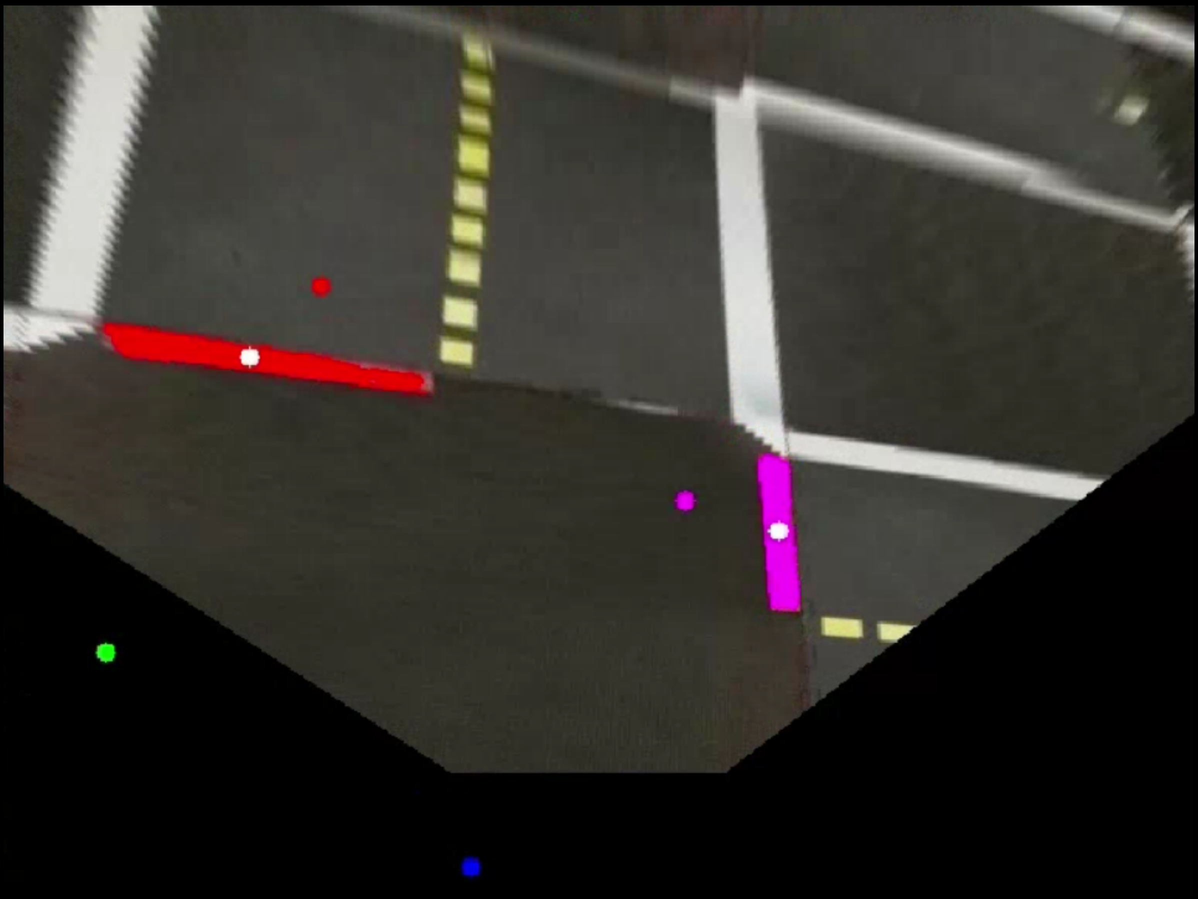
Intersection Navigation utilizing Bird's Eye View representations from 3D Image Features



Jasper Mulder

Cover illustration: Identified stop-lines in a Bird's Eye View of an intersection.

Intersection Navigation utilizing Bird's Eye View representations from 3D Image Features

Jasper Mulder
12567299

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute Faculty of Science
University of Amsterdam
Science Park 900
1098 XH Amsterdam

March 3rd, 2023

Abstract

This research examines the effectiveness of using a 3D-encoded image feature representation for navigating unprotected intersections with autonomous vehicles, specifically focusing on DuckieBots. The current solution proposed by Giles, Leopoldseder, and Wieland (2019) utilizes a Bird's Eye View (BEV) representation, which transforms a camera image to a BEV plane using camera intrinsics. However, due to the lack of depth information in a 2D camera image, this approach has limitations. MILE (Hu et al., 2022) addresses this limitation by encoding 3D image features before transforming to BEV space, achieving state-of-the-art driving scores in the CARLA simulator (Dosovitskiy, Ros, Codevilla, Lopez, & Koltun, 2017). This study aims to integrate the BEV encoding from MILE into the intersection navigation approach proposed by Giles et al. (2019), and evaluates its performance. While MILE achieves state-of-the-art performance in the CARLA simulator, this thesis shows that, without additional training, its generalization to other applications is limited.

Contents

Chapter 1

Introduction

Autonomous vehicles (AVs) can observe the world around them using many different methods, such as cameras, LiDAR, and radar (Ignatious, Khan, et al., 2022). Each of those methods has its strengths and weaknesses, as seen in Figure 1.1, but regardless of the method used, the surroundings of the vehicle must be represented in a way that is suited for path planning, obstacle recognition, and other possible self driving tasks. These tasks must be executed well in order to avoid collisions (see Figure 1.2 and reach a destination. For camera input, a possible representation is a Bird's Eye View (BEV), which displays the observed surroundings in a top-down view with the vehicle at the center. BEVs offer a spatial layout which is useful for tasks such as path planning and trajectory prediction, something the an image directly taken by a camera lacks.

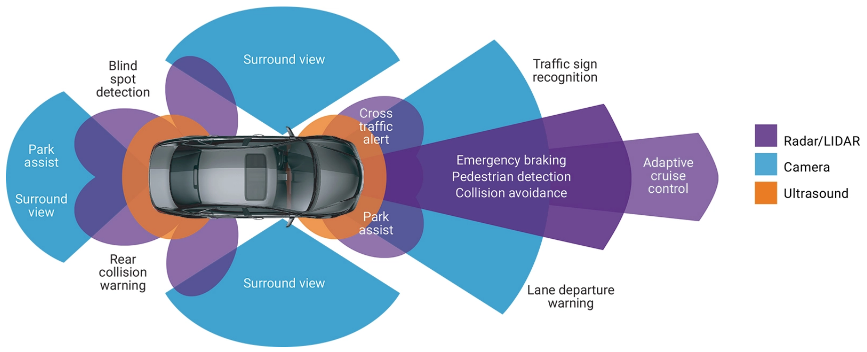


Figure 1.1: Different sensor technologies employed in autonomous driving and their strengths. *Source:* EETimes

There are several methods of constructing a BEV from 2D images, which can be split up into 2 categories (Saha, Mendez, Russell, & Bowden, 2021): Using the explicit camera geometry to construct the necessary transformation (Giles et al.,

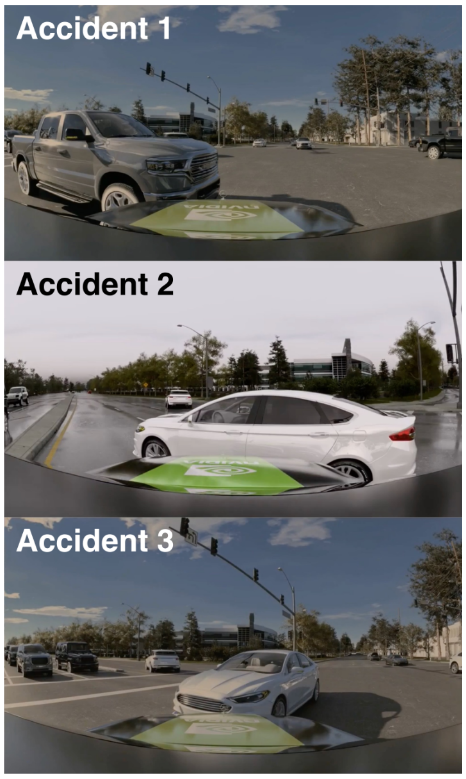


Figure 1.2: Possible simulated accidents starting from the same scenario. *Source:* NVIDIA

2019), or learning the representation implicitly without knowledge of the camera geometry.

This thesis will research the use of BEVs for autonomous driving with the Duckietown platform [1], more specifically, using DuckieBots (DBs) and the Duckietown simulator [2]. DBs mainly rely on a single front-facing camera to observe their surroundings, making the use of a suitable representation of the camera observations more important.

As autonomous driving is a broad subject with many different challenges, this thesis will be focused on crossing unprotected intersections. Giles et al. (2019) have proposed an approach to this problem, specifically targeted at DuckieBots. Their implementation of BEVs is a direct projective transformation from image-plane to BEV-plane. This fairly simple approach leaves plenty of room for improvement.

Hu et al. (2022) propose a method of constructing BEVs which involves learning 3D features from the 2D image before converting to a BEV representation. The

[1] `https://www.duckietown.org/`

[2] `https://github.com/duckietown/gym-duckietown`

improvement of performance using this method shows the importance of using 3D features to navigate a 3D world. Implementing this approach in the solution proposed by Giles et al. (2019), could therefore also show improvement in navigating unprotected intersections with DuckieBots.
Thus the following research question is posed: **To what extent can the navigation of unprotected intersections for DuckieBots be improved by utilizing a Bird's Eye View (BEV) representation created from 3D-encoded image features?**

Along with the research question, this study also aims to answer the following subquestions:
SQ1: Does MILE trained on data from the CARLA simulator (Dosovitskiy et al., 2017) generalize well to the Duckietown simulator?

SQ2: Are the metrics proposed by Caesar et al. (2021) well suited for evaluating intersection navigation in the duckietown simulator?

Chapter 2

Theoretical Background

Creating bird's eye view representations from monocular camera images is not a new field of research and many different methods of achieving this goal already exist (de Marez Oyens, 2022). This chapter will present the theoretical background for the two different methods employed in this thesis. First, in order to understand the implemented methods, a high-level summary of the pinhole camera model is provided. The theory of homography transformations and the lift-splat technique is then outlined, as well as the adaptations to the latter proposed by Hu et al. (2022).

2.1 The Pinhole Camera Model

The pinhole camera model is the simplest version of a camera model, a model which describes the projection of a 3d world onto a 2d plane, such as a camera sensor.

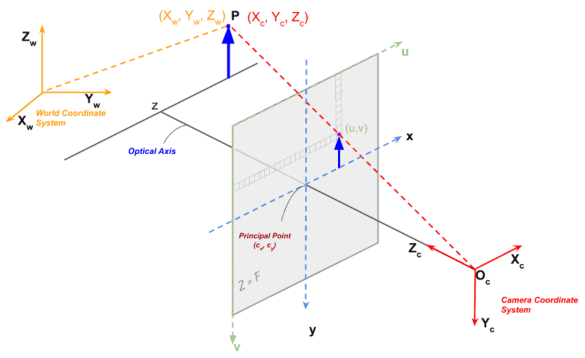


Figure 2.1: The pinhole camera model maps a point P in world space to a pixel (u,v) on the image plane F

2.2 Homography Transformations

When utilizing a pinhole camera model, and assuming the world space to be a flat plane, the world plane and the camera projection plane can be seen as two planes within the same 3D space. A projective transformation between two such planes is defined as a homography transformation. When the exact position of the camera plane in the world space is known, the appropriate homography matrix H for the transformation can be constructed, thus enabling the creation of a homographic BEV representation from an image.

2.3 Lift-Splat

Philion and Fidler (2020) propose the Lift-Splat technique, which does not make assumptions about the world space, unlike a homographic BEV. The images are first encoded with an Image Encoder (He, Zhang, Ren, & Sun, 2016), and then "lifted" to 3D. Lifting entails the per image feature prediction of a categorical depth distribution using a CNN (see Figure 2.2), which enables the encoding of 3D image features. These 3D features are then sum-pooled to BEV space

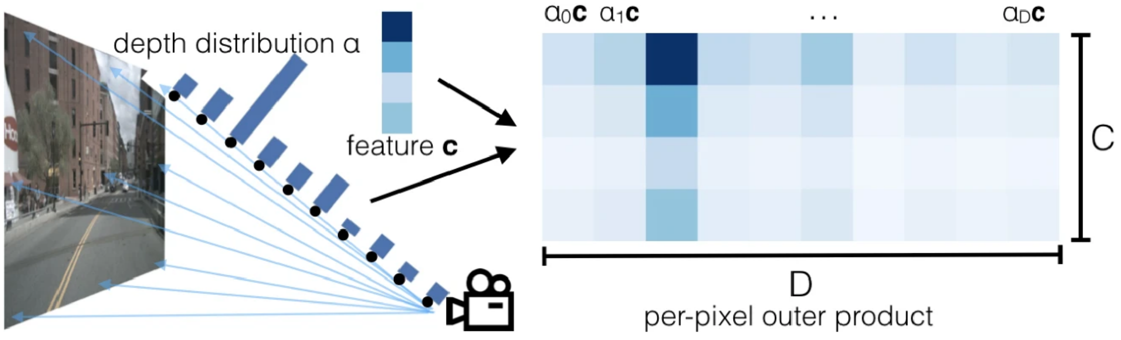


Figure 2.2: "Lift" step of Lift-Splat

Chapter 3

Methods

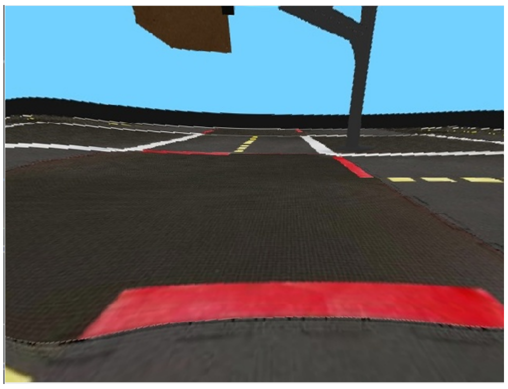
In this chapter, the two methods implemented for creating BEVs from a monocular camera are described, along with their integration with the intersection navigation solution proposed by Giles et al. (2019). Firstly, the method proposed by Giles et al. (2019) is outlined, along with its limitations. Secondly, the method proposed by Hu et al. (2022) is presented. The lane following architecture provided by the Duckietown environment is then outlined, and finally the intersection navigation architecture is described.

3.1 Homographic BEVs

The original method used by Giles et al. (2019) to create their BEV's uses the intrinsic and extrinsic properties of the camera to create a homography matrix. This is then used to perform a projective transformation. The Duckietown development documentation provides a homography matrix H defined as:

$$H = \begin{bmatrix} -4.89775 * 10^{-5} & -0.0002150858 & -0.1818273 \\ 0.00099274 & 1.202336 * 10^{-6} & -0.3280241 \\ -0.0004281805 & -0.007185673 & 1 \end{bmatrix} \tag{3.1}$$

Applying this transformation to a camera input provided by the Duckietown simulator results in a BEV representation as seen in Figure 3.1b. An apparent disadvantage of this approach is its assumption of a flat world plane, which results in the distortion of all objects which are not perfectly flat. This is not a significant issue in the experiments of this thesis, which will be illustrated in Section 4.3.


(a) Example camera input.

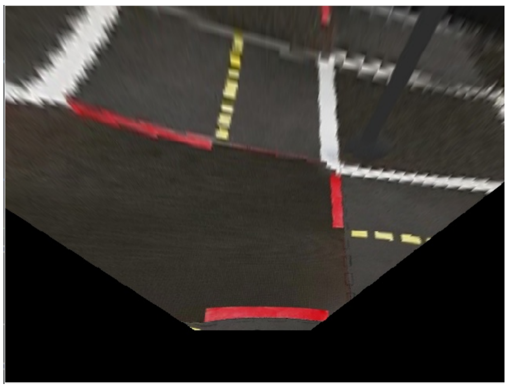

(b) BEV generated using homographic transformation.(Hu et al., 2022)

Figure 3.1: Example of transforming a camera input to BEV space using homography matrix H (Equation 3.1).

3.2 MILE

The method used by Hu et al. (2022) for generating BEV's is an adaptation of the Lift-Splat technique (Philion & Fidler, 2020). The camera input images are first encoded by the Image encoder (Table 3.1), and compressed further to a 1D vector using a pretrained ResNet18 model (He et al., 2016). To construct the segmented BEV, this 1D vector is decoded by the BEV Decoder (Table 3.2), which has a similar architecture to StyleGAN (Karras, Laine, & Aila, 2019)

Layer	Output size	Parameters
Input	3 x 320 x 832	0
ResNet18	[128 $\times$ 40 $\times$ 104, 256 $\times$ 20 $\times$ 52, 512 $\times$ 10 $\times$ 26]	11.2M
Feature aggregation	64 $\times$ 40 $\times$ 104	0.5M
Depth	37 $\times$ 40 $\times$ 104	0.5M
Lifting to 3D	64 $\times$ 37 $\times$ 40 $\times$ 104	0
Pooling to BeV	64 $\times$ 48 $\times$ 48	0

Table 3.1: Image encoder architecture of MILE model

3.3 Lane Following

The Duckiebot environment provides basic lane following behaviour, which navigates the vehicle until it approaches a stop line, and keeps it within the lane

Layer	Output size	Parameters
Input	[512 × 3 × 3, 1024, 512]	0
Adaptive instance norm	512 × 3 × 3	1.6M
Conv. instance norm	512 × 3 × 3	3.9M
Upsample conv. instance norm	512 × 6 × 6	7.9M
Upsample conv. instance norm	512 × 12 × 12	7.9M
Upsample conv. instance norm	512 × 24 × 24	7.9M
Upsample conv. instance norm	256 × 48 × 48	3.3M
Upsample conv. instance norm	128 × 96 × 96	1.2M
Upsample conv. instance norm	64 × 192 × 192	0.5M
Output layer	[8 × 192 × 192, 1 × 192 × 192, 2 × 192 × 192]	715

Table 3.2: BEV decoder architecture of MILE model

markings. When a stop line is detected within 0.1 meters in front of the vehicle, intersection navigation is activated.

3.4 Intersection navigation

The generated BEV representations are then utilized by the intersection navigation model proposed by Giles et al. (2019). First, stop lines are detected by filtering red pixels and clustering them with the DBSCAN algorithm (Hinneburg, 1996). The estimated position and orientation of the Duckiebot is constantly calculated based on the naive assumption that the initial position of the vehicle is straight in front of a stop line. This estimation is utilized to predict the location of stop lines, enabling the classification of the stop line clusters in the BEV based on the nearest predicted stop line. An example of this can be seen in Figure 3.2. If the number of pixels within a cluster is below a set quality threshold, the cluster is ignored.
The intersection is then navigated by following a precomputed trajectory which is mapped to the intersection using the estimated stop line positions. If there are no detected stop lines, the trajectory is based on the predicted stop line positions instead.

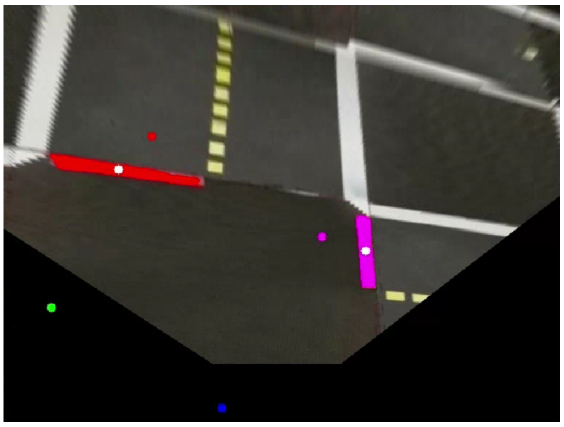


Figure 3.2: Classified stop line clusters in a BEV. The coloured dots represent the predicted stopline locations, each detected stop line is coloured according to the closest predicted stop line, and its center represented by a white dot.

Chapter 4

Experiments

This chapter presents the experiments conducted to evaluate the two methods described in the previous chapter, the metrics used for evaluation, and the results of these experiments.

4.1 Experimental setup

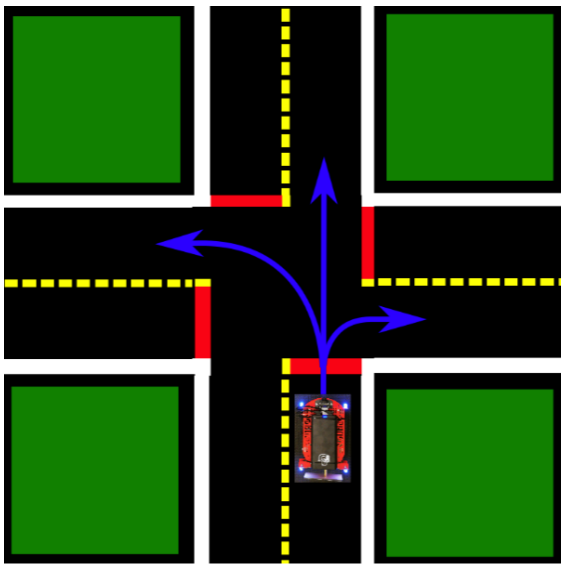


Figure 4.1: Three possible trajectories for navigating the intersection.

Both methods are evaluated on the same experiment, which consists of navigating a 4-way unprotected intersection 20 times per possible trajectory (left, straight, right), as seen in Figure 4.1, from a position close to the stop line. There are no other vehicles or obstacles present in the intersection. To ensure variability between starting positions, the DB is first driven from a randomized position on the road to the stop line using the existing lane following architecture described in Section 3.3. Hu et al. (2022) provide pre-trained weights, trained on the CARLA Simulator (Dosovitskiy et al., 2017), which are utilized to construct a trained MILE model, and no additional training is conducted. The experiments are run with a camera input size of 320 x 832, which is the input size of the pre-trained MILE model. Figure 4.2 shows an example of a camera input during the experiment.

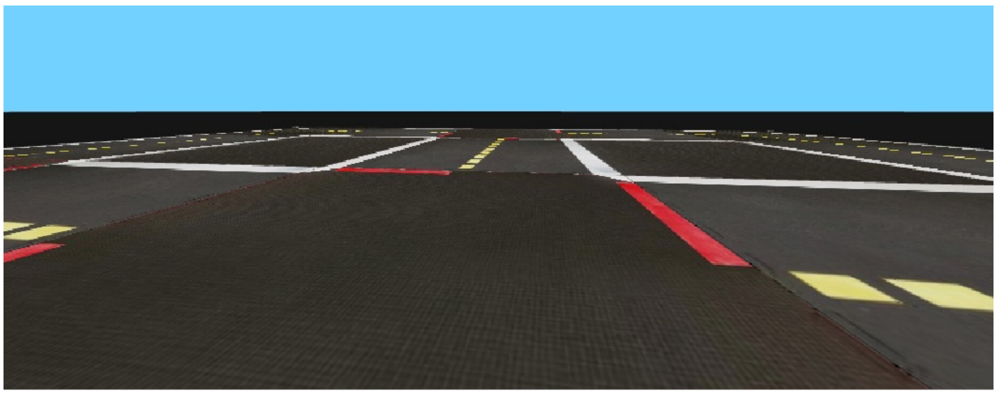


Figure 4.2: Example of camera input utilized for generating BEVs during intersection navigation

During intersection navigation, BEVs are generated and utilized by the intersection navigation architecture to detect stop lines. The estimated stop line positions are then used to reconstruct the position and orientation of the vehicle within the intersection, which in turn is used to calculate the optimal trajectory through the intersection. When there are no clear stop lines detected, the trajectory of the DB within the intersection is estimated based on the velocity and heading of the vehicle, assuming the start position to be straight in front of a stop line.

4.2 Metrics

The success rate for both methods is evaluated by measuring the percentage of intersection navigations that result in the test vehicle exiting the intersection correctly and in the correct lane. In addition, metrics proposed by Caesar et al. (2021) for evaluating the navigation of autonomous vehicles are employed to assess the experimental results. The drivable area compliance and ride comfort metrics are selected as they are applicable to the experiments and do not require an expert's trajectory for comparison. Given the absence of expert driving data for these experiments, it is not feasible to compare the autonomous driving data with that of an expert driver.

Drivable area compliance (DAC) is simply whether or not the vehicle stays on a drivable area (road surface) during the entire navigation. Ride comfort consists of evaluating the minimum and maximum longitudinal accelerations, maximum absolute value of lateral acceleration, maximum absolute value of yaw rate, maximum absolute value of yaw acceleration, maximum absolute value of longitudinal component of jerk, and maximum magnitude of jerk vector. If all of those variables are within thresholds defined by Caesar et al. (2021) (see Table 4.1), the navigation is evaluated as comfortable. The metrics definitions and threshold values are obtained from the Nuplan devkit documentation. Both DAC and Comfort are evaluated as a percentage of test runs which achieved a positive score.

Metric	Threshold
min_lon_accel	$-4.05\ m/s^2$
max_lon_accel	$2.40\ m/s^2$
max_abs_lat_accel	$4.89\ m/s^2$
max_abs_yaw_accel	$1.93\ rad/s^2$
max_abs_yaw_rate	$0.95\ rad/s$
max_abs_lon_jerk	$4.13\ m/s^3$
max_abs_mag_jerk	$8.37\ m/s^3$

Table 4.1: Thresholds for ride comfort metrics, obtained from Nuplan metrics. When all metrics are within their thresholds, the comfort metric is evaluated as *true*.

4.3 Results

The experiments as described in the previous section resulted in the evaluations given in Table 4.2. For both methods, the success rate is equal to the DAC percentage, meaning the vehicle consistently either successfully exited the intersection, or navigated over a non-drivable area. Comfort percentages are 0 for all experiments, meaning for each test run, at least one of the metrics presented in Table 4.1 was not within its respective threshold.

As illustrated by the results presented in Table 4.2a, navigation utilizing Proj-LFI BEVs achieves a significantly lower success rate for the right trajectory compared to the left and straight trajectories.

metric	trajectory Left	Straight	Right
Success rate	100%	100%	55%
DAC	100%	100%	55%
Comfort	0%	0%	0%

(a) Results using Proj-LFI

metric	trajectory Left	Straight	Right
Success rate	85%	100%	90%
DAC	85%	100%	90%
Comfort	0%	0%	0%

(b) Results using MILE

Table 4.2: The evaluated metrics of the experiments for the two implemented methods; Drivable Area Compliance percentage and Comfort percentage.

Navigation utilizing MILE BEVs demonstrates an equivalent success rate for the straight trajectory and a superior success rate for the right trajectory in comparison to Proj-LFI. This is despite the MILE-generated BEVs being non-representative of the actual road surface. Figure 4.3 illustrates that the BEV generated by Proj-LFI clearly represents the intersection, whereas the MILE BEV does not.
As outlined in Section 4.1, when the BEV fails to detect any clear stop lines, navigation is exclusively based on the trajectory calculated from the DB's velocity and heading. This assumes the start position of the intersection navigation to be straight in front of the stop line, which is not necessarily the case. Thus, the results presented in Table 4.2b essentially reflect the outcomes achieved when navigating based on intrinsic vehicle position estimation.

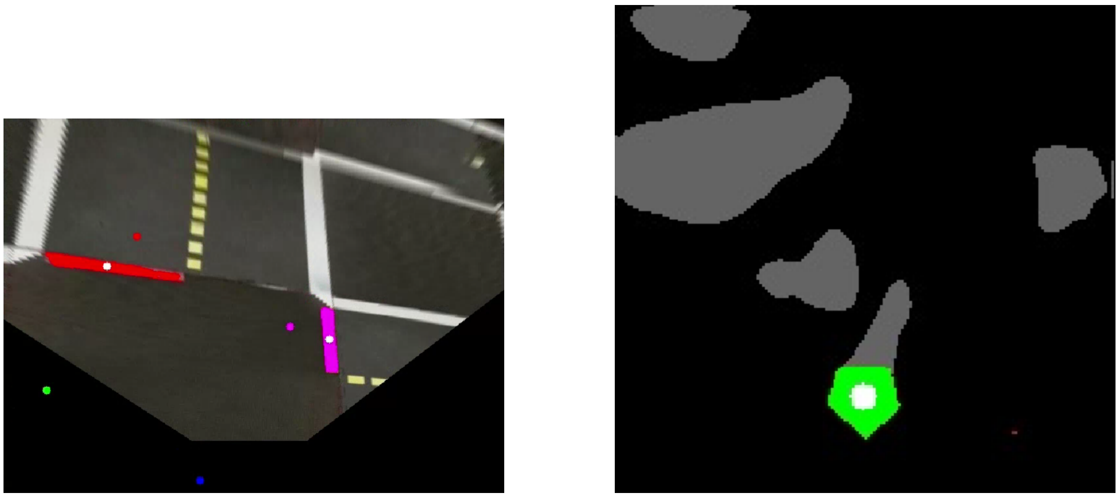

(a) BEV generated by Proj-LFI (b) BEV generated by MILE

Figure 4.3: BEVs from both methods, generated at the same position during intersection navigation. Each estimated stop line is colored according to the closest predicted stop line position. The center of the estimated stop lines are indicated by the white dots

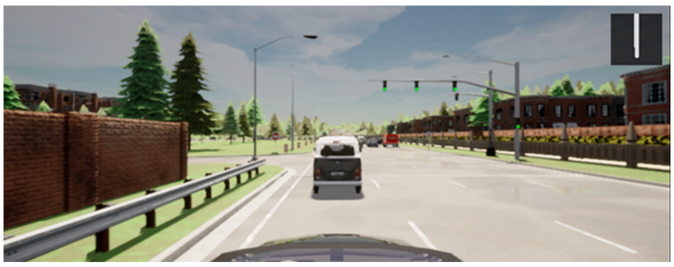
Chapter 5

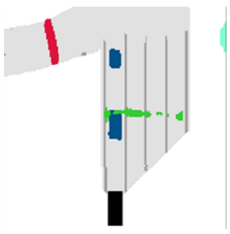
Discussion

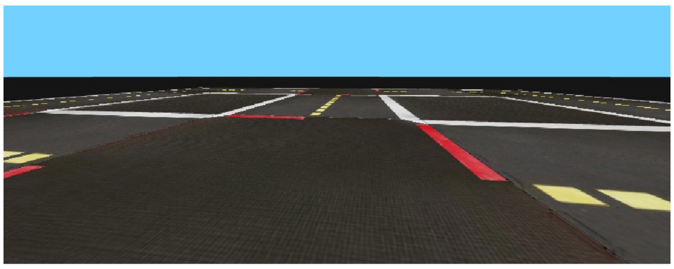
The results presented in Section 4.3 will now be evaluated in relation to the research question and subquestions posed in Chapter 1. Secondly, the opportunities for future work will be outlined.

SQ1: Does MILE trained on data from the CARLA simulator generalize well to the duckietown simulator?
As was clearly illustrated by Figure 4.3b, the BEVs generated by MILE are not a

sufficient representation of the actual intersection. Figure 5.1 demonstrates that MILE is significantly more accurate at representing the road with an input from the CARLA simulator. This outcome is to be anticipated given the model's training solely on data from the CARLA simulator. The colorization of the road surface varies significantly between the input from the CARLA and Duckietown simulator, resulting in MILE barely recognizing any road surface when presented with an input from the latter.


(a) Example camera input from CARLA simulator.


(b) Example BEV generated from (a).


(c) Example camera input from Duckietown simulator.

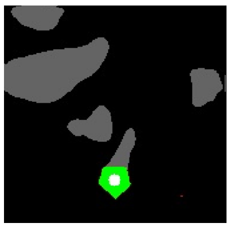

(d) Example BEV generated from (c).

Figure 5.1: Comparison of MILE-generated BEVs from input data obtained from the CARLA simulator (a) and the Duckietown simulator (c). The differences in colors are purely aesthetic, with gray indicating the road surface in both BEVs. White in (b) and black in (d) represent non-drivable areas. The figure highlights the significant difference in the accuracy of MILE's representation of the road surface between the two simulators. (a) and (b) are provided by Hu et al. (2022)

SQ2: Are the metrics proposed by Nuplan well suited for evaluating intersection navigation in the Duckietown simulator?
The metrics presented in Table 4.2 display both methods failing to achieve a single

comfortable ride, according to the comfort metric proposed by Nuplan (Caesar et al., 2021). Per definition of this metric, a navigated trajectory is only deemed comfortable if all of the metrics presented in Table 4.1 are within their respective threshold. This is not the case for a single experiment run, suggesting the intersection navigation in all experiments was uncomfortable according to Nuplan's metrics. However, the thresholds used for evaluating these metrics are based on expert driving behaviour in the CARLA simulator, which is designed with real-world vehicles in mind. This is in contrast to the Duckietown simulator, which is designed with Duckiebots in mind. Duckiebots have a significantly lower maximum speed and acceleration, and a significantly smaller turn radius. Thus it is to be expected that the longitudinal and lateral accelerations of the DB stay within their respective thresholds, and that turns are significantly sharper resulting in higher yaw and jerk metric values than expected from a real-world vehicle.
Adequate evaluation of ride comfort in Duckiebot navigation requires expert driving data collected specifically with Duckiebots to establish suitable thresholds.

RQ: To what extent can the navigation of unprotected intersections for DuckieBots be improved by utilizing a Bird's Eye View (BEV) representation created from 3D-encoded image features?
Despite the limitations of MILE's BEV representation when applied to the Duckietown simulator, the intersection navigation results obtained with this model, as presented in Table 4.2b, exceed expectations considering the non-representative state of the BEV. The success rate achieved utilizing MILE averaged over all trajectories is higher than Proj-LFI's average success rate. These results seem to suggest that using a model that creates BEVs from 3D encoded image features can achieve more generalized navigation compared to Proj-LFI.
However, as outlined in Section 4.3, this is clarified by the understanding that when stop lines can not be clearly detected, intersection navigation depends on the vehicle's internal velocity and heading to estimate its location within the intersection. This demonstrates that even when relying solely on intrinsic vehicle position estimation, satisfactory intersection navigation in the Duckietown simulator can still be achieved, assuming there are no other vehicles or obstacles present in the intersection.
Considering this, the results obtained suggest a different conclusion: achieving successful intersection navigation in Duckietown, assuming no road obstructions exist, does not necessarily require an accurate BEV representation of the surroundings.

5.1 Future work

To conclusively determine whether BEVs generated using MILE could improve intersection navigation in Duckietown, a follow up study could gather training data specifically from the Duckietown environment. This data could then be utilized to train MILE, enabling it to accurately predict BEVs in that setting. Such an approach could help determine the effectiveness of MILE in the Duckietown simulator and potentially pave the way for further improvements in intersection navigation. In order to accurately evaluate the efficacy of MILE in this context, experiments would need to incorporate other vehicles and obstacles on the road, which would thoroughly test the model's capabilities.

Another possible expansion of this thesis could be construct more suitable thresholds for the metrics proposed by Caesar et al. (2021), by analyzing training data with expert trajectories created within the Duckietown environment. This would greatly expand the effectiveness of development for this environment, as such planning specific metrics do not currently exist for Duckiebots.

Chapter 6

Conclusions

The implementation of MILE presented in this thesis does not generalize well to the Duckietown environment, and was unable to generate more accurate BEV representations when compared to a homographic BEV. Despite this, intersection navigation success rates utilizing such limited BEV representations are higher on average when compared to a more representative homographic BEV. This demonstrates that navigating a Duckietown intersection not containing any obstructions is a reasonably simple problem not requiring detailed representations of the immediate surroundings. Employing a pre-determined trajectory from a set start position is arguably a more reliable solution in such cases, as opposed to constructing BEVs from a singular low-resolution camera.

References

Caesar, H., Kabzan, J., Tan, K. S., Fong, W. K., Wolff, E., Lang, A., … Omari, S. (2021). *Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles.* arXiv. Retrieved from https://arxiv.org/abs/2106.11810 doi: 10.48550/ARXIV.2106.11810

de Marez Oyens, P. (2022). Predicting abstract bird's-eye-view representations from monocular camera images using deep learning. https://scripties.uba.uva.nl/search?id=record_28905.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning* (pp. 1–16).

Giles, S. N., Leopoldseder, C., & Wieland, M. (2019). *Project lane following intersection (project-lfi).* Project report. Eidgenössische Technische Hochschule Zürich (28 pages).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hinneburg, A. (1996). A density based algorithm for discovering clusters in large spatial databases with noise. In *Kdd conference, 1996.*

Hu, A., Corrado, G., Griffiths, N., Murez, Z., Gurau, C., Yeo, H., … Shotton, J.

(2022). Model-based imitation learning for urban driving. In *Advances in neural information processing systems (nips).* (paper 662).

Ignatious, H. A., Khan, M., et al. (2022). An overview of sensors in autonomous vehicles. *Procedia Computer Science*, *198*, 736–741.

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 4401–4410).

Philion, J., & Fidler, S. (2020). Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *16th european conference on computer vision (eccv)* (Vol. Part XIV 16, pp. 194–210).

Saha, A., Mendez, O., Russell, C., & Bowden, R. (2021). Enabling spatio-temporal aggregation in birds-eye-view vehicle estimation. In *2021 ieee international conference on robotics and automation (icra)* (p. 5133-5139).