector machine (SVM) classifier. The algorithm was compared with a solution using only one camera for detection. The comparison shows an increase in the algorithm's accuracy from approximately 78% to 85.4% at the cost of higher computation time, which increased from approximately 20 ms to 30 ms.

Another research study [18] uses deep learning techniques and a ZED stereo camera with two horizontally placed lenses that mimic dual human vision. The trained model was tested on marked and unmarked speed bumps with partially faded paintings. The detection accuracy was 97.44% with a false positive rate of 0.0427 for marked speed bumps and 93.83% with a false positive rate of 0.0909 for unmarked speed bumps. The distance towards speed bumps was estimated with an accuracy of ±20 cm in the 2-10 meters range. The limitation of this approach is that "the detection of a bump is poor at very low-light environments" [18].

The publication [19] uses a pre-trained convolutional neural network (CNN) and stereo vision for supervised automatic classification. It describes the proposed methodology to detect speed bumps using CNN without needing

their good marking. It can estimate a reconstruction of a 3D space from images of two cameras and make it an input for the supervised classifier to detect speed bumps by analyzing surface elevations.

Speed bumps are hugely different in terms of dimensions and shapes. Thus, more robust methods using CNN should perform better for visual speed bump detection than classifiers such as SVM. The proposed algorithms in discussed research studies were unsuitable for this assignment because of the need for a longer detection distance in every possible ambient condition. Furthermore, the methods applying neural networks directly to the obtained point clouds from LiDAR were considered unnecessarily complex, causing complicated fine-tuning and a need for labeled training data. Moreover, the research studies show that using only one sensor can be complicated and limited to a specific environment. This thesis proposes a solution combining an algorithm processing scans from LiDAR and a convolutional neural network classifying images taken from a camera.

Chapter 3
Proposed solution

This chapter will discuss the proposed solution, consisting of a fusion of two distinct approaches. The first mentioned method uses LiDAR, and a detailed explanation is in Section 3.1. The second approach uses a camera and pre-trained object detection classifier and is deeply explained in Section 3.2. Section 3.3 focuses on the fusion of these two assessments and discusses the advantages achieved by their combination.

3.1 LiDAR detector

This section will discuss an algorithm for speed bump detection using LiDAR. The baseline of the proposed solution is explained in Section 3.1.1, and its improvements made throughout the development are described in Section 3.1.2.

3.1.1 Baseline solution

The algorithm can be divided into several parts. First, the scanned points from LiDAR, which are more than 16 meters from the front of the vehicle, are removed due to their sparsity, which poses a problem for subsequent plane creation. Then a plane is fit to the points using the RANSAC (Random Sample Consensus) algorithm [3], leading to obtaining its coefficients. Further, points are assigned to 10 cm wide bins based on the distance from the vehicle. The following calculation of bin values is divided into three cases. For better clarity, notations are introduced in Table 3.1.

P	The set of all points obtained from LiDAR.
$point_i = (x_i, y_i, z_i) \in \mathbb{R}^3$,	where $point_i \in P,\ \forall i \in \{1, ..., \lvert P \rvert\}$.
$f : P \to ax + by + cz + d = 0$,	where $a, b, c, d \in \mathbb{R}$ are the coefficients of the plane, $x, y, z \in point_i$, $\forall point_i$ lying in the plane. (RANSAC function)
$dist : (point_i, a, b, c, d) \to \mathbb{R}$	The distance of $point_i$ from a plane with coefficients a, b, c, d.

$g : (point_i, a, b, c, d) \to \{0, 1\}.$

$$g(point_i, a, b, c, d) = \begin{cases} 1, & if\ point_i \text{ lies in a plane with coefficients } a, b, c, d, \\ 0, & otherwise. \end{cases}$$

$$number\_of\_outliers = |\{point_i \,|\, g(point_i, a, b, c, d) = 0, \forall point_i \in P\}|.$$

$$number\_of\_inliers = |\{point_i \,|\, g(point_i, a, b, c, d) = 1, \forall point_i \in P\}|.$$

Table 3.1: Notations for LiDAR detector.

First, if no outliers are assigned to the bin, the value of the bin is zero. Second, if the number of inliers assigned to the bin is zero:

$$bin\_val = \frac{20 \times \sum_i z_i}{number\_of\_outliers}. \tag{1}$$

In the last case, the calculation is as follows:

$$bin\_val = \frac{30 \times \sum_i z_i}{number\_of\_outliers\ +\ number\_of\_inliers}. \tag{2}$$

A bin is marked as possible speed bump detection if its value is greater than or equal to 1. After this calculation, the detection itself is performed by a state machine shown in Figure 3.1. The states (values) are: No detection (0) - initial state, Unknown (1), Probable (2) and Detected (3). When a speed bump is detected in the bin, the corresponding state machine updates its value (state) by increasing it by two (the maximum is 3). Otherwise, the value is decremented by one (minimum is 0). Then, after each frame, a value (state) of 2 is propagated from the farthest bin from the vehicle with a successful speed bump detection to the following closer bins with a value (state) less than 2. The algorithm detects the speed bump in bins that are in state Detected. Moreover, the algorithm can be described in Algorithm 1.

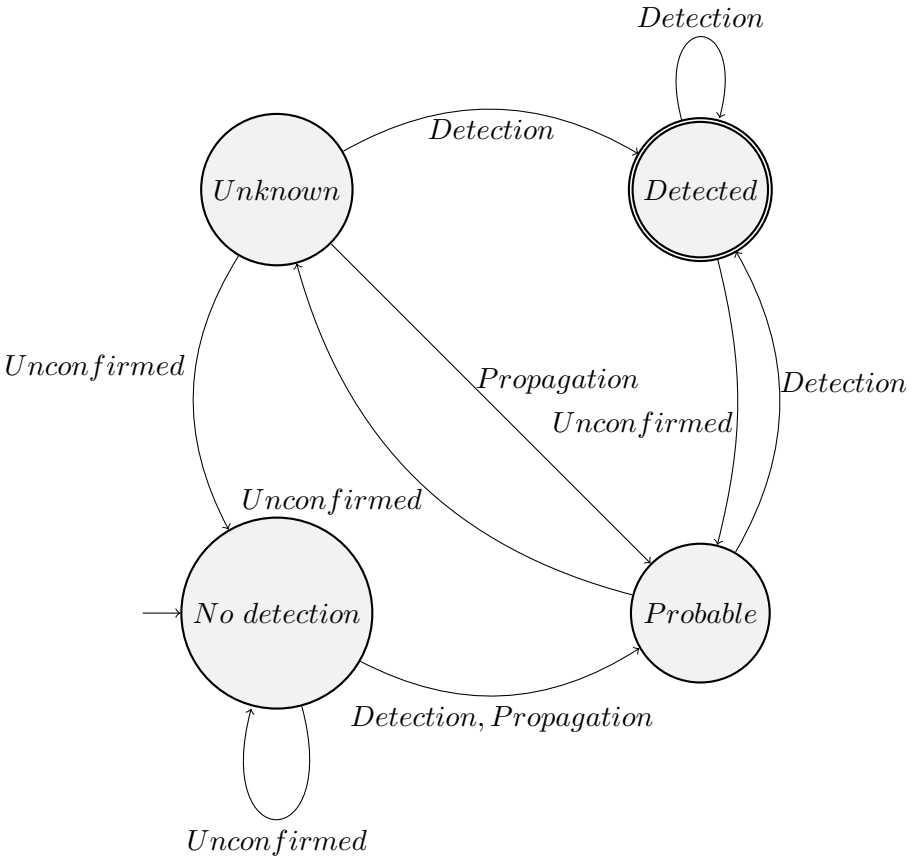


Figure 3.1: Non-deterministic Finite Automata (NFA) diagram of the detection state machine.

Algorithm 1 Pseudocode of the baseline of the algorithm.

for `each frame captured by LiDAR` **do**
  $points \leftarrow$ all points from the frame which are closer than 16m
  $indices\_of\_inliers, coefficients \leftarrow$ RANSAC($points$)
  $bins \leftarrow [0] \times number\_of\_bins$
  for `each` bin `in` $bins$ **do**
    if $number\_of\_outliers$ in bin is 0 **then**
      continue
    else if $number\_of\_inliers$ in bin is 0 **then**
      $bin \leftarrow \dfrac{20 \times \sum_i z_i}{number\_of\_outliers}$
    else
      $bin \leftarrow \dfrac{30 \times \sum_i z_i}{number\_of\_outliers \ + \ number\_of\_inliers}$
    end if
  end for
  STATE_MACHINE.$update(bins) \rightarrow$ possible DETECTION
end for

The proposed detection method has issues reporting false positive detection of sidewalks or diagonal roads with slightly different heights. Thus, the points cut out before the vehicle are narrowed into a cone form which slightly suppresses the false positive detection when the car turns on a narrow road. However, this false positive detection is not trouble from an application perspective. Since it occurs when the vehicle speed is low, when the vehicle is turning or entering an intersection, and the entire algorithm is stopped.

Three examples of detecting various speed bumps are visualized in Figures 3.2–3.4. All points scanned by LiDAR are colored in black, and if they lie in the space from which the plane is computed, they also contain blue, yellow, or red. The points in the calculated plane are colored blue, and the ones representing speed bump detection are highlighted in red. Otherwise, they are colored yellow.

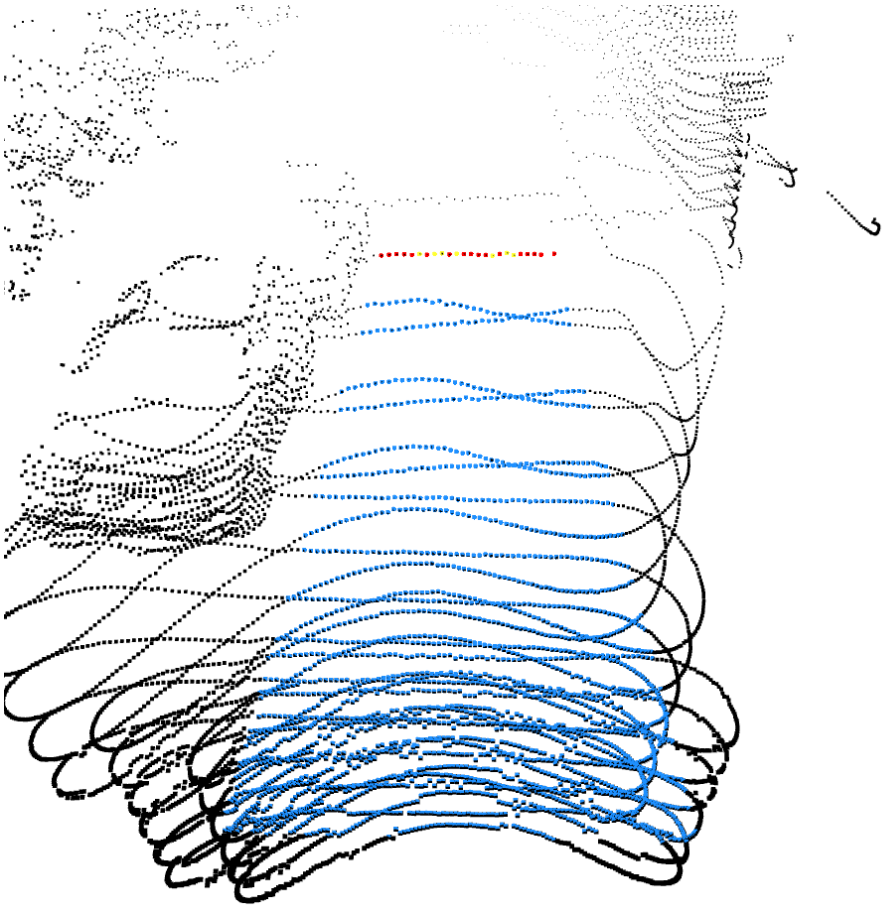


Figure 3.2: The first example of LiDAR detection.

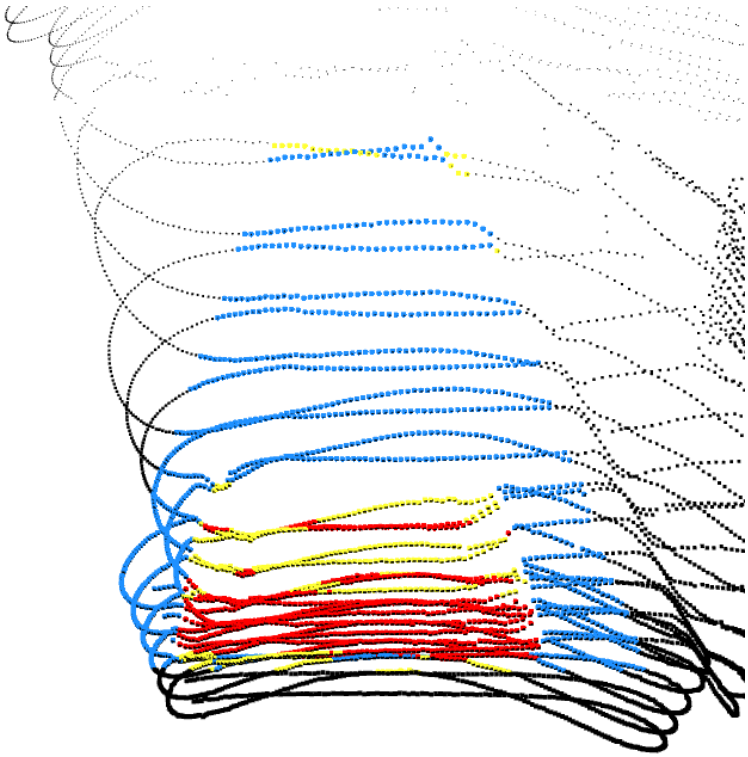


Figure 3.3: The second example of LiDAR detection.

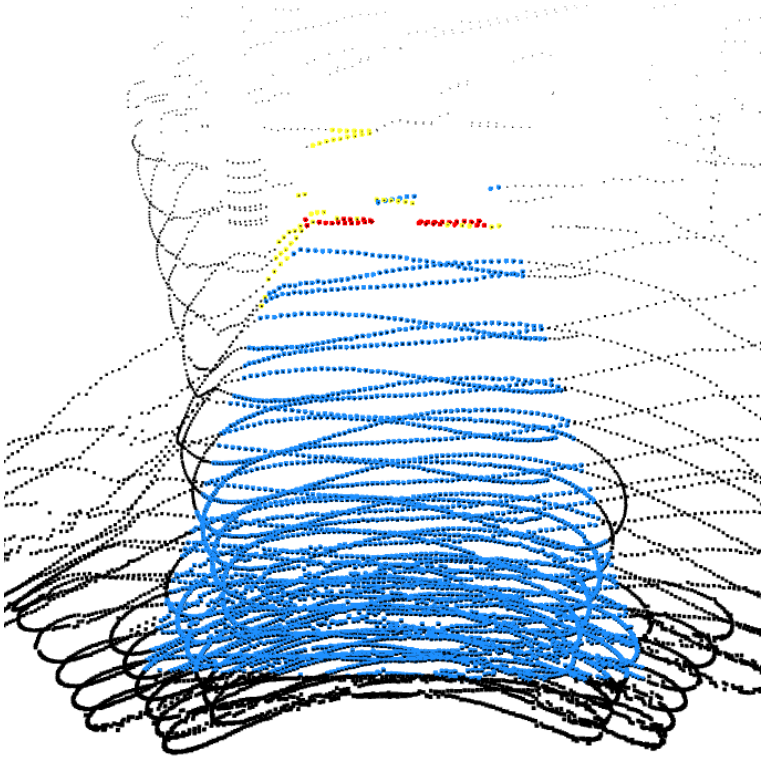


Figure 3.4: The third example of LiDAR detection.

3.1.2 Solution improvements

This section describes all improvements made to the algorithm during development, leading to a usable detector for speed bump detection. Some were done based on the thoughts from testing on recordings, and others were initiated after testing in the test vehicle.

3.1.2.1 Filtering - enhanced

During development, it was found that the result of the algorithm was inaccurate while calculating the plane for points distant more than 13 meters from the front of the vehicle. For this reason, the deletion of points at the beginning of the algorithm was adjusted.

Another problem with the first version of the algorithm was detecting objects such as vehicles or people in front of the car. The key to a more robust algorithm was the addition of a filter based on calculating the distances of the points from the plane using the obtained plane coefficients. The distance is calculated as:

$$dist(point_i, a, b, c, d) = \frac{|a \times x_i + b \times y_i + c \times z_i + d|}{\sqrt{a^2 + b^2 + c^2}}. \tag{3}$$

This filter is used before assigning the calculated value to the bin, and the algorithm is described in Algorithm 2.

Algorithm 2 Pseudocode of the improved version of the algorithm.

for `each frame captured by LiDAR` **do**
  $points \leftarrow$ all points from the frame which are closer than 13m
  $indices\_of\_inliers, coefficients \leftarrow$ RANSAC($points$)
  $bins \leftarrow [0] \times number\_of\_bins$
  for `each` bin `in` $bins$ **do**
    if $number\_of\_outliers$ in bin is 0 **then**
      continue
    else if $number\_of\_inliers$ in bin is 0 **then**
      $bin \leftarrow \frac{20 \times \sum_i dist(point_i, a, b, c, d)}{number\_of\_outliers}$
    else
      $value \leftarrow \frac{30 \times \sum_i dist(points_i, a, b, c, d)}{number\_of\_outliers \ + \ number\_of\_inliers}$
      $bin \leftarrow$ FILTER($value$)
    end if
  end for
  STATE_MACHINE.$update(bins) \rightarrow$ possible DETECTION
end for

3.1.2.2 Combining multiple scans

The proposed approach's most significant limitation was the far points' low density which can be considered a property of Livox Horizon LiDAR. This constraint was bypassed by combining multiple scans with relevant transformations of points. The stacking of scans consists of computation of the required LiDAR-Inertial odometry with FasterLIO [20] and the subsequent join of scans using ROS 1 package laser_assembler. The areas with high point density in the final scan are subsampled with the VoxelGrid filter. This particular improvement was not a part of the thesis assignment, and the work advisor provided the solution for the computation implemented in C++.

An example of a scan acquired from 50 ms lasting exposition from Livox Horizon LiDAR with colorful information about intensity is in Figure 3.5.

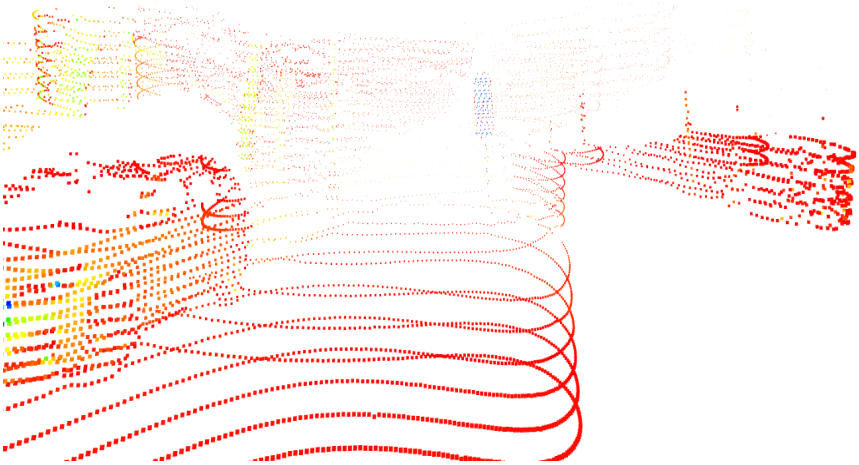


Figure 3.5: A single scan acquired from Livox Horizon LiDAR.

For the comparison, the combination of 10 subsequent LiDAR scans is shown in Figure 3.6. The color represents the intensity of the collected points.

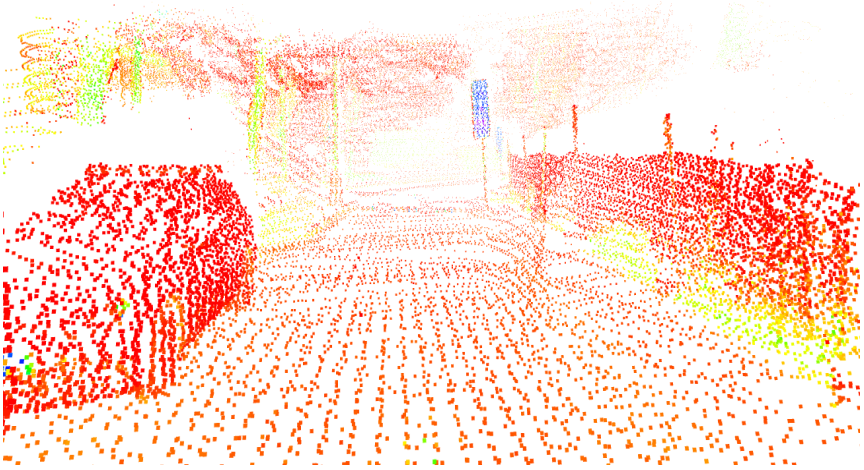


Figure 3.6: A combination of 10 scans acquired from Livox Horizon LiDAR with the application of a VoxelGrid filter.

This point cloud modification allowed the extension of the computed plane to 18 meters from the front of the vehicle. On the other hand, this adjustment of the plane's length caused a false positive detection in case of a change in road height. Therefore, the computation of the whole plane was divided into three parts. The same algorithm using the RANSAC function with many parameters is used in each but on a different overlapping section of the plane. Parameters such as a distance threshold, the maximum number of iterations or the model type of plane, where the parallel plane with a defined maximum angle of 15 degrees from a horizontal plane was selected, were set. The sections of the plane are as follows, starting with a range of 3 to 10 meters, then a range of 7 to 14 meters and 11 to 18 meters from the front of the vehicle.

After this adjustment, the algorithm was able to detect a speed bump at a distance longer than 17 meters. However, during testing, it was found that immediately after a significant change in acceleration in z axis occurred after crossing over a speed bump, the combination of LiDAR scans was inaccurate, and the algorithm reported a false positive detection. Thus it was necessary to filter scans with a low percentage representation of points lying in the computed plane, which decreased the distance of speed bump detection to a range of 12 to 17 meters, depending on the type of speed bump.

Figures 3.7–3.9 show the visualizations of speed bump detection on different speed bumps using the proposed algorithm applied to the combination of ten subsequent scans acquired by LiDAR. All collected points are colored in black, and the computed plane is colorfully highlighted. Points in the computed plane are blue, and the plane's outliers are colored yellow or red, where the red color denotes the speed bump.

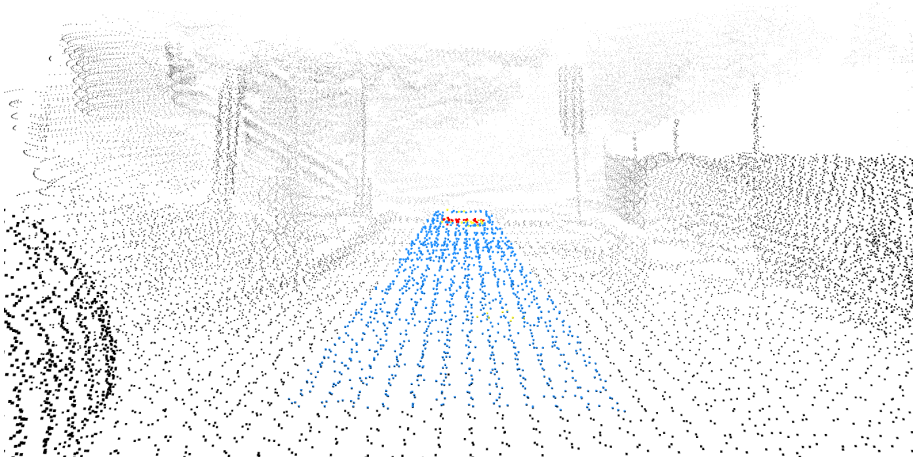


Figure 3.7: The first example of detection from a combination of LiDAR scans.

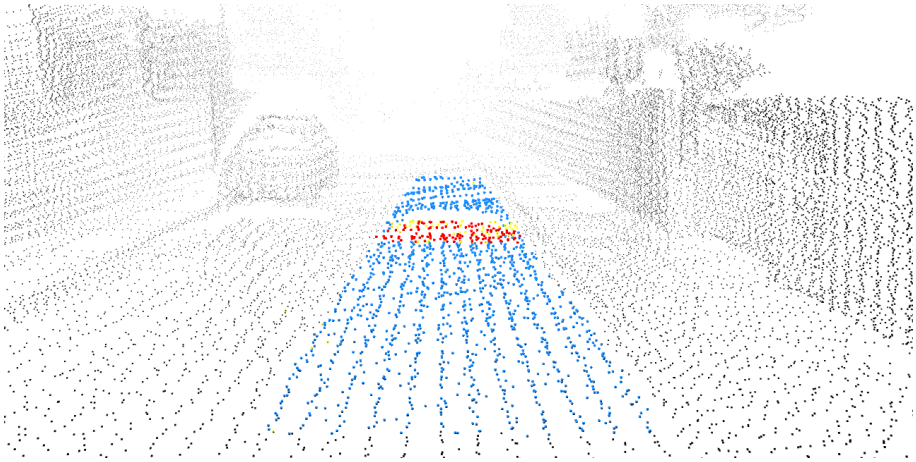


Figure 3.8: The second example of detection from a combination of LiDAR scans.

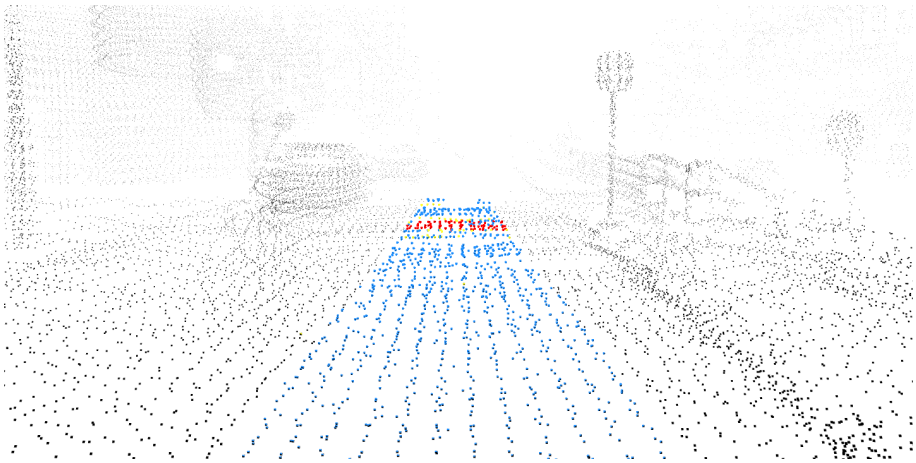


Figure 3.9: The third example of detection from a combination of LiDAR scans.

3.1.2.3 Algorithm speed up

During the first testing in the vehicle, the low speed of the algorithm was found, and the slowest part, the state machine, was removed. The functionality was replaced with a different algorithm implemented in the ROS node, deciding whether detection occurred. The algorithm's running time varies from 10 to 70 ms, depending on the convergence speed of the RANSAC algorithm. The LiDAR frequency is set to 20 Hz. Due to that, it is necessary to process scans every 50 ms and hence, the algorithm was remade to the multiprocess application where each scan is processed in a different thread. The time synchronization of scan processing is handled in the previously mentioned ROS node, discussed in Section 3.3.1.

3.1.2.4 Accuracy adjustment

The proposed algorithm reported false positive detection of cars despite using a filter on the distance of points. Specifically, the algorithm detected the car's back wheels, which appeared as an obstacle or possible speed bump. Therefore, another check on the distance from the plane was added, which removed the false positive detection. When an object is higher than 20 centimeters, other more distant points from the front of the vehicle are not considered for detection.

Another adjustment was in the distance calculation from the computed plane, where the absolute value was deleted. Only objects in the same half-plane as a vehicle are considered for detection, and this half-plane is found according to the signum value after the insertion of point [0, 0, 10] to the plane. The distance without absolute value is calculated as:

$$dist\_raw(point_i, a, b, c, d) = \frac{a \times x_i + b \times y_i + c \times z_i + d}{\sqrt{a^2 + b^2 + c^2}}. \quad (4)$$

And then, the final distance from the plane is calculated as:

$$dist(point_i, a, b, c, d) = sgn(10 \times c + d) \times dist\_raw(point_i, a, b, c, d). \quad (5)$$

3.2 Camera detector

A camera was the second sensor used to create a more robust speed bump detector and complement the LiDAR. The ZED 2 stereo camera was installed in the test vehicle for these and similar purposes, and its left lens was utilized for training and detection.

During the collection and preparation of the data from the test vehicle, it was decided to collect not only speed bumps but also traffic signs, which are much earlier visible. Some speed bumps are hard to recognize from the cockpit of a moving vehicle, even for humans. Usually, they are highlighted with a speed bump sign next to them or even a speed bump warning sign far ahead. Furthermore, detecting the speed bump sign placed before the speed bump provides a more fluent and comfortable slow down and crossing of the speed bump. If the speed bump sign is missing, the driver is usually forced to slow down based on the road's surroundings, or the speed bump is visible enough. The speed bump sign and the speed bump warning sign are shown in Figure 3.10.

Figure 3.10: The speed bump sign and the speed bump warning sign.

3.2.1 Data preparation

The research for an available public dataset among datasets for semantic segmentation such as SemanticKITTI [21], KITTI-360 [22] or SemanticPOSS [23] was done. There was not found any suitable dataset for training purposes for the speed bump classifier since none contained the speed bump class. Therefore, collecting data with the test vehicle and making a new dataset was necessary. The collection was done multiple times in different weather and various light conditions. After each, the images were extracted from recorded rosbags and uploaded to a CVAT annotation tool server [24]. For annotation, three classes were prepared: speed bump, speed bump sign and speed bump warning sign. Then the images were manually annotated, and it was possible to train an object classifier from them. An example of annotation is in Figure 3.11.

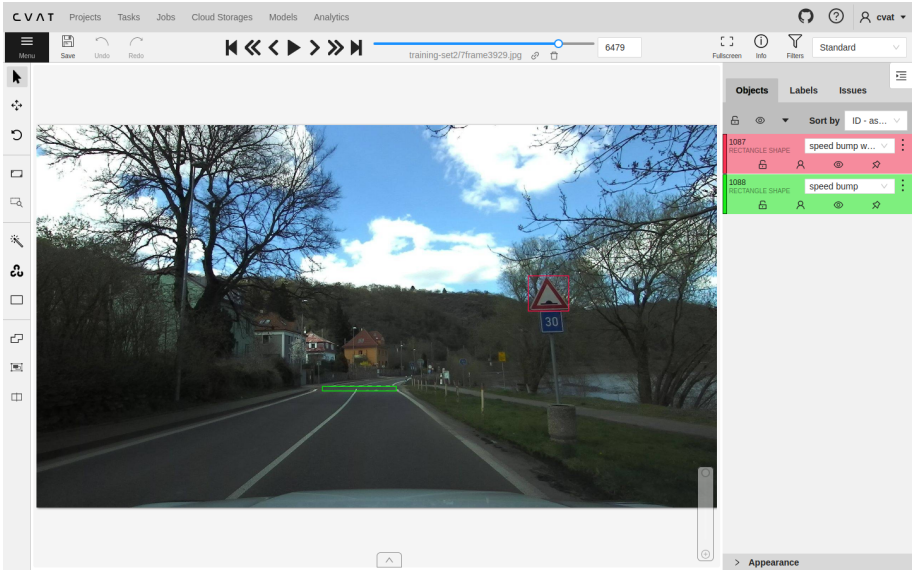


Figure 3.11: An example of annotation in CVAT environment.

After several collections and improvements of the trained classifier model, the auto-annotation feature was used four times to speed up the labeling

process by letting the pre-trained model auto-annotate the new images. The annotation was fast and efficient since the CVAT server had access to a graphics processing unit (GPU).

The camera in the test vehicle used for image collection was set to record at 10hz. It collected more than 50 000 relevant images over the several hours of driving, including people, cars, speed bumps, signs and other traffic elements. Only the meaningful ones for the speed bump detection assignment were selected for annotation. Together it was annotated more than 37 000 images, and the collections took place in Prague and its surroundings, at Dejvice, Nové Butovice, Prague-Suchdol and many others.

3.2.2 Baseline solution via SVM and HOG

One of the mentioned techniques in [25] and used method in [17] combines the histogram of oriented gradient (HOG) feature [5] with a support vector machine (SVM) classifier [6].

Before a dataset of speed bumps was created, the method mentioned above was tried on the dataset of cars. The classifier was trained and employed using the dlib library [26], and the training was performed on the Cars Dataset [27]. An example of car detection with an SVM classifier is in Figure 3.12.

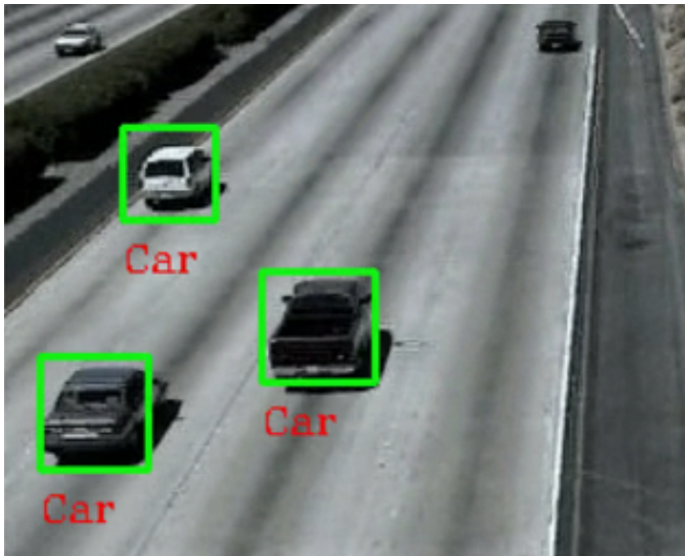


Figure 3.12: An example of car detection using an SVM classifier.

The SVM classifier was performing well in the case of car detection. However, the annotated labels on the training images had to have the same shape, which would cause trouble when applying to a speed bump dataset with a considerable difference between individual speed bumps. Therefore, for speed bump detection, the mentioned solution was abandoned and replaced with the convolutional neural network discussed in Section 3.2.3.

3.2.3 Solution using convolutional neural network

Another mentioned technique for object detection in [25] is the You Only Look Once (YOLO) detector. It is a fast single-stage detector that applies a single neural network to the full image. The first versions of these detectors had difficulties with incorrect localization accuracy, especially for small objects [14]. The most recent releases of these detectors have paid more attention to these issues, resulting in outperforming a significant majority of existing object detectors concerning speed and accuracy.

For the case of speed bump detection, the YOLOv8 [28] was used, released in January 2023 by Ultralytics. The YOLOv8 has five versions with different numbers of parameters. The two smallest are YOLOv8n, with 3.2 million parameters, and YOLOv8s, with 11.2 million parameters. Due to the need for fast detection, the smaller YOLOv8n with the best inference time from all versions was selected. This model is also the most manageable to fine-tune based on the number of parameters.

It was fine-tuned entirely on custom object classes and camera parameters to detect speed bumps and signs mentioned in Section 3.2.1. The training was done several times, each time with a different dataset with the addition of newly collected data. Moreover, the model's training in the prepared pipeline contains additional data augmentation. The last version of the custom YOLO model was deployed and used in the test vehicle.

3.3 Fusion of detectors

The proposed solution of the fusion of LiDAR and the camera secures independence between sensors, thus improving the functionality of detection in all ambient conditions. Due to an overexposed or too-dark image from the camera, the camera detector may not work correctly, and the detection must be substituted fully by the LiDAR detector. On the other hand, in the case of a low number of points scanned by a LiDAR, sometimes occurring in the rain, the LiDAR detector can compute fewer planes or none of them. This behavior is probably caused by a high layer of water on the road causing poor reflection of the laser beam back to the LiDAR. For such cases, the detectors could have assigned confidence influencing how they impact the detection. For example, when the rain-light sensor reports rain, the confidence of the LiDAR detector can be reduced, and the detection will depend more on the camera detectors. A few examples of LiDAR scans taken from the test vehicle in rainy weather are shown in Figures 3.13 and 3.14.

Figure 3.13: An example of visualization of a LiDAR scan with a missing road plane in rainy weather.

Figure 3.14: An example of visualization of a LiDAR scan with very few points in front of the vehicle.

Another possible solution is a direct fusion of points from LiDAR to the coordinate system of a camera or vice versa. However, for automotive applications, it has a disadvantage with the calibration of this transformation caused by car shaking and movement. The calibration would have to be automatic and monitored to be usable. Moreover, the independence between detectors brings an advantage for the manufacturer in the possibility of having more independent vendors and thus being less dependent on only one supplier.

The suggested fusion of LiDAR and the camera is achieved using four ROS nodes. One deal with LiDAR detection, two with camera detection of speed bumps and speed bump signs, and the last with a final combination of these nodes. The diagram representing the fusion of the sensors is in Figure 3.15, and the functionality is demonstrated by slowing the test vehicle based on

the sent target speed, discussed in Chapter 4.

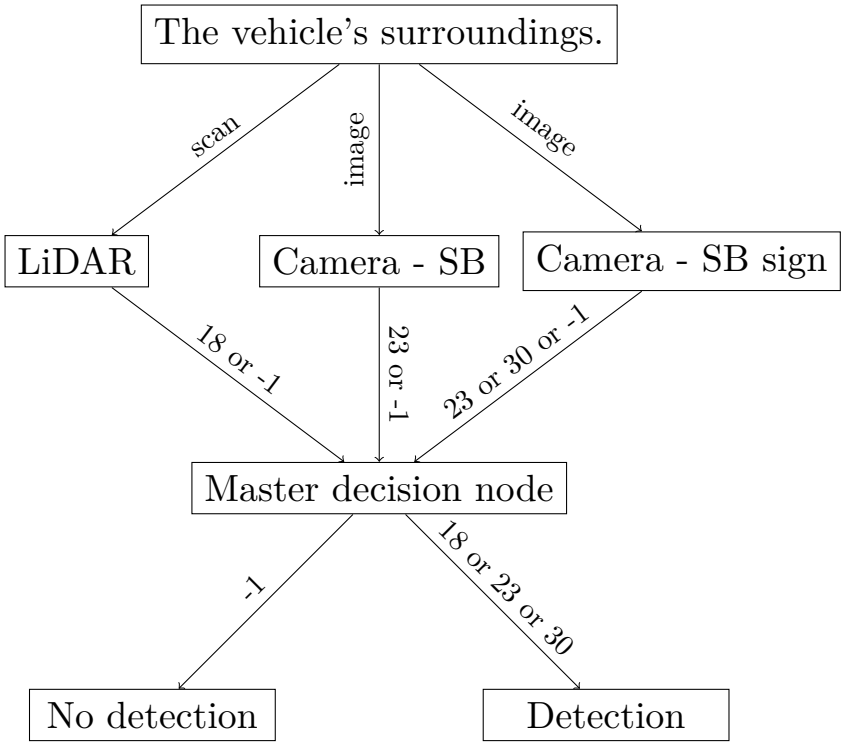


Figure 3.15: The diagram of the fusion of detectors.

3.3.1 LiDAR detection

The ROS node securing a detection using LiDAR decides based on ROS messages sent by an algorithm processing scans. These messages contain the timestamp and possibly the distance of the detected speed bump. The computation also takes into account a possible scattering of messages based on their timestamps caused by a different processing time of scans.

The detection timestamp is stored after receiving a message with the speed bump information. Saved timestamps older than 1.5 seconds from the last received timestamp are removed. Furthermore, the detection is decided based on the computed value from the division of 80% quantile, calculated from the subtraction of the current timestamp and saved timestamps, and the number of stored timestamps. This whole computation executes if the number of saved timestamps is at least three. If a successful detection occurs, the algorithm pauses for two seconds and waits on timeout.

When changing from the state without detection to a state with it, the ROS node sends the ROS message containing a target speed value set to 18 to the master decision node. This value was chosen after several tests and is considered sufficiently comfortable for crossing over a speed bump. In the case of detection termination, the target speed value in the message is replaced with the value -1.

3.3.2 Camera classification

The camera detector performs detection on three classes with the confidence threshold set, based on the performance of the object detection model and testing in the test vehicle, to 0.5 for each. According to the reliability of these classes, the handling ROS node was divided into two nodes. The first providing the speed bump signs detection is less strict and is similar for both speed bump signs and speed bump warning signs, only with different parameters. The second ensuring detection of speed bumps includes additional verification.

The detection of signs is accepted if the expected sign is recognized in two consecutive frames. The parameters for both signs were tuned after testing in the test vehicle and regarding their usual position on the road. The speed bump sign is commonly placed next to the speed bump, and the warning sign is typically between 50 to 100 meters before the speed bump.

The target speed of the warning sign detection is 30 kilometers per hour, and the detection timeout is set to 6 seconds. In the case of a speed bump sign, the speed is established to 23 kilometers per hour and the timeout to 4 seconds.

Since the reliability of speed bump detection is worse than at speed bump signs, the detection is accepted when the speed bump is recognized in four consecutive frames. Other properties as the target speed and the timeout, are the same as for the speed bump sign. All classes' values and parameters were selected after testing in the test vehicle to ensure the most comfortable deceleration and crossing over a speed bump.

3.3.3 Master decision node

The master decision node provides a combination of all independent detectors. It chooses the minimum value from all requested target speeds. The value -1 is considered an infinity value in the comparison.

The functionality of the speed bump detection is demonstrated by slowing the approaching test vehicle based on the reprogrammed cruise control. In the case of a change in the minimum requested target speeds of detectors, the ROS node publishes the new value to the ROS topic from which the cruise control reads. The whole algorithm is started with the activation of cruise control, which maintains the vehicle's speed and slows down on request. If the detection of all detectors terminates, the master decision node publishes -1, and the cruise control accelerates the vehicle back to the previous set speed.

The individual target speeds of each detector were set according to the expected braking process. The relation between velocity and traveled distance when decelerating using the cruise control was calculated for this estimation by the advisor of this work. The calculation reckons with delays of all computations, including delay of the signal from cruise control to brakes, and are shown in Figures 3.16 and 3.17. System delay was lower than expected when tested in a test vehicle, so the estimated deceleration curves are slightly pessimistic.

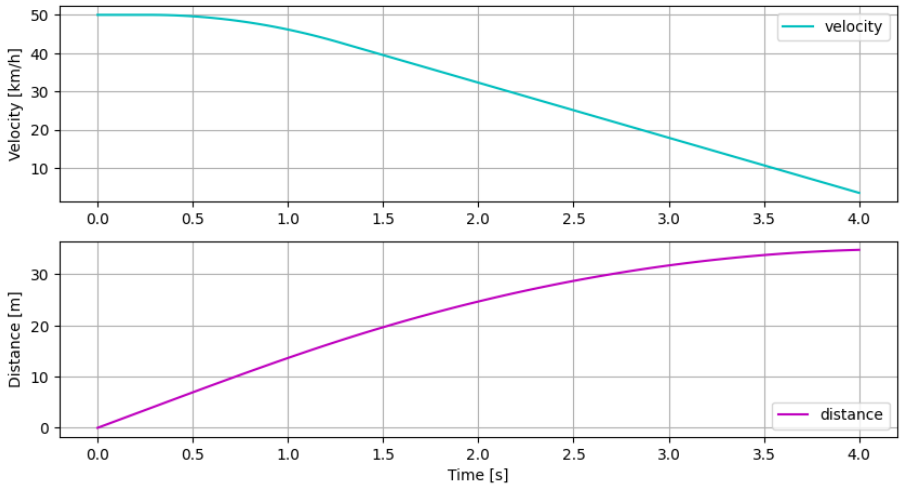


Figure 3.16: The calculated relationship between velocity and traveled distance when decelerating from 50 kilometers per hour.

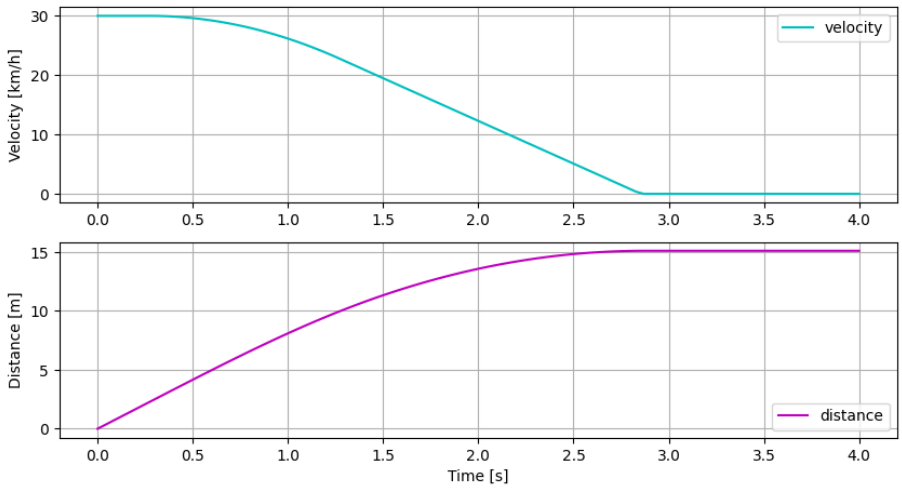


Figure 3.17: The calculated relationship between velocity and traveled distance when decelerating from 30 kilometers per hour.

The expected braking process of the vehicle will start with detecting a warning sign far before the speed bump and begin decelerating to 30 kilometers per hour. At this velocity, the detectors can detect a speed bump at enough distance for the vehicle to decelerate to 18 kilometers per hour in front of the speed bump. Which was found to be a comfortable speed for crossing over a speed bump, and it was set as the target speed for the LiDAR detector. The target speed of the camera speed bump detector and the speed bump sign detector was set to 23 kilometers per hour based on their worse reliability than the LiDAR detector.

Furthermore, the vehicle's driver is provided with the image output of a pre-trained detection model and processed points collected with LiDAR with the addition of computed planes with colorfully highlighted detection.

The image output from the YOLOv8 model contains detection labels with class descriptions and the detection's confidence. Moreover, the visualization includes the target speed of all detectors and the more noticeable requested speed chosen by the master decision node. An example of detection visualization is shown in Figure 3.18, where the camera detectors detect a speed bump sign with a confidence of 75.3% and a speed bump with a confidence of 63.9%. The LiDAR detector also successfully detects a speed bump in the visualization highlighted with red points. The camera detectors request a speed of 23 kilometers per hour, and the LiDAR detector requests a speed of 18 kilometers per hour which is also the minimum speed selected by the master decision node.

Figure 3.18: The visualization of the detection.

Chapter 4

Evaluation

This chapter discusses the functionality and processing quality of the proposed solution. Section 4.1 describes the LiDAR detector's functionality based on the test vehicle recordings and the camera detector's performance evaluated on the validation dataset. The evaluation of quality, reliability and earliness of the proposed fusion of detectors is described in Section 4.2.

4.1 Detectors performance

Before deploying the LiDAR detector into the test vehicle, it was tested and tuned on captured recordings. The test vehicle collected dozens of recordings containing many speed bumps of various types. Based on the tests, the distance in which the LiDAR detector detects a speed bump is around 12 to 17 meters. The exact distance depends on the type of speed bump, especially on its height and width. The performance of the improved version of the algorithm, which uses a combination of 10 scans, was compared to the previous single-scan version. The comparison of algorithms was evaluated on the recordings containing different types of speed bumps. It is presented in Table 4.1 and shows that the multi-scan algorithm performed better regarding the detection distance in all tests. The presented speed bump types refer to the types introduced in Figure 1.2. Moreover, it was found that the improved version of the algorithm is much more robust to a change in road slope or potholes than the single-scan algorithm.

Type of speed bump	Single-scan detector $[m]$	Multi-scan detector $[m]$
Type 1	12.5	17.1
Type 3	11.1	16.8
Type 5	10.9	14.3
Type 6	12.6	14.4
Type 7	11.5	12.7
Type 9	11.4	13.1

Table 4.1: The comparison of detection distance of the single-scan algorithm and the algorithm combining 10 scans.

During the testing, one disadvantage of the proposed method of the LiDAR detector was found. The detector reports a false positive detection of sidewalks, misinterpreting them as speed bumps. However, this attribute of the algorithm does not matter from the application's point of view since it is reported when the vehicle has low speed and the whole algorithm is stopped.

The custom YOLOv8 model was evaluated on the validation dataset to get its performance. The validation dataset was built from images collected by the test vehicle. The collected data were divided into the training and validation datasets so that sets are distinct and contain different places and road situations. The images in the validation dataset were manually annotated regardless of the detector capabilities, which caused slight inaccuracy in the computed results. It was measured in the test vehicle that the tuned custom model can detect speed bumps and signs at a distance of fewer than 18.5 meters. However, the validation dataset contains images with labeled classes even from a longer distance than that, which worsens the numerical results and makes the entire performance estimation pessimistic.

The performance of the object detection model is computed based on the concept of Intersection over Union (IoU) of two bounding boxes, the annotated ground truth and the prediction. The IoU threshold value of the computation was set to 0.5, the default value. The prediction is marked as True Positive (TP) if the value of IoU is higher than its threshold. Otherwise, it is taken as False Positive (FP). If there is no prediction on the ground truth of the image containing searched object, it is marked as False Negative (FN). The Precision of the detection model is computed as $\frac{TP}{TP+FP}$ and the Recall as $\frac{TP}{TP+FN}$. The Precision-Confidence Curve is shown in Figure 4.1 and the Recall-Confidence Curve in Figure 4.2. The unsuitably chosen validation dataset significantly affects the result of the Recall-Confidence Curve.

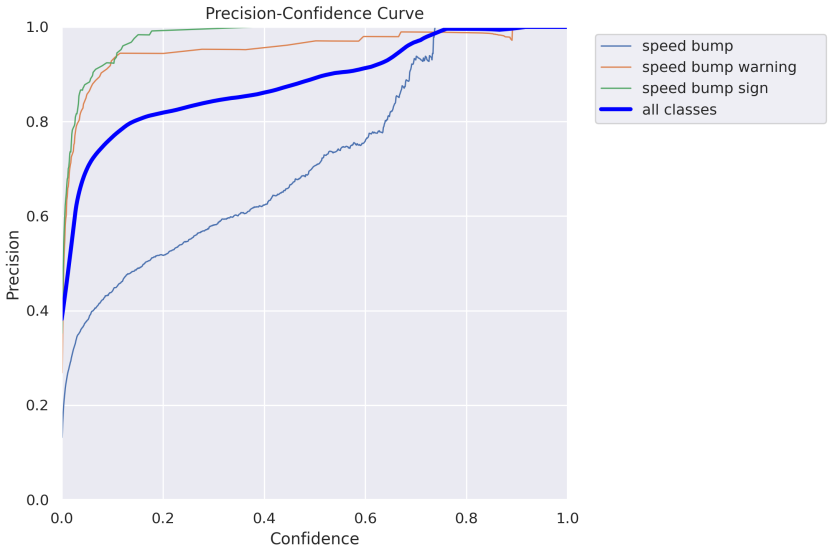


Figure 4.1: Precision-Confidence curve of the custom YOLOv8 model.

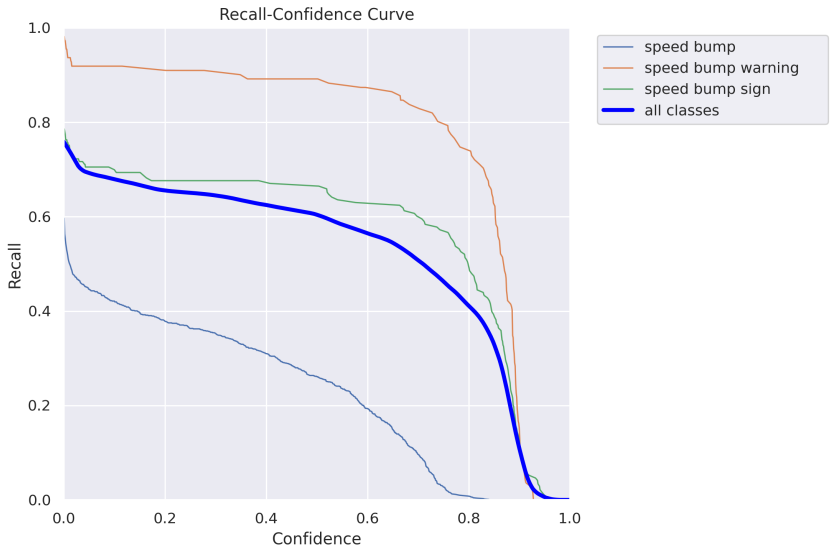


Figure 4.2: Recall-Confidence curve of the custom YOLOv8 model.

Based on the estimated performance and conducted testing in the test vehicle, the confidence threshold of detections was set to 0.5 for each class. The main focus of tuning was to increase the detection distance at the cost of worse accuracy, which is further handled in the ROS nodes. The performance of the custom YOLOv8 model was compared to the mentioned studies in 2.3. The third of these works only presented the methodological proposal for speed bump automatic detection without evaluating the functionality. Thus only the first two works [17], [18] were compared regarding Precision, Recall and the maximum detection range. All values used for the comparison are average values achieved by individual solutions. The performance of the custom YOLOv8 model was calculated for each class, and the comparison is shown in Table 4.2. The Recall values of the custom YOLOv8 model are highly affected by the unsuitably chosen validation dataset.

	Detection performance		
	Precision [%]	Recall [%]	Distance $[m]$
Yun H.-S. et al., (2019)	85.4	92.6	< 8
Varma V.S.K.P. et al., (2018)	96	98	< 14
Custom YOLOv8 model (speed bump)	70	27	< 18.2
Custom YOLOv8 model (speed bump sign)	99	67	< 18.5
Custom YOLOv8 model (warning sign)	96	89	< 18.5

Table 4.2: The comparison of the YOLOv8 model performance with related research solutions.

The comparison shows that the related studies focus more and are better in the detection accuracy but have fewer detection capabilities regarding distance. In addition, the proposed solution also uses the LiDAR detector that can detect speed bumps at around 12 to 17 meters and is much more reliable than the camera detectors. Moreover, the Recall value of the custom YOLOv8 model is far higher when capturing the objects from a closer distance, and the Precision is improved by adding a filter for more subsequent detections, reckoning with the difference in reliability between traffic sign detections and speed bump detections.

4.2 Experimental evaluation

The testing and the evaluation of the proposed algorithm were performed in the traffic under various weather conditions using the test vehicle. During testing, a few complications that made the tuning and the evaluation of the proposed solution more difficult were encountered. For example, the battery in the test vehicle broke down, and the vehicle was disabled to run for a few days. Furthermore, there was an issue with the computer in the vehicle, which started to restart based on a broken connection between the graphics card and a motherboard, damaged by car vibrations and movement since the components of the computer were not made to be in a car. However, these issues were resolved, and the algorithm functionality was tested and evaluated.

The algorithm's functionality and earliness were demonstrated by decelerating the test vehicle using the reprogrammed cruise control set by the requests of the algorithm. The maximum distance from which the LiDAR detector can detect the speed bump was measured at 12 to 17 meters, similar to the values computed from the recordings. Furthermore, its property of false positive detection of sidewalks found by testing on recordings was confirmed. Camera detectors can detect speed bumps and speed bump signs from around 18 meters. The installed camera in the test vehicle highly limits them, as it covers an improperly wide angle for detection tasks. Thus, every object in the captured image appears further away and smaller than it is in reality.

Furthermore, some limitations of the proposed solution were found. In rainy weather, the LiDAR sensor sometimes has difficulties scanning points for the LiDAR detector caused by a high layer of water on the road. In these situations, camera detectors fully cover detection, making the process less reliable. Further, in the case of an overexposed or too-dark image captured from the camera, the camera detector cannot work correctly, and only the LiDAR detector can be used, resulting in lower earliness of the detection.

However, when the weather conditions allow the use of all detectors, the proposed algorithm can slow the vehicle from the speed limit of the areas where the speed bumps are located to a comfortable crossing speed before reaching the speed bump. The comfortable speed for crossing over a speed bump was determined after tests in a test vehicle to 18 kilometers per hour. Four independent observers evaluated the functionality of the speed bump

detection algorithm. They stated that the car slowed down nicely and sufficiently before the speed bump and accelerated again after the crossing.

The vehicle's deceleration consists of several parts depending on the type of speed bump and traffic signs. It usually starts by detecting the warning sign and deceleration to 30 kilometers per hour. This situation, when the car starts slowing based on the request of the camera detector, is shown in Figure 4.3.

Figure 4.3: Speed limited by detecting speed bump warning sign by the camera detector.

Then the vehicle slows down or maintains a speed of 30 kilometers and approaches closer to the speed bump. After a while, the speed bump is detected with the camera detector and the vehicle's speed is limited to 23 kilometers. The speed bump detection by the camera detector is shown in Figure 4.4.

Figure 4.4: Speed limited by detecting speed bump by the camera detector.

After that, the detection with the LiDAR detector occurs, and the vehicle's speed is restricted to 18 kilometers per hour. The vehicle slows down to this velocity and maintains it until timeout. Afterward, it accelerates back to the previously manually set speed on the cruise control. The speed bump detection by the LiDAR detector is shown in Figure 4.5.

Figure 4.5: Speed limited by detecting speed bump by the LiDAR detector.

Compared to related studies, the proposed solution focuses on the earliness of the detection and can detect speed bumps from a longer distance, as shown in Table 4.2. In addition, it provides independence between all algorithm components, thus securing reliable detection under different weather conditions. Furthermore, it was designed as an automotive application, and the functionality was demonstrated by decelerating the test vehicle.

Chapter 5

Conclusion

Over the past few years, significant progress has been made in integrating electronic devices into vehicles. Nowadays, cars are equipped with various sensors, cameras and radars that assist the driver in providing safer and more comfortable driving.

This thesis proposes an algorithm dealing with speed bump detection, which can be utilized to decelerate a vehicle or adjust its chassis, providing a much more comfortable crossing over a speed bump. The algorithm combines two independent methods, securing a more robust solution operating under various weather conditions. One method uses a technique that fits a plane to the obtained points from LiDAR and subsequently decides the detection based on the distances of points from the plane. The baseline solution and its further improvements made during development are described in Chapter 3.1. The second method uses the YOLOv8 neural network [28] applied to an image from a camera. It was trained on custom data collected by the test vehicle. The training data includes the speed bump class and extra classes of speed bump signs, providing more reliable and earlier detection.

The algorithm was tested and tuned on the recordings from the test vehicle. Furthermore, its functionality was demonstrated by the automatic deceleration of the test vehicle driving with activated cruise control when approaching a speed bump. The vehicle can decelerate fluently before going over the speed bump to a comfortable crossing speed selected after multiple tests to 18 kilometers per hour.

In future work, the following improvements and modifications that would increase the reliability and earliness of the algorithm could be included. The first modification would be to extract and use only points from the LiDAR scan in the wheels' direction, which would suppress the false positive detection of sidewalks. Another improvement could be to increase the density of more distant points, which could be achieved by using a different LiDAR that would scan farther or by trying a different LiDAR location, ideally somewhere on the vehicle's roof. Further modification could be using a different camera covering a narrower angle, leading to earlier detection based on larger objects in the image.

Bibliography

[1] R. Harrap, and M. Lato. An Overview of LIDAR: collection to applications. NGI publication, 2010. 1-9

[2] STANFORD ARTIFICIAL INTELLIGENCE LABORATORY ET AL. 2018. Robotic Operating System, ver.Noetic. [software]. [visited on 08-10-2022]. Available from: https://www.ros.org

[3] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24(6), 381–395 (1981). https://doi.org/10.1145/358669.358692

[4] Andrew Straw. Python-pcl [source code]. [visited on 15-10-2022]. Available from: https://github.com/strawlab/python-pcl

[5] Dalal N, Triggs B. (2007). Histograms of oriented gradients for human detection. Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), San Diego, USA, pp. 1–8

[6] Suykens JAK, Vandewalle J. (1999). Least squares support vector machine classifiers. J Neural Process Lett 9(3): 293–300

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014, pp. 580–587

[8] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. International journal of computer vision, vol. 104, no. 2, pp. 154–171, 2013

[9] R. Girshick. Fast r-cnn. ICCV, 2015, pp. 1440–1448

[10] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. ECCV. Springer, 2014, pp. 346–361

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. International journal of computer vision, vol. 88, no. 2, pp. 303–338, 2010

[12] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 2015, pp. 91–99

[13] Dai, J., Li, Y., He, K., and Sun, J. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. arxiv. https://doi.org/10.48550/arXiv.1605.06409

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. CVPR, 2016, pp. 779–788

[15] Triess, L. T., Peter, D., Rist, C. B., and Zöllner, J. M. (2020). Scan-based Semantic Segmentation of LiDAR Point Clouds: An Experimental Study. arxiv. https://doi.org/10.1109/IV47402.2020.9304631

[16] Yan, Y., Mao, Y., and Li, B. (2018). SECOND: Sparsely Embedded Convolutional Detection. Sensors, 18(10), 3337. https://doi.org/10.3390/s18103337

[17] H.-S. Yun, T.-H. Kim and T.-H. Park. Speed-bump detection for Autonomous vehicles by Lidar and Camera. Journal of Electrical Engineering and Technology, vol. 14, no. 5, pp. 2155-2162, 2019. https://doi.org/10.1007/s42835-019-00225-7

[18] V.S.K.P. Varma, A. Sasidharan, K. Ramachandran, B. Nair, Real time detection of speed hump/bump and distance estimation with deep learning using GPU and ZED stereo camera. Procedia Comput. Sci. 143, 988–997 (2018). https://doi.org/10.1016/j.procs.2018.10.335

[19] BALLINAS-HERNÁNDEZ, Ana Luisa; PINEDA, Iván Olmos; OLVERA-LÓPEZ, J. Arturo. Speed Bump Detection on Roads using Artificial Vision. Res. Comput. Sci. 148(9), 71-82 (2019)

[20] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang and X. Gao. Faster-LIO: Lightweight Tightly Coupled Lidar-Inertial Odometry Using Parallel Sparse Incremental Voxels. IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 4861-4868, April 2022. https://doi.org/10.1109/LRA.2022.3152830

[21] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. (2019). SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. arxiv. https://doi.org/10.48550/arXiv.1904.01416

[22] Liao, Y., Xie, J., and Geiger, A. (2021). KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D. arxiv. https://doi.org/10.48550/arXiv.2109.13410

[23] Pan, Y., Gao, B., Mei, J., Geng, S., Li, C., and Zhao, H. (2020). SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances. arXiv. https://doi.org/10.48550/arXiv.2002.09147

[24] CVAT.AI CORPORATION. Computer Vision Annotation Tool [online]. [visited on 03-02-2023]. Available from: https://www.cvat.ai/

[25] Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2019). Object Detection in 20 Years: A Survey. arXiv. https://doi.org/10.48550/arXiv.1905.05055

[26] Davis E. King. Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research 10, pp. 1755-1758, 2009

[27] Asraf, Amanullah. (2021). Cars Dataset (Positive and Negative). Mendeley Data, V1. https://doi.org/10.17632/dvnntk22h6.1

[28] G. Jocher, A. Chaurasia, and J. Qiu. YOLO by Ultralytics [source code]. 2023. [visited on 20-02-2023]. Available from: https://github.com/ultralytics/ultralytics

Appendix A

Attachments

The following Python scripts are attached to the electronic version of the thesis:

- lidar_detection.py
- lidar_node.py
- camera_bumps.py
- camera_signs.py
- master_node.py