Design Report

Solid-State LiDAR to Detect Cyclists

MINDHASH × University of Twente EEMCS

11 November 2022

Group 6

Jasper van Amerongen	s2384442	
Sebastiaan Hofstee	s2300257	
Omar Elkady	s2389541	
Remy Benitah	s2247372	
Malek Assaad	s2374544	

Supervisor

Dr. Duc V. Le

Word Count: [~18000]

Table of Contents

Introduction

As an integral part of the programme, students of the bachelor of Technical Computer Science at the University of Twente, Enschede have proposed, designed and implemented a product for a client of their choosing, with the intent of displaying mastery of the project design track. During a ten week period, Jasper van Amerongen, Sebastiaan Hofstee, Omar Elkady, Remy Benitah and Malek Assaad, hereinafter referred to as Group 6, have partnered with MINDHASH in materialising a portable solid-state LiDAR (SS-LiDAR) module and corresponding machine learning (ML) model to detect and count cyclist in dedicated bike lanes.

MINDHASH is a company focussed on developing their clients' ideas using hardware and software. Some of their clients include municipalities interested in gaining insight into traffic usage and (mis)conduct. To this end, a common practice is to deploy pneumatic tubes across the road and use spikes in internal pressure to detect crossing vehicles (Brosnan et al., 2015). However, MINDHASH is interested in finding a way to incorporate LiDAR and ML to accomplish the same goal of counting traffic, and bicycles in particular.

Given that MINDHASH has in-depth knowledge about the applications and implementations of rotational LiDAR (R-LiDAR) and corresponding machine learning models, in this document and the rest of the project this knowledge will be assumed. However, MINDHASH have indicated that solid-state LiDAR is a new area to them and thus do not have existing models. This is where they see the opportunity for Group 6 to find new insights.

Project Objectives & Scope

Stakeholder's Problem

Briefly mentioned in the introduction, current practices for detecting and counting vehicle throughput in a given lane is achieved by detecting changes in air pressure in pneumatic tubes across the road over which vehicles pass. This is a readily available, cheap and easily deployable solution. However, it has some drawbacks: This method is inaccurate especially when deployed across multiple lanes or lanes where overtaking is prevalent; insights other than throughput - such as size, velocity or direction cannot be measured; the tubes and other machinery are inherently exposed to wear and tear due to physical contact and moving parts (Brosnan et al., 2015). MINDHASH believes that LiDAR in combination with machine learning can provide a solution that circumvents these drawbacks, whilst providing a more accurate result. For those reasons they looked to start an exploratory project to see what the capabilities and limits are of SS-LiDAR in this application. As they put it themselves:

> *"Municipalities and traffic consultants are interested in the amount of traffic on roads given the time of day as this can indicate a certain level of safety. (…) At Mindhash we believe that LiDAR, and especially the solid state variant, can provide an affordable and more accurate alternative to existing methods for counting traffic."*

Support for LiDAR

As previously mentioned, the existing solution to this problem has many drawbacks that can be eliminated through the usage of a sensor such as a LiDAR. However, it could be argued that something such as a camera would provide for an even more cheap and robust solution. This is not the case as cameras have a few main issues which the LiDAR is able to circumvent. Firstly, cameras provide a major concern for privacy. This is a product being developed by a private company for the usage of municipalities and there would need to be high amounts of consideration for privacy with the usage of a camera. Based on the results of this project, and the resolution of the data being collected, it is not possible to identify individuals with the LiDAR which eliminates any privacy concerns. Furthermore, cameras are highly susceptible to weather conditions and visibility issues such as snow, rain, and lack of light. The LiDAR package which is being used includes settings that already account for this and because the sensor uses point cloud data it does not suffer from visibility issues.

Project Objectives

Based on the written project description and on the initial meeting between Group 6 and Lasse Licht and Sierd Meijer from MINDHASH on 9 September, we have identified two objectives to this project:

1. **Design and implement an integrated SS-LiDAR system to detect cycle lane traffic across a bidirectional cycling lane and a means of displaying real-time insights.**

   MINDHASH wants to explore the capabilities of an integrated SS-LiDAR system, based around the Livox Mid-40, in the application of capturing cyclist traffic data. The cycle lanes are standardised bidirectional lanes as found in The Netherlands and the greater Enschede area in particular. This system should be able to capture, preprocess, interpret and conclude the captured data in real-time (see Requirement Specification). The gained insights should be displayed digitally, preferably on a reactive mobile web app.

2. **Gain insights into a robust method for counting cycling lane traffic using a SS-LiDAR, emphasising on its feasibility with existing R-LiDAR trained models and different hosting platforms.**

   MINDHASH is experienced in applying R-LiDAR and possesses machine learning models to work with this device. However, seeing that they want to expand their expertise into SS-LiDAR applications as well, they want to know if they can use and adapt if necessary their existing models to use SS-LiDAR.

Project Scope

A formality of defining the project's scope is covering the project's uncertainty and complexity (Shenhar, 2001), as well as its knowledge building potential and strategic fit (Nobelius, 2001).

Uncertainty & complexity

Shenhar (2001) has outlined a framework for determining a project's technological uncertainty and its complexity in a two-dimensional plane. The team has implemented a system using a custom model, hosted on partially in the cloud and with data from an SS-LiDAR. These technologies are not widespread and require a certain level of expertise. For this reason both our hosting strategy and model strategy lift this system to high-tech (type C) technological uncertainty (Shenhar, 2001). The system can be characterised as a scope level 2 system (Shenhar, 2001).

Knowledge building potential & strategic fit

Another scope-defining metric is the project's knowledge building potential and strategic fit. These are the two primary factors of predicting project success. A project's knowledge building potential encompasses how much new knowledge the client can gain from a project. The strategic fit determines how well a project fits in the strategy - short and long term - of the client. Based on how well this project fits this two-dimensional scale determines whether to cancel, redirect, outsource or keep a project. (Nobelius, 2001).

As discussed in the Stakeholder's Problem, MINDHASH is actively looking for in-house research into the deployment of SS-LiDAR for traffic observation. As a hardware-software company looking for new opportunities and addressing the needs of their clients is a continuous strategy. As a matter of fact, this project was issued primarily for this reason. Based on that it is clear to see that this project has both high knowledge building potential, as seen by the lack of experience with SS-LiDAR, and strategic fit, since this project will be directly competing with existing solutions based on pneumatic tubes.

Requirements Specification

Having narrowed down the scope of this project, we will move on to formalising its details in the form of a requirements specification. Requirements can be organised on a goal-design scale (Lauesen, 2002), with four levels of specificity: goal-level, domain-level, product-level and design-level. Goal-level requirements refer to the requirement of the system to meet the project's objective or other high-level business goals. Domain-level requirements specify certain supporting tasks that the system will realise. Product-level requirements define individual functionalities of the system and design-level requirements describe details of the system's interface. For this project, we have extended Lauesen's scale (2002) with a fifth level of requirements, namely architecture-level requirements. These include certain architectural and otherwise technical requirements that play a crucial role in the implementation and therefore should be considered. Furthermore, each requirement has been assigned a priority. These priorities are labels derived from the MoSCoW method (Clegg & Barker, 1994). This is an abbreviation for 'Must', 'Should', 'Could' and 'Won't'. We have defined our MVP to consist of all requirements labelled 'Must' and we will omit 'Won't' labelled requirements. In addition to a priority badge, the requirements have been decorated with a badge indicating if they were met at the end of this project.

Goal-level Requirements

R1 Implementation of the system shall explore if SS-LiDAR is a viable option for replacing existing solutions for cycling lane traffic monitoring. Must ✓

Domain-level Requirements

R2 The system shall be able to identify bicycle lane traffic, being bicycles, scooters and pedestrians, with an accuracy of 80% or more. Must ✓

R3 The web application shall display data analytics of observed traffic. Must ✓

R4 The system shall be able to provide data analytics in near real-time, with a maximum delay after detection of one minute. Should ✓

R5 The web application shall support remotely accessible data analytics. Must ✓

R6 The system shall be remotely manageable. Could ✓

Product-level Requirements

R7 The system shall be portable. Must ✓

R8 The system shall be able to run off a battery, which itself can be charged. Could ✗

R9 The system's enclosure, including the LiDAR module itself and accompanying hardware, shall be weatherproof to IP34 or above. Could ✓

R10 The system shall be connected to the Internet. Must ✓

R11 The web application shall have the option of displaying analytics (absolute amount, average frequency, etc.) lively in the current session or displaying analytics over a previous time frame (last week, month, year etc.). Should ✓

R12 The web application requires authentication to operate. Should ✓

Design-level Requirements

R13 The web application shall be implemented with Mantine. Must ✓

R14 The web application shall maintain a certain level of design coherence. Should ✓

R15 Temporal data analytics will be displayed as a chart, with variable temporal scopes. Should ✓

Architecture-level Requirements

R16	The system shall use a Livox mid-40.	Must	✓
R17	The system shall run on a Raspberry Pi or otherwise similar microcomputer.	Must	✓
R18	The system shall use an FPGA board.	Could	✗
R19	The system shall use a machine learning model to infer cycling lane traffic objects.	Must	✓
R20	The web application shall be hosted in Firebase.	Must	✓
R21	The web application shall be developed with the React framework.	Must	✓

Governance

This chapter will shine a light on who is responsible for what aspects of the project, which stakeholders are important and at what level their cooperation and feedback is required, what project management approach is suitable and which communication media will be used throughout the project for communicating internally and with the stakeholders.

Responsibilities

The members of group six have varying skill sets. To maximise the result of this project, it is sensible to play on each other's strengths. In doing so, it follows naturally that Sebastian Hofstee, with prior experience in AI and ML consisting of a research paper and current enrollment in a ML Master's course at the University of Twente, takes the lead in achieving the first objective of this project, which is the creation of a working model used to detect and classify bicycles in point cloud data frames. Sebastiaan, Remy Benitah and Malek Assaad have worked with each other previously and did so in good cooperation. It is therefore that we believe that Remy and Malek are well suited to aid Sebastiaan in this track.

Jasper van Amerongen is experienced in building front-ends, in particular with React and backed by a Firebase environment, and has an eye for design. He used this strength to take the lead on the development of a web application for displaying the insights gained from the LiDAR data. Furthermore, he attended a minor focusing on project management and project proposals in particular, so this experience will be put towards writing this and other documents. Omar Elkady will work together with Jasper to research different hosting and model strategies as is prescribed in objective 2. Omar and Jasper have prior cooperation experience and it is for this and aforementioned reasons we believe they are a good fit for developing data visualisation and achieving objective 2.

This split does not imply that the members of Group 6 will not work closely together. Objective 1 and 2 are inherently tied together by the fact that objective 2 is likely to determine the strategy to be used to achieve objective 1. The split in Group 6, however, enables us to execute tasks in parallel. This is especially useful in cases where attendance to a problem by all five members is not required.

Some tools are also needed during the development process such as one for data collection before training an ML model and a LiDAR visualizer. Malek, Omar and Sebastiaan worked on these tools to aid in the development of the final system. Remy's time was very importantly spent labelling about 600 frames of point cloud data to help the group achieve a high AOS and mAP on the ML model.

With both project objectives matched to subsets of Group 6, responsibility for to be determined soft deadlines and successes lie in the hands of the corresponding people. To ensure everyone makes ends meet, a group contract has been drawn up and signed. This contract defines responsibilities on an objective level, rather than outlining individual tasks, as these are hard to define upfront (see Appendix A). This will be discussed further in the Project Management Approach. Since prior cooperation has been established, friction is not expected. However, this cannot be ruled out and is outlined in the contract as well.

Stakeholders

As has been demonstrated in the Introduction, this project is an internal and exploratory one with no business operations directly attached or dependent. This has the benefit of having very few stakeholders. The primary stakeholder is Lasse Licht, CEO and founder of MINDHASH. He has commissioned this project at the University of Twente and in turn to Group 6. Another two equivalent stakeholders are Sierd Meijer and Nadu Rebeja. Both are employees at MINDHASH and are expected to further develop this project or use findings in insights in other projects once completed. A fourth stakeholder in this project is Le Viet Duc. He is a researcher with the EEMCS Pervasive Systems group at the University of Twente and supervisor to this project. His stake in this project is purely supervisory and there is no tangible interest for him in terms of the outcome of this project.

To demand an appropriate level of interest and contribution of this project's stakeholders, we can order them in an Interest-Influence matrix (Mendelow, 1981). This matrix defines four quadrants along two axes. Each quadrant prescribed a level of bilateral engagement between the team and stakeholder:

- **Highly interested, highly influential** stakeholders should be managed closely and actively engaged and great effort must be made to satisfy them.
- **Highly interested, less influential** stakeholders should be kept informed. Active engagement is required and often these stakeholders are the ones to work out details of the project.
- **Less interested, highly influentia**l stakeholders should be kept satisfied. Active engagement is not required, but the team should focus on satisfying their needs.
- **Less interested, less influential** stakeholders should be monitoring the project and fill a more observatory role.

Traditionally a CEO or an equivalent role should be labelled as less interested, but highly influential as they are often not the experts to the project, but will see the result of its success or failure. MINDHASH CEO Lasse Licht would be better labelled as a highly interested and highly influential stakeholder, since he has quite some expertise on this topic and has appointed himself as point of contact. WIth similar logic it follows that other MINDHASH employees are highly interested, but less influential stakeholders. They are looking to continue this project afterwards and are experts in their fields. But of course the failure or success of this project is not their responsibility. Finally, Le Viet Duc is less interested and less influential, since his role is to supervise this project from an academic perspective and is other than that completely detached from this project and its successes.

Project Management Approach

Part of the proposed workflow has been laid out in terms of responsibilities. Another important factor to consider is the project management approach. Taking the responsibility and task division into account, one quickly looks at an agile project management approach. Agile is popular because it will bring about the most efficient way to execute software projects in terms of costs, quality and time under continuously changing circumstances and requirements (Ciric et al., 2019). This project will be executed by a small and nimble team, working closely together. Moreover, all members of this team have worked with scrum in the past as part of the bachelor's programme. A common scrum methodology is a sprint-based method, where design and product-level requirements can be revised and implemented in two to three week sprints. Given that the scope and requirements of this project are not expected to change much plus the relatively short execution time (about eight weeks), a sprint-based method does not make a lot of sense. Therefore, this project will be managed around an agile kanban method. In layman's terms, one long sprint. This has the benefit of an agile methodology, but allows for more fluid execution across different length timespans. Agile requires a product owner to be appointed. Naturally, MINDHASH's Lasse or another to be appointed employee will fulfil this role.

A common tool to manage software projects is Atlassian's Jira. Since we are opting for a kanban methodology, Jira is a good choice because of its interactive kanban board. Moreover, all group members are familiar with the software. Jira is accessed by all five group members through an educational licence.

Bloch et al. (2012) have determined four dimensions of project success: strategy and stakeholder; technology and content; building teams and capabilities; and project progress. Implementing their work, we would like to ensure that we stay on top of these dimensions. This gives us a formal and verifiable metric for measuring project success. For now, let's address each dimension in context. Strategy and stakeholders have been covered in Stakeholders. We have given plentiful thought to our strategy and we have also identified our primary stakeholder and product owner to be involved on multiple levels. The technology of this project is new to most of us, especially LiDAR. This might form an obstacle, since its trivial aspects may take us more time than it would an expert. As discussed in Responsibilities, we believe Group 6 to be a very capable team, based on prior working experiences and expertise on relevant topics. This makes us confident in building a capable team. Lastly there is project progress. This can not be measured as easily as would have been with a sprint methodology, as then requirements and goals could be set ahead of a sprint, multiple times during the project's life span. We measured project progress in ratios between expected time and spent time on requirements. It is hard to define a mitigation strategy in case we fall behind. Risks and their mitigation strategies will be discussed further in Risk & Feasibility.

Communication

WhatsApp & Microsoft Teams

Digital communication between the members of Group 6 primarily took place on WhatsApp and Microsoft Teams. For both media private groups have been created. Microsoft Teams is the medium used for online meetings, whereas WhatsApp is reserved for all other matters of communication. Being able to have online meetings is not a matter of convenience as much as a necessity. Malek Assaad was abroad and Omar Elkady has only been able to attend physically from 19 September on.

Slack & Google Meet

For communication between them, Group 6 employed the software MINDSHASH uses internally. These are Slack and Google Meet. For quick questions, planning meetings and otherwise non-urgent matters Slack is ideal, since all employees and group members have access to each other. Because the MINDHASH office is in Hengelo, commuting there is often rather time consuming. Therefore and for the same reasons as mentioned above, options for online meetings are necessary. MINDHASH is used to using Google Meet for their internal meetings and stand-ups and to that end, this project will employ Google Meet as well.

Risk & Feasibility

Risk Analysis

Projects are risky. They cost money, time and other resources. This allows a chance that they go over budget, don't meet deadlines or will fail altogether. For these reasons it is important to acknowledge possible risks and ideally pose mitigation strategies if these risks fall through. In this section we will omit project risk, reputational risk and change risk. Project (budget) risk is irrelevant for the reason that no budget is specified, the hardware is already allocated and not used in different MINDHASH projects and Group 6 is not financially compensated. Reputational risk is also minimal, since after completion MINDHASH can choose to deploy, cancel or further develop this project in-house without third party knowledge. For the same reason, we do not cover change risk; the risk of this project disturbing current (business) practices.

Human risk

Human risks are those that might harm the project by human error or the absence of a key individual halting progress across the board. In the latter case, this should not form a hard problem. The members actively worked on various parts of this project in parallel, but not one task has sole responsibility. This means that we can assume that sickness or leaves of absence would slow, but not halt, said tasks. In case a point of contact at MINDHASH is unavailable, we assume other employees to be able to (temporarily) fulfil the role of product owner. This risk needs no defined mitigation strategy for the aforementioned reasons.

Another human risk is that of man made errors in documentation or code. In the case of documentation, we expected at least three and ideally five group members to proofread all documentation that is shared with MINDHASH and vice versa to ensure clear and understandable communication between all parties. In the case of development errors, this is harder to mitigate.

Scope creep risk

A risk hardly influenced by sickness or other human error is scope creep risk. This is the risk of the scope of the project, its objectives and/or its requirements pushing work beyond the estimated amount and resulting in the project going over time or over budget. Often this is caused by changing or unclear requirements that push the budget or deadlines of a project out of bounds (Kendrick & Tom, 2015). We have mitigated this by keeping close contact with stakeholders and establishing well-defined requirements. Extra care has been taken when altering goal or domain level requirements and required confirmation from all requirements owners.

Black swan risk

In every project there persists the risk of unforeseen events altering the course of the project or its success in one way or another. We call these events black swan risks. Black swan risks are almost exclusively external to the project, like the COVID-19 pandemic or the Russian invasion of Ukraine this February. These events in particular do not pose a threat (not anymore, in case of COVID-19). Another pandemic or a surprise return of COVID-19 would not impact this project too much in terms of implementation. Working remotely is almost de facto, after all. In that case, however, testability will almost vanish when less people will make use of the bicycle lanes we would be testing on.

Impact & Feasibility Audit

Requirement Feasibility

Requirements are only valuable to a project's success if it is feasible. Especially when the stakeholder or any other requirements owners are not sufficiently proficient in the topic, it could happen they demand something that cannot be achieved at all, or within the set time frame or budget. To this end, it is important to audit the feasibility of the set Requirements. Ideally this happens early in the requirements specification. Established - but not met - requirements may lead to a lack of trust in the implementing bodies.

The requirements that we have established are subjective and abstract by nature. This makes an objective audit of their feasibility rather difficult. Currently, the only tool we possess to assess this is common sense. Luckily the stakeholders and group 6 hold a certain expertise on the different matters. Sebastiaan, being proficient with ML, has provided many insights that we were able to take into account. With his experience with web applications and cloud hosting, Jasper covered feasibility insights over those respective requirements. Moreover, the requirements have been set up in collaboration with MINDHASH and they did not see any major problems that may halt this project down the line. This lets us be confident in our requirements specification and do not expect major hiccups in implementing them.

Time Feasibility

For the same reasons as we considered the requirements feasibility of this project, we will look at the feasibility of the time planning. Though it cannot go unmentioned that this is rather more difficult. It is hard to define a time planning for the realisation of the system, both its hardware and software. This brings a certain level of unpredictability and hurts the feasibility of this project. To ensure we stay on top of things, close collaboration with MINDHASH was required and as issuers and experts, we trusted them in part to avoid scope creep drawing out our planning. It has previously been determined that the web app is not particularly complicated, no new or otherwise unfamiliar or immature technologies will be applied and the web app was liberally planned out, with an extra week to spare. We do not see any feasibility problems arising from the web app's planning.

Machine Learning & Sensing

Subsystem Overview

The data pipeline begins at the embedded system, the computer to which the LiDAR setup is connected by wire. The Webapp is connected to the embedded system through wireless connection or more precisely through a firebase database. Between the user and the three actors, calls and returned messages travel to provide full functionality. The following sequence diagram shows how the user and other systems interact between one another and the sequence of operations.

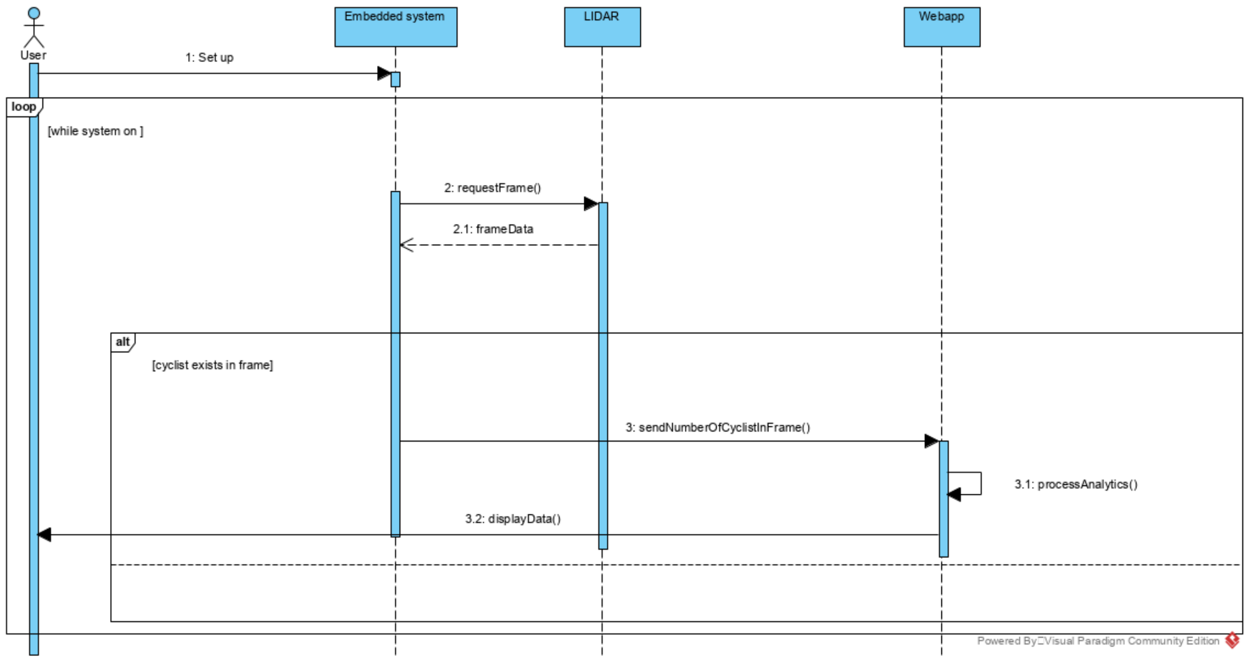


Figure 1: Sequence diagram for interacting with the system

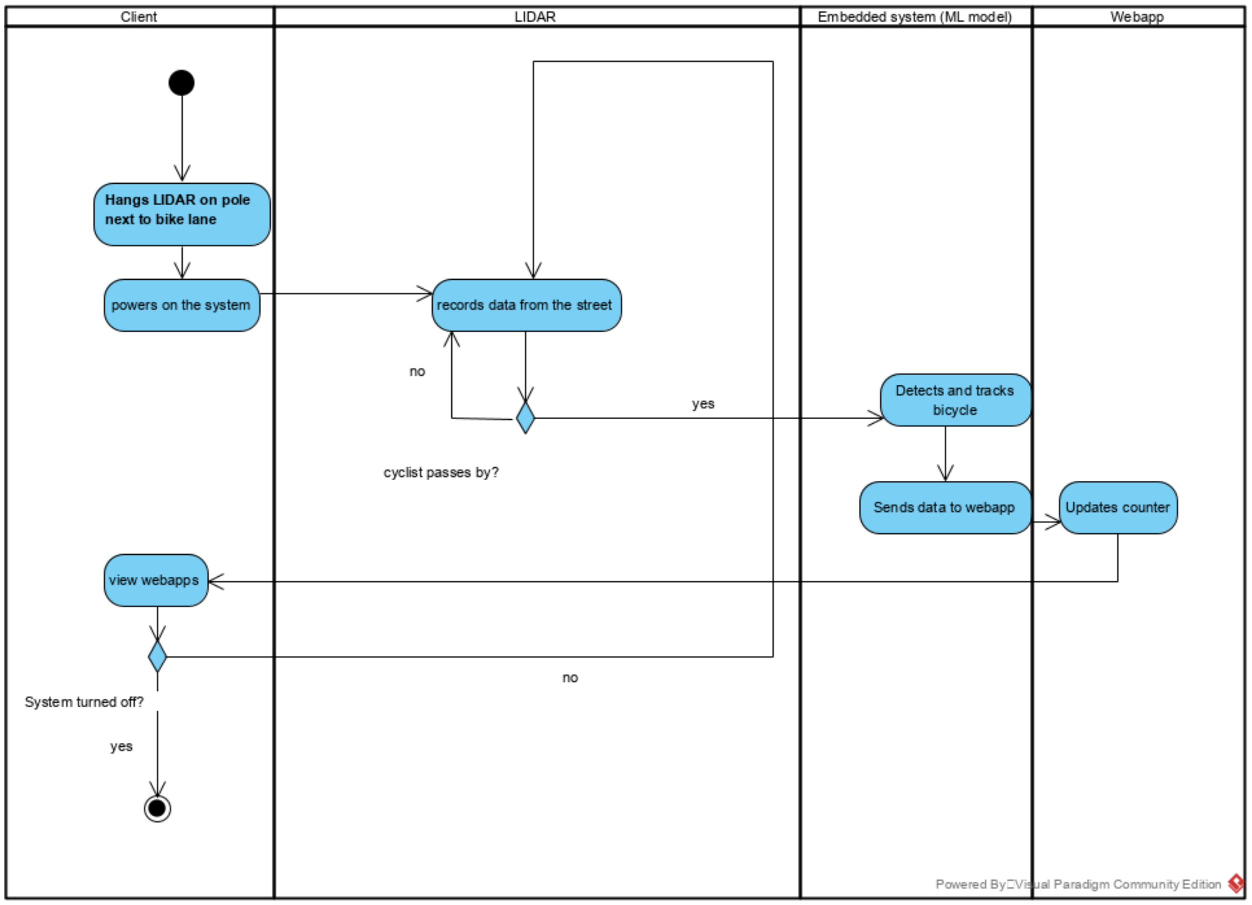

Client
LIDAR
Embedded system (ML model)
Webapp
Hangs LIDAR on pole next to bike lane
powers on the system
records data from the street
no
yes
cyclist passes by?
Detects and tracks bicycle
Sends data to webapp
Updates counter
view webapps
no
System turned off?
yes
Powered By Visual Paradigm Community Edition

Figure 2: Activity diagram for interacting with the system

Figure 2 shows the activity diagram for how the user/client interacts with the system and how decisions are made by the system throughout the process. What is notable is that there is not any network traffic when no cyclists are detected by the LiDAR sensor. Whenever there is a cyclist, the data is then passed to the database through a firebase database.

Data is collected in 0.5 second intervals from the lidar sensor (about 50,000 points) then stored in a file on the computing device. This frame is then run through the model for classification and detection. The result from the model is then sent to the current session in the Firebase so that the web application can display the results live.

Scientific Background & Contribution

We propose and display the feasibility of a method by which it is possible to detect and classify bicycles with a solid state LiDAR module. Furthermore, in the process of doing so, we have created a valuable dataset containing 620 labelled data frames, each consisting of approximately 50.000 data points, in which bounding boxes have been drawn for all bicycles in a frame. Such a solid state LiDAR bicycle dataset is unique in the field. Apart from this, a solid state LiDAR point cloud collection tool was created for the Livox Mid-40 unit, which could be used by researchers around the world.

Thirdly, we have shown that high Average Orientation Similarity (AOS) and mean average precision (mAP) scores, higher than those reported in the original paper of the approach used (Lang et al., 2019), can be obtained with the proposed architecture with only a limited dataset available.

Lastly, we have shown that with this data, it is possible to create a framework providing valuable insights regarding, for this use case, the detection and counting of bicycles on a bicycle lane.

Design Choices

When designing the system, one key aspect considered was the relation between the sensor and the traffic lane. Prior to data collection or testing, it was known that the Livox Mid-40 is limited to a 38.4 degree cone starting from about one meter away from the sensor. These are some limitations of using solid state lidar as opposed to its rotational counterpart. Three initial designs were proposed as can be seen in figure 3.

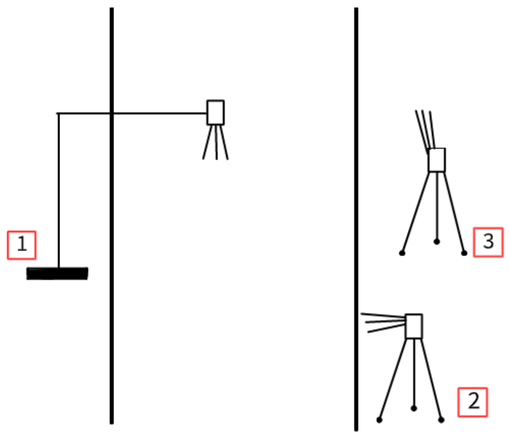


Figure 3: Sensor Placement Design

After testing the sensor using various positions such as the ones seen in the figure above, it was determined that the third design would be the most suitable. This is because the first obstructs traffic, and the second only allows for a very narrow picture of the traffic lane. The third design provides a long tunnel of vision pointed down the lane. This allows for points to be generated of the objects that are within point cloud data generated. Furthermore, this positioning was considered as the most adaptable to a real environment. It allows for simple and mobile installation of the device as long as it is provided with power.

After having decided on this design, the next steps were to refine the positioning of the sensor even more. In order to ensure quality and consistency, a standard was set for the positioning of the sensor. After some testing it was determined that the sensor would be rotated 67 degrees from the traffic lane. Additionally, the sensor would be 100 cm from the ground and 10 cm from the edge of the lane. This was decided in order to optimise the cone of vision of the observed traffic.

Data

As it became clear that it was necessary to create a model, the team began the process of data collection, labelling, and processing. In order to generate value from data it is important to ensure that it is of high quality (Cai & Zhu, 2015). The team made many considerations during each step of the process to achieve quality data. This is important because it will be used to train and test the model for the purpose of real world functionality. Therefore, it must reflect the data that the system will eventually operate on.

Data Collection

Data was collected using the design proposed in the placement of the sensor above. The sensor is able to capture data at a rate of approximately 100,000 points per seconds. Based on initial tests, it was found that collecting data for a total of 0.5 seconds provides a resolution for which a human can rather consistently identify bicycles. The idea is that if a human is able to detect these bicycles by looking at the raw point cloud, then a machine learning model should be able to do this as well. The collection interval could be compared to the exposure time of a camera: the longer the exposure time, the more points are collected in the frame and thus the higher the resolution. It was found that 0.5 seconds of collection time furthermore enables the system to consistently and reliably capture bicycles in a single frame. Although some trailing occurs (the act of a trail of points behind a bicycle being present due to the exposure time) dependent on the speed of the bicycle, this trail is not of such size that within a frame, no start and end of a bicycle object can be determined.

In addition to the LiDAR sensor, a camera was also used when gathering data. This choice was made in order to aid the process of labelling the data. It was found that the point cloud data can often be hard to decipher and thus the photos would supplement the point cloud data for clarification.

A proprietary program was made for collection. This program takes in data from both a camera and the LiDAR simultaneously and takes a snapshot of the current frame of data when the user presses a button. The program was designed this way such that frames with the desired content could be saved on-demand by the operator of the attached computer. Thus, data was collected efficiently to include quality data frames that had data of interest which reduced time spent in post-processing.

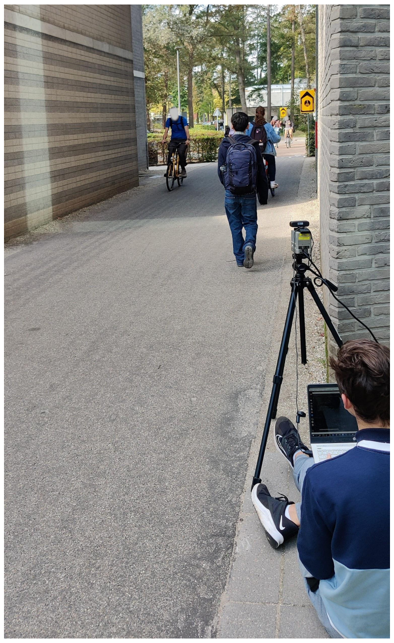


Figure 4: Data being collected

This took place in the area between the Ravelijn and Citadel buildings which intersects the De Zul road. It was chosen to collect data in this location due to the high amount of bicycle traffic in addition to the availability of a nearby power source. By the end of the session a total of ~1000 frames of data were collected.

Data Labelling

After having collected the data, the next step was to label it for training and testing purposes. The objective of labelling frames of point cloud data is to draw bounding boxes around the object which the model should be trained to detect. A bounding box is simply a three dimensional box which surrounds exactly the dimensions of the object such as in figure 5. This box then has a label which indicates what is contained within it. For the purpose of this project, bounding boxes were made using the 'bimo' (bicycle/motor) label. They were only made around cyclists and rarely scooters that were in the frame.

In order to create these bounding boxes, a tool in Matlab was used (see Matlab). The process of labelling using this tool was not very complicated. First, the user must import the frame of data and create the bounding box. Next, it was necessary to change how the point cloud points were colored in order to better recognize the objects in the frame. Finally, bounding boxes were placed and adjusted to any cyclists that could be recognized. If there were no recognizable cyclists in the frame, no bounding boxes were placed.

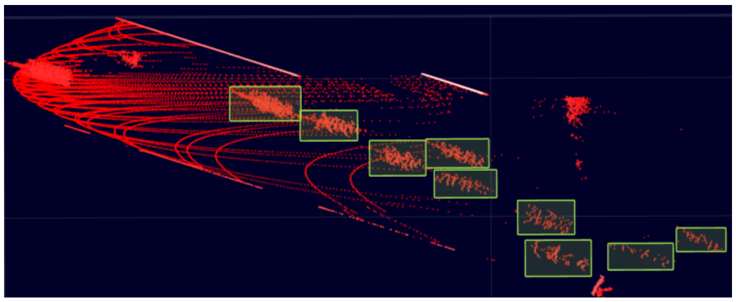


Figure 5. Labelled Data Frame

The bounding boxes created during the labelling process were intentionally chosen to be axis-aligned, that is to say that each of the three dimensions of each box is parallel to one of the global axes. The choice to label the data as such helps the group not only label faster but it also allows the centroid-based tracking algorithm to function (further details in Tracker). Another positive of axis-aligned bounding boxes is that they do not require matrix operations to calculate the vertices and thus run faster during runtime, reducing delay between inferences. However, some data about the traffic may be lost when there is no rotation to the bounding boxes, detrimental only if bicycle orientation is important, which it isn't for the purposes of this project.

Data Processing

Preprocessing

As the range of the sensor used is reported to be 260 meters (Livox, 2019), the captured point cloud frames consist of more points than necessary to detect nearby bicycles. As to decrease the computational complexity of the machine learning model, the point cloud is cropped according to the below table. The chosen domains for the x, y, and z variables are such that a 3D domain is created of such size that nearby bicycles ahead of the sensor will remain in the point cloud with a certain margin, while further away datapoints (such as those of trees in the background) are discarded.

Variable	x-axis	y-axis	z-axis
min	0.0	-39.68	-5.0
max	69.12	39.68	5.0
step	0.16	0.16	-

Table 1: Point cloud range definition

Important for the interpretation of table 1 is the definition of the axes in space with respect to the LiDAR sensor. The x-axis goes 'into' the front-face of the LiDAR sensor and hence the x-axis is the axis determining the depth of an object (front or back) as seen from the front-face of the lidar. The y-axis is the axis which defines what is 'left' and 'right', and together with the x-axis forms a horizontal plane in space. Lastly, the z-axis defines what is 'up' and 'down' with respect to the LiDAR, and creates a vertical plane in space together with the x-axis, along the x-axis. figure 6 and figure 7 visualise the axes in space with regard to the LiDAR sensor.

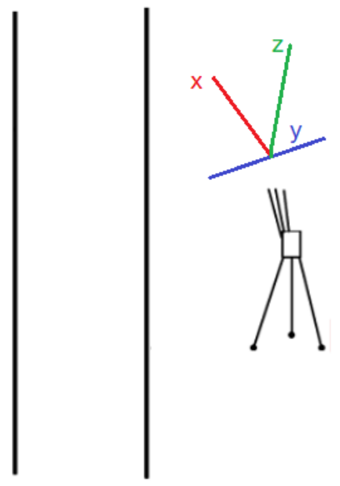


Figure 6: Orientation of axes with respect to the position of the LiDAR sensor in space

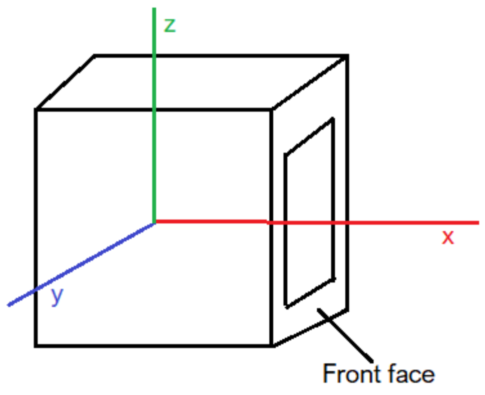


Figure 7: Orientation of axes with respect to the LiDAR sensor in space

As shown in table 1, the point cloud is cropped in such a way that only points that are no further than 69.12 meters away (x-axis component of the distance) from the front of the LiDAR sensor, that are no more than 39.68 meters to the left or to the right of the sensor (y-axis component) and are no more than 5 meters lower or higher compared to the height of the sensor (z-axis component). Furthermore, the resolution along both the x- and y-axis is set to be 0.16. These step values define the size of the voxels in which the point cloud will be organised, as per the common paradigm such as implemented in, among others, Complex Yolo (Simon et al., 2019), AVOD (Ku et al,, 2018) as well as the PointPillars encoder (Lang et al., 2019) used in this work.

The difference between the cropped and uncropped point clouds can be observed in the example given in figure 8 (uncropped) and figure 9 (cropped).

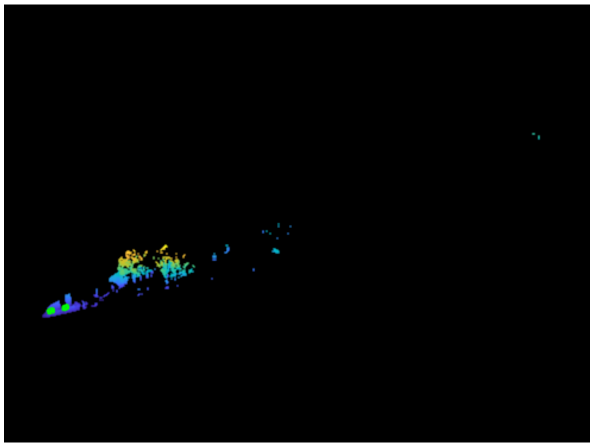


Figure 8: Point cloud before cropping

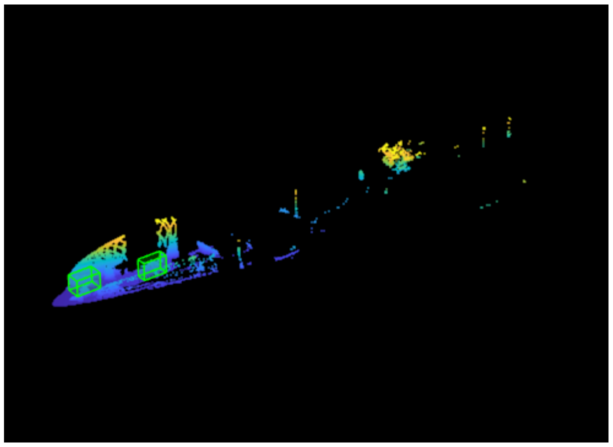


Figure 9: Point cloud after cropping

Data augmentation

Data augmentation is used to further enhance the quality and quantity of the dataset. The augmentation methods used are a way to decrease the risk of overfitting by attempting to mitigate certain biases, such as the exact position of a bike on the road, the size of bicycles and the direction from which bikes are observed. The augmentation methods used are rather standard (Lang et al., 2019; Zhou & Tuzel, 2018; Yan et al., 2018) and have been proposed in literature numerous times. According to literature (Zhou & Tuzel, 2018; Yan et al., 2018), data augmentation such as the techniques used in our work are critical for having a model perform well on the KITTI dataset (Geiger et al., 2012), which is a dataset containing labelled point clouds from a rotational LiDAR module.

Firstly, up to ten bicycles are randomly added from one frame to another frame to increase the total number of bicycles in a frame. This way, the model has more bicycle "examples", and hence model performance improvement is expected. Although not applicable to the implementation in this work, as the model in this work is only trained to classify a single bicycle, one could furthermore use this technique to alleviate class imbalance problems by artificially adding more objects of a certain class to frames by randomly sampling these objects from other frames. However, by having the exact same point signature for an object in multiple frames (that is, the same exact object being in multiple frames), the risk of overfitting increases.

To mitigate this overfitting risk to an extent, further augmentations are applied by which the data becomes more varied and hence the model is less prone to introducing biases in the model for a common pattern specific for this dataset. The standard augmentations used in this work as used in literature (Lang et al., 2019; Zhou & Tuzel, 2018; Yan et al., 2018) are random flipping along the x-axis, random rotation along the z-axis $[-\frac{\pi}{4}, \frac{\pi}{4}]$, random scaling by at most 5% and random translations by [0.2, 0.2, 0.1] metres along the x, y and z axes respectively. An example of the effect of the augmentation applied to a random point cloud data frame can be seen in figure 10 and figure 11.

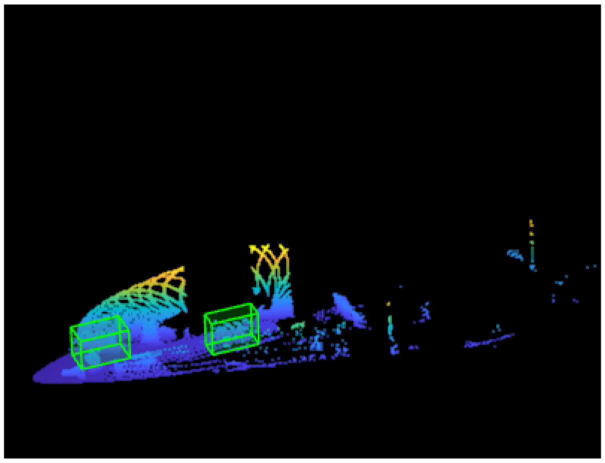


Figure 10: Point cloud before augmentation

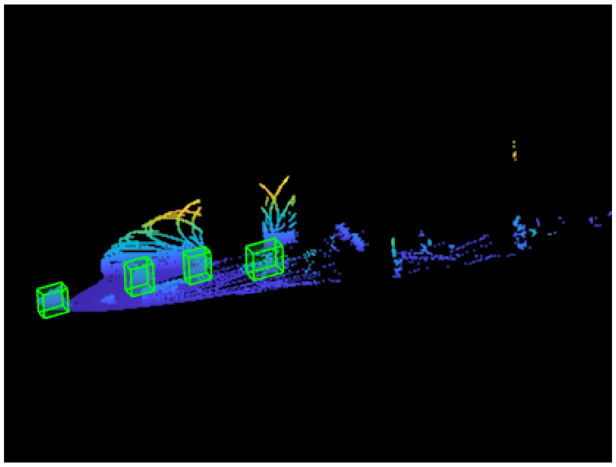


Figure 11: Point cloud after augmentation

Model

Architecture & Related Research

To be able to robustly detect bicycles with acceptable precision, a well designed machine learning model architecture is required. In literature, there are a number of model architectures proposed for the detection and classification of objects in space.

Context

For the detection and classification of objects in a point cloud, literature suggests two types of inputs – commonly referred to as modalities – to the model. The first is simply the points in the point cloud itself without any additional context, which is the modality used in for example VoxelNet (Zhou et al., 2018), SECOND (Yan et al., 2018) and PointPillars. On the other hand, certain models use additional context by means of 2D images taken by a camera, which more often than not allows one to achieve a higher accuracy in standard (well lighted and fair weather) conditions as compared to the models that do not make use of such additional context. Model architectures that make use of such additional camera input are MV3D (Chen et al., 2017), F-PointNet (Qui et al., 2018) and Roarnet (Shin et al., 2018) to name a few.

In addition, there are two main performance contexts with regard to the detection and classification of objects in point cloud data. One can namely detect objects from a bird's eye view (BEV) or from a 3D view perspective. BEV and 3D view benchmarks appear to be of similar popularity for benchmarking rotational LiDAR point cloud data performance (Zhou et al., 2018;Yan et al., 2018;Qui et al., 2018;Shin et al., 2018). As the unit used for this system is solid state rather than rotational, it is inherently limited to a 3D view benchmark, which often performs slightly worse than BEV benchmarks (Lang et al., 2019), as no BEV can be constructed with the limited FOV and range of the solid state LiDAR unit used.

Model choice

With the goal of the system specifically being investigating the feasibility of using a solid state LiDAR itself, it was decided to pick a model architecture that does not rely on context other than the point cloud data itself.

Furthermore, as to provide scientifically relevant results, it was chosen to narrow down the choice of model architecture to models used for the system to only those that predict bounding boxes around detected objects in space rather than solely detecting the number of them. This way, the approach used in this work can be applied to other applications as well, such as for example the detection of bicycles in space in autonomous vehicle applications.

To be able to make a decision on what model architecture to use for the system, three state of the art models that use only point cloud data frames as an input were considered. These models have all been benchmarked with the KITTI dataset, introduced by Geiger et al. in 2012. The KITTI dataset is a well known dataset consisting of, among other things, point cloud data captured with a Velodyne rotational LiDAR sensor attached to the top of a vehicle. Table 2 shows the performance of these models on said KITTI dataset for the 3D detection benchmark on easy mode. The difficulty mode, in this case easy mode, tells one something about the minimal bounding box height measure of occlusion and maximum truncation percentage of objects. The exact values for these parameters for the different modes has been neatly presented by Luo et al. in their 2018 paper, and can be seen in Table 3.

Method	Modality	mAP Car (easy)	mAP Pedestrian (easy)	mAP Cyclist (easy)
VoxelNet (Zhou et al., 2018)	Lidar (point cloud data)	0.7747	0.3948	0.6122
SECOND (Yan et al., 2018)	Lidar (point cloud data)	0.8313	0.5107	0.7051
PointPillars (Lang et al., 2019)	Lidar (point cloud data)	0.7905	0.5208	**0.7578**

Table 2: Performance of different model architectures on KITTI dataset (easy mode) 3D detection benchmark. Adapted from Lang et al. in their 2019 paper

Difficulty	Minimum bounding box height	Maximum occlusion level	Maximum truncation
Easy	40px	Fully visible	15%
Moderate	25px	Partly occluded	30%
Hard	25px	Difficult to see	50%

Table 3: KITTI dataset difficulty levels with regard to maximum truncation- and occlusion levels. Adapted from Xie et al. in their 2018 paper

As can be seen from table 2, the PointPillars model architecture is able to achieve the highest mean average precision (mAP) among the three candidate architectures on the KITTI 3D detection benchmark (easy mode), which is the reason why this PointPillars model architecture was chosen as the model used in the system. The reason why the easy mode benchmark was used to base the architecture choice on is two fold. For one, regarding bicycles, the difficulty level simply appears to scale the resulting mAP rather equally among the model architectures, as presented by Lang et al. in 2019, hence not altering the decision made based on the performance for detecting and classifying bicycles. Secondly, as the custom dataset created for this work does not have a set level of maximum occlusion or maximum truncation, it is not possible to assign a difficulty level one can use for comparison with other work. From empirical analysis of the novel created dataset, there certainly is (significant) occlusion and to a lesser extent, truncation present. However, due to a lack of quantification of said occlusion and truncation levels, it is only fair to compare results from our work with the KITTI benchmarks on easy mode. At best, our novel dataset is of greater difficulty, which only strengthens claims regarding the robustness of the model when compared to benchmarks on easy mode in literature.

PointPillars

The PointPillars model architecture, proposed by Lang et al. in 2019, is made up of three distinct parts, as can be seen in figure 12, which was presented in the original paper (Lang et al., 2019).

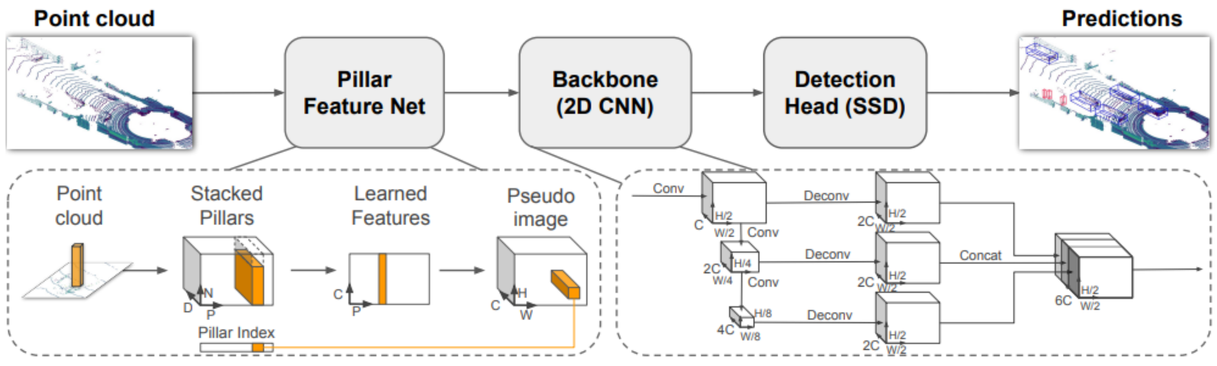


Figure 12 An overview of the PointPillars network architecture

The first part of the network architecture consists of a feature encoder network which takes a point cloud as an input, and converts this to a sparse pseudo image. Next, a 2D Convolutional Neural Network consisting of three channels is used to convert said pseudo image into a higher level representation. Lastly, a detection head (single shot detector, SSD) is used to detect and regress the 3D bounding boxes around the detected objects.

Pillar Feature Net

To be able to convert the point cloud data to a pseudo image, the point cloud is first discretized into an evenly spaced grid in the x-y plane, creating so-called pillars. The number of non-empty pillars in a point cloud data frame ***P*** as well the number of points per pillar ***N*** are defined as hyperparameters. Note that in case there are more points than allowed in a pillar or pillars in a frame, points or pillars are randomly sampled to fit the requirement. In case there would be too few points in a pillar or frame, zero padding is utilised. In addition, a bin size (also referred to as voxel size) is defined to reduce the point cloud to having only one point per voxel, but only in the x-y plane as we use pillars. This is done to reduce the computational complexity of training as well as inferring the model. After this, the points in the created pillars are augmented, resulting in the augmented lidar points having exactly 9 dimensions, creating a (***D***,***P***,***N***)-sized tensor. Following this, a simplified PointNet is used where a linear layer, Batch Norm (Ioffe et al., 2015) and ReLU (Nair et al., 2010) are applied in order to generate a (***C***,***P***,***N***)-sized tensor. Next, a max operation is applied over the channels, creating an output tensor of size (***C***,***P***). The features defined in this tensor are then scattered back to the locations of the original pillars to create a pseudo-image. The implementation used is exactly as proposed in the original paper (Lang et al., 2019), hence the number of output features from the network ***C = 64***.

2D-CNN

The 2D-CNN backbone proposed in the utilised PointPillars (Lang et al., 2019) paper is similar to the one used in VoxelNet proposed by Zhou et al. in 2017 and consists of a top-down network producing features as well as a network performing concatenation and upsampling of the produced top-down features. The implementation of the networks is furthermore exactly as proposed in the original paper (Lang et al., 2019), namely, the networks each consist of three individual blocks, each of size (***S***, ***L***, ***F***) of sizes: (***S***, 4, ***C***), (2***S***, 6, 2***C***) and (4***S***, 6, 4***C***) respectively, where ***S = 1*** denotes the stride used, ***L*** denotes the number of 3x3 2D convolution layers in the block and ***F*** denotes the number of output channels. Next, each block is upsampled by the steps: (***S***, ***S***, 2***C***), (2***S***, ***S***, 2***C***) and (4***S***, ***S***, 2***C***) respectively, after which the features of these steps are concatenated, providing ***6C*** features for the detection head.

Detection head

The detection head used in the system is furthermore as proposed in the original PointPillars paper (Lang et al., 2019), where a Single Shot Detector (SSD), proposed by Liu et al. in 2016. SSD is greatly useful, as it is able to localise (bounding box regression) and classify objects in a single forward pass. An overview of the SSD architecture with an example input of size 300x300x3 is given in figure 13.

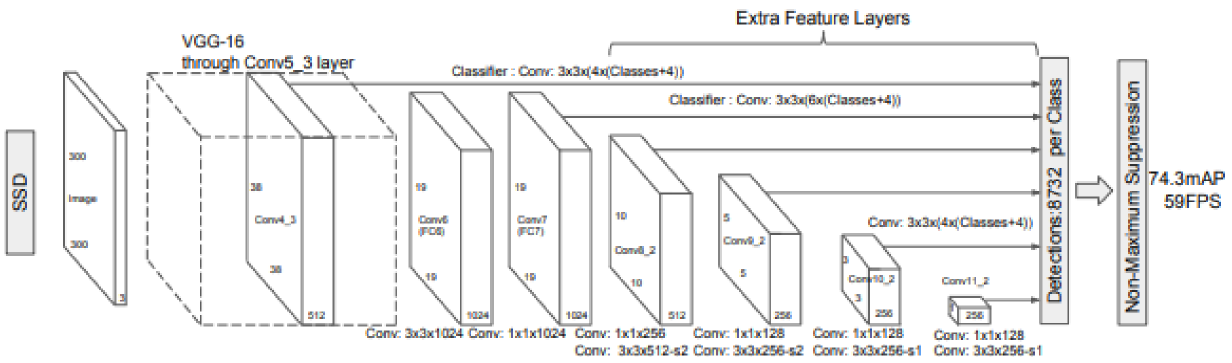


Figure 13. An overview of the SSD architecture with an example input of size 300x300x3.

Implementation

The model was implemented in MatLab and is an adaptation of the MatLab PointPillars implementation (Matlabengine, 2022). figure 14 consists of an overview of the MatLab implementation of the model architecture.

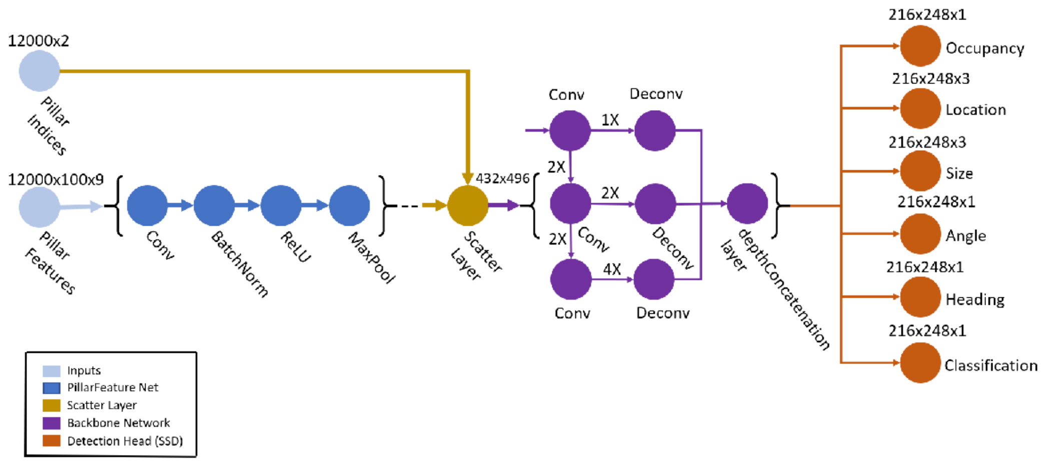
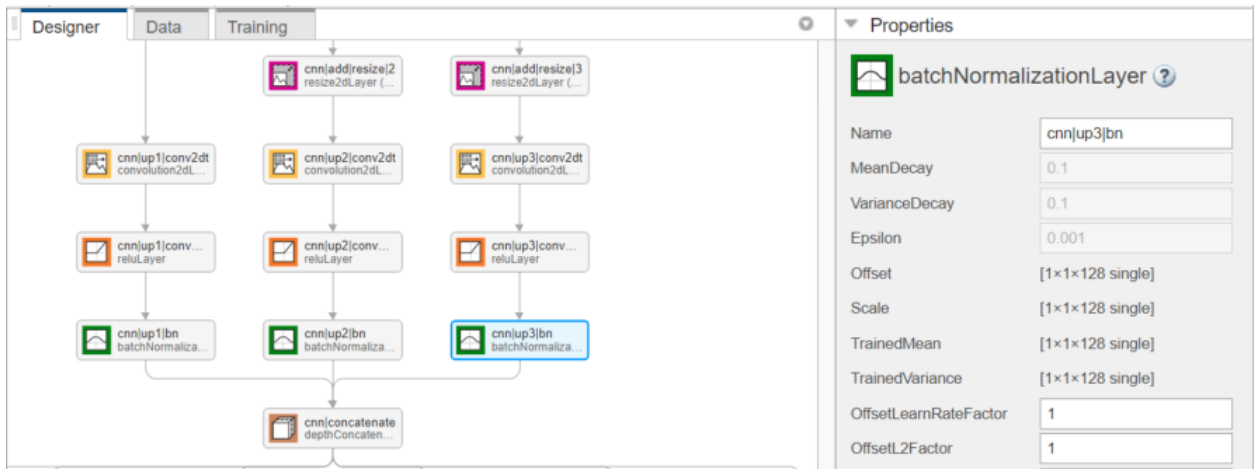


Figure 14. MatLab PointPillars (Lang et al., 2019) implementation overview. Sourced from the MatLab website ("Lidar 3-D Object Detection..", 2022)

The architecture shown in Figure 14 corresponds exactly to the model architecture proposed in the original PointPillars paper (Lang et al., 2019), hence its technical details correspond with those presented in above sections regarding the model architecture, as well as the original PointPillars paper.

As a 'sanity check', figure 15, taken from the "Deep Network Designer" tool in MatLab with the used model loaded into it, one can see that the input to the detection head consists of three batchNorm layers with 128 features each, this corresponds to the theoretical number of features fed to the detection head, namely 6***C***, where ***C*** was defined as being the number of output features from the Pillar Feature Net, which was defined as being 64 (128x3 = 64x6).



Name	cnn|up3|bn
MeanDecay	0.1
VarianceDecay	0.1
Epsilon	0.001
Offset	[1×1×128 single]
Scale	[1×1×128 single]
TrainedMean	[1×1×128 single]
TrainedVariance	[1×1×128 single]
OffsetLearnRateFactor	1
OffsetL2Factor	1

Figure 15. Network at concatenation of features after upsampling.

The MatLab Engine API ("Install MATLAB Engine API for Python...") is used to interface MatLab with Python, required to interface the model with the data collection part of the application as well as the forwarding of detected objects to the WebApp backend.

Performance metrics

In literature (Lang et al., 2019; Zhou & Tuzel, 2018; Yan et al., 2018), for the detection and classification of objects in point clouds, the two standard metrics are the Average Orientation Similarity (AOS) and mean Average Precision (mAP). The AOS metric was introduced by Geiger et al. in their 2012 paper, the same metric which introduced the widespread KITTI benchmark dataset. The AOS is defined as follows:

$$AOS = \frac{1}{11} \times \sum_{r \in \{0, 0.1, ..., 1\}} max_{\tilde{r}:\tilde{r} \geq r} \; s(\tilde{r})$$

$r = \frac{TP}{TP + FN}$, which is the recall in which detected bounding boxes are deemed correct when the overlap (intersection over union) with the ground truth bounding box is larger than the set threshold (in case of the threshold for the system, this is 0.5). Furthermore, the recall r is a normalised cosine similarity $s(r) = \frac{1}{|D(r)|} \times \sum_{i \in D(r)} \frac{1 + cos\Delta_{\theta}(i)}{2} \delta_i$. in which $D(r)$ is the object set of all detections for recall rate *r* and $\Delta_{\theta}^{(i)}$ is the difference between the orientation of the ground truth and the predicted orientation, for the ith detection. The AOS is a powerful metric as it does not only require a predicted bounding box to overlap with the ground truth bounding box by at least the threshold percentage, it also takes into account the recall and the orientation of the predicted bounding box as compared to the ground truth.

Next, the mAP is defined as being $mAP = \frac{1}{N} \times \sum_{c \in N} \frac{TP_c}{FP_c + TP_c}$, where *N* is defined as the number of classes.

Training and Results

The training of the model is done in MatLab, and training was executed on the University of Twente remote desktop, which is equivalent to a small-scale cluster. The results of the different training runs done are shown in table 3.

Epochs	Batch size	Train/test split	Contains empty frames	Confidence threshold	IoU Threshold	**AOS**	**mAP**	Dataset size
1	3	90/10	Yes	0.25	0.5	0.0055268	0.0055430	
60	3	90/10	No	0.25	0.5	0.47576	0.47644	300 (minus empty frames)
60	3	90/10	Yes	0.25	0.5	0.68677	0.68684	620
160	3	90/10	Yes	0.25	0.5	0.78774	0.78778	620
175	3	90/10	Yes	0.25	0.5	0.84523	0.84525	620
200	3	90/10	Yes	0.25	0.5	0.85762	0.85764	620
220	3	90/10	Yes	0.25	0.5	🟩 0.86962	🟩 0.86965	620

Table 3. Results obtained from different training runs, the best result is marked in green

The train/test split utilised for all frames is 90%/10%, this is done as to maximise the potential of the rather small dataset at hand. Furthermore, all evaluation metrics are obtained using the test split.

Training was started with a dataset of 300 labelled images without empty frames and with 60 epochs, as proposed in the MatLab PointPillars implementation ("Lidar 3-D Object Detection..", 2022). The mAP and AOS obtained from this training run was acceptable (48%, see Table 3) for the purpose of counting bicycles (one could use the error to possibly give a range of bikes that passed, given the error is constant enough), although not optimal.

Notably, as the AOS and mAP have almost exactly the same value, this indicates that the main problem of the model is in fact false positives.

In order to improve performance, experiments were conducted with more data as a way to reduce the risk of overfitting, more epochs as a way to improve the model performance, as well as training the model with empty frames as a way to potentially decrease the amount of false positives. One can see in Table 3 that when significantly increasing the dataset size as well as training with empty dataframes, the model AOS and mAP improved drastically to 69% mAP and AOS. Still, the main problem of the model remained the detection of bicycles in places where there were in fact no bicycles; false positives.

Further increasing the amount of training epochs appeared to increase performance on the test set, up to a certain point. It was observed that from approximately 175 epochs onwards, the AOS and mAP of the model only very slightly increased further. This observation has been visualised in Figure 16. Because the AOS and mAP of the model on the test set did not appear to increase with any significance after 220 epochs anymore, it was decided to not train the model further than this.

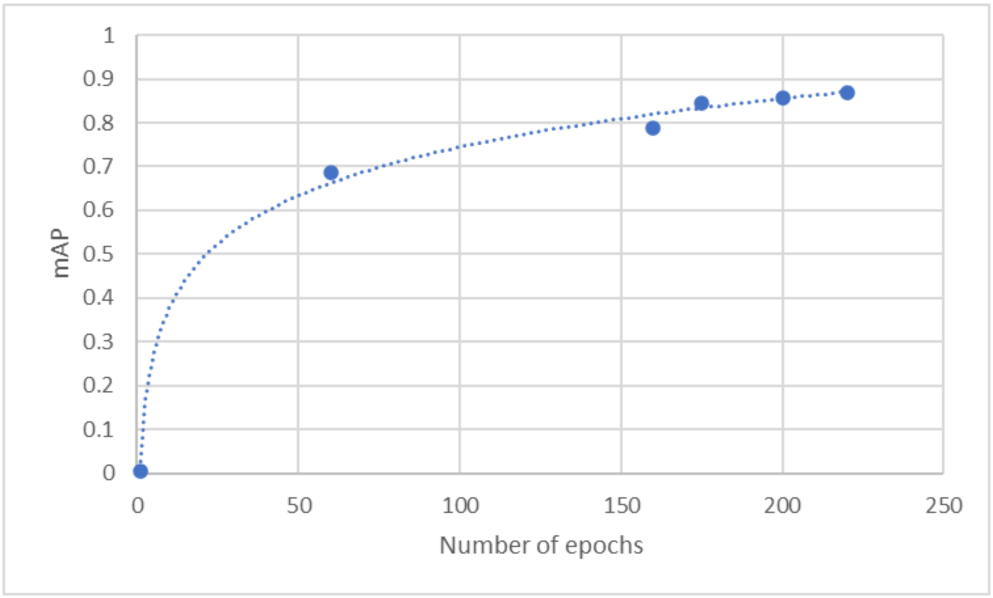
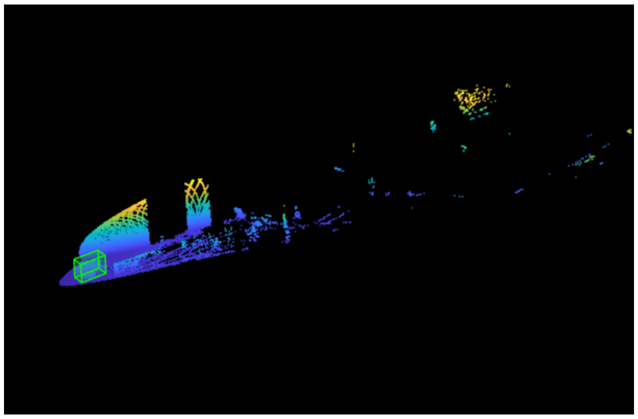


Figure 16. mAP after training for a number of epochs



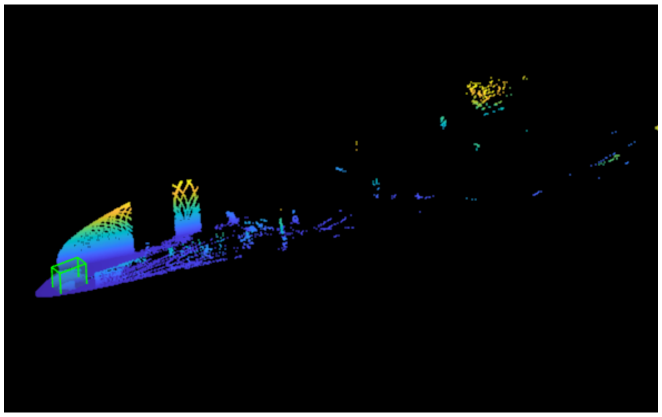


Figure 17. Ground truth bounding boxes

Figure 18. Bounding box predicted by model

After all, an AOS and mAP of 0.87 was achieved on the test set, which is in itself higher than the mAP and mAOS reported in the original PointPillars paper (Lang et al., 2019), hence providing evidence supporting that it does in fact seem feasible to utilise Solid State LiDAR data for robust detection of bicycles, as well as supporting the quality of the collected dataset. To provide an example of the capabilities of the model, an example prediction done by the model on a point cloud data frame from the test set is given in Figure 18, the ground truth of said frame is provided in Figure 17. Furthermore, notably, in certain cases when the labeler forgot to label certain bicycles in a frame, the model as in fact able to detect these bicycles and predict a bounding box around these, as can be seen in the prediction made by the model for the frame shown in Figure 20 (right side of frame, as there is a false negative on the left side of the same frame regarding occlusion). The ground truth bounding boxes of said frame are shown in figure 19. The fact that the model appears to be able to detect bicycles that were forgotten to be labelled by the labeler, indicated that the model might be used as an assistance for labelling future point cloud data frames.

Model limitations

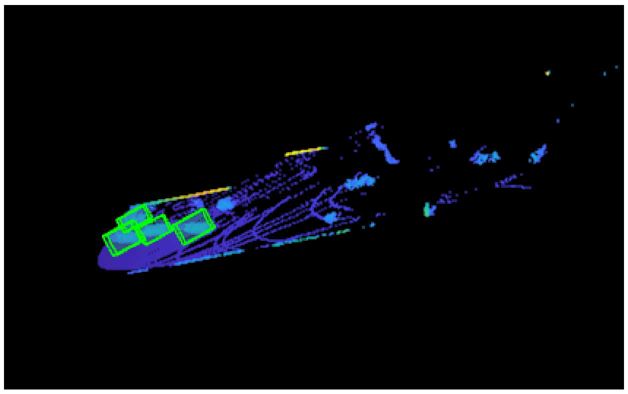


Figure 19: Ground truth bounding boxes

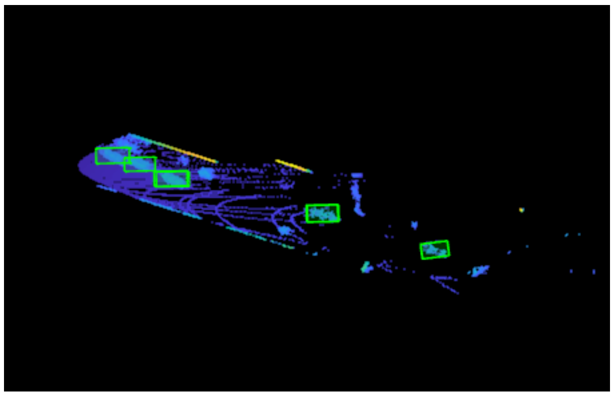


Figure 20: Predicted bounding boxes by model; containing one false negative (left) and two missed bicycles (right)

Despite the model proving to be very promising given its high AOS and mAP, there are still certain limitations to its abilities, especially in case of (partial) occlusion. This limitation can be clearly seen in the prediction made by the model in Figure 20 (left side), where despite correctly having classified two bikes that were not labelled by the labeler (right), the partially occluded bicycle behind two other bicycles was not detected as being a bicycle. Likely, given more partially occluded labelled training data, the model would perform better, hence specifically adding occluded bicycles could be a useful addition to the dataset in future work.

Tools

Matlab

Matlab was one of the fundamental tools used throughout the development process. To begin, Matlab's LiDAR toolbox has a visualizer that was used to visualise LiDAR frames, whether captured within the program or imported from csv files. What makes Matlab a versatile visualizer is its ability to change colour schemes based on LiDAR information. Multiple colour schemes depending on resolution, distance and reflectivity allowed the group members to better perceive the information captured by the sensor. This kind of visualisation was not only important for manual testing of the setup, but also for the labelling of data needed to train the machine learning model.

Matlab's LiDAR toolbox is equipped with a labelling tool, which allowed the group to label frames more quickly. A proprietary script was made to capture frames from the LiDAR sensor as .csv files and then import them into the labelling tool, which allowed the group to manually draw a bounding box around each bicycle in the frame. The tool would then export the labelled frames as a Matlab file, which was in turn used to train the machine learning model.

The group's final use of Matlab was the most essential part of the development process, namely the training of the machine learning model. Matlab is not only pre-equipped with all the tools needed to process data labelled through it, but also has support for standard machine learning libraries such as PyTorch and TensorFlow. The group decided to stick with the Matlab ecosystem because there exists many and comprehensive materials available for learning and applying skills in its suite of tools.

GitLab

There are not many solutions for source code management out there, but it has been agreed upon to use GitLab. Managing code with the help of git's tools made it possible to collaborate and split tasks more hassle-free. Utilising git's branches and version control opened the possibility to iterate on code, create different versions and restore working files seamlessly. For example, multiple versions of the frame capturing tool were created, one that needed to be operated manually for more precise frame capturing and one that could be toggled on and off for automated collection; with the help of branching, multiple developers were able to work simultaneously to save time. Milestones and issues were also intended to be used but due to the course the development process took and the exploratory nature of it, it was difficult to make good use of them.

Visualisation

Visualisation during development has been a recurring problem that we had to tackle. Livox developers have provided proprietary software to show the output of the sensor but it is just that. It has no api or methods to hook into it besides exporting .csv files and reading them using other programs such as Matlab. A visualizer had to be developed to make up for the shortcomings of Livox software. Open3D was the Python library used to create the visualizer because it provided a voxel based renderer that could handle 100,000 point data frames. It was used during the capturing process to quickly and efficiently read .csv files to see what was being captured without jumping between software solutions. However, during the labelling process, Matlab's visualizer was used instead because it included a built-in labeler and was more equipped to export labelled frames.

UT Remote Desktop

One tool that was used extensively was the university's remote access tool to university hosted hardware. University machines are equipped with hardware and software that are built for extensive data crunching that are ideal for the purpose of model training and inference. These tools gave the group autonomy to quickly reiterate on models by providing power equivalent to two NVIDIA GeForce RTX 2080 GPUs which were otherwise unattainable. Moreover, most existing projects used for LiDAR equipment were built for Unix based operating systems and having machines that could run them without virtualization allowed for more efficient workflows.

Tracker

One of the problems associated with the counting problem is being able to track which bicycles were counted and which were not. In any given two succeeding frames captured by the LiDAR sensor, one and the same bicycle may appear in both. This is either due to lower cycling speed or high lane traffic. This may cause the system to count said bicycle multiple times, artificially inflating traffic count and decreasing the accuracy of the final solution. This problem prompted the creation of a custom tool for tracking bicycles across multiple frames. The solution is a Python class forked from the *Centroid-Based_Object_Tracking* repository by *lev1khachatryan*, which was then adapted to support the tracking of 3D bounding boxes. The algorithm assigns an ID to each bicycle in frame and uses its centroid (the average of bounding box vertices) to follow it from one frame to another. Unfortunately, this method of tracking only works on higher frame rates because it is difficult to tell whether a certain bike is the same from a previous frame or a new one given a long enough delay in between two frames; the model is only able to make inference every 2 seconds. This is a limitation of LiDAR data because there are hardly any identification details for bicycles besides position and speed. It was concluded that without a secondary model running to identify bicycles, it would not be feasible to attain a reliable tracker. This is a source of inaccuracy for the project.

Data Collector

The data collection tool is a tool custom built by the group to quickly and efficiently collect data for labelling. The tool captures a full frame (50,000 LiDAR dots) over half a second from the LiDAR sensor, along with a picture from a camera. Too short of a capturing window leads to dispersed data points (i.e., lower resolution images) whereas a longer window leads to ghosting (i.e., motion blur). It was discovered that a half second window is a feasible middle ground that balances between resolution and the ghosting effect. The data collection tool is built with the help of the Livox Python SDK and OpenCV. It has two modes for operation, manual and automatic collection. Manual mode requires a user to press a button every time a new data point is wanted whereas the automatic mode only requires toggling on and off and the system will automatically collect data points when it is on. These two modes allow versatile methods for data collection where one can let the system automatically collect or can pick and choose when to collect a data point. These modes can help balance the number of data points for each machine learning categorization and avoid biases.

Materials

Many of the materials used by Group 6 were already in the possession of the team members or were otherwise supplied by MINDHASH or the university. However, for the purpose of analysis and replication it is good to understand the materials involved and their costs. These costs can be seen below in table 5.

Material	Explanation	Cost
Livox Mid-40	The sensor around which this project revolves. Used to gather data for testing as well as provide data to the ML model.	€606.80
Camera Sensor	Used to gather data for the purpose of ensuring high quality data labelling.	€40.00
Tripod	As a prototype, a simple tripod is used to mount the system for testing and usage.	€28.00
Extension Cable	This allows for the system to have more flexibility in the location that it is deployed.	€ 27.00
Computer / Processing Device	A minimum cost is based on the Raspberry Pi 4	€75.00
Total		**€830.00**

Table 5: Material Costs

Web Application

Earlier in this document, a web application to display traffic analytics was conceptualised. In this section we will explore how this was achieved and justify design choices that were made. A high-level walk-through of the system is followed with more in-depth documentation and references. Furthermore, a comprehensive manual is made available in Appendix A.

Subsystem Overview

The web application is quite simple to understand on paper. It is a single page application (SPA) built with common JavaScript framework React, TypeScript and component library Mantine. The entire application does not have a traditional backend, but is rather hosted on Firebase, a serverless cloud environment by Google. It is a backend as a service (BaaS) that offers out-of-the-box solutions for a proprietary NoSQL database service called Firestore, web hosting, authentication and cloud functions. The advantages of Firebase and why we chose this platform will be discussed later on.

Backend Architecture

Firestore

In Firestore, there exist three collections: `sessions`, `sessionData` and `stats`. A session is used across the application to describe a continuous capture of successive traffic where the LiDAR station is stationary. It is described by a document in `sessions` with a start time, end time if the session has finished and a location. Important is that the id of said document serves as the primary key for each session across all collections, although this is not enforced. This means that `sessionData` and `stats` each contain documents that correspond one-to-one to those in `sessions`.

Documents in `sessionData` contain the raw captured data in the form of an array of JSON objects describing the inference object and the capture timestamp. The inference object is described as a string and generally `"bicycle"` and `"pedestrian"` are used, but this project is designed with indeterminate object ids in mind.

`stats` contains documents that have pre-compiled statistic values based on the corresponding data in `sessionData`. Moreover, `stats` contains a special document with id `"avg"` with the same structure that contains the average values for all defined metrics across the documents in `stats`. It is the documents in `stats` that are requested by the frontend. The reason for this split will be discussed in Design Choices & Process.

Cloud Functions

A cloud function (CF) is deployed as an observer of the `sessionData` collection. It will be executed on every update to each document in the collection. Only if data is added, corresponding statistics are recompiled with the new data and the average statistics are updated and the changes are written to `stats`. As mentioned before, it is the documents in stats that are used for displaying in the frontend, so this CF takes care of lively recomputing these values on every new inference.

Latter Data Pipeline

The pipeline introduced in ML Data Pipeline will continue after the webhook has written an object and timestamp. As described above, the CF will be called and the respective and average statistics will be updated. The frontend opens in turn an observer on the `stats` document corresponding to the session that is shown on screen. This observer will update the local copy of the statistics document. More details will be provided in the next section. The two observers and the nature of React make it so that the entire application displays data in real-time, where the latter data pipeline takes no more than ten seconds to complete, depending on the allocated resources to the CF by Firebase. A detailed overview of the pipeline is given below. Note that the frontend has been abstracted with a cube, as its internal details will be discussed in Frontend Flow and Documentation.

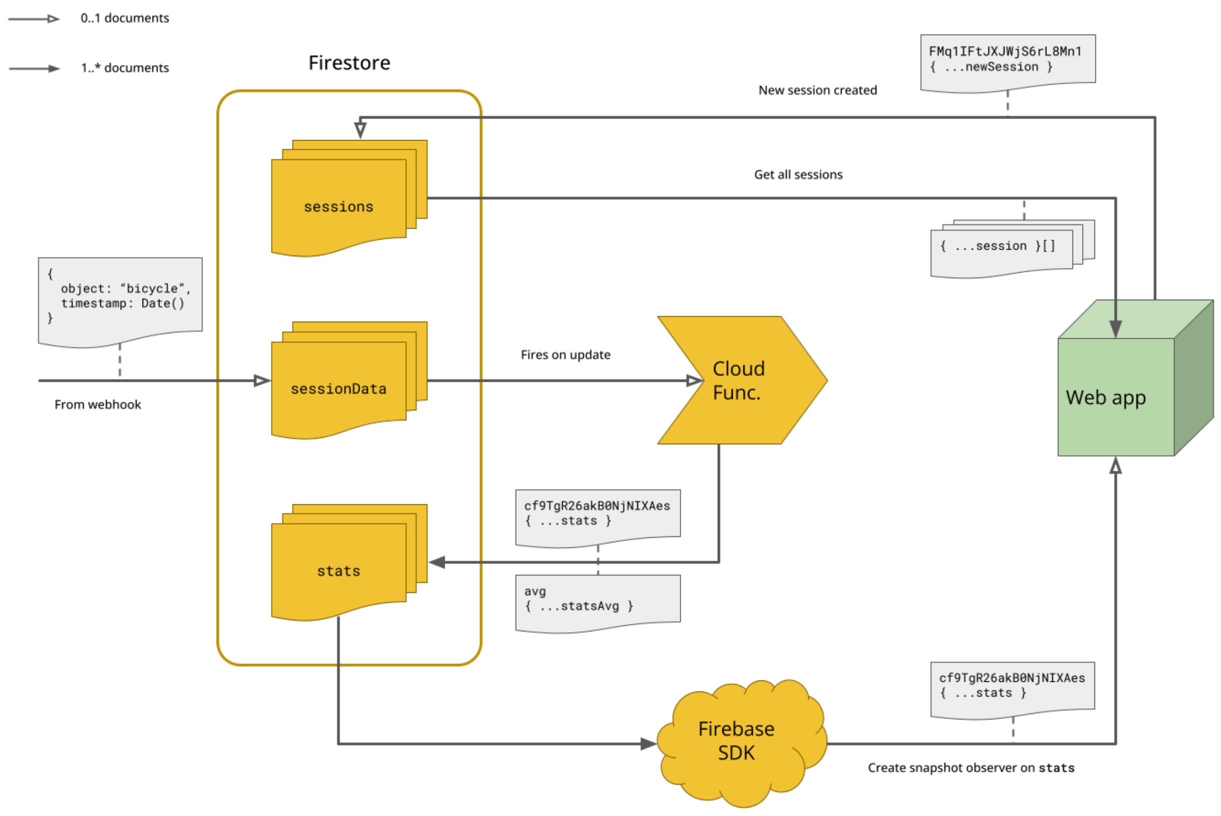


Figure 21: Flow chart of latter data pipeline

Frontend Flow

Authorisation

This section will provide a brief visual roadmap the user can take. A pre-authorised email address and password in Firebase are required to login into the system. To this end, the first page the user is greeted with is a login page prompting an email address and password. The user is notified in the case either or both of the fields are empty, the email address value is not an email address or does not match any known email addresses or when the password is incorrect. If a correct combination is provided, the web application will authenticate the user with Firebase and redirect them to the dashboard. The user remains authenticated within the current browser session.

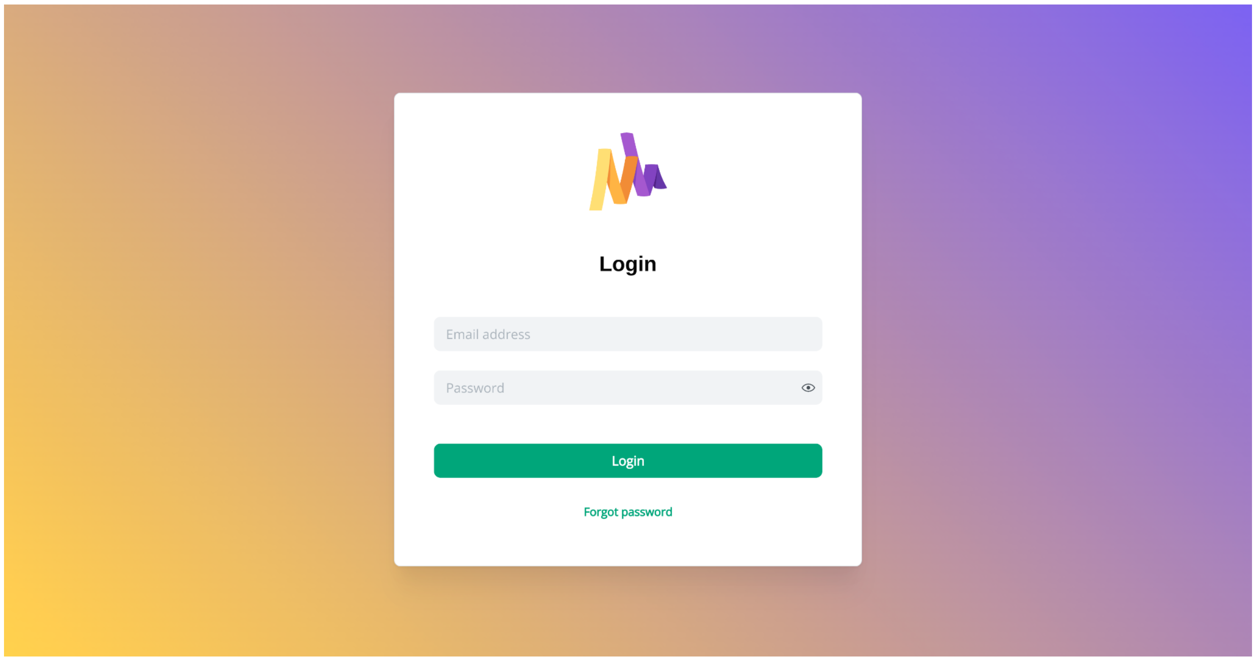
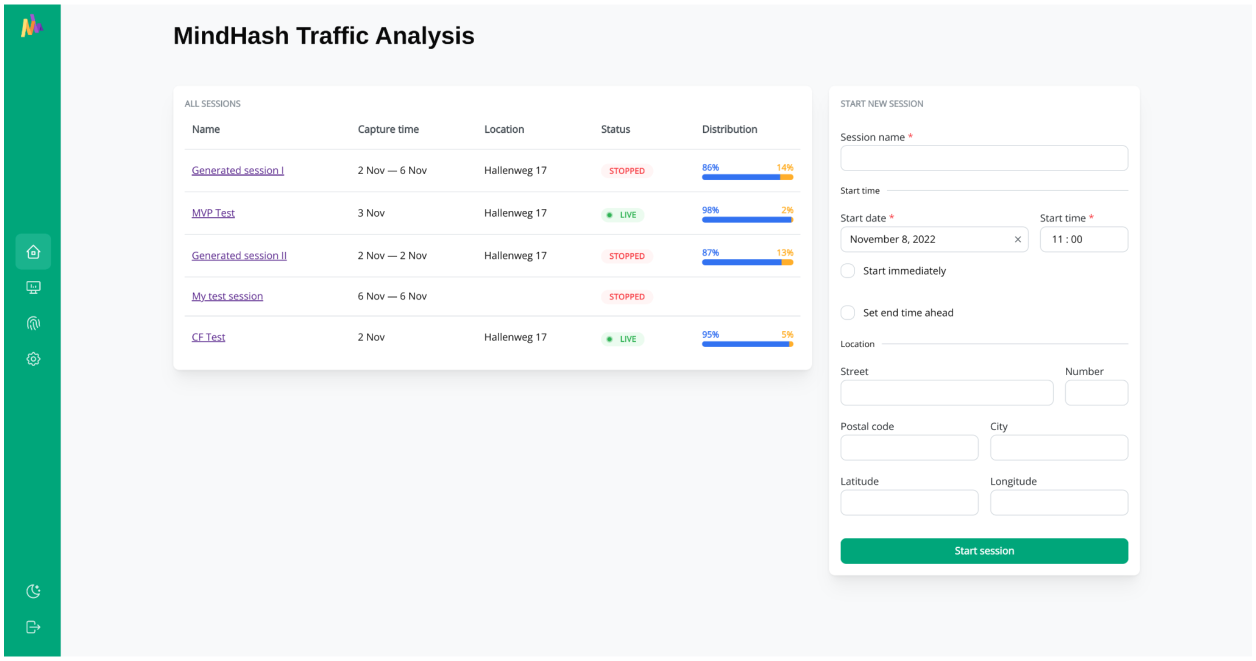


Figure 22: Web application Login page (/login)

Home page

After logging in, the user is redirected to the dashboard. The dashboard has a simple navigation bar on the left with buttons to each page, in addition to a colour theme switch (light/dark) and a logout button. In the body of the dashboard an overview of all sessions that are currently live and those that have finished in the past is displayed. Along with the exact dates, the location and the status, each row of this overview also gives a very brief insight into the traffic distribution of that session. Hovering over each of the aforementioned fields will show more information. Finally, clicking on the session name will redirect the user to the analytics page of that session.

Next to this overview, a form allows the user to create and start a new session. A new session requires a name and a start time, which can be now or in the future. An end time can also be set and will stop the session automatically. Finally, a deeply optional location can be provided, consisting of an address and a geopoint.


MindHash Traffic Analysis

ALL SESSIONS

Name	Capture time	Location	Status	Distribution
Generated session I	2 Nov — 6 Nov	Hallenweg 17	STOPPED	86% 14%
MVP Test	3 Nov	Hallenweg 17	LIVE	98% 2%
Generated session II	2 Nov — 2 Nov	Hallenweg 17	STOPPED	87% 13%
My test session	6 Nov — 6 Nov		STOPPED	
CF Test	2 Nov	Hallenweg 17	LIVE	95% 5%

START NEW SESSION

Session name *

Start time

Start date * November 8, 2022

Start time * 11 : 00

Start immediately

Set end time ahead

Location

Street

Number

Postal code

City

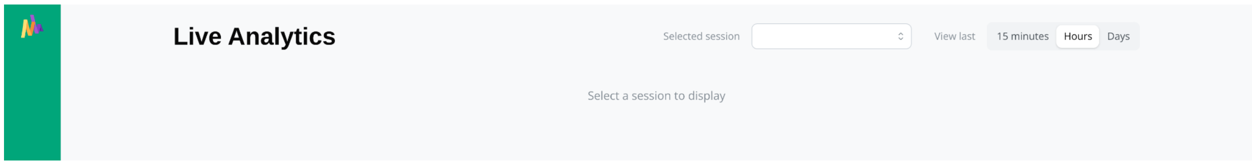
Latitude

Longitude

Figure 23: Web application dashboard Home (/)

Analytics page

The page most interesting to a traffic analyst is the Analytics page. This page will prompt the user to select a session from a dropdown menu if none is selected.


Live Analytics

Selected session

View last 15 minutes Hours Days

Select a session to display

Figure 24: Web application Analytics page, select session (/analytics)

When a session is selected, the application will redirect to a dynamic URL with the id of the selected session. This will show a page with the analytical overview this project provides and is fed directly by the documents from `stats`. Next to the dropdown menu to change sessions, there is a segmented control that allows the user to change the time scope of the overview between fifteen minutes, hours and days. This changes the granularity of all non-absolute statistics. This is useful when the system is deployed continuously for a week and the user wants to see a per day granularity.

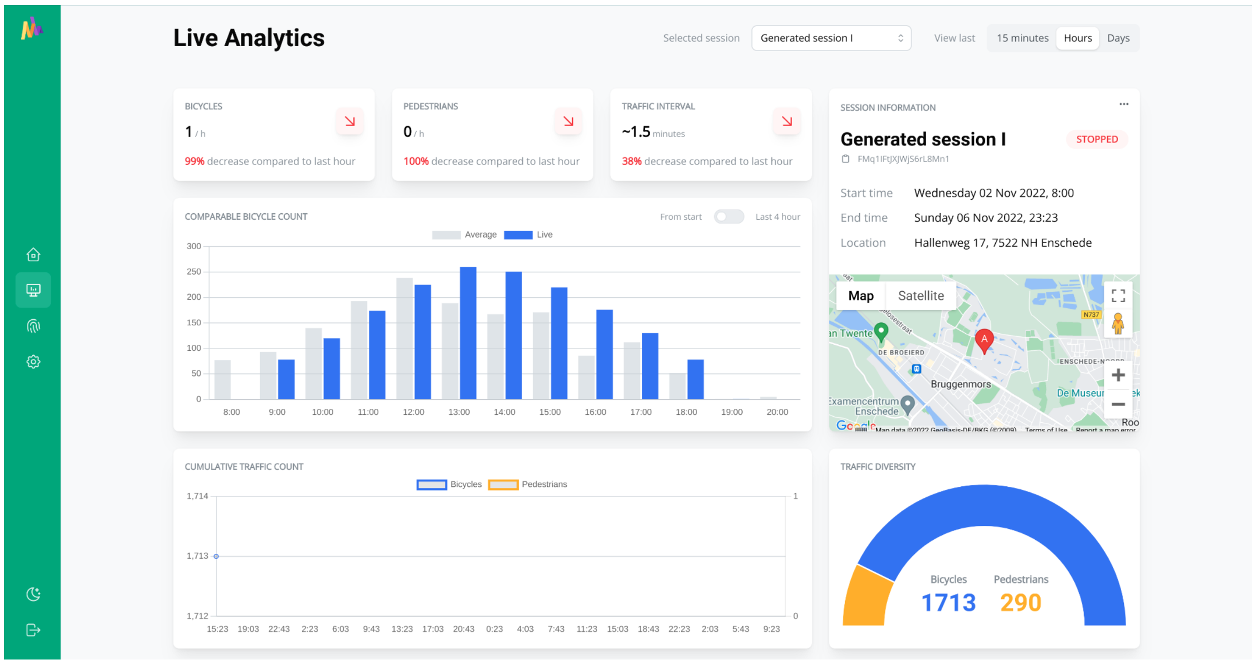
As it was mentioned before, the web application initiates an observer to the respective document in the `stats` collection. It was established earlier that the statistics document of the session being written to and the average statistics document are updated in real-time. This is combined with the local observer, enabling the web application to display live updates in real-time. This happens across all points where the statistical data is used.

On top of the page are three cards that show the number of traffic objects (here bicycles and pedestrians) in the current time scope. There is also a statistic that shows the interval between detections of all sorts. These three cards compare their respective statistics to that of the previous time scope and change style accordingly.

Below the three statistic cards is a graph that shows the distribution of the traffic since the start of the session in blue. This graph also has the option to follow along in time and trailing behind four units. The units can be changed with the time control mentioned before. Once again this data is fetched directly from the `stats` collection, where the cumulative values are summed per time interval in order to be displayed in this graph. Notable as well are the grey bars. They are the average value that has been captured in other sessions at that time unit. That is to say that the average at 13:00 is the average of all sessions that have data between 13:00 and 13:59. The same logic is applied to fifteen minute intervals and days.

Next to the live graph is a card that contains metadata about the session. In particular, the name and id; start and optionally end time; and the address and an interactive map with a pin. Next to the name is also an indicator to show the status of the session. The id is copied to the user's clipboard when clicked as the id is required by the webhook of the LiDAR system.

On the last row, two graphs remain. The first is a live tracker of the cumulative traffic, separated into each object class. This stepped line graph follows time and trails back one hour to show the recent history. A doughnut graph is displayed next to it that depicts the distribution between the detected traffic objects.


Live Analytics

Selected session: Generated session I
View last: 15 minutes | Hours | Days

BICYCLES
1 / h
99% decrease compared to last hour

PEDESTRIANS
0 / h
100% decrease compared to last hour

TRAFFIC INTERVAL
~1.5 minutes
38% decrease compared to last hour

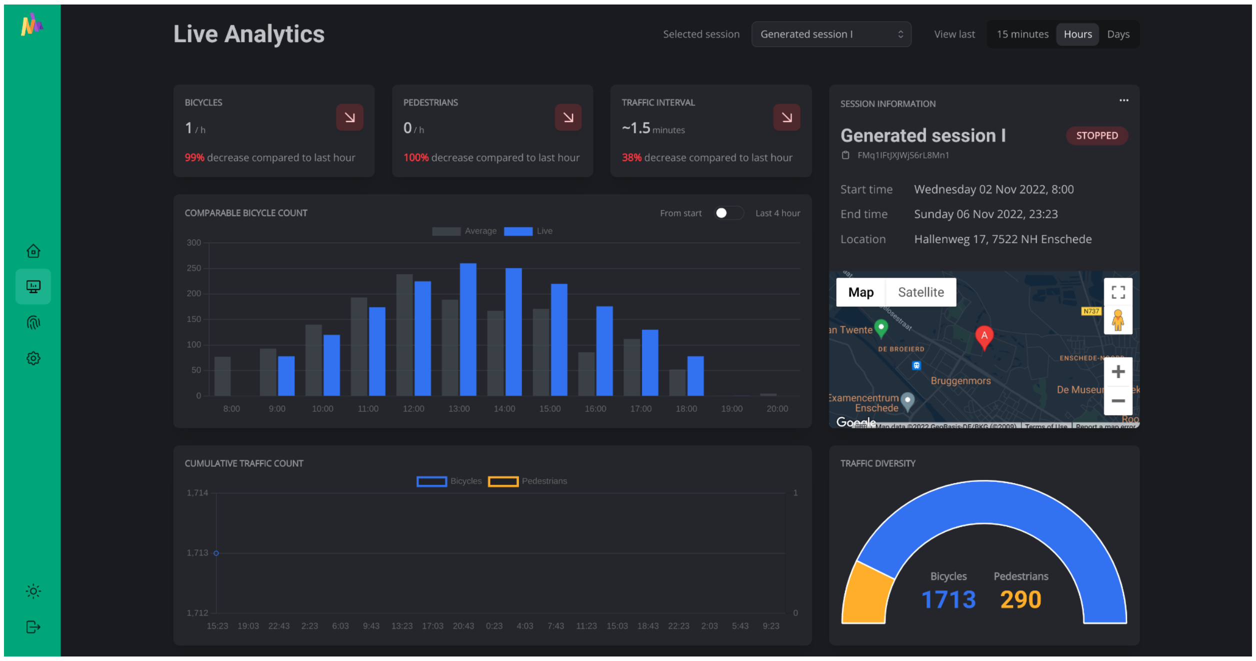
SESSION INFORMATION
Generated session I — STOPPED
FMq1IFtJXJWjS6rL8Mn1

Start time	Wednesday 02 Nov 2022, 8:00
End time	Sunday 06 Nov 2022, 23:23
Location	Hallenweg 17, 7522 NH Enschede

Figure 25: Web application Analytics page, with session (/analytics/:id)

Dark mode

One of the features of Mantine, the React component library that was used, is native dark mode support. Naturally, this made implementing a dark theme straightforward.



Selected session: Generated session I
View last: Hours

BICYCLES 1 / h — 99% decrease compared to last hour

PEDESTRIANS 0 / h — 100% decrease compared to last hour

TRAFFIC INTERVAL ~1.5 minutes — 38% decrease compared to last hour

SESSION INFORMATION

Generated session I — STOPPED

FMq1IFtJXJWj56rL8Mn1

Start time	Wednesday 02 Nov 2022, 8:00
End time	Sunday 06 Nov 2022, 23:23
Location	Hallenweg 17, 7522 NH Enschede

TRAFFIC DIVERSITY: Bicycles 1713, Pedestrians 290

Figure 26: Web application Analytics page, with session, dark mode (/analytics/:id)

Design Choices & Process

In the process of developing this software, plentiful choices were made at various levels of criticality and with various levels of confidence. Nevertheless, all choices are made in the best interest and are justifiable as such.

Technology Stack & Tools

The most critical choices of software development, especially that in a tight timeframe, are made at the beginning. Likewise here. Across the entire stack, technologies have been chosen based on at least two factors: prior experiences and maturity. This is because a functional MVP has to be realised within the allocated ten weeks. Taking away time for planning and designing beforehand, only a few weeks remain. It is for this reason that familiarity with the tools and software to be used in particular was the leading determinant in choice making.

It commonly is, but for this project we deem performance not important. Performance in terms of the backend has the only requirement to handle minimal JSON document writes at most every second or so. It should also serve a handful of connections to users. Both tasks are lightweight and likely to cause big performance hits. For the frontend is not affected by a loss in loading or operating performance because an application of sorts will be accessed by a few users per time at the most and other than UX, such performance does not inhibit the workflow.

Backend

Traditional would be to create a server to run on Node or Python, like Express or Django. They would expose a RESTful API which our frontend could use. Next to the HTTP server could be a database server, relational or not, like Postgresql or MongoDB. Despite existing experience with these technologies and them having a fair share of advantages, our point against them is that it is comparatively difficult and tedious. The backend as envisioned is very simple; receive data on one end, process it and push it to the frontend. Spending time setting up and hosting such an environment and managing it during the project would be time wasted. That's why we started looking towards a more integrated solution and decided on Firebase. As outlined before, Firebase is a serverless, managed backend environment with minimal setup and hosting out of the box. Jasper has experience with deploying Firebase in medium sized projects and Firebase is accessible through an excellent JavaScript SDK and has outstanding documentation. These perks have made the obvious choice above a proprietary setup or even other cloud providers like Azure or AWS.

Frontend

Nowadays it is hard to create a web application (do not read sites or pages) without pondering what JavaScript framework to use. Traditional choices are Angular, Vue or React compared to modern frameworks like Solid or Svelte. Note that the line between traditional and modern here is no more than three years in the past. Because of a lack of experience and a young and immature ecosystem, the latter choices are deemed unsuitable. React has matured over the past ten years since it was open-sourced by Facebook in 2013 into the most popular JavaScript framework and has a vast ecosystem. Moreover, Jasper has extensive experience with it. A downside of React is that it takes a performance hit compared to more modern frameworks. This is not a problem, however, since performance is not a priority.

Another question that needs answering beforehand is whether to use a component library and if the answer is yes, which one. We showed before that speed and DX have a high priority and therefore the use of a component library is recommended. A component library provides out-of-the-box customisable components like buttons, text inputs, checkboxes and more, but also structural components like grids and popovers. It also generalises colours, font families, spacings, shadows and more properties across the entire application where the library is a single source of truth. A component library of sorts makes building layouts and forms very rapid. We settled on Mantine for their extensive documentation, modern look and great DX. Mantine also offers a great deal of utility hooks that speed up development as well and supports dark themes out of the box.

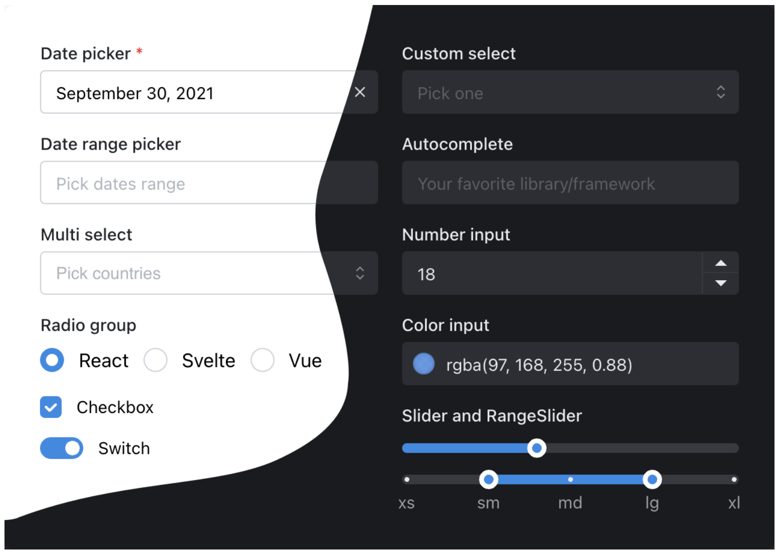

Date picker *
September 30, 2021
Date range picker
Pick dates range
Multi select
Pick countries
Radio group
React Svelte Vue
Checkbox
Switch
Custom select
Pick one
Autocomplete
Your favorite library/framework
Number input
18
Color input
rgba(97, 168, 255, 0.88)
Slider and RangeSlider

Figure 27: An example of Mantine components

JavaScript is great partly because it is extended with features like modules, a type system, JavaScript XML, top-level await and more. Unfortunately, those are not understood by the JavaScript engines in browsers or Node. To this end, this abstract JavaScript code needs to be transpiled into a simpler version, like ES5. The TypeScript compiler does this, along with bringing types to an otherwise weakly typed language. This project is written entirely in TypeScript and for the sake of this document, TypeScript and JavaScript are interchangeable. It is natural and hardly arguable that TypeScript is the language of choice.

Lastly, a bundler must be chosen. A bundler takes all separate JavaScript modules and bundles them into one file that can be linked to the `index.html` that the server exposes. This process happens when the project is being built for production, but also development. Most bundlers also provide a built-in development and preview server and takes care of hot module replacement (HMR)[1]. Webpack is commonly used, but also notorious for being incredibly slow. Vite is a toolkit

[1] Frameworks with HMR capabilities can leverage the API to provide instant, precise updates without reloading the page or blowing away application state (Vite, n.d.)

that includes a bundler with the aforementioned features, but also a built-in TypeScript compiler and TSX support. Vite is extremely fast because it has been built on top of esbuild, making 10 to 100 times faster than Webpack (E. Wallace, n.d.). Moreover, project setup is very intuitive with a single NPM command.

Tools

The choice for IDE is not trivial to the project, as long as they are rich in functionality relevant to the implementation process. WebStorm and VSCode are both contenders and having experience with JetBrains' environment, WebStorm was decided on. Nevertheless, both are available for free under an educational licence. For all intents and purposes, this project is performed in an educational context and not a commercial one.

The use of a version control system (VCS) is non negotiable. The university hosts a GitLab server and this server is also used for the machine learning part of this project. It only makes sense to use the same server here. We did decide that it is better to host a separate repository for the web application, however. There are several points against a monorepo, most notable of which is the lack of interdependence between the subsystems, because the only connection between them is Firebase. Another point to be made is the fact that only one person has been allocated to the development of the web application, so a shared repository is not compulsory.

Other tools include code linters and formatters. Industry standards are ESLint and Prettier. For both a custom configuration was used across the entire project to ensure high quality code and equal format across the entire repository. The configurations of these tools are declared in `./package.json`.

Graphs

The decision at hand after having determined what stack to use was what data we need to display and in what form. We were inspired by the live busyness overview that Google Maps displays for certain stores and locations, where the live busyness is compared to a daily overview of the average busyness. This captures a few important analytical values in one graph, namely the live traffic per 15 minutes/hour/day, the distribution of live traffic over time, the average traffic per 15 minutes/hour/day and the distribution of the average traffic. For those reasons we decided to make this graph the main event on the Analytics page.

To embrace the liveness of the web application and showcase the real-time updates best, the live-average bar chart is followed by a historic cumulative traffic count line chart. The right bound moves along with time as the traffic flows in. To prevent this graph from becoming too crowded, only the past hour is shown. Combined, that makes this graph great for seeing real-time detailed overview as opposed to the time-aggregated values in the graph above.

Aesthetics

Design is inherently subjective and comes mostly with experience. But there are certain rules and guidelines that one ought to follow to create a professional and meaningful product. As stated before, Mantine allows us to outsource minor design concerns such as buttons and text inputs without neglecting their importance. This left us room to focus on major design choices, such as the layout of different pages and what graphs to display.

Documentation & Reference

In this section a brief documental overview will be presented. Not every file and component will be covered for the reasons that that is outside the scope of this project and the fact that inline documentation and descriptive component names should clue future developers in sufficiently. Moreover, knowledge of React and specifically React Router V6 will be assumed in order to lay out the implemented routing strategy. All of the external packages have their own respective documentation, which will be assumed too. All path references are relative to the repository root.

Folder Structure

In the root of the folder, there are six folders and a bunch of configuration files for Firebase (`./.firebaserc`, `./firebase.json` and `./firestore.rules`), NPM (`./package.json`), Vite (`./vite.config.js`) and TS (`./tsconfig.json`) and also an `./index.html`. The folders of `./node_modules`, `./dist` and `./.firebase` are irrelevant to development as they are libraries and the build destination and their contents are generated automatically. The folders `./src`, `./public` and `./functions` contain the source code. `./public` contains static files that are shipped with the rest of the application to the browser. `./functions` is a separate NPM package that is generated by the Firebase CLI and contains the source code for the cloud functions. `./src` contains all the components and the root of the React app (`./src/views/App.tsx`).

`./package.json` describes this project as an NPM package. This allows commands defined in `./package.json/scripts` to be executed by NPM. So can the Vite development server be started with

```
> npm run dev
```

and deploy directly to Firebase through the Firebase CLI with

```
> npm run deploy[:hosting|:functions|:firestore]
```

Design Pattern: `./src`

The web application has been built according to a more advanced design pattern that would be generally required. This has been done with scalability in mind, where the codebase can seamlessly accept more complex and interdependent extensions. As `./src` contains the course code of the app, this is where the VACS (views, assets, components, services) design pattern unfolds. `./views` contains one or more components directly tied to a `<Route />` in `./src/views/App.tsx`. `./assets` contains static content like images and RegEx's. `./components` contains the majority of the React components and is further divided based on component type (e.g., layout, modal, auth). `./services` contains contexts, custom API connectors, like for Firebase (`./src/services/firebase.ts`) and Google Maps (`./src/services/GoogleMaps`). Each of the aforementioned folders and their subfolders may have a `utils` folder with TS files that export one default utility function which can only be imported by the siblings of the `utils` folder. Global and general utility functions should be put in `./src/services/utils`.

As stated, `./src/views` contains a loose resemblance of the `<Router />` in `<App />` (`./src/views/App.tsx`) in the form of the folder structure. So does `./src/views/home` contain the page that is rendered to the `<Outlet />` in `<AppShell />` (`./src/components/layout/AppShell.tsx`) as the index of the `<Router />`. Notable is that `<AnalyticsPage />` (`./src/views/analytics/AnalyticsPage.tsx`) itself also renders an `<Outlet />`. This is so when the dynamic URL `/analytics/:id` is accessed, an `<AnalyticsPage />` is rendered to the outer `<Outlet />`, and then a `<SessionAnalytics />` (`./src/views/analytics/SessionAnalytics.tsx`) is rendered to the inner `<Outlet />`, creating an application that renders only what is necessary based on what has been requested.

That is to say that changing to another session with `/analytics/:otherId`, the `<AnalyticsPage />` is not actually rerendered; merely it's inner `<Outlet />`. All other components in `./src/views` are rendered directly to the outer `<Outlet />`, except `<LoginPage />` (`./src/views/analytics/SessionAnalytics.tsx`), as this is rendered outside `<AppShell />`.

Cloud Functions: `./functions`

There exists another NPM package inside the repository, namely under `./functions`. This is an automatically generated subproject by the Firebase CLI, because it requires a TypeScript configuration that is slightly different from the root one and is tailored for deployment to the cloud. `./functions/package.json` defines scripts that are used in the linting and building of the source folder, but should not be used individually. These separate scripts are combined by the `firebase deploy` command, which should be used through `deploy` in `./package.json`. The build directory is `./functions/lib`.

The folder `./functions/src` contains the source code for the Cloud Functions, which is currently only one and is defined in `./functions/src/index.ts`. The exported function imports utility functions from `./functions/src/utils`. These utility functions are the logic for computing the specific and average statistics.

Simulator

Early on in the development process there was a need for representative data for developing the statistical caching logic, building proper looking graphs that are performant enough on large datasets. This data should be generated automatically to remove the necessity to maintain a static collection of data points, which obviously would be rather time consuming to do. To this end, we wrote a mathematical model that can generate traffic objects within a specified time frame, usually 08:00 to 18:00. This data had to be distributed close to a normal distribution in order to simulate the expected trends we would observe within this timeframe. The crucial step here is calculating the Δt based on the absolute t value (i.e., t's close to the mean of the timeframe should generate tighter Δt's and vice versa). An interface to this simulator is on the otherwise hidden route `/sim`.

This was achieved by starting with a normal distribution and transforming it in such a way that the t-axis represents legible time. This means that one step equals one second, which makes a 10 hour domain $[0,\ 36000]$. Define $m = 0$ to be the lower bound and $M = 36000$ to be the upper bound. We also didn't use a strict normal distribution, because some terms do not matter as much on the previously defined domain. Starting off with the initial normal distribution

$$N_{init}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

we omit $\sqrt{2\pi}$ from the numerator of the first multiplicand and the $\frac{1}{2}$ from the exponent, as they are insignificant in comparison. Moreover, we define $\mu = \frac{1}{2}M = 18000$ so the optimum is in the middle of the specified domain, but this need not specifically be the case. The definition of $\sigma = 15000$ is derived from the shape of the curve we would like to see and in particular, the non-zero value we need at $N(m)$ and $N(M)$. Lastly, we need to control how many traffic objects we generate. Therefore, the entire function is scaled such that its integral over $[m, M]$ approximates 2000, or any number otherwise if desired. The scaling factor turns out to be 1000. This leaves us with the final function

$$N(t) = \frac{1000}{\sigma} e^{-\left(\frac{2t-M}{2\sigma}\right)^2} \sim \boldsymbol{\mathcal{N}}(\mu, \sigma^2)$$

This function does not give us the tools we need, however. We are more interested in the area under the curve as that defines the amount of traffic. We need to solve the $\Delta t = t_2 - t_1$ for any given t_1 and t_2 such that $\int_{t_1}^{t_2} N(t)dt = 1$. To this end, we look towards the origins of the integral and implement a Riemann sum across the domain $[m, M]$. We create rectangles with height $h = N(t_i)$, base $b = t_{i+1} - t_i$ and area $A = 1$ in a way such that the next rectangle has width $t_{i+2} - t_{i+1}$. If $A = bh$, then $b = \frac{A}{h}$ and so $\Delta t_i = t_{i+1} - t_i = \frac{1}{N(t_i)}$ and $t_{i+1} = t_i - \frac{1}{N(t_i)} = I(t)$, which is a recurrence relation on t and $t_0 = m$. We now have a tool to calculate consecutive t_i's based on the relative timestamp for each traffic object we want to generate. Mathematically, this is hard if not impossible to solve and certainly outside the scope of this programme. Luckily, we can approach this problem programmatically with a loop, keeping track of the intermediate value of t_i, using it in the next iteration t_{i+1} and stopping when $t_i \geq M$.

In `./src/views/simulator/util/writeData.ts`, $I(t)$ is implemented by creating a `DataPoint[]` (`./src/views/analytics/AnalyticsPage.tsx`) whose timestamp values are shifter from $t = 0$ to 8:00. With the parameters tuned as above, 10 hours will be generated, so 8:00 to 18:00. In each iteration, a record $T\colon i \rightarrow t$ is kept track of for the next iteration. After the loop has finished, a new document in `sessions` is made and the entire `DataPoint[]` is written to the `sessionData` collection.

Security

Firebase provides authentication out of the box. Although various OAuth providers are supported, we opted for a simple email/password combination. All routes are protected if no user is authenticated by means of not rendering the contents to relevant `<Route />` components. To this end, a `<RequiresAuth />` (`./src/components/auth/RequiresAuth.tsx`) components wraps `<AppShell />`. The login logic itself is contained in `<LoginPage />` (`src/views/auth/LoginPage.tsx`). Firebase implements the OAuth 2.0 standard and for further documentation and implementation details we refer to their documentation.

Time Zones

In the case that future work is performed it is important for those to understand how time zones are dealt with. Chart.js requires a time adapter. Moment has been chosen for this and is used by itself throughout the project as well. Moment extends the functionality of the native Date API and is relatively timed to the local machine. Firebase, however, has its own implementation for datetime and executes cloud functions at UTC. This creates the situation where the inference is mate at UTC+1, but the statistics are run at UTC. The frontend is developed to deal with this by accepting timestamps in UTC and converting them to local time; UTC+1 in this case.

Testing & Results

Machine Learning Model

Throughout the development process, two main methods for testing were used: manual testing and full system testing. At every stage of the development of the machine learning model, extensive manual testing had to be carried out to ensure a robust system to comply with Mindhash's requirements. The tests used did not only make the system more robust and free of errors but it also helped speed up the development process which was critical due to the long waiting time associated with labelling and training for a model.

During the labelling process, a data collection tool was created, before using the tool, it was tested against file overrides, ensuring every LiDAR frame is accompanied by an image and making sure every frame is saved from memory to the filesystem. This was done by running the tool for an hour and running a script to check if any of the expected files were missing. At first, some errors were observed leading to duplicate frames and mismatched image-to-data mapping, and as a result of these tests, they were quickly fixed in preparation for a data collection run.

To test the trained model without being burdened by the training time, the group capitalised on the time needed to label the training data. At about every 50 labels interval, a model was trained on the labelled data to quickly test for flaws and reiterate on the model using updated training parameters. To test the accuracy of the iterations, a group member would go and carry out real life tests which they reported back to the group for immediate reiteration. Testing in this manner did not only allow the group to fix errors but also was a major time saver.

Full System Testing

Finally, the model was tested with the rest of the system to complete a full system test. All of the components of the system were integrated together and tested in a simulated and then real environment. In the simulated environment, a bicycle pass was generated at a random interval between 5 and 10 seconds and was communicated to the web application. Many issues were uncovered during the testing like timezone mismatch and protocol errors but they were quickly resolved in between testing sessions. More than error fixing, the full system test allowed the group to measure delays and reiterate to improve on total inference time.

Legal Considerations & Compliance

Important to companies of any kind and size seeking to implement projects are their legal implications. MINDHASH is a small company and does not have a dedicated legal department. It is for this reason we will have to consider legal implications and compliance ourselves. There are three topics of law that we will consider: data protection, licences and intellectual property.

Data Protection Law (GDPR)

In the territory of the European Union the General Data Protection Regulation holds. Two important definitions to consider are that of the Controller and Processor. In simple terms, a data Controller is the body that determines the purposes and means of the processing of personal data (Wolford, 2018). The data Processor is the body who processes personal data on behalf of the Controller (Wolford, 2018). In terms of this project, MINDHASH and group 6 are jointly the Controller. Google Cloud is the Processor (Google Cloud, n.d.-a).

Another important aspect of the GDPR is the territorial scope. It states to apply to data processing of the personal data of EU data subjects, regardless of whether the Processor or Controller are in the EU themselves (Wolford, 2018a), and in particular behavioural data as long as this behaviour is observed in the EU (Wolford, 2018a). For this project, that means that the fact that MINDHASH is located in The Netherlands is not definitive for GDPR compliance. Neither is using Google Cloud and/or Firebase, located in Frankfurt, Germany (europe-west3) or Middenmeer, The Netherlands (europe-west4) respectively (Google Cloud, n.d.).

The GDPR states that personal data is any information that is identified or identifiable to a natural person (Wolford, 2018). The data that the system will be collecting is LiDAR data, in the shape of a point cloud and paired with timestamps. This can in no way be used to identify EU citizens, in particular bicycle lane traffic in the greater Enschede area. This leaves us with the fact that the system does not collect personal data as defined in the GDPR and thus it need not be compliant.

From the moment other data will be collected, such as account info in the web app, this system must be GDPR compliant.

Intellectual Property Right

Important to this project, maybe more than others, is the consideration of intellectual property (IP) rights. Issued by MINDHASH to the University of Twente and implemented by students in an educational context, IP right of the proposed system is not as straightforward as usually would be the case. Engaging in courses at the University of Twente issued by an external company legally places the IP right out of the hands of the students developing the IP and into the hands of said company. This can be overruled by the appointed Examination Board at the students' request. (VSNU & NFU, 2020).

Software Licences

Since it has been determined that this project is executed in an educational setting (VSNU & NFU, 2020), most software licences that are needed for this project are available and justifiable under their educational licence. Others are available under an open source licence like Apache 2.0 (↗) or MIT (↗).

We required licences to the following software: Livox SDK, livox_detection_simu, Ubuntu, gcc, TensorFlow, SocketIO and PyCharm. All of these, apart from PyCharm, are open course and commercially usable, since Livox SDK and SocketIO are published under the MIT licence (Livox, 2020; SocketIO, 2018), livox_detection_simu and gcc are published under the GNU General Public licence (Livox, 2021; GCC, n.d.), TensorFlow is published under the Apache 2.0 licence (TensorFlow, 2021) and Ubuntu is published under Canonical's proprietary intellectual property rights licence (Canonical, 2015).

Software required for the web app are React (↗), TypeScript (↗), ESLint (↗), Prettier (↗) and Mantine (↗). Luckily, all but one project are published under the MIT licence (Ramos, 2018; Zakas, 2021; Fox, 2019; Rtishchev, 2021), with TypeScript being published under the Apache Commons 2.0 licence (Hegazy, 2014). The web app has been developed in JetBrains' WebStorm IDE and version control will be managed on the university's GitLab, both of which hold an educational licence (JetBrains, 2021; GitLab, n.d.). Lastly, the webapp's backend and deployment environment Firebase comes in a free tier and paid tier, the latter of which is required for this project. Both are eligible for commercial use (Google Cloud, 2022; Google Developers, 2021).

Discussion & Future Work

Applying machine learning to point cloud data for classification is a complex problem which was difficult to solve within ten weeks. However, it is a very interesting problem with many points which could be further improved or investigated.

Additional Metrics

Given more time, a high priority would be to gain further insights into the traffic that is being monitored by the sensor as defined by the first objective of the project. The speed of the cyclists is one metric which would be of interest to municipalities as it is important to consider for safety. In the point cloud frames collected, a trail can be seen behind the cyclists. This is due to the fact that the data is collected over a period of time (in this case 0.5 seconds) and the object is moving. The team believes that this trail could be used to infer through regression the speed of the detected objects. Additionally, the distance between cyclists and their postures are also interesting points of data to measure and display. Both can be used to measure safety and gain insight into the behaviours of cyclists.

Edge Implementation

Hardware

The current solution relies on a laptop with the implemented software to be connected to the sensor in order to process the data. According to the first and second project objectives, the solution should be both robust and integrated. While the current solution is portable and can be integrated into a single device, it is quite large and contains unnecessary components which are not needed for the device. This could be improved through the usage of a smaller edge computing device such as a Raspberry Pi 4 with an FPGA module. This not only allows for the overall system to be smaller and more easily integrated, but also less expensive. While it has not been tested, in order to be able to use this type of computing device, the software would require some optimisation. One large bottle neck of the system is the way in which files are being stored.

Once this problem is solved the system will be much closer to real-time and should be able to work with less computational resources.

Occlusion

FInally, one prominent limitation of the current system is when (almost) complete occlusion occurs. Because the sensor uses light to gather information, it is possible for objects with high reflectivity (such as bicycles) to block other objects that are behind them. In the majority of cases this is not a problem, as it is rare for an object to be fully blocked by another. Usually at least some portion of the object will still be ‘visible’ to the sensor and can still be detected. However, it is not possible to detect an object that is fully occluded. This edge case can be solved through the usage of multiple sensors at different angles. This would allow for higher coverage such that even if occlusion occurs at one angle, the other sensor should still be able to gather data that the other cannot. Furthermore, to reduce the risk of obtaining false negative detections when partial occlusion occurs, it might be beneficial to add more frames containing partially occluded bicycles to the dataset.

Web Application

As outlined in the Design Project course description, we are required to deliver an MVP. By all means have we achieved this. Despite the web application being a complete and fully integrated system, there are a few improvements that we would have liked to implement given a more generous timeframe.

First of those would be to finish the other pages on the dashboard Security and Settings. We would have liked moderators to be able to manage and create their accounts through the Security page, but the only way of doing so currently is through the Firebase Console. Various settings could have been implemented, ranging from theme colour to the granularity of the graphs. This would require an extensive overhaul, however. It would have been good practice to take these changes into account when developing, but the choice was made to focus on speed more than extensibility.

The second recommendation is to implement more data visualisers, as has been later asked to do by MINDHASH. On the Live Analytics page, there are two cards that have no content which are dedicated to this inquiry. In order to do so, one will have to create a new key in the stats cache and update the stats logic accordingly. Then filling a table of sorts goes in the same way as data is fed elsewhere.

Thirdly, performance could be improved with the use of proper React contexts and lazy loaded components. Implementing this is not hard by any means and will drastically improve loading performance, especially when the other pages grow in complexity. However, as has been argued earlier, performance is not an important metric for this dashboard and as such we have not done so.

References

1. Andreas Geiger, Philip Lenz, & Raquel Urtasun (2012). Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Conference on Computer Vision and Pattern Recognition (CVPR).

2. Bloch, Blumberg, Laarts (2012) Delivering large-scale IT projects on time, on budget, and on value. McKinsey Quarterly, January 2012.

3. Brosnan, M., Petesch, M., Pieper, J., Schumacher, S., & Lindsey, G. (2015). Validation of Bicycle Counts from Pneumatic Tube Counters in Mixed Traffic Flows. Transportation Research Record: Journal of the Transportation Research Board, 2527(1), 80–89. https://doi.org/10.3141/2527-09

4. Cai, L., & Zhu, Y. (2015, May 22). Data Science Journal. https://datascience.codata.org/articles/10.5334/dsj-2015-002

5. Canonical. (2015, July 15). Intellectual property rights policy. Ubuntu. Retrieved September 22, 2022, from https://ubuntu.com/legal/intellectual-property-policy

6. Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1907-1915).

7. Ciric, D., Lalic, B., Gracanin, D.,Tasic, N., Delic, M., & Medic, N. (2019). Agile vs. Traditional approach in project management: Strategies, challenges and reasons to introduce agile. Procedia Manufacturing, 39, 1407-1414.

8. Clegg, D., & Barker, R. (1994). Case Method Fast-Track: A Rad Approach (Computer Aided System Engineering) (1st ed.). Addison-Wesley.

9. Fox, J. (2019, January 3). prettier/LICENSE at main · prettier/prettier. GitHub. Retrieved September 22, 2022, from https://github.com/prettier/prettier/blob/main/LICENSE

10. GitLab. (n.d.). GitLab Pricing. Retrieved June 22, 2022, from https://about.gitlab.com/pricing/

11. GNU. (n.d.). License. GNU GCC. Retrieved September 22, 2022, from https://gcc.gnu.org/onlinedocs/libstdc++/manual/license.html

12. Google Developers. (2021, November 9). Google APIs Terms of Service. Retrieved September 22, 2022, from https://developers.google.com/terms/

13. Google Cloud. (2022, September 20). Google Cloud Platform Terms Of Service. Retrieved September 22, 2022, from https://cloud.google.com/terms/

14. Google Cloud. (n.d.-a). Google Cloud & GDPR. Retrieved September 19, 2022, from https://cloud.google.com/privacy/gdpr

15. Google Cloud. (n.d.). Select locations for your project | Firebase Documentation. Firebase. Retrieved September 19, 2022, from https://firebase.google.com/docs/projects/locations

16. Google Cloud. (n.d.). Vertex AI. Retrieved September 19, 2022, from https://cloud.google.com/vertex-ai

17. H. Ramos. (2018, September 8). react/LICENSE at main · facebook/react. GitHub. Retrieved September 22, 2022, from https://github.com/facebook/react/blob/main/LICENSE

18. Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

19. JetBrains. (2021, September 1). TOOLBOX SUBSCRIPTION AGREEMENT FOR STUDENTS AND TEACHERS. Retrieved September 22, 2022, from https://www.jetbrains.com/legal/docs/toolbox/license_educational/

20. Kendrick, Tom (2015). "Chapter 3. Identifying Project Scope Risk". Identifying and Managing Project Risk: Essential Tools for Failure-Proofing Your Project (3rd ed.). AMACOM. pp. 50–52.

21. Ku, J., Mozifian, M., Lee, J., Harakeh, A., & Waslander, S. L. (2018, October). Joint 3d proposal generation and object detection from view aggregation. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1-8). IEEE.

22. Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. (2019). Pointpillars: Fast encoders for object detection from point clouds. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 12697-12705).

23. Lidar 3-D Object Detection Using PointPillars Deep Learning - MATLAB & Simulink - MathWorks Benelux. (n.d.). https://nl.mathworks.com/help/lidar/ug/object-detection-using-pointpillars-network.html

24. Livox. (2020, December 4). Livox-SDK/LICENSE.txt at master · Livox-SDK/Livox-SDK. GitHub. Retrieved September 22, 2022, from https://github.com/Livox-SDK/Livox-SDK/blob/master/LICENSE.txt

25. Livox. (2021, January 20). livox_detection_simu/LICENSE at master · Livox-SDK/livox_detection_simu. GitHub. Retrieved September 22, 2022, from https://github.com/Livox-SDK/livox_detection_simu/blob/master/LICENSE

26. Livox. (2019). Livox Mid Series User manual. (n.d.). https://www.livoxtech.com/3296f540ecf5458a8829e01cf429798e/downloads/Livox%20Mid%20Series%20User%20Manual%20EN%2020190118.pdf

27. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

28. Luo, J. Q., Shi, Y., & Xie, C. J. FSSPD: Fast Single Stage Pedestrian Detector for Autonomous Driving. Dim, 3, 3.

29. matlabengine. (2022, September 13). PyPI. https://pypi.org/project/matlabengine/

30. Mendelow, A. L., "Environmental Scanning--The Impact of the Stakeholder Concept" (1981). ICIS 1981 Proceedings. 20. https://aisel.aisnet.org/icis1981/20

31. M. Hegazy. (2014, July 14). TypeScript/LICENSE at main · Microsoft/TypeScript. GitHub. Retrieved September 22, 2022, from https://github.com/microsoft/TypeScript/blob/main/LICENSE.txt

32. Nair, V., & Hinton, G. E. (2010, January). Rectified linear units improve restricted boltzmann machines. In Icml.

33. Nobelius, D. (2001). Empowering project scope decisions: introducing R&D content graphs. R&D Management, 31(3), 265–274. https://doi.org/10.1111/1467-9310.00215

34. Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum pointnets for 3d object detection from rgb-d data. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 918-927).

35. Rtishchev, V. (2021, January 7). mantine/LICENSE at master · mantinedev/mantine. GitHub. Retrieved September 22, 2022, from https://github.com/mantinedev/mantine/blob/master/LICENSE

36. Shenhar, A. J. (2001). One Size Does Not Fit All Projects: Exploring Classical Contingency Domains. Management Science, 47(3), 265–274. https://doi.org/10.1287/mnsc.47.3.394.9772

37. S. Lauesen, "Software Requirements: Styles and Techniques," Addison-Wesley, Boston, 2002.

38. Shin, K., Kwon, Y. P., & Tomizuka, M. (2019, June). Roarnet: A robust 3d object detection based on region approximation refinement. In 2019 IEEE intelligent vehicles symposium (IV) (pp. 2510-2515). IEEE.

39. Simon, M., Amende, K., Kraus, A., Honer, J., Samann, T., Kaulbersch, H., ... & Michael Gross, H. (2019). Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).

40. SocketIO. (2018, February 28). socket.io/LICENSE at main · socketio/socket.io. GitHub. Retrieved September 22, 2022, from https://github.com/socketio/socket.io/blob/main/LICENSE

41. TensorFlow. (2021, November 29). tensorflow/LICENSE at master · tensorflow/tensorflow. GitHub. Retrieved September 22, 2022, from https://github.com/tensorflow/tensorflow/blob/master/LICENSE

42. Vite JS. (n.d.). Vite Features. Vitejs.dev. Retrieved November 8, 2022, from https://vitejs.dev/guide/features.html

43. VSNU & NFU. (2020, September). Addendum to the Set of Guidelines Dealing with IPR and Students. UTwente. Retrieved September 19, 2022, from https://www.utwente.nl/.uc/f1196c20b01025b23ca01f2f78302a5c804c27b72c83500/Addendum%20to%20the%20Set%20of%20Guidelines%20Dealing%20with%20IPR%20and%20Students.pdf

44. Wallace, E. (n.d.). esbuild - An extremely fast JavaScript bundler. Esbuild. Retrieved November 8, 2022, from https://esbuild.github.io/

45. Wolford, B. (2018, November 14). Art. 3 GDPR – Territorial scope. GDPR.Eu. Retrieved September 19, 2022, from https://gdpr.eu/article-4-definitions/

46. Wolford, B. (2018a, November 14). Art. 4 GDPR – Definitions. GDPR.Eu. Retrieved September 19, 2022, from https://gdpr.eu/article-3-requirements-of-handling-personal-data-of-subjects-in-the-union/

47. Wolford, B. (2018b, November 14). Art. 5 GDPR – Principles relating to processing of personal data. GDPR.Eu. Retrieved June 21, 2022, from https://gdpr.eu/article-5-how-to-process-personal-data/

48. Yan, Y., Mao, Y., & Li, B. (2018). Second: Sparsely embedded convolutional detection. Sensors, 18(10), 3337.

49. Zakas, N. C. (2021, August 5). eslint/LICENSE at main · eslint/eslint. GitHub. Retrieved September 22, 2022, from https://github.com/eslint/eslint/blob/main/LICENSE

50. Zhou, Y., & Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4490-4499).

Appendix A: Manual

Web App

The following is a manual to get started with the bicycle dashboard and starting up the inference module. Go to https://bicycle-counter-b71c4.web.app/. If you are accessing for the first time or if your session has expired, you will see a login screen. Use this to log in with

- **Username:** bikes@mindhash.nl
- **Password:** 123456

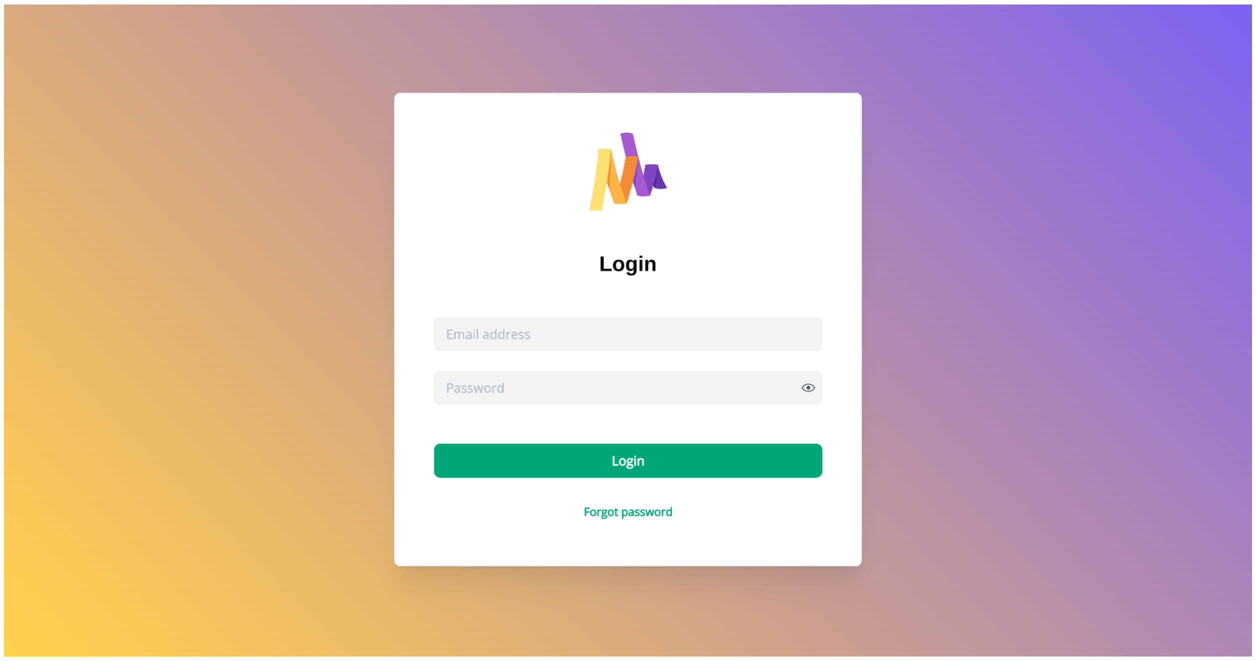
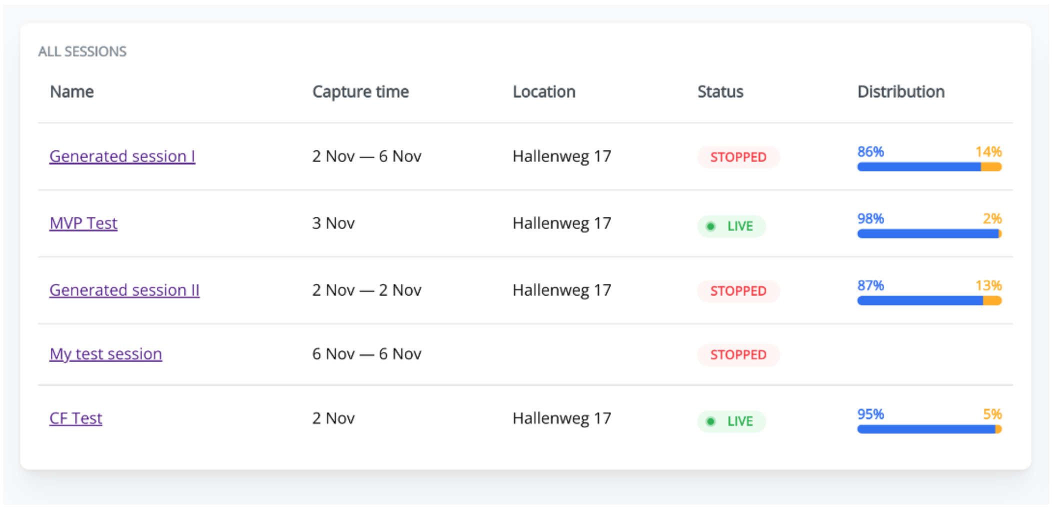


Figure 28: Web application Login page (/login)

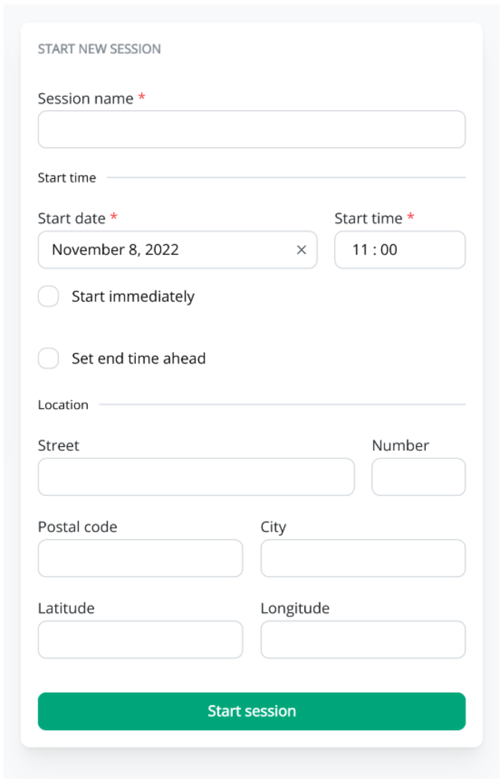
Once you are logged in or if you were previously, you will see the homepage of the dashboard. Here is an overview of all the sessions that have been recorded so far. You can also start a new session with the form on the right. Visit a session's analytics page by clicking on their name.


ALL SESSIONS

Name	Capture time	Location	Status	Distribution
Generated session I	2 Nov — 6 Nov	Hallenweg 17	STOPPED	86% 14%
MVP Test	3 Nov	Hallenweg 17	LIVE	98% 2%
Generated session II	2 Nov — 2 Nov	Hallenweg 17	STOPPED	87% 13%
My test session	6 Nov — 6 Nov		STOPPED	
CF Test	2 Nov	Hallenweg 17	LIVE	95% 5%

Figure 29: Web application dashboard Home (/)

If you want to create a new session, the session name and start date are required. You can choose a date and time in the future or click the checkbox "Start immediately". Initially, the session will last indefinitely or until the session is stopped manually. More about this later. If the end time is known beforehand, it can be set by clicking the "Set end time ahead" checkbox. Below it, an end time and date field will appear which can be filled accordingly. Optional is the location. The session can be created by clicking the big button "Start session". If an input is given that is not acceptable, an error message will appear below the field that generated the error.


START NEW SESSION

Session name *

Start time

Start date * November 8, 2022

Start time * 11 : 00

Start immediately

Set end time ahead

Location

Street

Number

Postal code

City

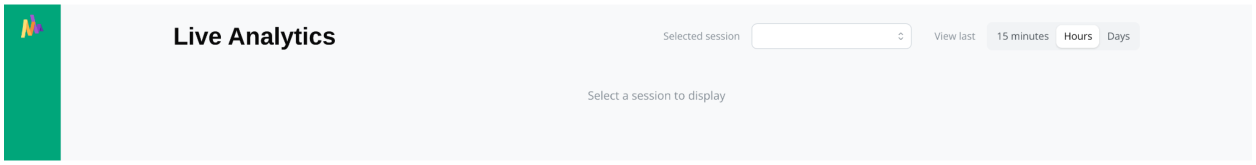
Latitude

Longitude

Start session

Figure 30: Web application dashboard Home (/)

An existing or new session’s analytics can be viewed. To do so, click on the monitor icon called Analytics on the left. This will take you to the analytics page. From the selector in the top right, select a session. Once a session has been selected, the analytics will be loaded. Next to the selector is a segmented control for setting the time scope. Changing this will change the granularity of the graphs and of the comparisons made in the three cards on the top of the page. Set it to “days” if you want an overview of a week and “15 minutes” if you want an overview of the last few hours.



Live Analytics

Selected session

View last 15 minutes Hours Days

Select a session to display

Figure 31: Web application Analytics page, select session (/analytics)

The bar chart in the middle of the screen has a switch labelled “From start” and “Last 4 hours”. This changes the view of this graph to be from the start until the last recorded instance or it will show you the last 4 units of time and move along live.

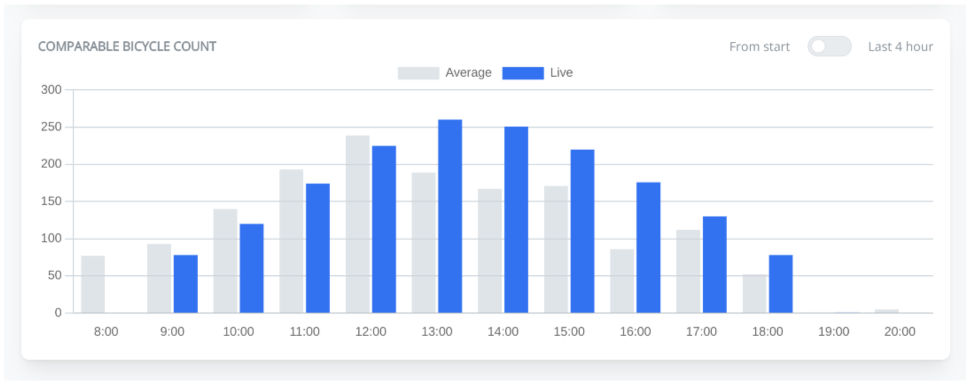
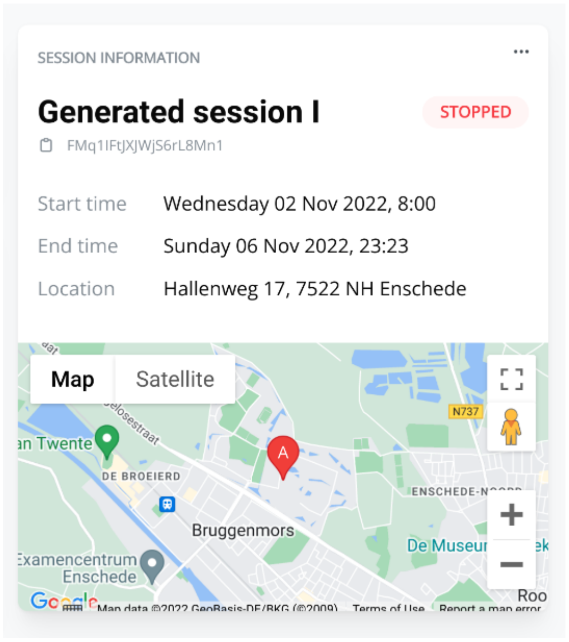


Figure 32: Web application Analytics page, with session (/analytics/:id)

If the session needs to be stopped, click the menu in the top right of the card that shows the session data (name, start time, etc) and click “Stop session”. This will be disabled if the session is already stopped. A green or red badge will indicate if the session is live or stopped respectively. This card shows the session’s unique ID underneath the name. Click this text to copy it to your clipboard. The text will become green on success. This ID is needed to start the inference module in the next step. In the navigation bar on the left, there are two buttons in the bottom left.

Use the sun/moon icon to change the theme of the dashboard to light or dark and click the door with the arrow to log out.



SESSION INFORMATION

Generated session I STOPPED

FMq1IFtJXJWjS6rL8Mn1

Start time	Wednesday 02 Nov 2022, 8:00
End time	Sunday 06 Nov 2022, 23:23
Location	Hallenweg 17, 7522 NH Enschede

Figure 33: Web application Analytics page, with session (/analytics/:id)

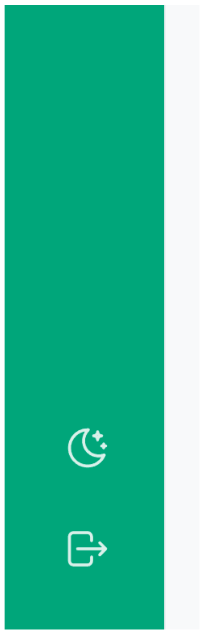
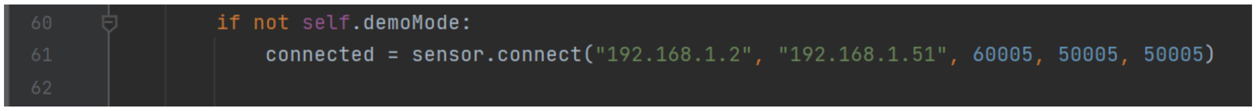


Figure 34: Web application change theme button (top) and logout button (bottom)

Client

Firstly, it is of importance to clone the *liveInference* branch from the GitLab repository of the project. Next, install dependencies required to run the *main.py*. After this, make sure that you have installed the 2022b version of MatLab and furthermore have installed the MatLab Engine API for Python for the Python version that you would like to use (>= 3.9 is recommended).

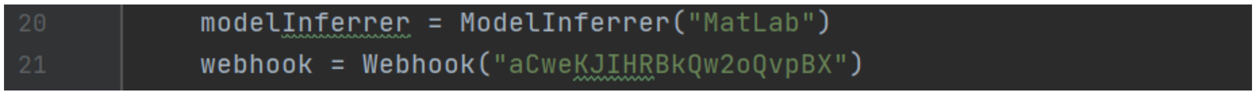
Before being able to connect to the LiDAR unit, one has to make sure that the ethernet adapter to which the unit is connected is on the same subnet as the ip address of the LiDAR sensor (default is 192.168.1.51), i.e. “192.168.1.2” and make sure the subnet mask is set accordingly (i.e. “255.255.0.0”). The address can be changed in the source code of the application in the */OpenPyLivox/Lidar.py* file, on the line shown in Figure 35.



```
        if not self.demoMode:
            connected = sensor.connect("192.168.1.2", "192.168.1.51", 60005, 50005, 50005)
```

Figure 35: The line where one should manually change the addresses of the host (first) and sensor (second).

In order to successfully deploy the system, it is of importance to change the session ID in the source code of the application. This should be done on the line shown in Figure 36, which is furthermore located in the /OpenPyLivox/Lidar.py.



```
        modelInferrer = ModelInferrer("MatLab")
        webhook = Webhook("aCweKJIHRBkQw2oQvpBX")
```

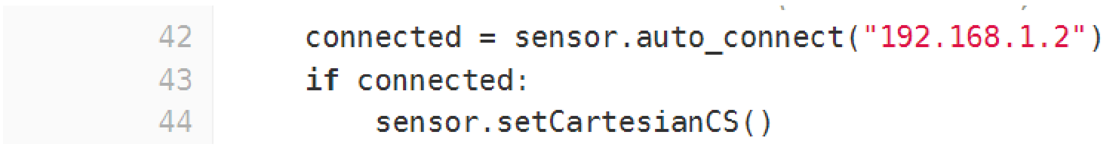
Figure 36. The line where one should manually change the session ID of the created session.

After this, simply run the *main.py* file with your Python version that has the MatLab Engine API for Python and the other dependencies installed and the detection results will be automatically sent to the WebApp.

Data collection

If you would like to simply collect point cloud data and store this data in a .csv file, first clone the *CollectionAdaptation* branch from the GitLab repository. Next, install dependencies required to run the main.py.

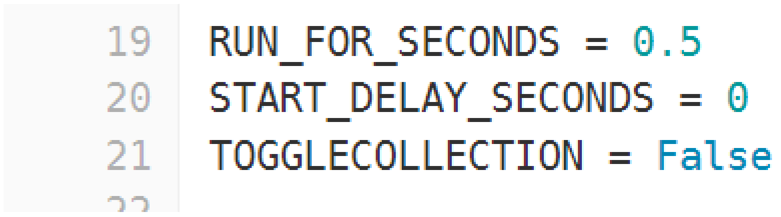
Before being able to connect to the LiDAR unit, one has to make sure that the ethernet adapter to which the unit is connected is on the same subnet as the ip address of the LiDAR sensor (default is 192.168.1.51), i.e. “192.168.1.2” and make sure the subnet mask is set accordingly (i.e. “255.255.0.0”). Make sure to change the ip address of the local machine in the code to the actual local address of the machine in use (from the perspective of the ethernet adapter to which the LiDAR sensor is connected). This should be done in the *main.py* file on the line shown in Figure 37.



```
    connected = sensor.auto_connect("192.168.1.2")
    if connected:
        sensor.setCartesianCS()
```

Figure 37. The line where one should manually change the ip address to the one of the local machine on the network with which the LiDAR sensor is connected

Next, set for how many seconds a single point cloud data frame should take (and thus how many points a single data frame should contain, note that the sensor collects data points at 100,000 pts/s) by setting the global "RUN_FOR_SECONDS" variable. Next, set how many seconds you want the sensor to wait with the collection of data points by setting the global "START_DELAY_SECONDS". Lastly, set the collection toggle mode to be either "True" or "False" by setting the global variable "TOGGLECOLLECTION". The global variables used to set above parameters are on the lines shown in Figure 38.



```
RUN_FOR_SECONDS = 0.5
START_DELAY_SECONDS = 0
TOGGLECOLLECTION = False
```

Figure 38. The lines at which one can set a number of collection parameters.

When the "TOGGLECOLLECTION" variable is set to "True", the Lidar will continuously collect point cloud data frames of the specified length after pressing the spacebar. When the "TOGGLECOLLECTION" variable is set to "False", the sensor will only collect a single point cloud data frame when one presses the spacebar.

The collected point cloud data frames are stored as a .csv file, with each point having the following properties: *x, y, z, intensity, time, returnNum*. The time is accurate up to a microsecond.

Appendix B: Team Contract

Project Title: LiDAR Traffic Monitoring

Client: Mindhash

Supervisor: Dr. Duc V. Le

Team information

Name	Grade expectations		Preferable day, time, and place to have extra meetings	Preferred method of communication
	Minimum grade	Expected grade		
Remy	7.5	8	Any time of week except Weds during lunch.	WhatsApp
Sebastiaan	8.0	9.5	Any time, preferably not on Thursdays however	WhatsApp, Discord, Teams, in-person
Malek	7.5	9.0	Anytime except Fridays 11:00 - 15:00	Text: Whatsapp Voice: Discord/Teams
Jasper	7.0	8.0	Anytime, given notice	WhatsApp, Teams
Omar	7.0	8.0	Anytime, given notice	WhatsApp, Teams, Discord

Project Goals

What are your team goals for the project? What do you want to achieve? What skills do you want to improve or work?

- To gain knowledge in the usage of a LiDAR sensor
- To develop a product that meets all the requirements.
- To enhance experience of machine learning in embedded systems
- To practise interacting with a client to develop a product
- To demonstrate mastery of the design process

Team Expectations

1. Participation
    1.1. We agree not to miss any meeting unless something is very important.
    1.2. We agree to actively listen to the other members and try not to lose focus.
    1.3. We agree to put effort into contributing towards the topics or issues being discussed.
2. Meetings
    2.1. We agree to come to the meetings on time.
    2.2. We agree to prepare ourselves with the ideas, and the tasks to be done before the meeting.
    2.3. We agree to conduct ourselves formally during meetings to stay productive
3. Communication
    3.1. We agree to use the following tools for communication: WhatsApp, Slack, Microsoft Teams
    3.2. We agree to decide a venue and time for the physical meetings.
    3.3. We agree to communicate with the team if we are unable to come to meetings beforehand.
    3.4. We agree to communicate with the team if we are going to be late to meetings beforehand.
4. Individual Accountability
    4.1. We agree that if a team member does not complete the assigned task in time without informing us, this is their responsibility.
    4.2. We agree to communicate our to-do tasks within the team, and if possible we send reminders to each other to have a submission on time.
    4.3. We agree to use Jira to assign and manage tasks.
    4.4. We agree to communicate with the others in the case that we are.
5. Conduct
    5.1. We will not tolerate rudeness in meetings as it affects the quality of the project.
    5.2. We will be disciplined and honest throughout the project.
    5.3. We agree to follow the orange card procedure for a warning to the team member(s).

6. Conflict

6.1. We agree to discuss the entire team's problem and mutually find a solution.

6.2. We agree to discuss the problem with our mentor(s) and project coordinator if we couldn't identify the solution.

6.3. We agree to conduct any disagreements in a civil manner, and allow for both sides to present their argument.

Team Agreements

I undersigned, hereby declare that I have read, agreed, and accepted all the rules and commitments mentioned in this document and will try my best to follow these rules during the Design Project. If the rules are not being followed by us, we will bear the consequences as mentioned.

Name:	Remy Benitah	[illegible]
Student number:	s2247372	
Date:	21 September 2022	
Name:	Malek Assaad	[illegible]
Student number:	s2374544	
Date:	22 September 2022	
Name:	Sebastiaan Hofstee	[illegible]
Student number:	s2300257	
Date:	22 September 2022	
Name:	Jasper van Amerongen	[illegible]
Student number:	s2384442	
Date:	23 September 2022	
Name:	Omar Mohamed Anwar Mohamed Elkady	Omar Elkady
Student number:	s2389541	
Date:	23 September 2022	