universidade de aveiro

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

Data Centric Machine Learning ops for Autonomous Driving

(Colaboration is censored, due to contratual terms, until 2025)

Supervisor
Miguel Vidal Drummond, [redacted]

Students
Victor Souza (Team manager), 89330
Tiago Coelho, 98385
Frederico Vieira, 98518
Diogo Aguiar, 81020
Leonardo Freitas, 89131

Abstract

This project proposes a new way of obtaining data for training object detection models. The goal is to improve the model's quality by increasing data quality. A new process of synthetic data generation was used to achieve this goal.
For this approach, the game GTA V was used for real world simulation. After driving a car around the map, 2 datasets were taken. One dataset is the raw training dataset and the other one is the raw validation dataset. From the raw training dataset, the frames with empty scenarios (no vehicles) were separated into one folder and the vehicles into another folder. Subsequently, the empty scenes were optimally populated with vehicles, thus giving rise to the synthetic dataset.
Finally, a model (of object detection) was trained with the synthetic dataset and with the original training dataset, separately. Subsequently, both trained models were validated with the same dataset. By comparing the results obtained in the validations, it could be concluded that the model trained on the synthetic dataset was more accurate than the model trained on the original dataset, thus showing that this new approach to data processing is promising to improve object detection in autonomous driving.

Chapter 1

Introduction

Driving is a much needed activity in modern days, people rely on cars to get them fast and comfortably from point A to point B. However, every day, hundreds of thousands of car accidents happen worldwide and at least 90% of those accidents are caused by human error[1], resulting on more than 1 million deaths per year[2]. Part of the solution to this problem may come with autonomous driving cars, since human error gets almost neutralized with it.
Autonomous driving cars run by themselves through sensors and machine learning models (algorithms) that read data from sensors and takes actions, according to the training they had. Even though this concept seems promising, researchers and engineers all over the world struggle to get safe and reliable machine learning models for autonomous driving.
Machine learning translates into 2 concepts, data and models. Models are algorithms that make decisions to solve a certain ML problem, but in order for these algorithms to make good decisions, they need to be trained with good datasets. In the autonomous driving universe, datasets are collections of roads and traffic representations (pictures, point clouds, ...).
When trying to improve models, there are 2 strategies to focus the research on, those being the model-centric, in which the research focus on improving the algorithm itself, and the data-centric, in which the research focus on improving the quality of the data used to train the model. This work uses the data-centric approach, by taking point clouds of the empty environments and 3D scans of cars, into 2 separate datasets and fusing them into a third (augmented) dataset, which will be of better quality than a normal dataset of original scenarios.

1.1 Problem Statement

Using datasets made of original data isn't enough, as explained further in this document (2.2.1), so synthetic data is needed. However, when creating a synthetic dataset for Autonomous driving Machine Learning, it is important to perform such process in the most flexible and realistic way possible. For that, this project approaches the problem from an unprecedented way.

1.1.1 Conventional method

The process that has been used around the world until now is represented in 1.1. Basically, a vehicle captures point clouds with a LiDAR, then, those point clouds are sifted and the data contained in them is labelled. Optionally, that data can be augmented (modified in a way that increases its quality).

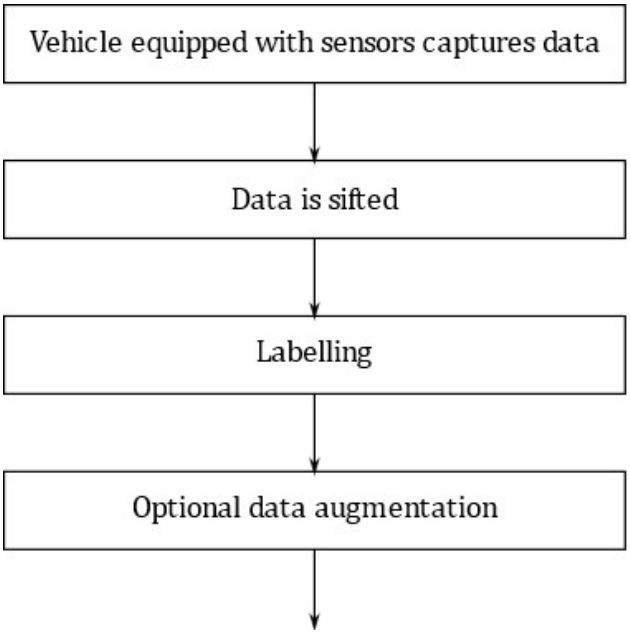

Figure 1.1: Conventional data acquiring method, for autonomous driving machine learning

1.1.2 New method

This project proposes a very simple, but still brand new, way to obtain synthetic data, and it works as in 1.2. This is, point clouds of empty scenes are merged with point clouds of objects (vehicles in this case), thus creating a synthetic dataset.

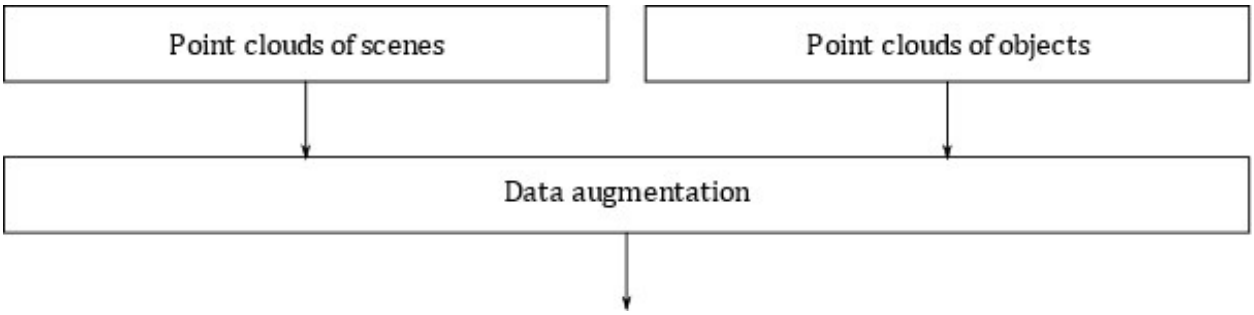

Figure 1.2: New synthetic data acquiring method, for autonomous driving machine learning

1.2 Procedure summary

Tools referred next in this section are detailed under section 2.3.
The procedure begins with the capture of point clouds by circulating in GTA V, using PreSIL[3], consequently creating the the raw datasets (1 for training and 1 for validation). Data from the raw dataset is then separated into 2 folders, scenarios with no cars go to one folder and point of cars, acquired from the raw point clouds, go to another folder. Synthetic data is then created by populating empty scenarios with cars, using our Python scripts. After having a synthetic dataset, PointPillars[4] (ML model) was trained in openPCDet using the different datasets, therefore creating a different trained models. The trained models are tested and then evaluated using IoU and Average Precision. In the end, all trained models are validated with the same validation set and the results are compared between them.

Chapter 2

State of the art

2.1 Background Theory

As the demand for better machine learning models, focused on autonomous driving, increases, researchers try to find solutions to improve them. The most widely adopted strategy to approach this has been the Model-centric one.
That strategy focuses on creating more and better algorithms, in a search for the most accurate and precise implementation for this autonomous driving problem. It's a strategy that works, though, researchers are reaching an almost optimal algorithmic solution already, a large amount of those solutions are open source and still, results are nowhere near the desired.
In this project, there is a shift in strategy. It approaches the problem from a Data-centric perspective, in which the research's focus turns to the Data, i.e., the goal becomes to improve the quality of the datasets, used to train the object detection models (algorithms).
In order to achieve this, Point Clouds were captured synthetic data was created and used.
This project was motivated by Bosch and has a very high industrial interest, since quality data is extremely valuable these days.

2.1.1 Point Cloud

Point Cloud is the type of data used for these datasets. A Point Cloud is a set of dots in a 3D space, which represent a scenario and/or object in 4D, the 4th dimension being the reflectiveness of a material, i.e., bright colored dots represent very reflective objects and black dots represent almost non-reflective ones.
In a real world situation, a car is equipped with a LiDAR on it's roof and it is responsible for taking point clouds from the car's surroundings. For the sake of this project, a simulation environment is used and the game, GTA V, is the platform for that simulation. PreSIL is used to capture the information from GTA V. PreSIL does use a 4th coordinate, however, in this case, it represents an ID that identifies at which object the point, from a point cloud, belongs to.

Figure 2.1: Torc's car mounted with LiDAR

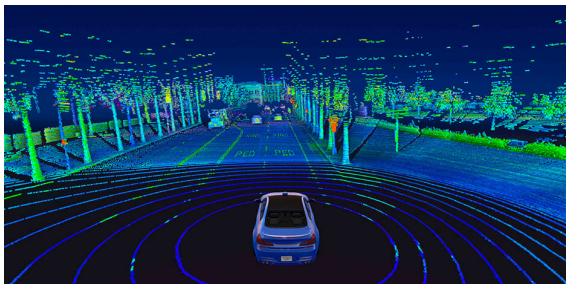

Figure 2.2: Point cloud taken from a Velodyne's LiDAR

2.1.2 Synthetic Data

In this project's case, synthetically populated point clouds are created from meshing original point clouds of empty scenarios (empty roads) with original point clouds of vehicles. The label of the new point cloud has to contain the car's info.
Follows an example, where a point cloud of an empty scenario is populated with a point cloud of a car.

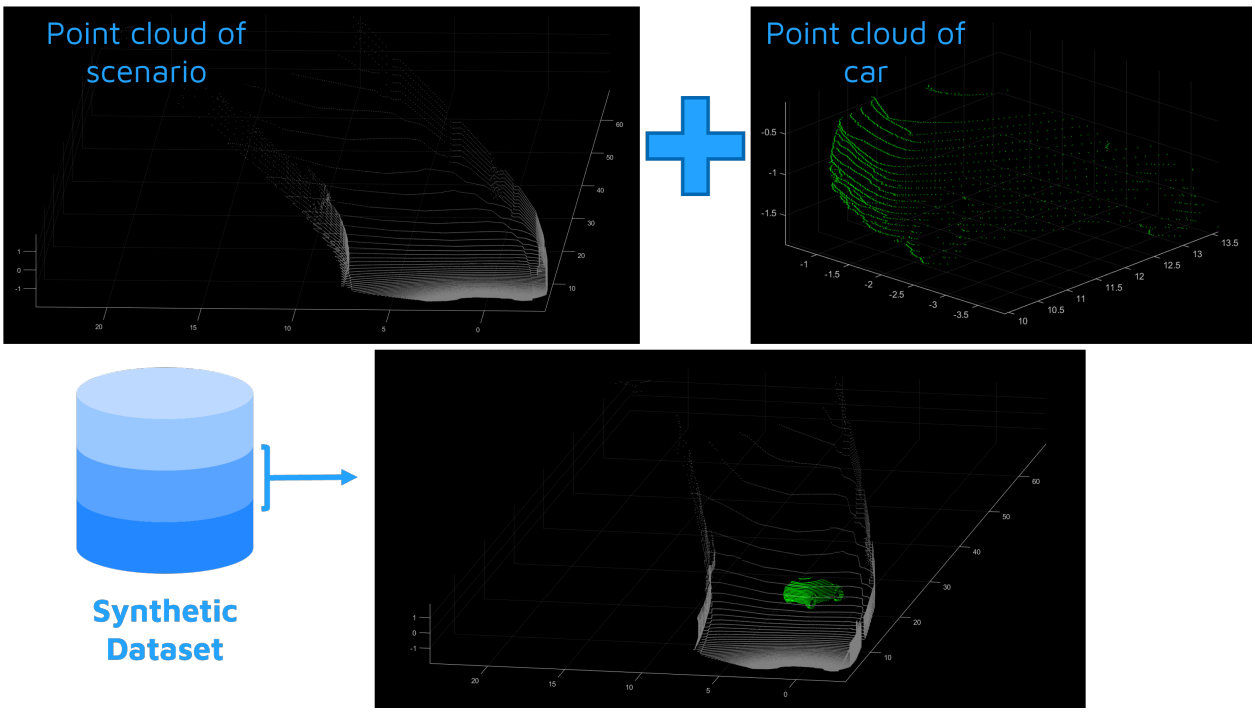

Figure 2.3: Process of synthetic data creation

2.2 Related Work

This project's theory is based of other previous work.
Many tools and procedures were also adopted from external sources and it is important to mention them, hence this section.

2.2.1 Thesis: Evaluation of data augmentation techniques in 3D object detection for self-driving vehicles

Written by Xavier Santos, the thesis[5] reports the first-ever detailed quantification of the impact that the inclusion of augmented samples in a dataset has in 3D Object Detection, in the context of autonomous driving. Xavier's work concludes that, datasets consisting of only augmented data, present better results than any other type of dataset.
The following tables represent the same trained models tested in 3 different environment difficulties, easy, moderate and hard. For these tests, 12 different datasets were used to train the model (which means that 12 differently trained models were used). The first column of each table represents the number of augmented scenarios used in a certain dataset, while the first line represents the number of original scenarios inside those same datasets. The float numbers in the colored cells represent the calculated average precision (2.4.2) result for each test and dataset.
Table reading example: A model trained with a dataset constituted of 50 augmented scenarios and 25 original scenarios (75 scenarios in total for this case), tested on an easy difficulty ambient, had a calculated average precision of 85.2144%.

Easy					
Augmented\Original	**0**	**25**	**50**	**75**	**100**
0	✗	65.9853	69.4208	68.9562	73.2588
25	✗	84.4446	84.5747	85.4031	✗
50	✗	85.2144	86.1979	✗	✗
75	✗	86.1419	✗	✗	✗
100	90.4832	✗	✗	✗	85.1988

Moderate					
Augmented\Original	**0**	**25**	**50**	**75**	**100**
0	✗	54.786	58.5927	58.185	61.4443
25	✗	72.3696	74.0089	74.6415	✗
50	✗	74.442	75.3042	✗	✗
75	✗	73.4236	✗	✗	✗
100	79.546	✗	✗	✗	75.1763

Hard					
Augmented\Original	**0**	**25**	**50**	**75**	**100**
0	✗	52.206	55.5152	56.7972	58.069
25	✗	70.0475	70.3622	72.0315	✗
50	✗	70.0822	72.8043	✗	✗
75	✗	70.7387	✗	✗	✗
100	75.7203	✗	✗	✗	72.9046

Figure 2.4: AP@40 3D object detection test results for cars in each difficulty. (Table taken from Xavier's Thesis).

As seen in 2.4, datasets made of 100 augmented scenarios and 0 original scenarios show better results than any other dataset with different combinations of scenarios, independent of the ambient difficulty. The present project takes that premise and tests it in a practical environment (using GTA V).

2.2.2 Thesis: Simplified 3D object detection for self-driving vehicles based on the removal of background points

In this thesis[6], João Gomes proves that removing background points from point clouds decreases the computational effort involved in 3D object identification while increasing its average precision.
For that, the author used 2 datasets, a real one (from KITTI) and a simulation one captured from GTA-V, using PreSIL. The GTA-V one is formatted similarly to KITTI rules, since it is the best configuration to approach OpenPCDet.
He then uses OpenPCDet to train the PointPillars model with both datasets separately and proceeds to evaluate the results using Intersection Over Unit.
(Referred tools in this paragraph are presented in the next section)
João's thesis was very useful for this project on a procedural way, since it goes through many shared processes, from data collection and data labelling, to model training and result evaluation.

2.3 Tools

Inside this section, information about specific tools used in the course of the project, can be found.

2.3.1 PreSIL

Precise Synthetic Image and LiDAR (PreSIL[3]), is an open source tool that creates a precise LiDAR simulator within GTA-V, which returns, among other things, point clouds of the game. PreSil point clouds are the original data in this project.
The authors of PreSIL also made available an already existing dataset of over 50,000 frames acquired through this tool. A fraction of those frames were used as validation set.

2.3.2 KITTI

KITTI[7] is a high quality, open source, real world dataset, capture from a VW Passat station wagon in Germany.
For this project, it was used as a mold for datasets labelling and size, the sizes being the following:

- Training set - 3712 frames;
- Validation set - 3769 frames;
- Test set - 7518 frames.

It was also used for some OpenPCDet debugging, since it is highly compatible with such software.

2.3.3 OpenPCDet

OpenPCDet[8] is an open-source project for LiDAR-based 3D object detection that supports different LiDAR-based perception models, PointPillars being one of them.
OpenPCDet facilitates the training and testing of different models used on the KITTI benchmark.
Since it is a popular tool and supports PointPillars, it was used to train and test the model in this work and evaluate it precision-wise.

2.3.4 PointPillars

Simple definition, PointPillars[4] is a machine learning model (algorithm) for autonomous driving, that uses point clouds. The model is also very popular for being fast, hence why it was chosen for this project.
More specific definition, PointPillars is an encoder network that uses PointNet to learn a point cloud representation organized in vertical columns (pillars). This pillar's features are used to predict the oriented 3D bounding boxes and allow end-to-end learning by only relying on 2D convolutional layers and, consequently, it can outperform other models when it comes to speed.

2.3.5 FPGA - Xilinx (Future work)

Field Programmable Gate Arrays (FPGAs) are programmable boards, that can be implemented in a car, for the purpose of this project and carry the trained machine learning model. Xilinx was chosen for it's portability and compatibility with machine learning models for autonomous driving.

2.4 Metrics

This section is intended to present the metrics used for result evaluation.

2.4.1 Intersection over Union (IoU)

IoU is used to calculate the percentage of overlap between the ground truth (GT) bounding boxes and the predicted ones. The calculation of this metric is done as follows

$$IoU = \frac{GTvolume \cap predictedvolume}{GTvolume \cup predictedvolume} \tag{2.1}$$

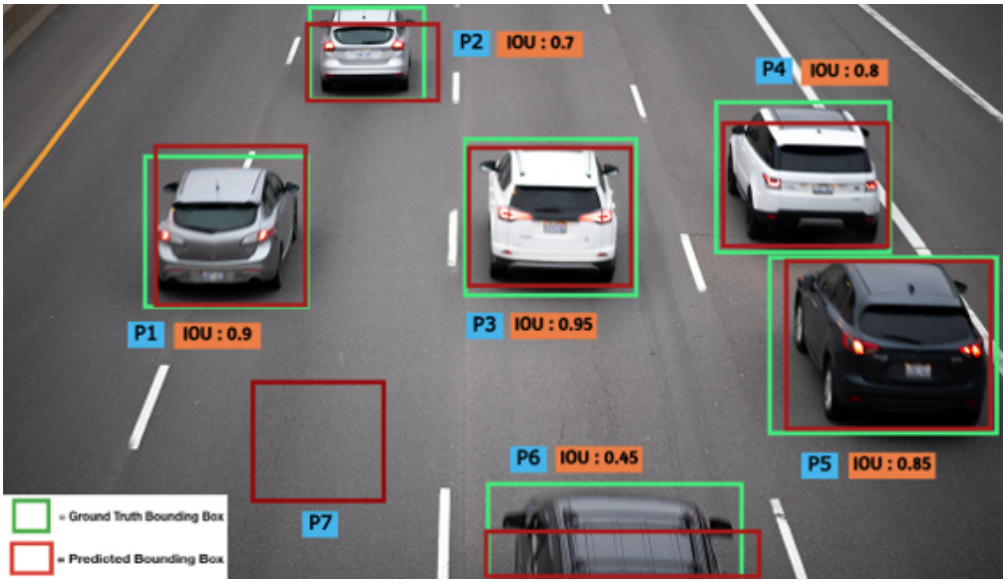

Figure 2.5: Representation of IoU

2.4.2 Average Precision

Average Precision (AP) summarizes the Precision-Recall curve as the weighted mean of precision achieved at each IoU threshold.
Interpolated Average Precision is one way to calculate it and is the one used to evaluate the model on the KITTI's dataset.
It is calculated as follows:

$$AP = \frac{1}{|R|} \sum_{r \epsilon R} \rho_{interp}(r) \tag{2.2}$$

$\rho_{interp}(r)$ is calculated as follows, where $\rho(r)$ represents the precision at recall r.

$$\rho_{interp}(r) = \max_{r':r' \geq r} \rho(r') \tag{2.3}$$

Chapter 3

System Requirements and Architecture

3.1 System Requirements

The goal of this project, is to document, for the first time ever, how to use the new data augmentation process, in a practical environment simulation, when working with machine learning for autonomous driving.
In order to better understand the problem and it's possible solutions, several meetings, with different personnel, took place, from lectures (about autonomous driving and data augmentation), Industry presentations and discussions (about possible approaches), to material financing.
Namely:

- Supervisor, Miguel Vidal Drummond, gave lectures for insights on the problem and the new solution to be implemented. Summoned meetings to present new people involved in the field.
- Master Students, Leandro Alexandrino, João Gomes, Xavier Santos, traded information, documents an experience with tools.
- UA Professor and [redacted] Arnaldo Oliveira, gave insights on FPGAs and their possible use in the context of this porject.
  He recommended the Xilinx FPGA and provided support during deployment.
- Event NEXT, a partnership between UMinho and Bosch that boosted several research projects, many of them in the autonomous driving field.
  On one particular presentation, "NEXT, Driving Tomorrow", they showed a car mounted and running with an FPGA, which revealed that the deployment phase of this work could be done.
  The engineer Pedro Barbosa from Bosch also recommended the use of a Xilinx FPGA and Bosch themselves provided the FPGA that was used in this project.

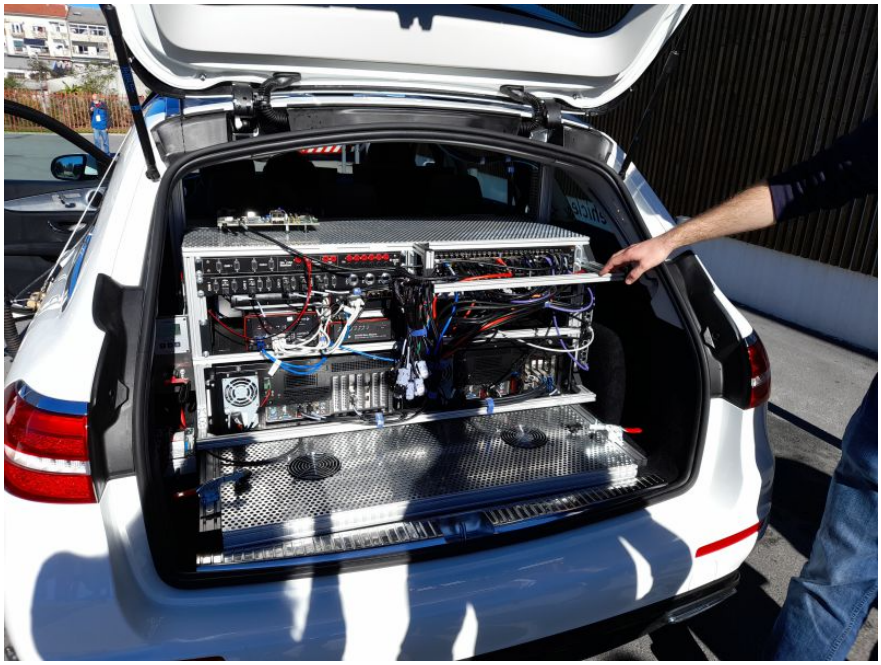

Figure 3.1: Bosch's car, at NEXT, mounted with an FPGA in the back

3.1.1 Functional requirements

The current project is a MLOps work, which, contrary to DevOps, is really hard to plan and predict, since the progress depends on results.

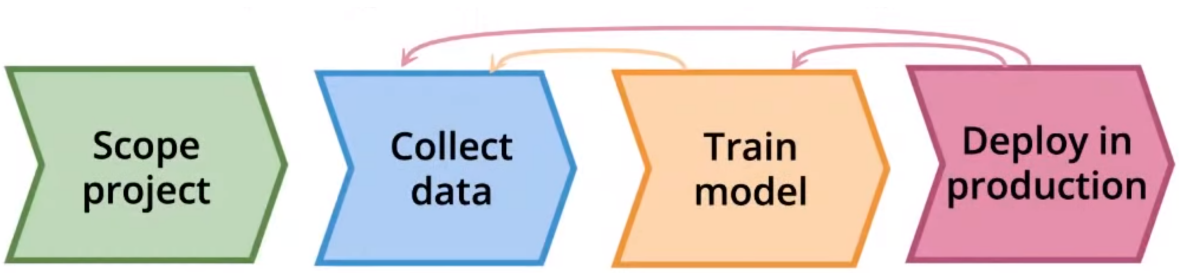

Figure 3.2: MLOps usual project evolution

The functional requirement is to improve results (over time) of the model trained with the test set. Models trained with the original dataset and models trained with the validation set are used as benchmark for lowest desirable result and highest desirable result, respectively.

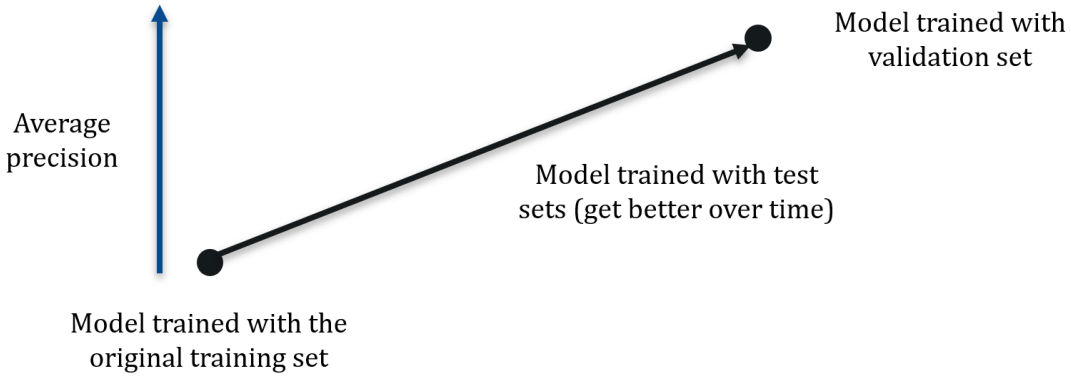

Figure 3.3: Representation of the functional requirement

Chapter 4

Methodology

4.1 Data science standard methodology

This is a data science work, therefore a data science methodology was used. Such can be summarized in figure 4.1.

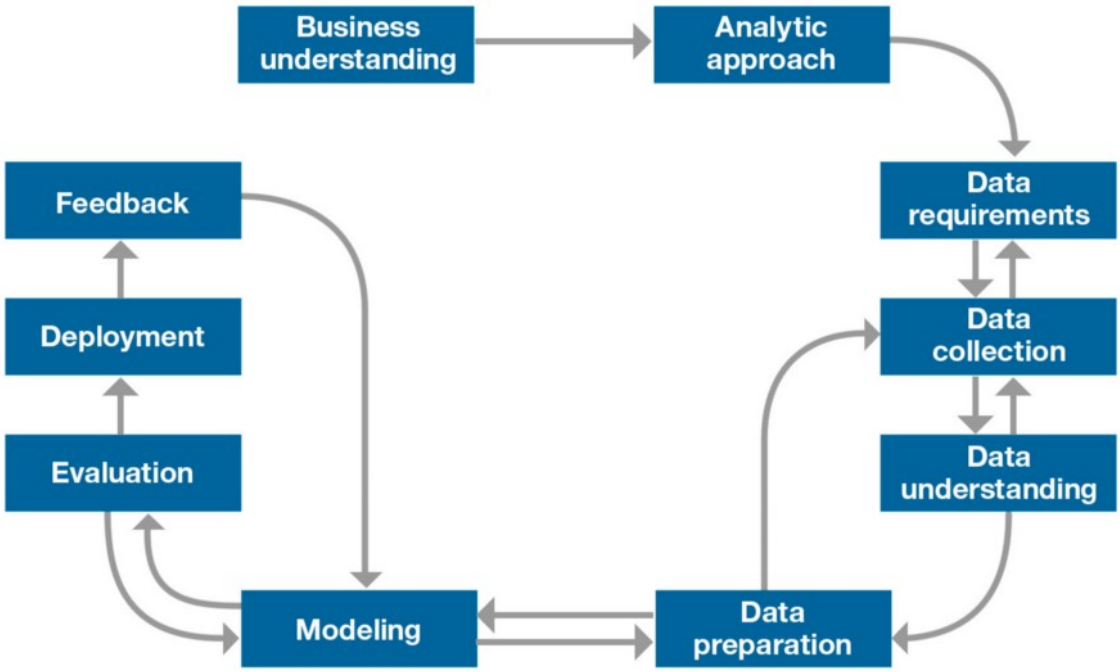

Figure 4.1: Data science methodology

Explaining 4.1:

- Business understanding consisted on learning about the autonomous driving world and how it works.
- Analytic approach was the understanding of the problem and solution proposed by the supervisor.
- From the proposed solution there was a requirement gathering for data to be collected.
- Data was collected.
- Data was then prepared (sorted and augmented) to insert in the model.
- The model was trained (Modeling).
- The trained model was evaluated, and according to the evaluation, data was collected again. (This cycle was repeated several times).
- Finally, the trained model was deployed into an FPGA.
- Feedback would be useful for future work.

4.2 Detailed methodology

This section contains all the technicalities of the project.

4.2.1 Data gathering and data treatment

First step was to capture data, for that, we installed GTA V (the simulation environment) and PreSIL[3] (the tool to extract data). PreSIL has many problems that had to be solved, those technicalities aren't in this report, but can be found on an attachment named "PreSIL_installation_guide.pdf" being sent alongside it. Next step was to start capturing data, to this end, we had to start GTA and then start PreSIL. PreSIL would automatically run across the map, for hours and save all data (frames and their information) on a specific folder called "Object", this folder had, among other things, screen captures of each frame, the point cloud of each frame and a label (containing important info about objects) of each frame. These three were connected by having the same file names (excluding extensions). We repeated the process until we got 7000 frames, which were divided into 2 datasets of 3500 frames each, one of them being the raw training set and the other one being the raw validation set. In order to get the synthetic datasets, we first had to have empty scenarios (with no cars) and individual point clouds of cars, separately.
To this end, we created a script that would read the labels from all frames of the raw training set and copy to an "empty_scenarios" folder, all point clouds corresponding to labels where no vehicles were described. However, for the labels where vehicles were described, all points from the point clouds were examined and those belonging to vehicles were grouped together by ID, i.e, by vehicle and then written into new point clouds, separately, thus creating a point cloud for each car. These point clouds (.bin files at this point) were then placed in a folder called "cars" and named after their ID and their mother frame number (the reason why will be ahead).
There was a problem with the car point clouds though. For some unknown reason, PreSIL assigned IDs to points that didn't belong to the vehicles, this meant that most point clouds of cars, had outlier points. Example in 4.2

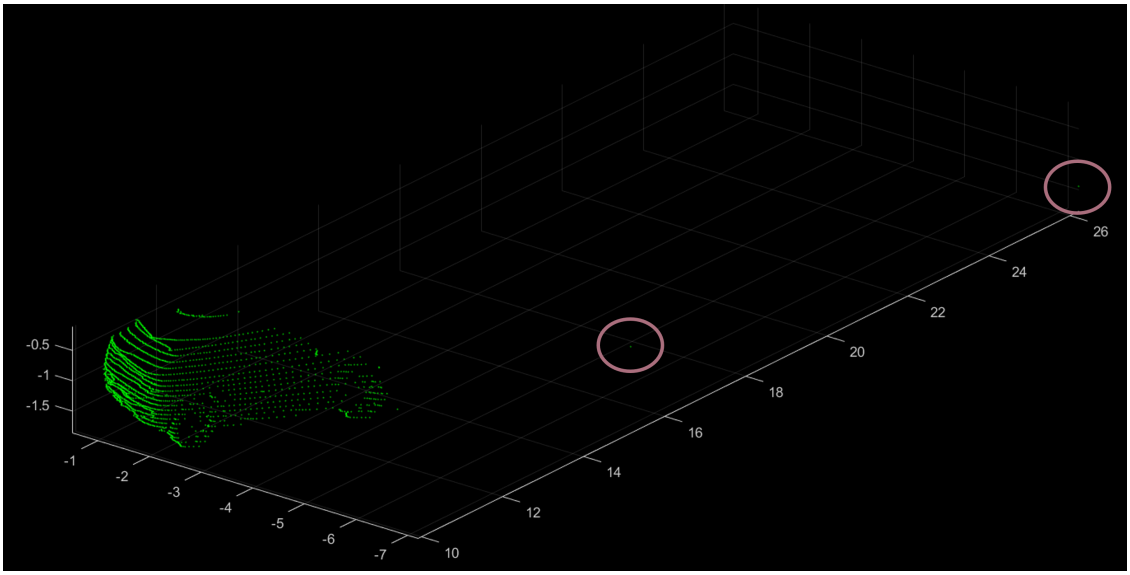

Figure 4.2: Example of car point cloud with 2 outlier points (circled in pink)

To solve this issue, we relied on labels, from where we got information about vehicle height, width, length and coordinates of floor central point. With this information, we calculated a circumferential perimeter around the car and points outside this circumference or over and under the car (with a threshold of +/-0.5), were not considered. The math was the following:

$$r = \sqrt{\left(\frac{width}{2}\right)^2 + \left(\frac{length}{2}\right)^2} \tag{4.1}$$

where "r" is a hypotenuse but also the radius of the circumference that will be used .

(xc,yc) being the 2D coordinates of the car's center point and (x,y) being the coordinates of the point being tested.
If:

$$(x - xc)^2 + (y - yc)^2 - r^2 > 0 \tag{4.2}$$

Or If:

$$(x > xc + height + 0.5) or (x < xc - 0.5) \tag{4.3}$$

It means that the point doesn't get included in the car's point cloud.

Point clouds corrected, it was then time to create the synthetic dataset. First approach was to simply add the points from car point clouds to the scenario point clouds and update the label of the empty frame with the information from the new added cars. However, only adding the points isn't a good solution, since cars would be out of bounds and flying, like in 4.3.

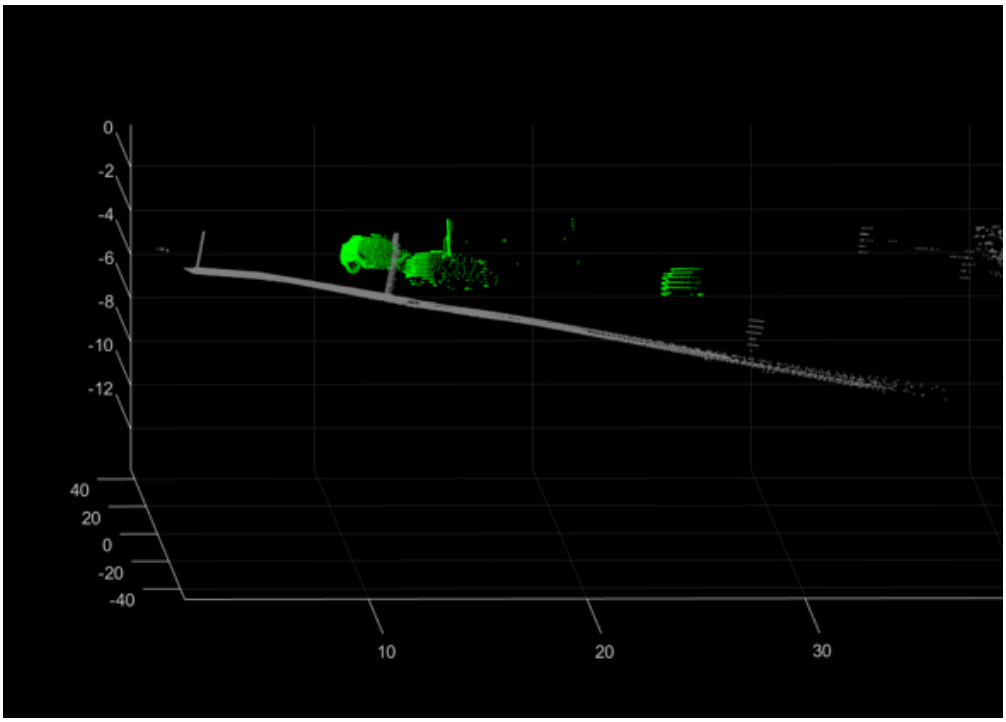

Figure 4.3: cars (in green) flying over road (in grey)

The solution was to read the coordinates from the points under the car, find the lowest z coordinate, get the difference between the car's floor and that point and then translate all the points in the car by that difference.

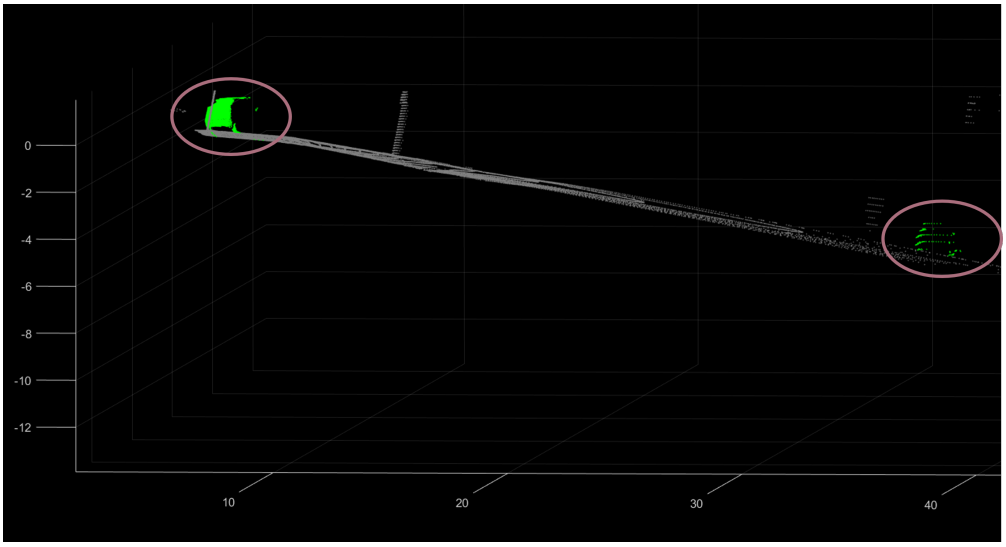

Figure 4.4: Cars now placed on the ground

The figure 4.4 shows that the problem was solved. This solution also partially solves the problem of having cars out of bounds, because the cars are only placed if there are points under them and places out of bounds don't have points. Another problem came with cars being overlapped and to so solve this problem

we used the radius calculated with 4.1 and cars that had a distance between centers, bigger than the sum of their radius, weren't placed. The math for this means that the following conditions has to be true:

$$\sqrt{(x1 - x2)^2 + (y1 - y2)^2} > (r1 + r2) \tag{4.4}$$

With all these problems solved, we were finally able to create the synthetic datasets. We created 2 synthetic datasets with 3500 frames each, one had 8 to 15 cars per frame and another had 15 to 23 cars per frame. Cars were chosen and placed randomly, complying, of course, with the previously mentioned rules.

4.2.2 Model training and validation

The next stop was then to train and validate the models with the datasets we had and we just relied on OpenPCDet[8] to do so. First step was to place PointPillar[4] on OpenPCDet, then set the amount of frames desired to train and validate (3500 for each) and turn off a data augmentation option that OpenPCDet has by default (because it would interfere with the premise of this project). Finally, we gave our validation set for OpenPCDet to use for validation and gave our other datasets (raw training set, synthetic dataset V1 and synthetic dataset V2) for it to train with PointPillar and validate separately. OpenPCDet and PointPillar devide cars into 3 categories of detection difficulty, that rely on the kitti[7] format, i.e., using values of truncation (percentage of the vehicle outside the limits of point cloud), occlusion (Fully visible, Partly occluded, Difficult to see) and bounding box heigh, in the following way:

- **Easy**: Min. bounding box height: 40 Px, Max. occlusion level: Fully visible, Max. truncation: 15
- **Moderate**: Min. bounding box height: 25 Px, Max. occlusion level: Partly occluded, Max. truncation: 30
- **Hard**: Min. bounding box height: 25 Px, Max. occlusion level: Difficult to see, Max. truncation: 50

Figure 4.5 has a good graphic representation of such metrics.

Figure 4.5: Example of easy(1), medium(2) and hard(3) cars

Chapter 5

Results

After validation, results are presented for the three categories (easy, medium and hard) as an average precision of the intersection over unit (IoU) between the bounding boxes predicted by the trained models and the real bounding boxes from the validation dataset (ground truth). The final results were the ones depicted in 5.1.

Trained with\Car difficulty	Easy	Medium	Hard
Raw training dataset	93.12%	85.77%	80.87%
Synthetic dataset V1 (8 to 15 cars per frame)	+0.79%	+2.14%	+0.05%
Synthetic dataset V2 (15 to 23 cars per frame)	+0.58%	+2.52%	+1.96%
Validation dataset (cheat)	+2.54%	+6.67%	+6.92%
Cropped dataset (Only easy cars, for comparison) **NOT SUCCESSFUL**	-2.52%	-10.25%	-17.37%

Figure 5.1: Results for the different trained models, in comparison to the model trained with the raw training dataset

One model was trained with the same dataset as the validation set (cheat), just to show the maximum potential that we could get from a trained model. Another model was trained with a dataset only containing easy cars, just to show that bad results would happen if we weren't careful enough with the creation process of the synthetic datasets. Finally, by analysing the results in 5.1, we can notice that the synthetic dataset with 8 to 15 cars per frame, trained a better model than the raw training set and the synthetic dataset with 15 to 23 cars per frame, trained an even better model that could predict hard and medium and hard cars with much more precision.
From these results, we can conclude that this new approach to data processing is, in fact, very promising for improving object detection in autonomous driving.

Chapter 6

Future Work

Something we tried to implement in this project was porting these models to the FPGA and due to a lack of time and problems with frame resolution, we couldn't quite make it. However, we were able to train a model with the Kitti dataset and have it run on the FPGA. So, even though we couldn't accomplish this extra step, we think it would be interesting to see how it would work. Also, this project was produced in a simulation environment, but regards a very real problem, so the next big step we would like to see, would be to have this process being ported to real data and real world scenarios.

Bibliography

[1] Bryant Walker Smith. *HUMAN ERROR AS A CAUSE OF VEHICLE CRASHES*. URL: http://cyberlaw.stanford.edu/blog/2013/12/human-error-cause-vehicle-crashes.

[2] World Health Organization. *Road traffic injuries*. URL: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[3] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. *Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception*. URL: https://arxiv.org/abs/1905.00160.

[4] Alex H. Lang et al. *PointPillars: Fast Encoders for Object Detection from Point Clouds*. URL: https://arxiv.org/abs/1812.05784.

[5] Xavier Santos. "Evaluation of data augmentation techniques in 3D object detection for self-driving vehicles". MA thesis. Portugal: University of Aveiro, 2021.

[6] João Gomes. "Simplified 3D object detection for self-driving vehicles based on the removal of background points". MA thesis. Portugal: University of Aveiro, 2021.

[7] Yiyi Liao, Jun Xie, and Andreas Geiger. "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D". In: *arXiv.org* 2109.13410 (2021).

[8] OpenPCDet Development Team. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. https://github.com/open-mmlab/OpenPCDet. 2020.