TU WIEN Informatics

Traffic violations detection

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Tobias Müller
Registration Number 01417912

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Mag. Stefan Ohrhallinger, PhD

Vienna, July 8, 2025

Tobias Müller	Michael Wimmer

Erklärung zur Verfassung der Arbeit

Tobias Müller

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 8. Juli 2025

Tobias Müller

Kurzfassung

Mit dem Aufkommen autonomer Fahrzeuge und ihrer zunehmenden Integration in moderne Verkehrssysteme ist es unerlässlich, möglichst viele Informationen aus der unmittelbaren Fahrzeugumgebung zu sammeln. Dies ermöglicht es den Fahrzeugen, in einer Vielzahl komplexer Szenarien optimal zu agieren. Idealerweise halten sich alle Verkehrsteilnehmer strikt an die Verkehrsregeln und -vorschriften, was deren Berechenbarkeit erhöht. Dieser Idealfall bildet jedoch selten die realen Bedingungen ab. Diese Arbeit zielt darauf ab, dieses Problem zu lösen, indem sie mehrere vortrainierte neuronale Netzwerkmodelle nutzt, um eine Vielzahl von Objekten in der Fahrzeugumgebung zu erkennen, darunter Fahrzeuge, Verkehrszeichen und Ampeln. Aufgrund von Laufzeitbeschränkungen und der Verfügbarkeit von Open-Source-Erkennungsmodellen konzentriert sich die Implementierung auf statische Szenen. Durch die Konzentration auf statische Szenen ist das System so konzipiert, dass es Verstöße ohne Objektverfolgung oder zeitliche Analyse erkennt. Die finale Implementierung kann Verstöße wie ein Fahrzeug, das in falscher Richtung in einer Einbahnstraße fährt, und, unter bestimmten Annahmen, illegale Ampelüberquerungen erkennen.

Abstract

With the advent of autonomous vehicles and their growing integration into modern transportation systems, it is essential to gather as much information as possible from the vehicle's immediate surroundings. This allows the cars to take the best course of action in a variety of complex scenarios. Ideally, all road participants strictly adhere to traffic laws and regulations, making them more predictable. However, this ideal rarely depicts real-world conditions. This work aims to address this issue by leveraging multiple pre-trained neural network models to detect a multitude of entities in the vicinity of the vehicle, including vehicles, traffic signs, and traffic lights. Due to runtime constraints and the availability of open-source detection models, the implementation focuses on static scenes. By concentrating on static scenes, the system is designed to detect violations without the need of object tracking or temporal analysis. The final implementation can detect violations such as a vehicle traveling the wrong way on a one-way street and with some assumptions, illegal traffic light crossings.

Contents

CHAPTER 1

Introduction

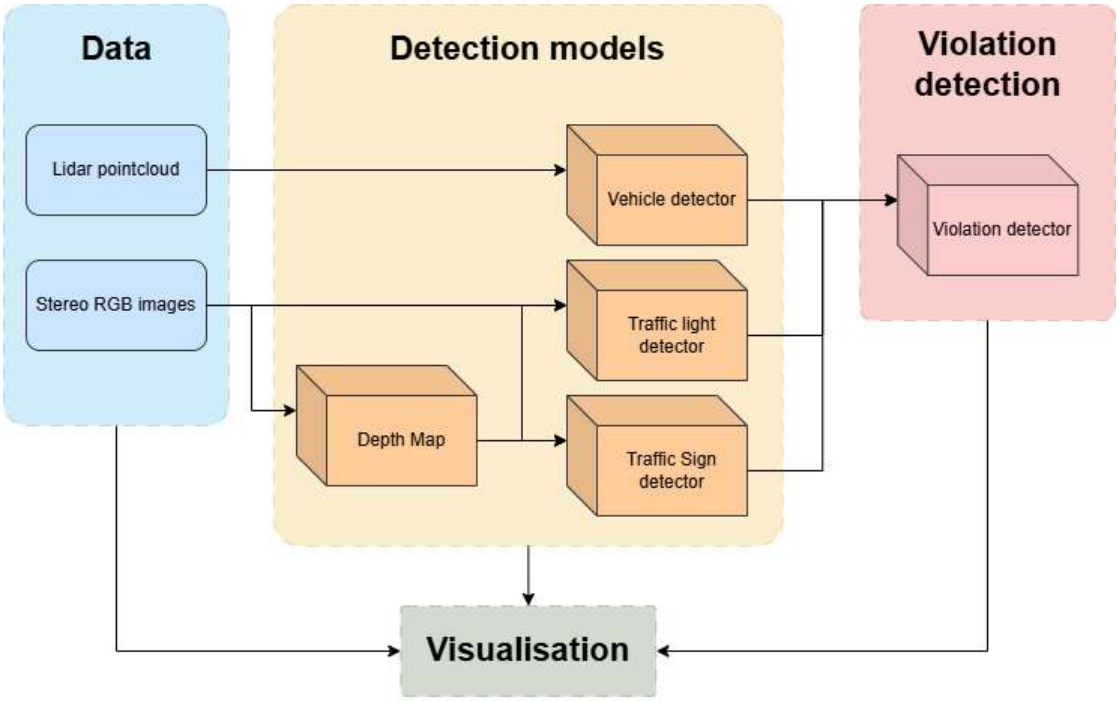


Figure 1.1: Diagram of the implemented pipeline

With the target of developing autonomous vehicles, significant progress has been made in the field of computer vision to detect a variety of entities in the street space. To ensure the robustness of such autonomous systems, it is not enough to believe all road participants adhere strictly to all laws and regulations. This is precisely where this paper comes in, aiming to detect various traffic violations in the immediate vicinity of a vehicle.

With the data collected by an array of sensors, vehicles, traffic signs, and traffic lights are detected, classified, and placed in relation to the vehicle. While existing models and pipelines primarily focused on detecting relevant data for the driven vehicle itself, the implemented pipeline also utilizes the data for all other detected vehicles to check if they violate any existing traffic rules.

Figure 1.1 depicts the rough overview of the implemented pipeline. In the first step, multiple data modalities are ingested. The various data modalities are then fed into a set of detection models. These detectors contain specialized pretrained models collected from open-source repositories. As these neural network models were not trained on the given dataset, the detectors transform the input data before feeding it into the models. The output of the models is then converted into a common dimension for further use in the violation detection step. As depicted by the "Depth Map" detector, some necessary information is not provided by the input data and, therefore, must first be generated or extracted. In this case, the available data provided by the LiDAR point cloud is insufficient; therefore, it is supplemented by a depth map. After all necessary entities are detected and classified, the violation detector tries to identify possible traffic violations. Finally, all the data is visualised in a graphical user interface.

This paper provides an in-depth examination of the implementation and the challenges that arose during the development process. The evaluation of the used detection models comprises both combined and isolated metrics. For models trained on the dataset used in the implementation, evaluation results can be found in the respective papers. In Chapter Five, "Conclusio & Future work" potential enhancements and improvements are discussed to extend the capabilities of the developed solution.

CHAPTER 2

Related Work

This chapter examines the various detection pipelines employed in the implementation, providing a unique perspective on the different input modalities. Different data modalities offer multiple advantages and can often resolve the shortcomings of other types of data.

2.1 Depth detection

Depth data is one of the most essential information for any robotic automation. While various sensors and data modalities attempt to capture dense and precise depth information, no optimal solution currently exists. As the used dataset provides multiple modalities from which the necessary information can be extracted, we take a look here at the most promising.

LiStereo [ZRVJR20] introduces a self-supervised framework that fuses stereo imagery and sparse LiDAR measurements to generate dense depth maps. By processing stereo image pairs from sparse LiDAR data LiStereo [ZRVJR20] allows for effective depth maps. This approach blends the strengths of dense but less stereo vision depth estimation with the accurate but sparse LiDAR data to produce high-quality results.

Wang et al. [QGW+20] proposed an end-to-end trainable framework that generates pseudo-LiDAR point clouds from images and integrates them directly into a 3D detector. By enabling joint optimization of depth estimation and object detection, their method significantly enhances the quality of the 3D point cloud and improves detection accuracy. This approach was chosen to be integrated into the framework as it was trained on the used KITTI dataset, and the resulting depth map provided excellent results.

2.2 Object Detection

The popularity of autonomous driving and ITS (Intelligent Transportation Systems) has heightened the need for fast and accurate object detection. Depending on the use case, various data modalities are available, each offering its own set of advantages and disadvantages. These modalities include, but are not limited to, mono RGB-images, Stereo RGB-images, LIDAR point clouds, and GPS data. This multitude of input data leads to various approaches to solving complex issues, such as vehicle detection. In this section, we describe different methods for vehicle, traffic light and traffic light detection.

2.2.1 Vehicle detection

Vehicle detection tries to resolve the issue of detecting vehicles from the given input data.

Vote3Deep [ERW+17] uses the point cloud data as input. This model converts the point cloud into a voxel-based 3D grid, which is put into the model. As the discretization of the point cloud into a 3D grid yields many zero values, a voting-based convolution is employed to enhance performance.

SFA3D [Dun20] is a model based on RTM3D [LZLC20]. It prepares the input LiDAR data by transforming it into a bird's-eye view image that contains descriptive information, such as height, density, and intensity. This 2D image is fed into a CNN model and outputs a list of directional bounding boxes.

For this implementation, SFA3D [Dun20] was chosen to detect vehicles and other entities, as it delivers high accuracy and fast runtimes. Additionally, the model offers an open-source repository, making integration significantly simpler.

2.2.2 Traffic sign detection & classification

This section examines various proposals for detecting and classifying traffic signs from RGB images.

2D Detection is identifying and locating objects within a 2-dimensional space. Traditionally, this was done with manual feature extraction, segmentation, and classification. One such approach was done via the use of a modified version of the Hough transformation to detect the shape of a traffic sign [YF15, SBA08].

Shao et al. [SWM+19] proposed an improved version of Faster R-CNN [RHGS16] for traffic sign detection, incorporating a second Region of Interest (ROI) and a Highly Possible Regions Proposal Network (HPRPN). The method enhances the efficiency and accuracy of the traditional Faster R-CNN by adding an extra ROI layer to refine the candidate regions for detection. The HPRPN further boosts performance by focusing on high-probability areas, reducing the number of irrelevant proposals, and improving overall detection speed.

A YOLOv8, a state-of-the-art object detection algorithm utilizing Convolutional Neural Networks (CNNs) for classification, has shown promising results. Singh et al. [S+23]

proposed a two-stage framework in which YOLOv8 efficiently localizes traffic signs in real-time, and a subsequent CNN classifies them into multiple categories. This approach is trained on widely used datasets, such as the German Traffic Sign Recognition Benchmark (GTSRB) [SSSI12] and German Traffic Sign Detection Benchmark (GTSDB) [HSS+13]. While the GTSDB provides full road scene images for training and testing traffic sign detection, the GTSRB focuses exclusively on traffic sign recognition, offering isolated pictures of signs at varying resolutions. This is also the main reason why this proposal was used in the implementation. As the KITTI dataset [GLU12] contains German traffic signs, a performant model trained on German signage was ideal for our use case.

2.2.3 Traffic light detection

Traffic light detection is similar to traffic sign detection in many ways, but it presents unique challenges. Unlike traffic signs, whose meaning is conveyed primarily by their shape, traffic lights require interpreting their active light states. This task is further complicated by varying lighting conditions, making accurate detection more difficult.

The aUToLights [CdAB+23] system demonstrates a robust approach to traffic light detection and tracking by utilizing multiple cameras combined with advanced object detection techniques and high-definition map data. This multimodal approach enhances detection accuracy and reliability in complex urban environments where occlusions and lighting variations are common.

Müller and Dietmayer et al. [MD18] propose an approach based on Single Shot Detection (SSD). Their method optimizes the SSD architecture specifically for traffic light characteristics, demonstrating significant improvements in both detection speed and reliability. As this approach relies on data modalities available in our dataset, it is well-suited for integration into the solution presented in this paper.

CHAPTER 3

Implementation

The goal of Traffic-violations-detector was to combine existing neural networks for vehicle and traffic light detection to detect and report traffic violations related to illegal crossings. Rather than developing new computer vision models, the focus was on integrating and orchestrating pre-trained ones. For this, the implementation focuses on making the process of incorporating additional neural networks as easy as possible.

The source code of the implementation can be found via: `https://gitlab.cg.tuwien.ac.at/stef/traffic-violations`

Traffic-violations-detector pipeline can be structured into four stages:

1. Data & input
2. Object detection & classification
3. Violation detection
4. Visualization

Accurate detection of traffic light violations requires precise positioning of entities in three-dimensional space. The KITTI dataset [GLU12] was selected for this purpose, as it provides predefined transformation matrices that map detections from point-of-view (POV) RGB images into 3D space with real-world metric scaling. This enables the formulation of precise detection rules based on actual physical distances. Beyond these transformation matrices, the KITTI dataset also offers a rich set of data modalities for each scene, further enhancing its utility.

For vehicle detection, the SFA3D[Dun20] neural network was chosen. It was designed explicitly for this dataset and delivers fast and accurate bounding boxes as well as the heading of vehicles.

As the KITTI[GLU12] dataset does not provide ground-truth data for traffic lights and lacks sufficient scenes to create a comprehensive training dataset ourselves, no model designed for this dataset can be found. For this reason, a YOLOv3[RF18] based model was used, trained on the LISA[JPM+16] traffic light dataset.

While developing Traffic-violations-detector , it was recognized that the provided LiDAR data was insufficient to position detected traffic lights in 3-dimensional space accurately. For this reason, an alternative solution was required. PseudoLiDAR++[YWC+20] allows for the creation of a precise depth map from stereo images. This depth map enables the extraction of Z values for all detections in the POV RGB images.

3.1 Data & Datainput

The KITTI[GLU12] dataset provides a multitude of different modalities usable for various detection pipelines and, as such, was chosen as the dataset for this implementation.

The data modalities provided by the KITTI[GLU12] for each scene:

- LiDAR pointcloud
- Left RGB image
- Right RGB image
- Calibration data
- GPS
- Labels

3.1.1 LiDAR

LiDAR data provides a set of X, Y, and Z coordinates representing points in LiDAR space, along with a reflection intensity value for each point. Because the points are not ordered in any meaningful way, additional processing is necessary to render them from arbitrary perspectives.

3.1.2 RGB images

Captured from the vehicle's point of view, the left and right RGB images are taken approximately half a meter apart. This stereo setup enables depth estimation even without relying on LiDAR data. However, the image dimensions are inconsistent across scenes, requiring normalization during preprocessing.

3.1.3 Calibration data

Calibration data includes transformation matrices that map detection points across different sensor modalities and coordinate dimensions. These matrices enable easy conversion of detection results into a shared reference frame, streamlining downstream computations.

3.1.4 Labels

In addition to the different sensor data, ground truth for different entities is given. They are described by a class, bounding box on left POV RGB image, center point in world coordinates and dimensions represented by height, width and length.

KITTI entity classes:

- Car
- Bike
- Pedestrian
- Misc (trailers, ...)

In addition to the classes provided by the KITTI[GLU12] dataset, additional entries were created for a subset of data for evaluation purposes.

Traffic signs classes:

- Ahead Only
- Bike
- Bumpy Road
- Children
- Dangerous Left Curve
- Dangerous Right Curve
- Deer
- Double Curve
- End of Limits
- End of No Overtaking
- End of No Overtaking for Heavy Vehicles
- End of speed limit (80km/h)
- General Caution
- Go Straight or Left
- Go Straight or Right
- Heavy Vehicles Prohibited
- Keep Left
- Keep Right
- Narrowing Road
- No Entry
- No Overtaking
- No Overtaking for Heavy Vehicles

- No Vehicles
- Pedestrian
- Priority Road
- Right-of-Way at next Intersection
- Road Work
- Roundabout Mandatory
- Slippery Road
- Snow
- Speed limit (100km/h)
- Speed limit (120km/h)
- Speed limit (20km/h)
- Speed limit (30km/h)
- Speed limit (50km/h)
- Speed limit (60km/h)
- Speed limit (70km/h)
- Speed limit (80km/h)
- Stop
- Traffic Signals
- Turn Left Ahead
- Turn Right Ahead
- Yield

Traffic light classes:

- Go
- Warning
- Stop

The data input stage reads a single set of input data on demand. As further calculations occur in world space, no transformation is applied to the ground truth for evaluation.

3.2 Object detection & classification

At this stage of the traffic violation detection pipeline, a collection of pre-trained neural networks is used to detect and classify various road objects.

The types of models used are:

- SFA3D[Dun20] (for Vehicle detection and classification)
- Pseudo LiDAR v.2[YWC+20] (Depth map creation)
- Traffic sign detection and classification[WCDG23]
- Traffic light detection and classification [Rat20]

LiDAR point clouds offer sufficient data for accurate and efficient vehicle detection, including basic classification. However, this modality alone is not suitable for all detection tasks required in this implementation. Specifically, traffic light classification relies on color information, which is only available in the POV RGB images.

Since RGB images lack inherent depth information, an attempt was made to supplement this using LiDAR data. However, LiDAR has significant limitations along the Y-axis; as illustrated in Figure 3.1, no points are captured above a height of 2 meters—making it unreliable for detecting elevated objects like traffic lights.

To address this limitation, a depth map is generated to provide depth information across the entire RGB image. This is achieved using stereo RGB images captured from slightly different viewpoints, allowing for accurate depth estimation even in regions where LiDAR data is sparse or absent.

Figure 3.1: Rendering of LiDAR data from POV

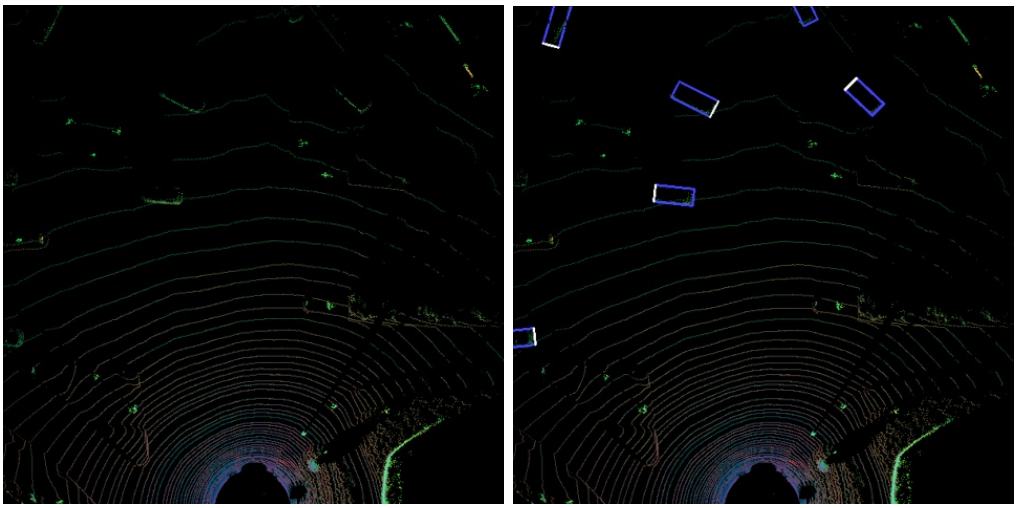
The main challenge here was to transform the available input modalities into a suitable form for each model, and then to convert the output into the same dimension that could be used for detecting traffic violations.

3.2.1 Vehicle detection

SFA3D[Dun20] is the model chosen for detecting vehicles. While SFA3D[Dun20] can detect not only cars but also pedestrians and bicycles, these classes are of no interest for the implementation. The code used is a slight adaptation provided by the SFA3D[Dun20] GitHub repository. Adaptations are limited to coordinate transformation and restructuring of code to fit the implementation's code layout. Although the available LiDAR data enables 360-degree detection, the implementation restricts the detection to the front half

of the vehicle. This is done to minimize runtime, as traffic light detection is limited to the front of the vehicle.

The input for the model consists of a 2D rendering of the available LiDAR data. The LiDAR data is first clipped to a set bounding box. With the provided transformation matrices, the data is then projected onto a 2D plane from a bird-eye perspective. The RGB values are used to encode information. The red channel contains information about the height (Z axis) of the highest point per pixel, the green channel holds information on the reflection intensity, and the blue channel encodes the density of LiDAR points per pixel. Finally, the input matrix is scaled to an image of size 608 x 608 pixels as seen in Figure 3.2a. This input is then fed into the pretrained model provided by the SFA3D[Dun20] GitHub repository.



(a) Input image for SFA3D model (b) Lidar BEV image with detected vehicles

Figure 3.2: Lidar BEV and detected vehicles

After the SFA3D[Dun20] model ingested the 2D LiDAR projection, the list of detected entities is filtered via a non-max suppression algorithm 3.1. This non-maximum suppression filter is limited to a maximum of 50 results. After eliminating duplicate detections, they are sorted by class and again filtered by confidence. The confidence threshold is set to 0.2. Every detection below this threshold is removed. As the list of results is still in LiDAR space, the values are transformed into the world coordinates as described earlier.

Algorithm 3.1: Non max suppression

Input: A list of labels with bounding boxes, a scalar for intersect over union threshold

Output: list of labels

```
labels ← sorted(self.labels, key = x.confidence, reverse = True)
for i ← 0 to len(labels) do
    for j ← 1 to len(labels) do
        if label.iou(labels[j]) > iou_threshhold then
            labels.pop(j)
        end
    end
end
return labels
```

3.2.2 Depthmap - Pseudo-LiDAR++

To generate the depth map, the Pseudo-LiDAR++ neural network [YWC+20] was used. Specifically, the initial stage of the Pseudo-LiDAR pipeline produces a 2D depth map. As previously mentioned, the available LiDAR data lacks the necessary detail to localize all detected objects in 3D space accurately. To address this, the left and right RGB images are first cropped to a resolution of 1200 × 352 pixels.

Figure 3.3: RGB Pov image & coresponding depthmap (Black = close | white = far)

Along with the stereo image pair, the calibration matrix of the left image is converted into a tensor and included as an input to the model. Figure 3.3 depicts the greyscale output of the network in which each pixel value represents depth in meters. These values can then be used to determine the depth of detected objects in the point-of-view (POV) RGB images.

3.2.3 Traffic light detection and classification

All previous models were trained on the KITTI[GLU12] dataset. So, the preparation of input data was minimal. As the KITTI[GLU12] dataset does not provide ground truth for traffic light positions and states no pretrained model could be found. Instead, a YOLOv3 [RF18] based model was chosen. The Traffic-Light-Detection-Using-YOLOv3[Rat20] model was trained on the LISA Traffic Light Dataset[JPM+16]. The possibility of training the model on the KITTI[GLU12] dataset was explored. However, the number of available scenes containing traffic lights is too limited, which is why the pre-trained model was chosen.

During testing, several limitations of the pretrained model were identified, particularly in detecting traffic lights at varying scales. To overcome this, a multi-scale inference approach was implemented. First, the input image is resized to fit within a 512×512 frame, with any remaining space padded in grey. After obtaining initial detections, the image is then resized to a height of 512 pixels and divided into five overlapping vertical slices, each covering one-third of the image width. Each version of the input is processed independently. Detections are filtered using non-maximum suppression (NMS) with an Intersection over Union (IoU) threshold of 0.6 and a confidence threshold of 0.1.

After running the detection pipeline across all six image versions, the results are consolidated into a single list. Bounding boxes are projected into world space: the Z-coordinate is estimated by averaging the depth values within the corresponding region of the depth map, while the 2D bounding box coordinates (xy_min and xy_max) are transformed into world coordinates. From these, the center and physical dimensions of each bounding box are computed and stored. Finally, a second round of non-maximum suppression 3.1 is applied to eliminate duplicate detections from overlapping regions.

3.2.4 Traffic sign detection and classification

Traffic sign detection and classification is a two-step pipeline and uses two models. One for traffic sign detection and the other for traffic sign classification. Like the traffic light detection and classification, these models were not trained on the KITTI dataset. The detection model was trained on the German Traffic Sign Detection Benchmark (GTSDB) dataset. The implementation of Traffic-violations-detector is similar to the implementation of the traffic light. Splitting the image into partitions and processing each partition separately.

The classification model was trained on the German Traffic Sign Recognition Benchmark (GTSRB) [SSSI12] dataset. As such, the model is designed to expect an image of 32

Figure 3.4: POV image with detected traffic lights and signs

times 32 pixels. For this, the bounding boxes of the detection steps are used and scaled to the required resolution. For better results, the traffic sign images are also subjected to an adaptive histogram equalization to enhance the contrast of the image.

3.3 Traffic lights violation detection

Since the selected data consists of static scenes, certain assumptions were necessary to enable the detection of traffic violations. Due to the lack of road marking detection—such as lane boundaries or stop lines—it is assumed that vehicles are required to stop at positions perpendicular to the traffic lights.

Additionally, the static nature of the data complicates the interpretation of traffic light states over time. To address this limitation, a strict legal interpretation was adopted, particularly regarding yellow (amber) signals, treating them as equivalent to red for the purpose of violation detection.

The Austrian law STVO.1960 Abs.38 states

> Gelbes nicht blinkendes Licht gilt unbeschadet der Vorschriften des § 53 Z 10a über das Einbiegen der Straßenbahn bei gelbem Licht als Zeichen für „Halt“. Bei diesem Zeichen haben die Lenker herannahender Fahrzeuge unbeschadet der Bestimmungen des Abs. 7 anzuhalten

Translation:

> A non-flashing yellow light shall be deemed to be a "stop" signal, without prejudice to the provisions of Section 53(10a) regarding the turning of a tram at a yellow light. At this signal, drivers of approaching vehicles must stop, without prejudice to the provisions of paragraph 7."

In addition to the interpretation that vehicles must stop at a yellow indicating traffic light, the set duration of the yellow state is a minimum of 2 seconds. With this in mind,

it was assumed that only in very rare cases is a car allowed to cross a yellow indication. For simplification purposes, we disregard this, and therefore, we can assume that every vehicle still located beyond the traffic light and a certain heading violated the law.

The traffic light detection does not differentiate between indicators meant for vehicles, pedestrians, or bikes. Therefore, the detection is limited to the one closest to the x (left or right) position of the POV vehicle. As there is no information on speed or time since the traffic light state switch, the detection is limited to vehicles within 15 meters of the traffic light.

3.3.1 Ego traffic

To detect this kind of violation, the state of the detected traffic light needs to be "stop". If this is the case, all vehicles beyond the position of the traffic light are fetched. If the vehicle's heading is within 45 degrees of the pov heading, it is assumed that the car crossed a red or yellow traffic sign and should have stopped. The heading detection is further extended by 45 degrees in one direction, depending on the position relative to the traffic sign. With this extension, vehicles that made an illegal turn can also be detected.

As seen in Figure 3.5, the car taking a right turn is marked in red to depict the violation of the red traffic sign. In this case, however, this is a false positive as the green traffic light is not detected.

3.3.2 Crossing traffic

If the state of the traffic light is green, we assume that all traffic crossing our lane is forbidden. To detect such violations, vehicles further away than the traffic light and within 5 meters to the left or right of it are investigated. If the detected heading is perpendicular to the pov heading, a traffic violation was detected.

3.3.3 Oncoming traffic

For oncoming traffic, the necessary traffic light data is missing. In this case, it is assumed that oncoming traffic is allowed to proceed simultaneously with the traffic from the ego vehicle. Every vehicle in the oncoming traffic beyond the traffic light is checked for heading. If the heading is within 22.5 degrees of a straight-oncoming direction, the car is marked as a possible traffic violation as seen in figure 3.6.

3.4 Traffic sign violation detection

In addition to traffic lights, traffic violations related to traffic signs were implemented. While violations connected to stop and yield signs require a non-static scene, the implementation was limited to detecting violations that can also be detected in static scenes.

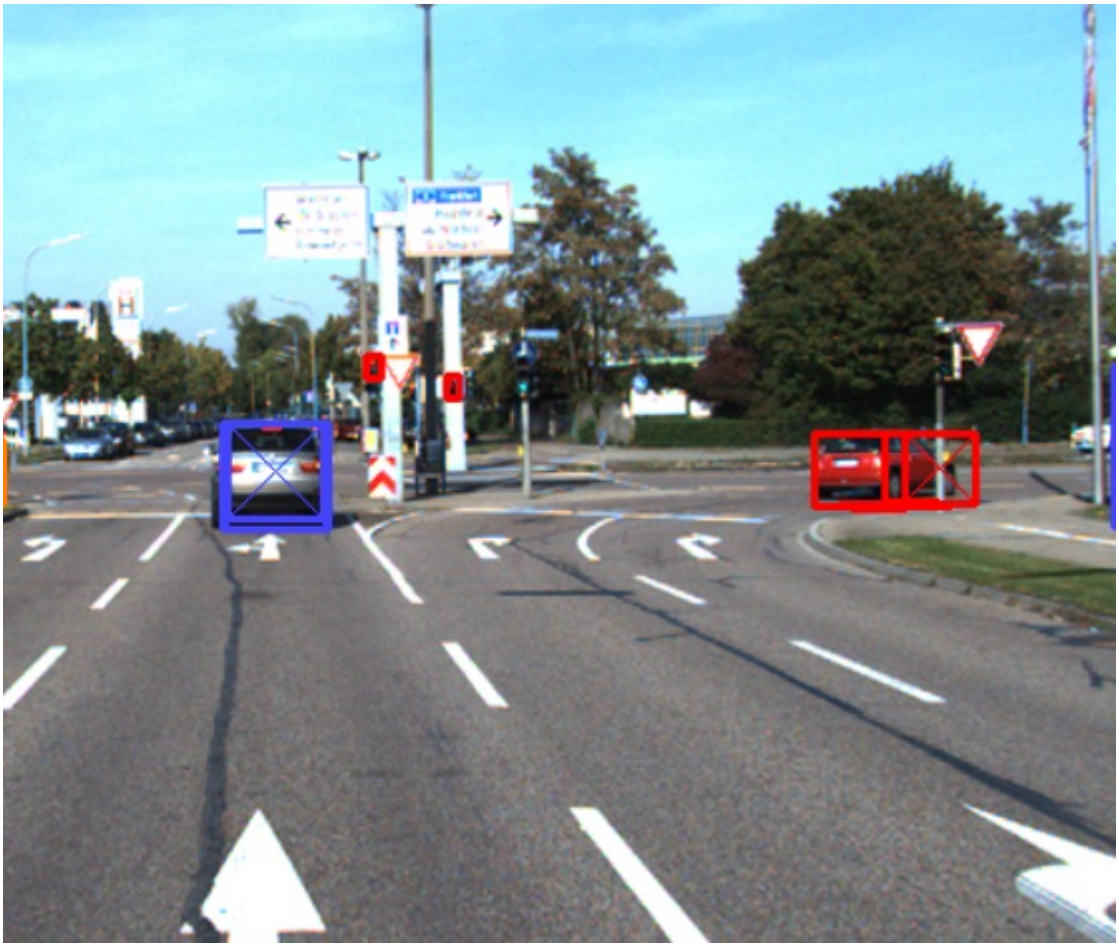


Figure 3.5: Violation (Red) for traffic going in ego direction

3.4.1 Keep right/left

When a Keep right/left traffic sign is detected, all vehicles beyond and opposite to the indicated side are examined for their direction. If they are directed from a point of view, they are marked as having violated the traffic sign.

3.4.2 No entry

Every vehicle beyond a no-entry sign is examined for direction. If it is in the point-of-view direction, the violation is marked.

3.5 Visualisation

To visualise the data of Traffic-violations-detector , Customtkinter was used. This is an extension of Tkinter. Each scene is rendered from two perspectives: point-of-view

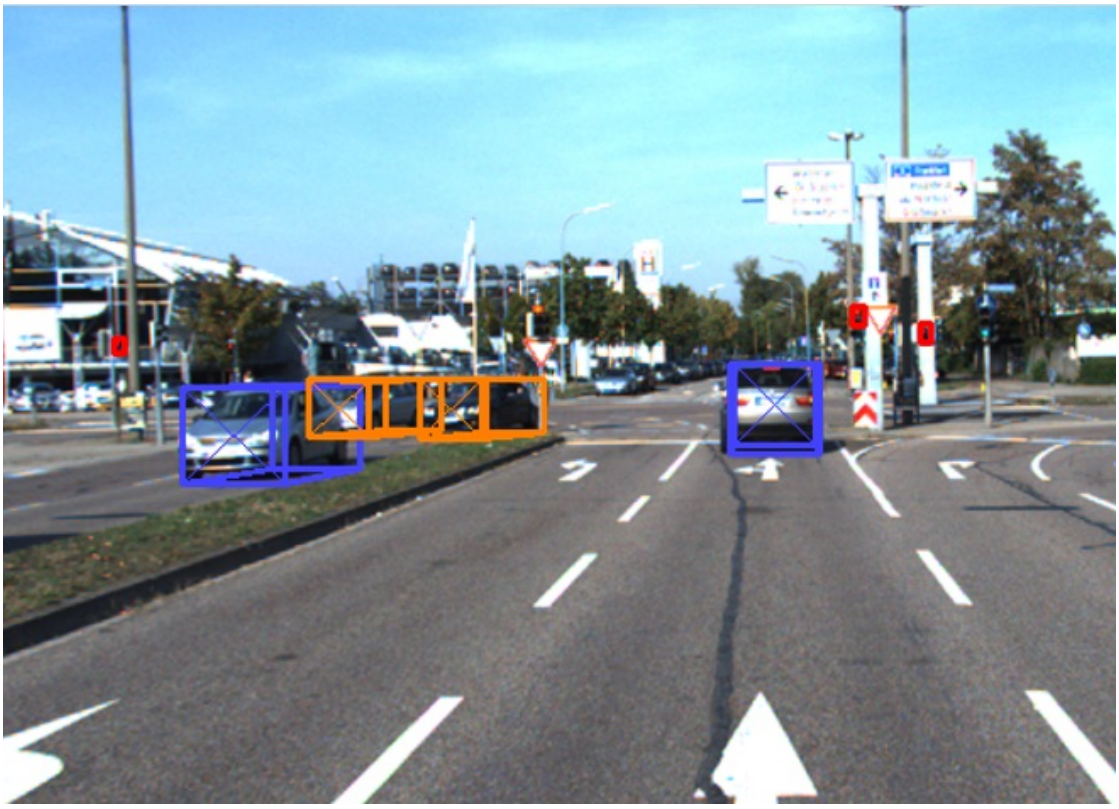


Figure 3.6: Possible violation (orange) for oncoming traffic

and birds-eye-view. As seen in Figure 3.7, some scenes may be overcrowded, and the UI allows for the rendering of each modularity and viewpoint.

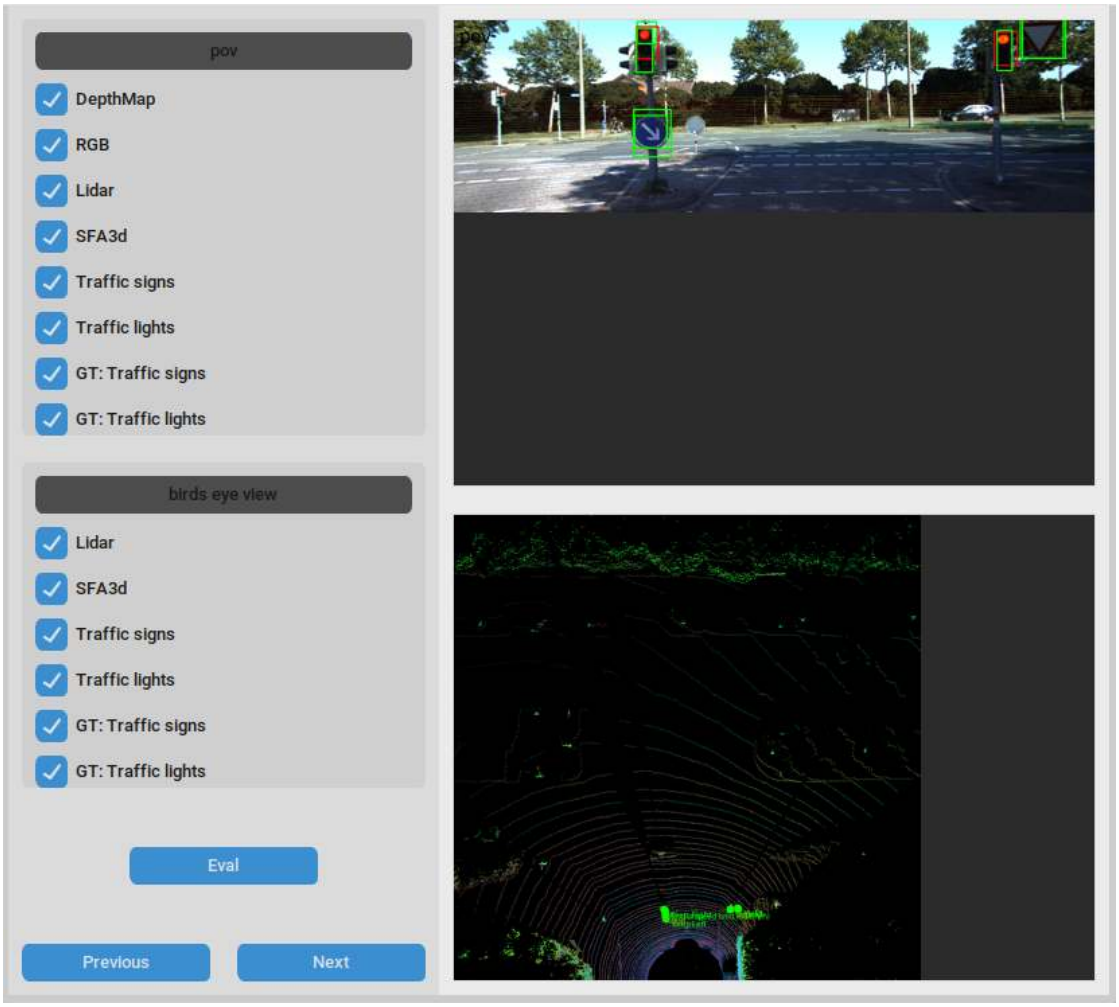


Figure 3.7: Screenshot of UI

CHAPTER 4

Evaluation & Results

In this chapter, we will take a look at the performance of the used pipelines and quality of detections.

4.1 Data & environment

4.1.1 Data

As the KITTI [GLU12] dataset does not provide ground truth for all entities of interest by this implementation, additional ground truth data was manually annotated. A subset of 49 Scenes was selected from the KITTI [GLU12] dataset that provided a mix of different traffic signs and traffic light states. In addition to the variety of objects, multiple scales of objects were also of interest to test the scale invariance of the implemented pipelines.

Figure 4.1: Rendering of ground truth and detected bounding boxes from POV. (Blue bounding boxes are ground truth)

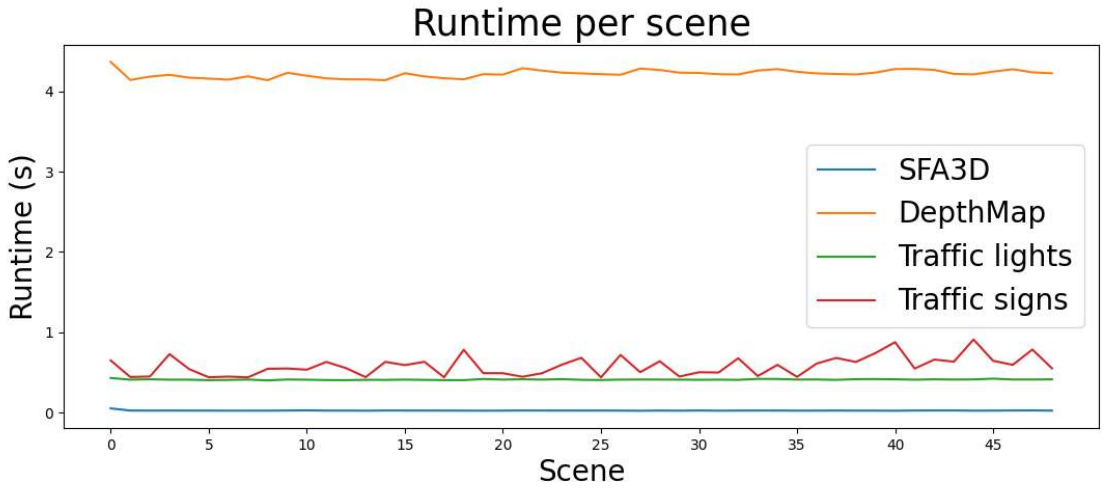


Figure 4.2: Runtime (seconds) of pipelines per scene

4.1.2 Test environment

The evaluation was run on a Linux machine with the following hardware specs:

- Processor: AMD Ryzen 7 5800X
- 32 GB of RAM
- NVIDIA RTX 3080

For OS and drivers, the following versions were used:

- Ubuntu 24.04.02
- CUDA driver 576.02

4.2 Runtimes

In this section, the runtime of the pipelines is measured. The runtime of the rendering for the GUI is ignored.

Average runtime:

- Vehicle detection: 31.479 milliseconds
- DepthMap: 4274.493 milliseconds
- Traffic lights: 442.081 milliseconds

- Traffic signs: 651.091 milliseconds

The vehicle detection via SFA3D[Dun20] is by far the fastest part of the pipeline and is way within the limits for a real-time detection pipeline. Traffic light and traffic sign detection require some optimization to be a viable option for such an application. The primary issue with the pipeline is the generation of the DepthMap. With a runtime of over four seconds per scene, it is significantly slower than the targeted runtimes outlined in this thesis.

4.3 Detection & 3D placing

This section will cover the quality of the traffic light and traffic sign detection pipelines used. As the vehicle detection uses a slightly adapted implementation and a pretrained model trained on the KITTI[GLU12] dataset, the evaluation can be seen in the SFA3D paper[Dun20].

To measure the performance of the detection and classification pipelines, the detected entities in the test data are matched with the annotated data. Matches are generated by calculating the Intersection over Union (IoU).

4.3.1 Intersection over Union

This subsection analyzes the Intersection over Union (IoU) values for detected matches. As all pipeline outputs are represented as coordinates in 3D world space, both sets of data are transformed into the point-of-view (POV) image space using the same transformation matrices. This yields 2D bounding boxes, for which the IoU value is calculated. Any IoU value above 0.4 signifies a match and is used for further analysis.

Since the inverse of the original transformation matrix is used to project the model's 2D bounding box outputs back into 3D space, the resulting Z-values may exhibit minor rounding errors due to floating-point operations. However, these errors are negligible, and the Z-coordinate is not relevant for the 2D comparison, allowing it to be safely ignored during evaluation.

IoU mean

- Traffic lights: 0.663
- Traffic signs: 0.698

Figure 4.3 depicts the IoU values per match with the minimal values for matches set to 0.3. It can be observed that matches with IoU values above 0.8 are rare, and in general, the IoU values of matches hover around 0.68.

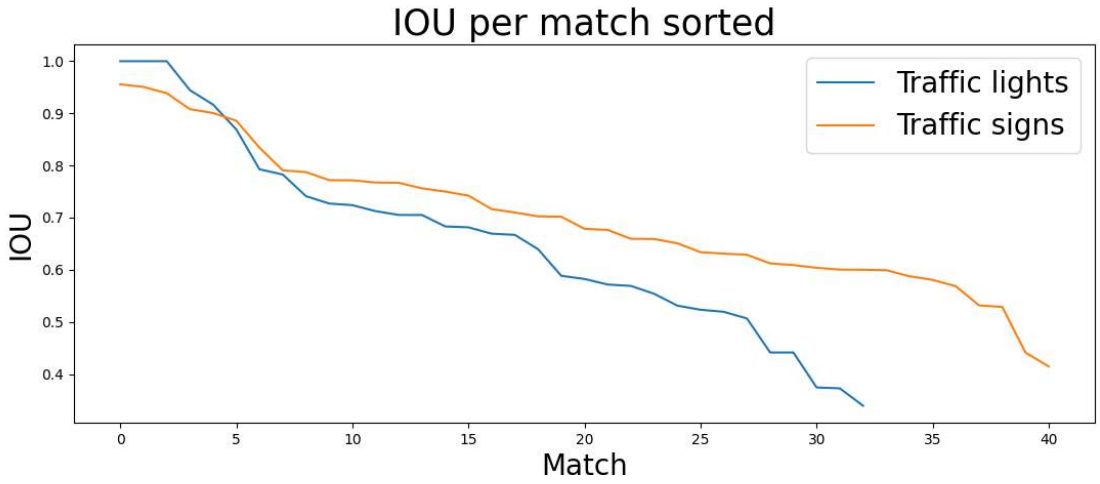


Figure 4.3: Intersection over union values for matches sorted desc.

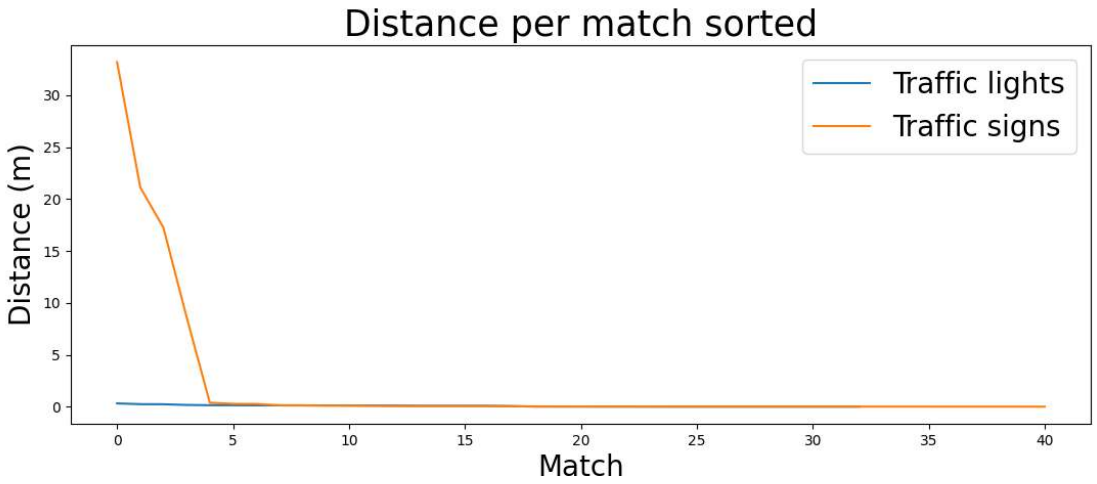


Figure 4.4: Distance in meters per matched ground truth and detected entity sorted desc.

4.3.2 Distance

This subsection analyzes the distance between detected entities and the matched ground truth label. The bounding box is ignored for these calculations, and only the center point is used to calculate the distance.

Traffic lights:

- Mean: 0.077 meters
- Median: 0.084 meters
- Root Mean Square Error: 0.11 meters

Traffic signs:

- Mean: 2.002 meters
- Median: 0.022 meters
- Root Mean Square Error: 6.84 meters

The results for the positional deviation of Traffic lights show auspicious results. The Root Mean Square Error is only at 11 Centimeters, which is more than enough for accurate positioning of the traffic lights.

For Traffic signs, however, some outliers significantly impact the distance between detected traffic signs and the ground truth, as can also be observed in figure 4.4. The figure depicts the distance between the ground truth and the detected entity per match. One observed explanation for such significant outliers is oversized bounding boxes, which result in depth values being assigned from the background rather than the actual sign.

4.4 Classification

This section will cover the quality of the detection pipelines used. A distinction is made between combined pipelines and isolated pipelines.

4.4.1 Traffic lights

Table 4.1 depicts the confusion matrix for the classification of traffic lights. We can see that the classification for traffic lights works very well and only shows one single error.

		Predicted Classes				
		GO	STOP	WARNING	STOP LEFT	STOP RIGHT
Actual Classes	GO	5	0	0	0	0
	STOP	0	25	0	1	0
	WARNING	0	0	1	0	0
	STOP LEFT	0	0	0	1	0
	STOP RIGHT	0	0	0	0	0

Table 4.1: Confusion matrix traffic lights

4.4.2 Traffic signs

The traffic sign pipeline consists of two models. The first model detects a 2D bounding box, which is then used to crop the image, serving as input for the second model, classifying the detected traffic sign. As the second model depends on the output of the first model, in addition to evaluating the combined pipeline, a second evaluation was conducted using the annotated ground truth bounding boxes to generate the input for the classification model.

The combined results, shown in Table 4.2, when the detection model result is used to generate input for the classification model, depict a false classification of traffic signs 80% of the time. Table 4.3 shows marginally better classification results when using ground truth bounding boxes for the classification model input. Even so, traffic signs are wrongly classified more than 60% of the time.

Combined metric

Actual Classes \ Predicted Classes	Unknown	Stop	Yield	Turn left ahead	Priority road	Right of way	Keep left	Keep right	No entry	No vehicles	End of speed limit	End of no overtaking	Heavy vehicles prohibited	Speed limit (30 km/h)	Speed limit (50 km/h)	Speed limit (60 km/h)	Speed limit (80 km/h)
Unknown	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-
Stop	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-
Yield	-	-	6	11	-	1	1	-	-	-	-	2	-	-	-	-	-
Turn left ahead	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
Priority road	-	-	-	2	-	-	2	-	-	-	-	2	-	-	-	-	-
Right of way	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-
Keep left	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Keep right	-	-	-	2	-	-	-	-	-	-	-	1	-	-	-	-	2
No entry	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
No vehicles	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
End of speed limit	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
End of no overtaking	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Heavy vehicles prohibited	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Speed limit (30 km/h)	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
Speed limit (50 km/h)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
Speed limit (60 km/h)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
Speed limit (80 km/h)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 4.2: Confusion matrix traffic signs

Isolated classification metric

Actual Classes \ Predicted Classes	Unknown	Stop	Yield	Turn left ahead	Priority road	Right of way	Keep left	Keep right	No entry	No vehicles	End of speed limit	End of no overtaking	Heavy vehicles prohibited	Speed limit (30 km/h)	Speed limit (50 km/h)	Speed limit (60 km/h)	Speed limit (80 km/h)
Unknown	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-
Stop	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-
Yield	-	-	10	8	-	-	1	-	-	-	-	2	-	-	-	-	-
Turn left ahead	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
Priority road	-	-	-	2	-	-	2	-	-	-	-	2	-	-	-	-	-
Right of way	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-
Keep left	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Keep right	-	-	-	2	-	-	-	-	-	-	-	1	-	-	-	-	2
No entry	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
No vehicles	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
End of speed limit	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
End of no overtaking	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Heavy vehicles prohibited	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Speed limit (30 km/h)	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-
Speed limit (50 km/h)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
Speed limit (60 km/h)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
Speed limit (80 km/h)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 4.3: Confusion matrix traffic signs isolated from detection model

CHAPTER 5

Conclusio & Future work

The initial objective of this thesis was to develop a framework for traffic violation detection. The current state can already detect vehicles, pedestrians, traffic lights, and traffic signs. While some detection pipelines yield acceptable results, others still suffer from limited precision or poor classification accuracy. In particular, vehicle and traffic light detection and classification have reached a reasonably reliable state. The traffic sign classification, however, will require more work as it rarely classifies the detected traffic signs correctly.

Significant improvements are still needed in terms of runtime. Vehicle detection currently operates in under 40 milliseconds, which is a promising result. In contrast, the other pipelines require further optimization. At present, traffic light and sign detection are executed multiple times per scene. If these models can be trained on the KITTI[GLU12] dataset and thus be executed only once per scene, their runtime could be reduced by more than 80%.

Depth map generation remains the most time-consuming component and, in its current form, is unsuitable for real-time applications. To resolve this, a more efficient solution must be implemented—either by generating depth maps in under 100 milliseconds or by developing a solution that utilizes LiDAR data even for areas not covered by it.

Once these pipelines are optimized—or replaced with more efficient alternatives—the system will be capable of processing dynamic (non-static) scenes.

For this purpose, synthetic data can be generated with the help of tools like Carla[DRC+17]. This would also allow for the creation of scenes to test the detection of traffic violations. Real scenes with traffic violations will require a lot of time to be captured, and as such, synthetic data offers a great alternative.

To expand the scope of detectable violations, additional elements should be incorporated into the detection framework. A key extension would be the detection of road markings, including lane dividers, pedestrian crossings, restricted zones, and stop lines. Integrating

these features would significantly enhance the system's ability to identify a broader range of traffic infractions.

Overview of Generative AI Tools Used

Microsoft Copilot was used in development for auto-completions

Grammarly AI-powered suggestions and spelling check was used in writing this paper

List of Figures

List of Tables

List of Algorithms

Bibliography

[CdAB+23] Pedro Carvalho, Thiago de Almeida, Anderson Botelho, et al. autolights: A robust multi-camera traffic light detection and tracking system. *arXiv preprint arXiv:2305.08673*, 2023. Accessed: 2025-06-22.

[DRC+17] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. Carla: An open urban driving simulator. In *CoRL*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 2017.

[Dun20] Nguyen Mau Dung. Super-Fast-Accurate-3D-Object-Detection-PyTorch. `https://github.com/maudzung/Super-Fast-Accurate-3D-Object-Detection`, 2020.

[ERW+17] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361. IEEE, 2017.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[HSS+13] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

[JPM+16] Morten Bornø Jensen, Mark Philip Philipsen, Andreas Møgelmose, Thomas Baltzer Moeslund, and Mohan Manubhai Trivedi. Vision for looking at traffic lights: Issues, survey, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):1800–1815, 2016.

[LZLC20] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In *European Conference on Computer Vision*, pages 644–660. Springer, 2020.

[MD18] Julian Müller and Klaus Dietmayer. Detecting traffic lights by single shot detection. *arXiv preprint arXiv:1805.02523*, 2018.

[QGW+20] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q. Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection, 2020.

[Rat20] Sovit Ranjan Rath. Traffic-Light-Detection-Using-YOLOv3. `https://github.com/sovit-123/traffic-light-detection-using-yolov3`, 2020.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[RHGS16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[S+23] Yashank Singh et al. Traffic sign recognition using yolov8 algorithm extended with cnn. *arXiv preprint / Conference Name (if known)*, 2023. Code available on GitHub.

[SBA08] MA Souki, L Boussaid, and M Abid. An embedded system for real-time traffic sign recognizing. In *2008 3rd International Design and Test Workshop*, pages 273–276. IEEE, 2008.

[SSSI12] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[SWM+19] Faming Shao, Xinqing Wang, Fanjie Meng, Jingwei Zhu, Dong Wang, and Juying Dai. Improved faster r-cnn traffic sign detection based on a second region of interest and highly possible regions proposal network. *Sensors*, 19(10):2288, 2019.

[WCDG23] Junfan Wang, Yi Chen, Zhekang Dong, and Mingyu Gao. Improved yolov5 network for real-time multi-scale traffic sign detection. *Neural Computing and Applications*, 35(10):7853–7865, 2023.

[YF15] Pavel Yakimov and Vladimir Fursov. Traffic signs detection and tracking using modified hough transform. In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, volume 5, pages 22–28. IEEE, 2015.

[YWC+20] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *ICLR*, 2020.

[ZRVJR20] Junming Zhang, Manikandasriram Srinivasan Ramanagopal, Ram Vasudevan, and Matthew Johnson-Roberson. Listereo: Generate dense depth maps from lidar and stereo imagery. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7829–7836, 2020.