Research Article

Over 60,000 km in a year: remotely collecting large-volume high-quality data from a logistics truck

Christian Berger[1] · **Arpit Karsolia**[2] · **Federico Giaimo**[1] · **Ola Benderius**[3]

Received: 4 November 2021 / Accepted: 7 September 2022
Published online: 24 September 2022

Abstract
After the first successful large-scale demonstration of eleven self-driving vehicles at the DARPA Urban Challenge in 2007, research results from the competing teams found their way into advanced driver systems (ADAS) that support typical driving tasks like adaptive cruise control and semi-automated parking. However, as of today, SAE Level 4 vehicles are not commercially available yet, which would allow the driver to be inattentive for longer periods. Hence, SAE Level 3, which represents partial automation yet continuously monitored by a human operator, may provide a step towards a viable SAE Level 4 product especially for commercial freight logistics. However, large amounts of data from such freight operations is needed to study the unique challenges in such use cases. In this paper, we present the system and software architecture of an end-to-end data logging solution, which is capable of recording large volumes of high-quality data. The system is installed in a commercial truck that is in daily operation by a logistics company and hence, the recorded data is only accessible remotely (i.e., over-the-air). We report about the fail-safe system design, initial findings from over one year of operation, as well as our lessons learned. During its first year of operation, the truck was used for 210 days by the logistics company, out of which 193 days were logged resulting in more than 4.5 TB of data from five cameras, two GNSS–IMU sensors, and six on-board vehicle controller area networks (CAN) busses. We demonstrate the value of the proposed end-to-end approach for traffic and driver behavior research by analyzing the uploaded data in the cloud to spot critical events such as unexpected harsh braking maneuvers caused by lane merging operations.

Keywords Automated driving · Autonomous driving · Large volume data logging · Remote data logging

1 Introduction

The automotive industry is focusing enormous amounts of resources in the areas of autonomous driving, external vehicular communication, and remote monitoring and deployment of software. These recent developments demonstrate the growing need of a better understanding of how products are being used in the field; not only retroactively during regular workshop visits where relevant data logs can be downloaded and handed back to the manufacturer, but rather in real-time allowing a closer tracking of the performance and system health.

1.1 Background

The self-driving vehicles that competed in the 2007 DARPA Urban Challenge were very tailored to that competition and could only succeed in a few, very specialized

✉ Christian Berger, christian.berger@gu.se; Arpit Karsolia, arpit.karsolia@chalmers.se; Federico Giaimo, federico.giaimo@gmail.com; Ola Benderius, ola.benderius@chalmers.se | [1]Department of Computer Science and Engineering, University of Gothenburg, Gothenburg, Sweden. [2]Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden. [3]Department of Mechanics and Maritime Sciences, University of Gothenburg, Gothenburg, Sweden.

and selected demonstration scenarios. Even though, the potential of the technology for improving safety on public roads by aiming at taking out the human driver was clearly visible. Nevertheless, the majority of the teams as outlined by Buehler et al. and Berger and Rumpe (cf. [1, 2]) realized their technical approach by a three stages, pipes-and-filters data processing architecture: (A) *perception layer* that is interfacing with all sensors and the vehicle network to fuse the data into objects, (B) *decision making* that is receiving the aforementioned objects for safe path planning, and (C) *low-level control* that is selecting the most suitable trajectory from the previous layer to actuate the vehicle according to criteria such as safety and comfort.

One advantage of the aforementioned architectural design resides in the clear *separation-of-concerns* of the various aspects for coming up with a driving decision that, back in those days, was primarily motivated by analytical decisions codified into software. While the three stages design enabled the teams to more easily debug and fine-tune their systems to better meet the expectations from the competition with its simplifications, none of the contestants' robot vehicles would be able to manage traffic on public roads anywhere on the world to meet SAE Level 5 requirements. However, a different architectural design was emerging enabled by a paradigm shift in hardware-accelerated mass data computation by using GPUs complementary to CPUs. Thereby, a different approach to better cope with the sheer unboundedness of the variability and unstructuredness of sensor data from real-world traffic situations was enabled and approached by letting the computer figure out an appropriate algorithm by providing partially annotated input data and expected output behaviour. This trend of using AI/ML to significantly improve the performance of the perception layer has shown remarkable success both, in scientific publications such as Bojarski et al. (cf. [3]), and in demonstrations on public roads.

The main driver, though, for a primarily data-oriented architectural design is the access to sufficient, high-quality data to systematically evaluate the quality of the automatically generated AI/ML algorithms. However, determining the aforementioned attributes for the data set in use to train and test such algorithms is so far academically hard and economically challenging. Nevertheless, start-ups and long-term, established vehicle OEMs are equipping prototypical vehicles to collect large amounts of data to fuel the engineering pipelines for such AI/ML algorithms. However, commercially available data logging solutions mainly specialize in high-frequency, low-volume signals as found on typical vehicle controller area networks (CAN) to collect data such as acceleration, temperature, velocities, or pressure. In addition, customized data loggers for high-volume data from cameras or lidars typically require additional logistical procedures for data ingress to a cloud-enabled data analytics environment such as swapping physical data disks or downloading data using wired connections.

1.2 Problem domain and motivation

As outlined in the previous section, commercially viable solutions for logging high-frequency and high-volume data by using only cellular connections are missing. Therefore, we are aiming at providing a design for a data logger that is especially addressing the aforementioned use case. The solution presented in this paper was designed in the project *Highly Automated Freight Transports* (AutoFreight), a research and technology transfer project between Chalmers University of Technology (Chalmers), Kerry Logistics Sweden AB, Volvo Group, Trafikverket, Ellos Group AB, Combitech, Borås stad, Speed Group AB, and GDL Transport AB in Sweden. The motivation for the research project "AutoFreight" is to better understand prerequisites and constraints towards highly automated driving (SAE Level 4) on public roads (for example: highways). The project is based on two trucks, where one is used for experimentation on confined test sites, while the other truck is primarily used for collecting data for an a posteriori data analysis. Chalmers led the design and installation of the logging system for the second truck that was handed back to the logistics company for daily operations since October 2019.

1.3 Research goal and research questions

The research goal for this paper was to design and evaluate a data logging solution that is able to reliably provide services to collect data from multiple cameras, GNSS–IMU, and vehicular on-board networks. A specific requirement for this solution was that the data exchange for post-processing needed to be realized solely via cellular connections as the truck was inaccessible to the research team due to its daily operation by a logistics company. Hence, the following research questions are addressed:

RQ-1 How does a reliable system design meet the aforementioned research goal?

RQ-2 What experiences and lessons learned can be reported from one full year of operation?

1.4 Contributions

This paper presents and discusses design drivers and resulting decisions behind the system setup and configuration. As the project benefitted extensively from previous research results obtained at the Chalmers vehicle laboratory Revere,[1] we provide links to the software that we made available as open source. Next to the system design, we also reflect upon our experiences and lessons learned from the system design as a second generation of the logging system was designed and implemented for a research collaboration between Sweden and India.[2]

1.5 Limitations

Due to project agreements as well as GDPR regulations, the collected data is currently not accessible and so far, only the core software to realize the system is available as open source. However, discussions around providing access to the data are initiated to also let the research community benefit from the effort made in this project.

1.6 Structure of the article

The rest of this article is structured as follows: Sect. 2 presents and briefly discusses related works. Our methodology is described in Sect. 3 followed by a detailed system description in Sect. 4 to address RQ-1. We present initial results in Sect. 5 and discuss our reflections in Sect. 6 to address RQ-2. The paper concludes with Sect. 7.

2 Related works

Data sets fostering research and development in aspects of autonomous driving such as reliable perception, robust path planning with real-time adaption, and safe and comfortable vehicle control have been created and presented in multiple contexts. Yin and Berger [4] as well as Kang et al. [5] presented an extensive overview of publicly accessible data sets and their respective properties in recent surveys.

Prominent examples for such data sets, which have been extensively used for research, include the first of its kind named Kitti as reported by Geiger et al. [6]. This data set was created by one of the teams from the 2007 DARPA Urban Challenge (Team AnnieWay from KIT, Germany).

More recent examples include a data set from UK as reported by Maddern et al. [7] who describe their design of a logging system used to capture 1000 km over one year. While their data set is pushing academically driven data sets to a new level with respect to covered distances, our project is pushing these limits even further by capturing nearly 60,000 km during one year.

Commercially supported data sets include Nuscenes from Caesar et al. [8], which covers Boston and Singapore, provide seven times more annotations and 100 times more images than the pioneering data set Kitti. Another commercially supported data set is provided by Waymo as reported by Sun et al. [9], which is providing more cities from the US.

While the aforementioned presentation and discussion of available data sets provide only a subset of many available data sets as covered by Kang et al. [5], our project is the first of its kind to the best knowledge of the authors that is creating a data set using a truck in daily operations. Hence, we are pioneering a unique system design, as most other data sets have been created using car-like platforms, to which the involved researchers had access to on a regular basis. In contrast, our team had to design and install a platform in a truck but could not physically access the truck regularly once the truck was back in daily operation as the logistics company.

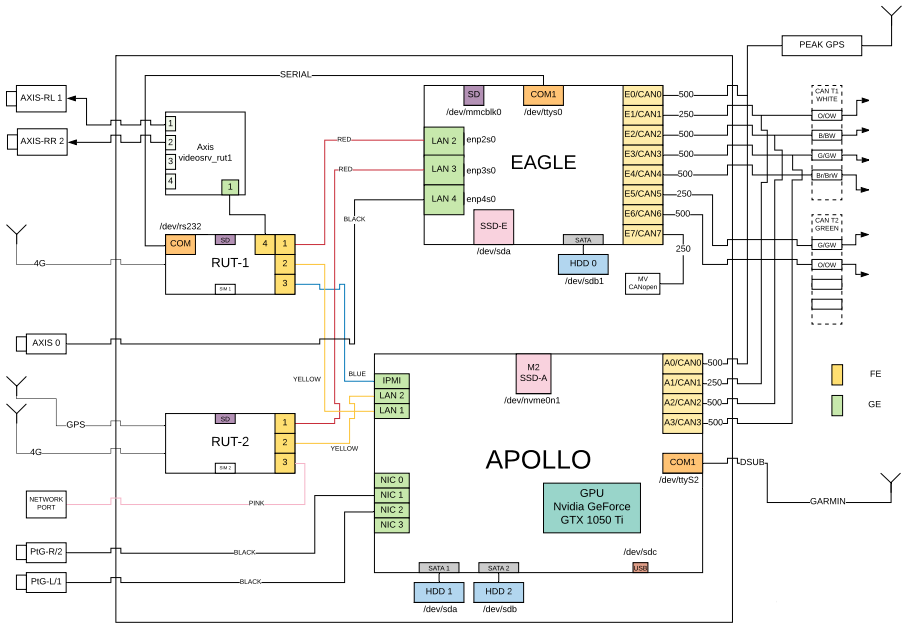
3 Methodology

In order to achieve the goals of the work, the adopted research methodology was design science [10], which concerns artifacts in context and their design and investigation. In the case of this work, the artifact is a solution for a reliable high-volume data logger using cellular connectivity for the automotive context.

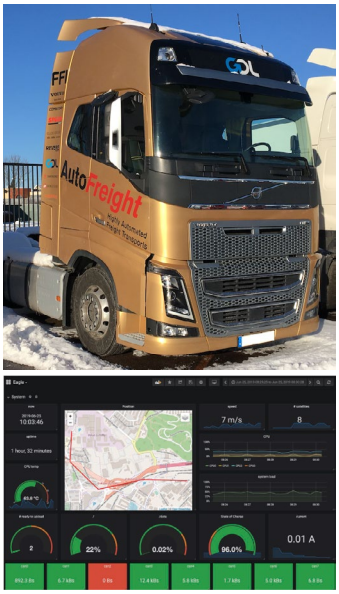
The artifact was designed, implemented, and evaluated to measure its effectiveness in achieving the research goal. In the design phase, the architecture of the system and its components, its power infrastructure, and even the software stack were devised so they would meet a number of functional and non-functional requirements as described in Sect.4. The resulting architecture was then implemented at Chalmers' vehicle laboratory Revere. Once the resulting system was tested successfully and handed over to the logistics company for daily operations, the continuous evaluation phase began by analyzing the performance of the data logger and the quality of the recorded data, in

[1] https://chalmers.se/en/researchinfrastructure/revere/Pages/default.aspx.

[2] https://volvogroup.com/en-en/news/2020/nov/sitis-connected-safety-bus-platform.html.



(a) Block diagram depicting the network architecture of the data logging system.



(b) The tractor used for installation, and a picture of the remote monitoring dashboard.

Fig. 1 Architecture of the system to meet the functional and non-functional requirements: The computing nodes are able to access the vehicular networks independently from each other as well as the GNSS–IMU system. Furthermore, at least one camera facing forward is always accessible in case of failure in the network connection or due to one failing computing node. The system is accessible only via an encrypted maintenance channel; in addition, a separate encrypted channel is used to upload the collected data for data analytics in the cloud

order to assess how closely the designed artifacts were fulfilling the research goal. Results and especially reflections therefrom are reported in Sect. 6.

4 Design of the system architecture

The project had to meet the following functional requirements that emerged during discussions with the project stakeholders:

FR-1 Accessing the system as well as the data during and after a data logging run must be possible only via secure remote wireless connections, as the truck is either parked in a fenced location or simply not close enough to Gothenburg to economically and timely swap disks.

FR-2 Visual forward facing camera footage, vehicle location, and vehicular network data must be captured on all trips in case of individually failing system components.

FR-3 The system must not interfere with the truck driver's usual routine and driving behavior.

FR-4 The system must only record data outside protected areas like Gothenburg harbour.

FR-5 The system must do a self power-off when not in use or when the state-of-charge of the battery is low.

The following non-functional requirements had to be met:

NFR-1 The system design shall not exhibit a single-point-of-failure (SPOF) compromising on the aforementioned functional requirements.

NFR-2 The system design shall allow for lossless data handling to not compromise on a posteriori data analysis.

4.1 System architecture and network design

These requirements were realized using the system architecture as depicted in Fig. 1 and listed in Table 1. To spot potential SPOFs in the system design, we created a directed, acyclic graph (DAG) for the network architecture of all involved hardware components to visualize the data, which is flowing from sensors at the bottom nodes of the DAG to the root nodes that represent the cellular modems

Table 1 System components for the data logging system

Component	Eagle	Apollo	Sensors and Connectivity	
Manufacturer	PCengines	custom-made	Cameras (forward facing)	
Mainboard	APU2D4	Supermicro X11SCZ-F	1x Axis M1124	1280x720 at 25FPS
CPU	AMD GX-412TC (4 cores)	Intel Core i9-9900K (8 cores)	2x PtGrey BFS-PGE-31S4C-C	2,048x1,536 at 17FPS
RAM	4GB	2x 8GB Samsung (M378A1K43CB2-CTD)	Cameras (rearward facing)	
Main OS disk	64GB m-SATA SSD	500GB Toshiba (KXG-50ZNV512G) NVMe M.2	2x Axis F1005-E	1280x720 at 25FPS
Fail-over OS disk	32GB SD-card	16GB USB pen-drive	Location & Vehicle State	
Data HDDs	1x 8TB HGST Ultrastar He10 (HUH721008ALE604)	2x 8TB HGST Ultrastar He10 (HUH721008ALE604) as RAID-1	PEAK-PCAN	position at 10Hz
GPU	–	Nvidia GeForce GTX 1050 Ti	GNSS–IMU	accelerations at 10Hz rotations at 10Hz
LAN	3x Intel I210AT 1GBit	2x Intel I210AT 1GBit onboard and 4-port Intel I350-T4 1GBit	Garmin 18LVC	position at 1Hz
CAN	2x PEAK 4-channel miniPCIe	PEAK 4-channel PCI	CAN	6x truck channels
Clock synchronicity	PTP client to Apollo, NTP	PTP Grand Master Clock using Garmin 18LVC	Connectivity	
Power supply	12V	M4-ATX 12V, 250W	2x Teltonika RUT955 3G/4G modem	

at its top. Then, we analyzed whether a single node or an edge between two nodes would break the data flow to meet FR-2 so that we could identify where to add redundancy along the data flows by adding an additional camera, computing node, or network link.

FR-1 requires an approach to only access the system securely using a cellular connection over 3G/4G as it could not be guaranteed that the truck may be in close proximity to a wifi access point or an Ethernet cable that is in reach overnight. In combination with NFR-1, a single cellular router was also deemed as a potential SPOF and hence, two independent 3G/4G routers with individual SIM cards were installed. The installation of two separate computing nodes to reduce the risk of a SPOF for a failing combination of computer and camera was also influential to the design decision of installing two separate cellular routers to meet NFR-1: The upper computing node (named Eagle) in Fig. 1 has a dedicated connection to an IP-camera (named Axis0); the same design was also chosen for the lower computing node (named Apollo) that has dedicated access to two cameras (labelled PtGrey0 and PtGrey1).

Each computing node is reachable from both cellular modems to mitigate a potential modem failure. Finally, both nodes have access to the truck's on-board vehicle networks via CAN as well as to a GNSS–IMU system that is also accessible via CAN. The possibility of a failing GNSS was compensated by adding a separate GNSS-receiver to Apollo that also enabled this node to act as IEEE1588 PTP grand master clock for the network to synchronize all clocks in the computing nodes and camera sensors. Furthermore, both cellular modems allow for low-sampling GNSS location tracking as well. NFR-2 was met by installing a GPU into Apollo to enable hardware-accelerated lossless video compression for the PtGrey cameras. Details about the system components as well as sensors are presented in the Table 1.

4.2 Design of the power architecture

As the system needs energy to complete its tasks and facilitate data transfer not only when the truck is powered by its engine but also when it is powered off, the truck was equipped with an auxiliary power supply using a lithium-ion battery to meet FR-3. This battery provides stable voltage to compensate for voltage drops that may occur when the engine is cranked, and hence, protecting sensitive electronic equipment from power fluctuations. The auxiliary power supply enables to operate the system even for several hours after the truck's engine has been shut off to provide operational time for uploading data without depleting the vehicle's batteries. While driving, the stock alternator in the vehicle provides enough current to charge the auxiliary battery.

Since lithium-ion batteries are sensitive to excessive depletion, it needs to be carefully monitored and protected. Therefore, the battery has been equipped with a battery shunt that keeps track of the state-of-charge. Approximately 80% of the battery capacity can be used before there is a risk of battery damage so, if the battery charge reaches 20% charge level, a safety relay will

disconnect the battery to protect it. The state of charge is additionally provided to the software via a CAN-open interface to safely shut down the computing systems. The entire system operation does not require any interaction from the truck driver.

4.3 Design of the software environment

The systems are powered by Arch Linux using Linux kernels 4.19.31-rt18-1-rt-lts on Eagle and 5.2.0-rt1-1-rt on Apollo. The Nvidia GPU is using the proprietary driver 430.40. Our software stack[3] is maintained and deployed as Docker services using Docker 19.03.1-ce on both systems. On Eagle, we are using 17 separate microservices to interface with the forward facing Axis camera and the two rearward facing Axis cameras, to record the data to disk, and to monitor the system health. On Apollo, 25 microservices are used to interface with the PtGrey cameras, to losslessly convert their video streams into compressed h264 NAL units using hardware-acceleration through the Nvidia GPU, to additionally compress their video feeds using Intel QuickSync into a lossy VP9 format, and to record the forward facing Axis camera. The Apollo system is also computationally powerful enough to allow for running ML-models to perform object detection (cf. Sect. 5) on the platform, which can be used to spot interesting events around the truck.

The recorded data is continuously uploaded whenever the system is connected to the Internet using an encrypted data link to an encrypted storage system for automatic data analysis to spot interesting events as described in Sect. 5. The system is continuously monitored using a dashboard as depicted in Fig. 1b, which is realized with Grafana, InfluxDB, and collectd to observe critical system diagnostics such as system temperatures, CPU and disk consumption, state-of-charge, vehicle speed and location, as well as number of satellites in sight. Manual over-the-air maintenance of the system is realized using an encrypted channel into the system.

5 Initial results

For one trip of approximately 80 mins between Gothenburg and Viared (close to Borås), the sensors and vehicle networks generate approximately 203 GB data per hour on the system, broken down as follows: Each of the two PtGrey cameras create 96 GB/h after lossless compression and each of the three Axis cameras generate approximately 3.2 GB/h. The six CAN channels from the truck generate approx. 1.4 GB/h. The two 3G/4G modems manage to upload approximately 6 GB/h each in the region where the truck is usually operated. Even if the system would have enough power for 24 h of operation time, it would approximately take more than 60 h to upload all data from one single day via 4G. Despite the fact that log files are buffered on the system, the lossless video feeds are only kept for *interesting* events to be identified on the truck, such as unexpectedly harsh braking maneuvers as explained in the following as this data is most valuable for training neural networks that require the best possible image quality. Longer stretches of a trip with a low event density like highway driving are primarily captured using lossy video compression as it was deemed sufficient for documentation and research purposes.

The signals to be logged were divided into different sets depending on data source, data grouping, and offloading priority. To consolidate our findings, a period between January 15, 2020 until January 15, 2021 was selected and an overview of the data is provided in Table 2.

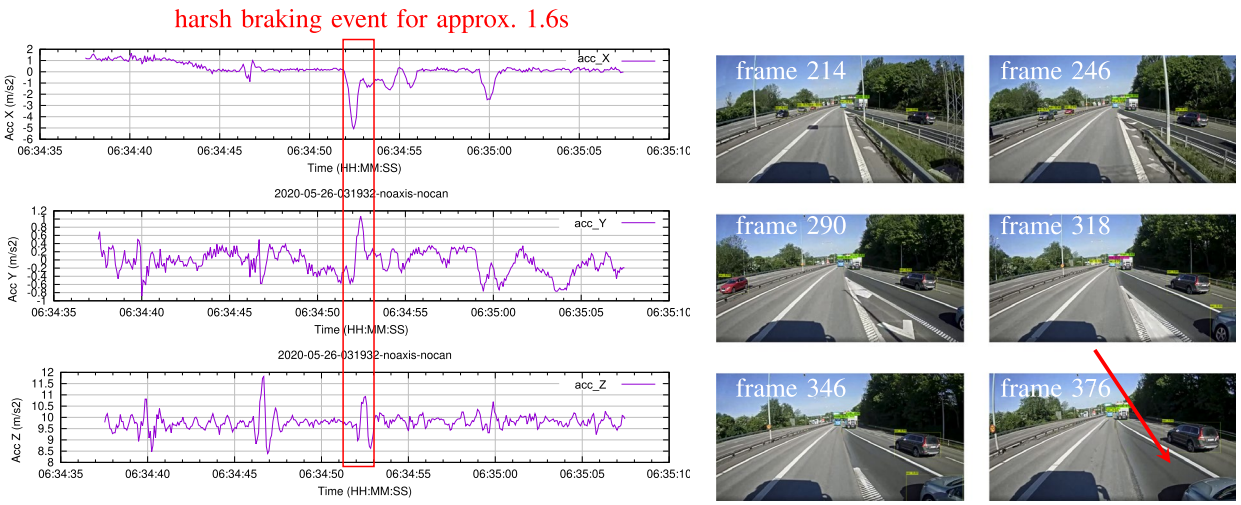
In order to conduct plausibility checks, each log file was automatically converted to CSV, PNG, and PDF for data analysis after uploading to the secure storage system. These checks include time synchronicity, offload rate, signal validity, and ground truth. Once the checks confirmed data validity, an analysis to spot events for *harsh braking scenarios* in the recorded data was conducted to demonstrate the value of the data set and the data processing chain.[4]

For our purposes, we chose to focus on longitudinal acceleration data and classified a harsh braking event as a data point less than or equal to −0.5 G (i.e., 4.9 $\frac{m}{s^2}$) [11]. While there are related metrics applicable to further classify such an event, we considered this threshold-based event as suitable and identified 23 events in total with only eight false positives. The following steps were conducted to detect, extract, and confirm a harsh braking event:

1. A script was written, which analyzed the acceleration data and extracted excerpts of 20s-30s of the data set satisfying the event condition.
2. The extracted data was plotted to identify relevant cases and to discard erroneous ones.
3. The start and end timestamps from the extracted data were used to trim the corresponding video data to visualize the identified event.
4. Using the video excerpts, a reported event was classified into three categories: (a) harsh braking (satis-

[3] Our software *OpenDLV* can be found on GitHub: https://github.com/chalmers-revere/opendlv.

[4] From the total driven distance, 59,699.5 km were captured in the analysis period January 15, 2020–January 15, 2021.



(a) Event in the acceleration data. **(b)** Traffic situation of the event with object detection.

Fig. 2 Event where the truck was forced to conduct an unintended deceleration maneuver; no collision happened

Table 2 Information about the log files

Number of days when data was logged	193
Total system uptime	9493 h
Total number of log files	18,340
Total size of uploaded data	4.5 TB
Total number of captured video frames	106,356,613
Average travelled distance/day	309.3 km
Total travelled distance	62,819.2 km[4]

Table 3 System health values during the operation

Component	Measurement	Values
Eagle	CPU temp. (°C)	min: 28, med.: 60, max: 88
Apollo	CPU temp. (°C)	min: 12, med.: 37, max: 61
Apollo	GPU temp. (°C)	min: 8, med.: 35, max: 59
Eagle	system load[5]	min: 0.08, med.: 2.91, max: 35.3
Apollo	system load	min: 0, med.: 0.58, max: 8.37
Power	state-of-charge (%)	min: 0, med.: 94, max: 100
Power	battery temp. (°C)	min: 0, med.: 19, max: 30
GNSS	visible satellites	min: 0, med.: 10, max: 12

fies event condition with video correlation) (b) data incomplete (signal error or video log not available), (c) false–positive (satisfies event condition, video evidence provides different perspective).

The threshold-based approach resulted in 18 days during the 193 days of logging, where 23 events satisfying the event condition were identified. From these events, five were classified as harsh braking, eight were classified as false–positive events, and ten had incomplete log data. An anonymized example of such an identified event is depicted in Fig. 2 showing a car merging from a right entry road forcing the truck to conduct a harsh braking maneuver to prevent an accident.

6 Analysis and reflections

In the following, we provide an analysis of the system design as well as report about our reflections. Table 3 summarizes the information from basic system health monitoring.

It can be stated that the system design can manage the computational demands from the software stack in a stable manner and the median system load is not exceeding the number of available CPU cores needed to run a process ready for execution. The passively cooled Eagle unit is typically much warmer than the Apollo unit that has active cooling. However, the climate conditions of the geographical region, where the system is in operation, have not caused in any operational errors so far. The median visibility of satellites when the truck is in service is around ten, allowing for good localization[5]

In Table 4, we list problems identified during operation, and upgrades to address them. Two unexpected software problems were identified: The problem regarding the SI-conversion could be corrected by a software upgrade to the platform and a software filter for the

[5] Cf. http://www.brendangregg.com/blog/2017-08-08/linux-load-averages.html.

Table 4 Changes and issues about the system

Type	Date	Description
fault/HW	2021-02-01	Power supply failure to Eagle HDD
info/HW	2021-01-28	Wear and tear NVMe disk: 4.92 TB written
fault/HW	2020-11-30	Power supply failure to Eagle HDD
fault/HW	2020-11-02	Power supply failure to Eagle HDD
fault/HW	2020-09-28	Power supply failure to Eagle HDD
fault/HW	2020-08-31	Power supply failure to PtGrey cameras
fault/HW	2020-08-17	Power supply failure to Eagle HDD
fault/SW	2020-07-06	Router 1 went offline but replied to SMS
fault/HW	2020-05-22	Power supply failure to Eagle HDD
fault/SW	2020-05-04	SI-conversion error for acceleration and rotation
fault/HW	2020-02-17	Connection failure to PtGrey cameras
fault/HW	2020-02-05	GNSS antenna of main GNSS unit broke down
fault/HW	2020-01-23 - 2020-01-31	Connection failure to Axis cameras
upgrade	2019-08-14	Increasing CPU frequency on Apollo to 1.1 GHz
upgrade	2019-09-27	Integration rear cameras
upgrade	2019-08-12	Upgrading to Nvidia 430.40
upgrade	2019-08-12	Upgrading to Linux kernel 5.2.0-rt
fault/HW	2019-04-24	CRC error reading data from disk

post-processing; the problem with the offline 3G/4G router could be traced to a software fault (root cause in external proprietary software) as the modem still replied to an SMS requesting its system status. So, it could be set into operation again by triggering a reboot of the modem via a special maintenance SMS.

The broken GNSS antenna needed to be replaced. However, the software stack could conduct a fail-over to one of the other GNSS sources until the scheduled workshop visit. Furthermore, as the IMU functionality of the preferred GNSS–IMU sensor was unaffected, only the update rate of the GNSS location data was temporarily reduced. The power-supply issues affecting the data disk resulted in the data being temporarily written to the OS disk until a power cycle of the logging system. The connectivity and power supply issues for the cameras temporarily resulted in missing log files for that particular camera. Overall, the MTTR was approximately one day for the majority of issues.

While the system already shows a decent performance, the ability to remotely power cycle any hardware component along the paths of the DAG is very valuable in case of an unexpected system behavior. Also, low-level system monitoring such as BIOS console over serial connection, Intel IPMI, or DMTF Redfish accessible wirelessly is essential for fault investigation. Both aspects have been incorporated in the second generation of the logging system that is implemented through the research collaboration in India.

7 Conclusions and future works

Data-driven engineering to realize, improve, and maintain AI/ML-enabled software systems require growing amounts of high-quality data from diverse traffic situations. Commercially available data loggers primarily focus on high-frequency, low-volume signals as typically found on vehicle CAN busses to capture velocities or accelerations for example. However, the AI/ML-enabled systems to improve the perception layer for upcoming generations of ADAS and highly automated systems require also large amounts of high-volume data from cameras and lidars. As no commercially available end-to-end solutions were available to address this task for collecting data from a logistics truck, which is in daily operations making swapping physical disks not an option, this paper presents the design of a fail-safe data logging solution, motivates the underlying design decisions based on functional and non-functional requirements, and discusses lessons learned after being in operation for more than one year. To the best knowledge of the authors, the collected data set is the first of its kind to cover nearly 60,000 km of inter-urban traffic situations from a truck's perspective.

The approach presented in this paper is filling a gap concerning the design of scalable data logging approaches for high-volume, high-frequency, high-quality data that is only accessible remotely due to operational constraints. Our future work is addressing the identified lessons learned primarily concerning fail-safety along the nodes and connections of the directed-acyclic graph, which is representing the data flow from the individual sensors to the wireless data transfer points in a research collaboration between Sweden and India, where a commercial bus is commuting daily between Bengaluru and Mysore. This 142 km route with an anticipated travel amount of 100,000 km during the planned operation will push the system design to even higher levels in terms of stability and endurance with respect to temperature and humidity, along with challenging cellular connectivity. Furthermore, new concepts to enable a swift data analysis within an academic context are also required in the back-end system to store, index, and post-process these amounts of incoming data.

Acknowledgements We would like to thank Fredrik von Corswant for his invaluable support with the design of the hardware and power architecture. Furthermore, we would like to acknowledge the support from involved project partners: Kerry Logistics Sweden AB, Trafikverket, Ellos Group AB, Combitech, Borås stad, Speed Group AB, GDL Transport AB, and Volvo Group.

Funding Open access funding provided by University of Gothenburg. This work was supported by Vinnova, Grant Number 2016-05413.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical approval The presented research did not involve human participants. The truck operators provided their informed consent to be part of the data collection.

References

1. Buehler M, Iagnemma K, Singh S (2009) The DARPA urban challenge: autonomous vehicles in city traffic, vol 56. Springer, New York
2. Berger C, Rumpe B (2012) Engineering autonomous driving software. In: Rouff C, Hinchey M (eds) Experience from the DARPA Urban Challenge. Springer-Verlag, London, UK, pp 243–271. https://arxiv.org/pdf/1409.6579
3. Bojarski M, Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel L. D, Monfort M, Muller U, Zhang J, Zhang X, Zhao J, Zieba K (2016) End to end learning for self-driving cars. NVIDIA Corporation, Holmdel. http://arxiv.org/pdf/1604.07316v1.pdf
4. Yin H, Berger C (2017) When to use what data set for your self-driving car algorithm: an overview of publicly available driving datasets. In: Proceedings of the 20th IEEE international conference on intelligent transportation systems (ITSC). Yokohama
5. Kang Y, Yin H, Berger C (2019) Test your self-driving algorithm: an overview of publicly available driving datasets and virtual testing environments. IEEE Trans Intell Veh 4(2):171–185
6. Geiger A, Lenz P, Stiller C, Urtasun R (2013) Vision meets robotics: the KITTI dataset. Int J Rob Res 32(11):1231–1237
7. Maddern W, Pascoe G, Linegar C, Newman P (2017) 1 year, 1000km: the Oxford RobotCar Dataset. Int J Robot Res (IJRR) 36(1):3–15
8. Caesar H, Bankiti V, Lang A. H, Vora S, Liong V. E, Xu Q, Krishnan A, Pan Y, Baldan G, Beijbom O (2020) nuscenes: a multimodal dataset for autonomous driving
9. Sun P, Kretzschmar H, Dotiwalla X, Chouard A, Patnaik V, Tsui P, Guo J, Zhou Y, Chai Y, Caine B, et al (2020) Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2446–2454
10. Wieringa RJ (2014) Design science methodology for information systems and software engineering. Springer, New York
11. Kamla J, Dawson A, Parry T (2017) Feasibility of using truck harsh braking incidents for predicting all vehicle accidents at roundabouts. In: Transportation Research Board 96th Annual Meeting, Jan, p 16