THE L3PILOT COMMON DATA FORMAT – ENABLING EFFICIENT AUTOMATED DRIVING DATA ANALYSIS

Johannes Hiller
Institute for Automotive Engineering, RWTH Aachen University
Germany

Erik Svanberg
SAFER Vehicle and Traffic Safety Center at Chalmers University of Technology
Sweden

Sami Koskinen
VTT
Finland

Francesco Bellotti, Nisrine Osman
DITEN – University of Genoa
Italy

Paper Number 19-0043

ABSTRACT

Analyzing road-test data is important for developing automated vehicles. L3Pilot is a European pilot project on level 3 automation, including 34 partners among manufacturers, suppliers and research institutions. Targeting around 100 cars and 1000 test subjects, the project will generate large amounts of data. We present a data format, allowing efficient data collection, handling and analysis by multiple organizations.
A project of the scope of L3Pilot involves various challenges. Data come from a multitude of heterogeneous sources and are processed by a variety of tools. Recorded data span all data types generated in various vehicular sensors/systems and are enriched with external data sources. Videos supplement time-series data as external files. Derived measures and performance indicators – required to answer research questions about effectiveness of automated driving – are processed by analysis partners and included for each test session.
As a file format, we chose HDF5, which offers a data model and software libraries for storing and managing data. HDF5 is designed for flexible and efficient I/O and for high volume and complex data. The usage of different computing environments for specific tasks is facilitated by the portability that comes with the format. Portability is also important for exploiting the rising potential within artificial intelligence (e.g. automatic scene detection and video annotation).
Based on lessons learned from past field tests, we defined a general frame for the common data format that is aligned with the data processing steps of FESTA "V" evaluation methodology. The definitions include representation of the source signals and a hierarchical structure for including multiple datasets that are gradually supplemented (post-processed or annotated) during the various analysis steps. By using the HDF5 format, analysis partners have the freedom to exploit their familiar tools: MATLAB, Java, Python, R, etc. First comparisons between time-series data in previous projects (e.g. AdaptIVe) and the proposed data format show a reduction in storage size of around 80 %, without losses in performance. Much of that is due to efficient internal compression and structuring of data. Considering the amount of objective data involved in automated driving, this leads to a great benefit, in terms of usability.
This paper presents a compact, portable, and extensible format aimed at handling extremely large amounts of field test data collected in automated driving pilots. As a harmonized format between tens of organizations performing tests in the L3Pilot project, the proposed format has the potential to promote data sharing as well as development of common tools and gain popularity for use in other projects. The format is designed to allow efficient storing of data and its iterative processing with analysis and evaluation tools. The format also considers the requirements of AI tools supporting neural network training and use.

INTRODUCTION

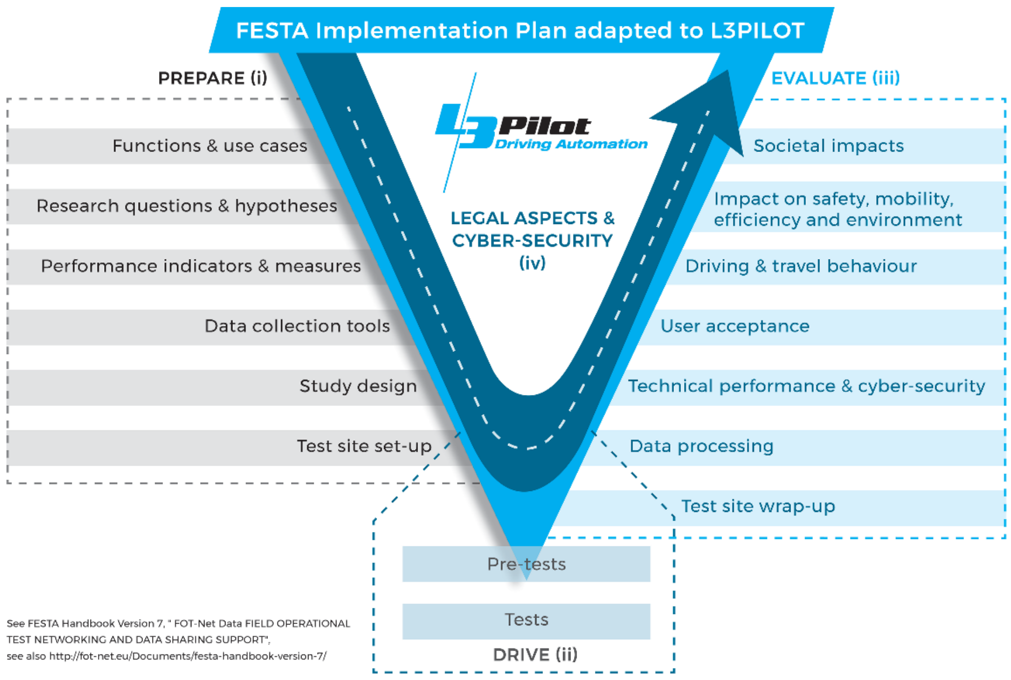
Automated Driving (AD) technology has matured to a level motivating large-scale road tests which can answer key open questions before market introduction. These newly-attained levels of maturity will ensure an appropriate assessment of the impact of AD. Of interest is what is happening both inside and outside of the vehicles. Also ensuring vehicle safety is of utmost importance as well as evaluating societal impacts. As a further point, the evaluation of emerging business models is of interest.
A point that has proven to be the crux in many previous projects is the data exchange between partners, as well as the evaluation. This led to devising a data format common to the whole project, thereby easing the exchange of data and the further development of evaluation processes and tools based on the data.
First, we will present the organization of data in previous projects and present the current project, L3Pilot [1]. After that, we will show how the process for deriving the requirements for the data format came together, followed by a few formats shortlisted for storage of the data. We will then describe the format itself and afterwards discuss it and its limitations.

PREVIOUS PROJECTS AND EFFORTS

Over the years, numerous projects have paved the way for advanced driver assistance systems (ADAS) and AD. Each of those projects had a slightly different approach to data acquisition, handling and evaluation. This section shortly picks out a few of these projects and gives some details on the used methods.
In 2008, a big European project was started with euroFOT [2]. It identified and coordinated an in-the-field testing of new intelligent vehicle systems with the potential for improving the quality of European road traffic. During the project, the effectiveness of various lateral and longitudinal control functions and active safety functions on public roads was assessed. Data collection and analysis was organized from test sites. Each test site was using similar (Matlab based), but still with differences, data formats and individual analysis tools [3].
With AdaptIVe [4] in 2014, the focus moved from ADAS to AD. In the three and a half years of the project, AD functions for scenarios such as parking and motorway driving were developed and demonstrated. Raw data for the evaluation was delivered by the vehicle owners. At the evaluation partner, the needed signals (cf. [5], Annex 3) were extracted and converted to an internal evaluation format, a mixture of CSV and MATLAB.
From past EU Field Operational Test (FOT) projects, at least TeleFOT [6] and DRIVE C2X [7] used fixed formats when gathering data from several test sites to a central data storage. These projects assessed, in respective order, in-vehicle navigation systems and short-range vehicle communication prototypes.
In 2017, the L3Pilot project was kicked off. L3Pilot will test automated driving functions (ADFs) in 100 cars with 1,000 test subjects across 10 different countries in Europe. The tested functions will be mainly of SAE automation level 3, some of them of level 4 [8]. Together, European automotive industry, suppliers and researchers will pave the way for large-scale field operational tests on public roads creating a harmonized Europe-wide testing environment. The overall objective of L3Pilot is to test and study the viability of AD as a safe and efficient means of transportation, explore and promote new service concepts to provide inclusive mobility.

SIGNAL DERIVATION / METHOD / REQUIREMENTS

L3Pilot follows the FESTA V-process methodology [9] of setting up and implementing tests with the four main pillars and adapting the methodology to suit L3Pilot needs (cf. Figure 1). The four pillars in L3Pilot are: Prepare, Drive, Evaluate and address legal and cyber-security aspects.
This paper focuses on the Prepare pillar with additional focus on the early phase of the Evaluate pillar. The Drive pillar is handled by the vehicle owners. As can be seen in the figure, the first step in L3Pilot is the definition of automation functions and use cases, with a major attention on motorways. In a further step, the research questions for this project are derived from the specified use cases. Accompanying the research questions are various hypotheses that this project will investigate. In order to do this, different data are needed. One part will be subjective data, i.e. data that originate from questionnaires and user evaluation. Particularly interesting from a data processing point of view, and therefore for a data format, are the data recorded in the vehicles. To specify what data to record, performance indicators and derived measures are defined, which are used to answer the research questions and confirm the hypotheses. Derived measures, in this context, are quantities that are directly calculated from source signal time-series data. These can be vehicular signals or information about the environment delivered by car sensors. Performance indicators, on the other hand, are no longer time-series data. They take different forms depending on the indicators. They can be single values giving a certain value or the average in a recording or in a specific scenario. However, they can also be histograms over some value in multiple occurrences of a driving scenario. The set of signals needed for the calculation of these measures results in a list of required signals, that are to be recorded during each session in the pilot vehicles. The full process following the FESTA-V up to the data can be found in [10].



FESTA Implementation Plan adapted to L3PILOT
PREPARE (i)
Functions & use cases
Research questions & hypotheses
Performance indicators & measures
Data collection tools
Study design
Test site set-up
L3Pilot Driving Automation
LEGAL ASPECTS & CYBER-SECURITY (iv)
EVALUATE (iii)
Societal impacts
Impact on safety, mobility, efficiency and environment
Driving & travel behaviour
User acceptance
Technical performance & cyber-security
Data processing
Test site wrap-up
Pre-tests
Tests
DRIVE (ii)
See FESTA Handbook Version 7, " FOT-Net Data FIELD OPERATIONAL TEST NETWORKING AND DATA SHARING SUPPORT", see also http://fot-net.eu/Documents/festa-handbook-version-7/

Figure 1. The FESTA implementation Plan adapted to L3Pilot

With the needed signals defined, the focus shifts towards the actual data. As stated before, different data are aggregated during the project. The three main sources are subjective data, objective data and video feeds. The data important for the L3Pilot Common Data Format (CDF) are the objective data. These include original vehicle signals and derived measures calculated from the source signals. In a project of the size of L3Pilot, there is not one platform running all the pilot vehicles and systems. One can think of platforms such as ADTF [11] or ROS [12] to just name two. Therefore, a simple export of the data collected in the car is not a viable option, since the export files of the different platforms are seldom compatible.
After the successful conversion, the Evaluate pillar starts with the data processing at the evaluation partner. Complementing the vehicle data will be data that originate from other, external, sources such as weather or map providers. Another factor is that multiple partners will be doing different analysis on the data, using different tools. One of the main programs used for post-processing and data analysis by the partners in this field is MATLAB. Additionally, in the previous years, Machine Learning has proven to be an important factor for the automatic detection of scenarios and video annotation. Therefore, a support of Python by the data format is of utmost importance. Considering statistical analysis of factors, some partners will also rely on R, SPSS or others. This leads to a requirement for a wide support of tools, platforms and programming languages.
Considering the aim of 1000 drivers in 100 cars, the amount of data recorded and transferred between vehicle owners and evaluation partners in terms of actual file sizes is another factor that should not be neglected. This leads to another criterion, the portability of files and results. For portability, memory efficiency is of course important.
Considering all these requirements, the common data format task force decided that a single file-based data format should be used in L3Pilot, to support an easy exchange of data between different partners. In order to reduce the amount of data that is transferred, compression was noted as another key feature to improve the process.
Within the L3Pilot project, it is agreed upon, that vehicle owners convert their datasets into the presented CDF. This enables the evaluation partners to use common tools to analyze the data, no matter which vehicle owner they work with.

CONSIDERED FILE FORMATS

As stated in the previous section, the decision upon a file-based format was taken quite early in the project. Therefore, going forward in the decision process, only file-based formats were considered. Solutions for big data storage were not further considered, although they can play an important role in the data management within an institution. As a first step towards the CDF, various file formats were evaluated and discussed. In various previous projects, that are partially listed in the section above, many different file formats were used. All of them have their own advantages and disadvantages. These differ strongly depending on the intended use of the formats. During the decision process for the CDF, many of them were evaluated and the pros and cons were compared. The

following formats were the ones taken into closer account during the decision process, as the task force already had experience with them, or they were deemed as promising.
A format that is commonly used among researchers in this field is the MATLAB file format [13]. The newest iteration is v7.3 that was introduced with MATLAB R2006b, however, the default version for files is v7. In it, data is stored in a binary format. It is a proprietary format, that is supported by MATLAB across all platforms that MATLAB supports. All datatypes included in MATLAB are supported and can be loaded and saved, while also supporting compression of data. One limitation is its strong link to MATLAB, however, for many research projects this was not an issue, since MATLAB is commonly used. This offers the opportunity to re-use existing tools from other or previous projects. The MATLAB file format was used in euroFOT [3] and internally during the evaluation in AdaptIVe [5].
A commonly used file format for exchange of numerical data is Comma-Separated Values (CSV). In it, values are simply stored as text, separated by commas (or any other type of delimiter), thus the name. Time-series are represented by new lines for each element. It is easy to use, doesn't need any updates and can be read and written by almost any program. However, there are also drawbacks to this format. CSV doesn't support the use of metadata. Therefore, value formats and e.g. minimum and maximum values of a column must be defined in a separate supporting document. One of the main advantages of CSV, the textual basis, is also one of its disadvantages, because it doesn't directly provide any compression and therefore takes up a lot of memory for long recordings. CSV was partially used for data transfers during the AdaptIVe project.
The Hierarchical Data Format (HDF) [14] exists in different versions. The current version is HDF5 revision 1.10.4 (as of January 2019). HDF was developed with portability in mind. It is supported by various languages such as C/C++, Fortran, Java, MATLAB and Python. Due to its support by a wide selection of programming languages, it can be used in the various available operating systems. It can therefore be easily implemented into different scripts and programs. HDF supports the storing of a wide array of datatypes including doubles, integers and strings. Additional datatypes can easily be added. To support portability, HDF has features for data compression. Different compression algorithms can be applied in order to save storage space. Metadata is stored in attributes in HDF files. This supports portability and simplifies the management of many files. Starting from version 7.3, the MATLAB file standard is based upon HDF5, thereby becoming compatible with HDF5 tools. As a disadvantage, for readability, HDF5 uses a binary format. An easy peek into the data without using the access libraries is therefore not possible. However, several data viewers exist. Editing the data can be another issue, depending on the viewers' capabilities. Another disadvantage is, that there is one inner core module on which almost all implementations of HDF5 rely upon. An error in this module would be devastating.
During L3Pilot, the main drawback of HDF5 was the somewhat limited documentation of programming examples. Another shortcoming and even a related bug were later found in Java libraries, as the main data viewer provided by HDF Group, HDFView, could not, at that time, display an array of compound datatypes used in the L3Pilot CDF. This was fixed by the HDF Group upon a report by the team.

L3PILOT COMMON DATA FORMAT

Considering the previously stated requirements, a file format was selected. The selection came upon HDF5. This allows us to define different datasets for the needed signals.
Since HDF5 supports a wide array of programming languages, the vehicle owners can use their preferred language and platform to convert their data recorded in their proprietary format. On the evaluation side, the corresponding partner should be able to use existing or preferred tools with small modifications. This allows for an efficient use of resources, as more time can be committed to developing and implementing new features.
HDF5 offers two ways of organizing data: datasets and groups [15]. Groups can contain zero or more HDF5 objects and can be accessed together using the group name. They can be hierarchically organized and have circular references. Datasets are where the data is stored. They are a collection of data elements, or raw data, and metadata that stores a description of the data elements, data layout, and all other information necessary to read, write and interpret the stored data. These data values can be of various datatypes. Already defined are datatypes such as double, integer, etc. However, the library also offers the opportunity to define new datatypes, if needed. Included in the datasets as well are the metadata, which include attributes. These attributes can be used to describe the contained data, thus allowing a verbal description of the data and, e.g., providing the unit of a logged signal. These attributes are independent and can be read and written without loading the complete HDF file. Datasets can be on the root level or belong to one or more groups. One advantage of datasets and groups is that each of them can be loaded individually without loading the complete file.
Derived through various iterations from the use cases and research questions, various logged signals are available on vehicle owner side. In order to allow for an efficient and quick access to the data, the signals are coarsely grouped according to their origin. For this purpose, datasets and groups are used by the CDF. All vehicle signals are organized in datasets on the top level of the file ("/") (cf. Table 1). The "egoVehicle" dataset contains all signals originating directly from the ego vehicle itself. This would be signals such as the ABS status or the speed

of the vehicle. All information about the lane markings, e.g. the distance to the lane markings and their type, is contained in the "laneLines" dataset. Dynamic objects and their properties such as speed and distance are saved in the "objects" dataset. Information from a global navigation satellite system (GNSS), e.g. GPS or Galileo, is stored in the "positioning" dataset. The previously discussed derived measures and performance indicators are calculated at the evaluation partners, and then stored in the "derivedMeasures" and "performanceIndicators" datasets.
In order to gain more information about the recorded trips, external data can be useful. Two important external data sources that were identified for the L3Pilot project are weather and map information. Weather information can be provided by various weather services and contains information about temperature, precipitation and cloud coverage. Map data provides information about the number of lanes, speed limits or intersections. These data are saved in the datasets "map" and "weather". These are located hierarchically under the "/externalData" group.
Some data cannot, or only with major difficulties, be derived from vehicle signals. One of these signals is, for example the secondary task performed by the driver during different situations. These kinds of signals are added by annotations through human experts, or students supervised by experts, watching the time-synced video feed of the recording. These annotations are normally added at the evaluation partner and not supplied by the vehicle owner. Annotations are located hierarchically under the "/annotations" group. For each annotation a subgroup is added with the name of the annotation. In this example it is "/sceondaryTask". It includes two datasets "comments" and "enum". In "comments" all comments that are additionally made by the annotator are stored, i.e. if there is an extraordinary reason for this annotation. The dataset "enum" contains the annotation as numerical value so that it can easily be reused in subsequent scripts and calculations. For the example of the secondary task, this would contain the annotated secondary task masked as an enum and the file time referencing it to the recording. Groups are used here, in order to have the possibility to flexibly extend the annotations.

Table 1.
All datasets according to the L3Pilot Common Data Format and the associated groups.

Group	*Dataset*	*Description*
/	egoVehicle	Signals directly concerning the ego vehicle
	laneLines	Information on the lane markings
	objects	A list of (dynamic) objects
	positioning	Information from the positioning system (local or GNSS)
	derivedMeasures	Contains all the derived measures
	performanceIndicators	Contains all the performance indicators
/externalData	weather	Contains information about the weather
	map	Contains map information
/annotation	-	Group containing various annotations
/annotation/secondaryTask	comments	Comments on the annotation by the annotator
	enum	The annotation values

Considering the memory usage and a memory efficient storage of the recorded data, the datatypes of all signals are carefully selected. For this purpose, the desired and expected precision of all signals is reviewed. Many signals such as vehicle speeds or accelerations come in high precisions, with many digits to the right of the decimal point. For these signals, the "double" datatype is used, which allows for a high precision. Other signals such as the ID of an object, the speed limit or the number of lanes are not needed with high precision. Therefore, these values are saved as integers. Another common signal in recorded data is the status of a system. In general, it takes very few distinct values that are known beforehand. These status signals are saved as "enums" in the CDF. HDF5 allows the definition of arbitrary enums. For the CDF, they are based upon "uint8" and take few distinct values. Table 2 summarizes the commonly used datatypes.

Table 2.
Different used datatypes in the Common Data Format, their sizes and examples

Datatype	*Size in byte*	*Exemplary value*	*Exemplary N/A value*
Double	8	3.14159265359…	NaN
int64	8	1545572564000	-1
int32	4	42	-1
enum (uint8)	1	ON (1)	N/A (-1)

In a project of the size of L3Pilot and with the wide variety of vehicle owners and sensor setups, not all the signals will always be available or will fit the same format. For the CDF this means, that some signals will not be available in some recordings. For missing values, “N/A” values are defined, i.e. values that are used when the signal is not provided. This makes it easier for programs and scripts to run on the data anyway. For floating point numbers, “not a numer” (NaN) [16] is used. Since NaN is not defined for integer types, values that are not expected to appear are used here, e.g. “-1”.
All signals in the CDF are synchronized between the datasets. In order to achieve this, a frequency of 10 Hz was selected for the project. For reference, each dataset contains two different time signals. The first is the “FileTime” which simply counts up in discrete 10 Hz steps from the beginning of the recording. This can be used for easy reference in the file itself. The second time signal is the “posix” time in milliseconds, named “UTCTime” in the file. This allows references to external data sources such as weather or traffic services. The posix time is the time in seconds, milliseconds or nanoseconds (depending on the application) since 00:00:00 on 1 January 1970 in UTC. It doesn’t include any leap seconds and therefore differs from the atomic time used in GNSS systems by currently 37 seconds (as of January 2019) [17].
Since not all signals are always recorded with the requested frequency of 10 Hz, interpolation methods are defined per signal. For continuously available signals, a simple linear interpolation is defined for most cases. However, since a linear interpolation is not applicable for status signals with few distinct values, a zero-order-hold (ZOH) interpolation is defined for these signals. Thereby each signal is held for one sample interval and then changes. In addition, a maximum time of loss is defined per signal. This is to prevent unreasonable behavior in signals when data loss was too long. For high precision variables this might be very close to zero seconds, for other signals (e.g. GNSS) this could be up to 10 seconds.
Another step towards memory efficient storage is the utilization of the HDF5 built-in compression. One common algorithm here is the “DEFLATE” compression [18]. This algorithm is not restricted by patents. Many different implementations for almost all common programing languages are available. The algorithm works especially well on data that does not change often. In that case, it will only save the value for the first occurrence and save the next value only if it changes. This saves a lot of memory especially for Boolean values and other mostly static variables.
In order to support faster I/O and memory efficient computing, HDF has a feature called chunking. Here, data is not saved in one continuous block in the file, but in so called chunks. These chunks are specified when creating the file according to the data that is to be stored. When reading the file, only one chunk at a time is loaded into the memory. This is especially useful, when handling large amounts of data. For the implementations in the L3Pilot project, the chunk size is selected in a way to get chunks of about 1 MB. Chunks are applied to the respective datasets. Since the size of a single timestep is known due to the mandated format and signals, the chunk size can be set accordingly.

DISCUSSION

For a preliminary assessment, memory consumption was measured for data coming from 32 hours of motorway data recorded in a previous project. The mean duration of one trip from this project is roughly 52 minutes. For that purpose, the raw data size was calculated from the known sizes of the datatypes and the length of the recordings. This is given in Table 3 as “Raw data, calculated” and taken as the reference for all other file sizes. This would lead to an average data file size of 84.54 MB for a recording length of 52 minutes. Using HDF5 for storing the data and activating the compression. the average reduction in file size is around 89 %. This results in an average file size of 9.63 MB. In terms of absolute memory, we can now save the recordings with only around 395 MB instead of the ~3.5 GB that would have been needed without the compression.

Table 3.
Comparison of file sizes for a selection of different file formats.

Format	*Mean file size*	*Relative*
Raw data, calculated	84.54 MB	100 %
HDF5, compression, DEFLATE	9.63 MB	11.17 %
csv	54.91 MB	64.94 %
mat file, v7	8.86 MB	10.48 %
mat file, v7.3	9.29 MB	10.98 %

For benchmarking, a few comparisons to other formats are done. The first one is CSV. All data that is written to the HDF5 files is taken and written to a csv file using the MATLAB function *dlmwrite*. This simply writes a matrix to a csv file. This results in an average file size of 54.91 MB. Compared to the raw calculated data size, this is only ~65 %, which can be explained by the fact, that data is written as ASCII characters, which only uses

one byte per character. Even though some numbers take up multiple characters, the overall number is still smaller than having double datatypes with eight bytes.
As another comparison, the MATALB mat file format is considered. Here the commonly used version 7 is compared as well as the newer version 7.3 which is however not enabled by default. v7.3 is built upon HDF5 and can therefore also be read using HDF5 tools. As can be seen from the table, v7 offers the best compression in the sense of the smallest average file size. With v7.3, the file size slightly increases which MATLAB also notes in its documentation, which can happen due to overhead in the description of the file contents.
Overall it can be seen, that the proposed format offers a good performance in terms of memory. It does not quite reach the memory efficiency of the long-matured MATLAB mat file format; however, it is not dependent on a proprietary program and can be accessed using multiple languages and programs.
The binary nature of the format, which is one of its advantages, because it allows compression, is also one of its disadvantages. The binary format leads to the restriction that the data can only be accessed using the appropriate tools and programming APIs. This also hides the structure of the data from peeks and from an easy overview of contained signals without using additional tools. This is however not seen as a drawback in the L3Pilot project, since the structure of the data is known beforehand by all partners.

CONCLUSIONS

In this paper we present the CDF approach we decided to implement to manage the heterogeneous data sources in the L3Pilot project. The intention of the format is to make the process of data exchange and evaluation more flexible and efficient. The paper showed the methodology used to define the signals needed for the evaluation in the project and presented the considerations that went into the decisions on the file format.
A preliminary test showed that the L3Pilot CDF using HDF5 is more efficient than some previously used formats, such as csv. On the other hand, it performs almost as well in terms of memory efficiency as the MATLAB proprietary format, while being independent of the software used. The portability is already by now exemplified by various tools built using the format but in different environments: Windows or Linux, and using Python, R or Matlab.
In the next months, the format will be extensively used and tested in the piloting phase of the L3Pilot project and will constantly evolve and mature, leading to a proven format that could be applied to many other projects of similar scale and type. Various analysis tools will be developed and adapted to support the format.

ACKNOWLEDGEMENT

This project has received funding from the European Commission Horizon 2020 program under the project L3Pilot, grant agreement number 723051. Responsibility for the information and views set out in this publication lies entirely with the authors. The authors would like to thank all partners within L3Pilot for their cooperation and valuable contribution.

REFERENCES

[1] "L3Pilot," 2017-2021. [Online]. Available: https://l3pilot.eu/index.php. [Accessed 20. December 2018].

[2] euroFOT, 2008-2012.

[3] S. Selpi, S. Borgen, J. Bärgman, E. Svanberg, M. Dozza, R. Nisslert, C. Norell, J. Kovaceva, D. Sanchez, M. Saez, C. Val, J. Küfen, M. Benmimoun and B. M, "D3.3 Data Management in euroFOT," euroFOT, Aachen, 2011.

[4] "AdaptIVe," 2014-2017. [Online]. Available: https://adaptive-ip.eu/. [Accessed 20. December 2018].

[5] C. Rodarius, J. Duflis, F. Fahrenkrog, C. Rösener, A. Várhelyi, R. Fernandez, L. Wang, P. Seiniger, D. Willemsen and L. V. Rooij, "Deliverable D7.1 Test and Evaluation Plan," AdaptIVe, 2015.

[6] P. Mononen and e. al., "TeleFOT Final Report, D1.15," 2012.

[7] M. Schulze, T. Mäkinen, T. Kessel, S. Metzner and H. Stoyanov, "DRIVE C2X Final Report, D11.6," 2014.

[8] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Society of Automotive Engineers, 2018.

[9] Y. Barnard, H. Chan, S. Koskinen, S. Imnamaa, H. Gellerman, E. Svanberg, A. Zlocki, C. Val, K. Quintero and D. Brizzolara, D5.4 Updated Version of the FESTA Handbook, FOT-Net Data, 2017.

[10] D. Hibberd, T. Louw, E. Aittoniemi, R. Brouwer, M. Dotzauer, F. Fahrenkrog, S. Innamaa, S. Kuisma, N. Merat, B. Metz, N. Neila, M. Penttinen, P. P. G. C. Rösener, A. Silla and A. Zerbe, "D3.1 From Research Questions to Logging Requirements," L3Pilot, 2018.

[11] Elektrobit, "EB Assist ADTF," [Online]. Available: https://www.elektrobit.com/products/automated-driving/eb-assist/adtf/. [Accessed 10 January 2019].

[12] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: an open source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[13] The MathWorks, Inc., "MATLAB MAT-File Format," Natick, MA, 1999-2018.

[14] The HDF Group, Hierarchical Data Format, version 5, 1997 - 2019.

[15] The HDF Group, "Introduction to HDF5," 08. Februar 2018. [Online]. Available: https://portal.hdfgroup.org/display/HDF5/Introduction+to+HDF5. [Accessed 18. December 2018].

[16] IEEE, "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2008*, pp. 1-70, 2008.

[17] C. Bizouard, O. Becker, J.-Y. Richard, D. Gambis, S. Lambert, T. Carlucci and P. Baudoin, "INFORMATION ON UTC - TAI," 7 January 2019. [Online]. Available: ftp://hpiers.obspm.fr/iers/bul/bulc/bulletinc.dat. [Accessed 10 January 2019].

[18] L. P. Deutsch, "DEFLATE Compressed Data Format Specification version 1.3," 1996. [Online]. Available: https://tools.ietf.org/html/rfc1951. [Accessed 18. December 2018].