Adapting LESCOT Data-Analysis Tools to PATH Data Format: A First Step for Further Collaboration

Benoît Mathern[1,2], Alain Mille[2], Thierry Bellet[1],
Steve Shladover[3], Christopher Nowakowski[3]

[1] IFSTTAR, LESCOT, France
[2] Université de Lyon, CNRS
Université Lyon 1, LIRIS, France
[3] PATH, UC Berkeley, USA

October 2010 – January 2011

Abstract

The Explo'RA Doc grant from French Région Rhône-Alpes made possible for Benoît Mathern, doctoral student at IFSTTAR to stay 4 months at PATH for embodying the collaboration between PATH and the LESCOT laboratory of IFSTTAR. The goal of this collaboration was to see how LESCOT tools and methodologies could be used on PATH data sets. We show how we adapted those tools to support the data format used at PATH. Then we'll present a first proof of concept of the possible analysis, on the example of the PATH CACC experimental data.

Contents

Introduction

The work presented on this report is the fruit of the collaboration of PATH and LESCOT laboratories during the 4 months stay at PATH of Benoît Mathern, a Ph.D. student from LESCOT.

PATH and LESCOT laboratories has already been in contact for several years. This exchange has been the occasion for a closer collaboration. The goal was to investigate how far it would be possible for PATH and LESCOT to produce a joint data analysis.

This exchange shows that it is possible for PATH and LESCOT to work together on a same data set and to share tools and methods for data analysis.

This report first presents in chapter 1 the context of this collaboration between PATH and LESCOT. Then some organisational aspects are presented in chapter 2. The technical developments making it possible to load PATH's data into LESCOT's tools are introduced in chapter 3. The analysis aspect are developed in chapter 4. Finally, the chapter 5 discusses the different aspects of this collaboration and descibes how we can prepare a longer-term collaboration.

Chapter 1

Context

We introduce the context of PATH-LESCOT collaboration. First, we present a brief history of previous collaboration. Then we explains the context of current collaboration, both from PATH and LESCOT point of view.

About PATH

California Partners for Advanced Transit and Highways (PATH) is administered by the Institute of Transportation Studies (ITS) at the University of California, Berkeley, in collaboration with California Department of Transportation (Caltrans). PATH is a multi-disciplinary program with staff, faculty, and students from universities statewide, and cooperative projects with private industry, state and local agencies, and non-profit institutions.

About LESCOT

LESCOT (Laboratoire Ergonomie et Sciences COgnitives pour les Transports)[1] is administered by IFSTTAR (Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux)[2]. IFSTTAR is the new institute resulting from the merge of INRETS (Institut National de Recherche sur les Transports et leur Sécurité)[3] and LCPC (Laboratoire Central des Ponts et Chaussées). LESCOT is located in région Rhône-Alpes, near Lyon. Its team gathers researchers and engineers and students mainly in the field of cognitive science, and a technical team developing and maintaining experimental devices.

[1]Laboratory of Ergonomics and Cognitive Science for Transportation.

[2]The French Institute of Science and Technologies for Transport, Development and Networks.

[3]The French National Institute for Transport and Safety Research

1.1 PATH–LESCOT collaboration

California PATH laboratory and French LESCOT laboratory have been developing collaboration for many years. Both laboratories develop an interest in taking into account the human factor for designing better intelligent transportation systems.

Based on experimental data collected on driving simulators or instrumented vehicles, LESCOT laboratory wants to understand the driving activity and the driver's cognition. Understanding driver's cognition is a key issue for understanding critical situations and driving errors. It is also essential to understand the driver perception of the car and its driving assistance systems in order to design better and 'ìntelligent" driving assistance systems. In this context, LESCOT has been developing tools and methods for analysing and understanding the driver's behavior.

The first collaboration between LESCOT and PATH started back from 1999 to 2002, on the question of driver modeling. Further, from 2007 to 2009, an other significant collaboration took place in the context of CalFrance project. Two workshops were held, one in California, at PATH headquarter in February 2008, one in France in April 2008. This workshops made it possible for researchers of both laboratories to meet, and more importantly to investigate the possibilities for long-term collaboration.

During CalFrance project, both laboratories agreed that long-term collaboration would be of great interest. Further collaboration should start with the question of sharing tools, data and methods. Such exchange could be achieved by collaborating on current projects of one another laboratory in the first place, and then on finding founding for common projects. The CACC[4] experiment, held at PATH was already identified as a potential project for collaborating.

1.2 The current context

From the end of CalFrance project in 2009 until the beginning of this new exchange, the global landscape of possible collaboration has not changed, but new elements had to be taken into account. On one hand, at PATH the CACC experiment has been finished. On the other hand, at LESCOT new tools has been developed. And more importantly, a source of funding made it possible for a LESCOT PhD student to collaborate with PATH during his 4 months stay in California.

1.2.1 PATH and CACC experiment

The CACC, Cooperative Adaptive Cruise Control, is a new kind of driving assistance system allowing to maintain shorter gaps while driving on highways. The CACC project was funded by the Federal Highway Administration's Exploratory Advanced Research Program and Caltrans, with technical cooperation provided by Nissan. The aim of the project was to prototype this new piece of technology. The project focused on all the aspects from the hardware development of a prototype, to testing the limitations of the system on tracks, and finally to the test of acceptability of the system on

[4]Cooperative Adaptative Cruise Control

open road with 16 drivers. It is this last aspect, the actual test with drivers from the general public, which is relevant for the collaboration with LESCOT.

The CACC acceptability experiment started in 2008. The data collection took over one year, from October 2008 to November 2009.

The funding for the CACC stopped in 2010, before LESCOT could actually collaborate with PATH on the CACC data analysis. Fortunately, after the completion of the work, Nissan provided some funding for supplementary analyses of the data. Nissan was interested in the further analysis of the data collected during the CACC acceptability experiment. This deeper analysis matches perfectly to the timing and interest of this PATH–LESCOT collaboration.

1.2.2 LESCOT and tools development

As PATH does, LESCOT laboratory also collect and analyze car-driving data. Actually, as LESCOT is a laboratory studying ergonomics and cognitive psychology, the experiments designed by LESCOT are focused on learning about human behavior and cognition. In this context LESCOT has been developing data analysis tools to support the researchers, cognitive psychologist, to understand human behavior.

The first key point for studying human behavior is to know about the context of a particular situation. That's why the first tools focused on visualizing the video, and then on synchronizing the video to a visual representation of data. A first tool has been developed for several years.

In 2008, in order to improve the software and give it new and desirable functionalities a research engineer in computer science has been hired. Based on LESCOT's previous experience, a new tool has been specified, conceived and developed. This tool, called BIND, is a framework for manipulating car-driving data. This framework has two main purposes. For an engineer, willing to process the collected data, it allows to easily access and process the data. For an analyst, that is to say a cognitive scientist, the framework allows to build interfaces for easily visualizing data synchronized with video and manual coding.

BIND framework is extendable, which means new functionalities can be added to support the engineer in processing the data, and new interfaces can be added to support the analyst to understand the data.

The second key point for studying human behavior is to describe accurately the situations, the actions and the possible mental states of the subject. For this, cognitive science uses words. Words must be selected carefully, as they have to describe rigourously the situation and what is happening. Being careful about the meaning of words is just about guaranteeing the rigour of the scientific analysis.

In this context, the ABSTRACT approach and tool have been developed. ABSTRACT is the result of a collaboration between LESCOT and LIRIS[5] laboratories. LIRIS is a computer science laboratory, with a lot of experience on data handling and analysis.

The idea of ABSTRACT is to associate a name to a certain pattern in the data. Each type of pattern can be described precisely with a unique name. Therefore the chosen name for a pattern precisely describes the underlying meaning, or concept of

[5]Laboratoire d'InfoRmatique en Image et Systèmes d'information

the pattern. Those names–we can call them concepts–are stored and organised in an ontology.

The ABSTRACT tool allows to easily associate the concepts, defined in an ontology, to the actual data set. In other words, it helps to reformulate the data, by associating a name to the patterns found in the data. Then, from this reformulation of what happened, one can either refine the definition of the existing patterns, or one can build new and more complex patterns on top of the previous ones.

BIND and ABSTRACT have been designed to be compatible. Actually, they both participate in the same chain of analysis. BIND framework allows to easily process data and achieve low level analysis[6]. Then, with BIND visualization interfaces, it is easy for an analyst to grasp what is happening, and then give a meaning to the patterns seen in the data. Last, with ABSTRACT the analyst can actually and rigorously associate the patterns with their meaning and do further analysis.

Finally, LESCOT has also been developing specific ad hoc tools for visualizing navigation or GPS data. Both BIND and ABSTRACT allow to easily visualize this kind of data with Google Earth. And the development of new tools and approaches keeps interesting LESCOT, as Benoît Mathern has been starting a PhD on this topic in 2008.

1.2.3 Benoît Mathern's PhD and work

Benoît Mathern has been working at LESCOT since 2006. First as an engineer in computer science, he conceived and developed ABSTRACT software. Later he contributed to the specification of BIND. He started his PhD work in 2008 in *Knowledge Engineering*. He studies ways to help experts, here experts about the driving activity, to discover knowledge out of a set of data. In order to achieve this, he adapts current mining techniques in order to integrate the expert's knowledge as soon as possible, in an iterative and interactive loop of data mining and data visualisation. He has developed a tool called AUTOMATA demonstrating this concept.

Benoît Mathern works on the questions of Knowledge Discovery with LIRIS laboratory, where he is supervised by the computer scientist Professor Alain Mille. His work is applied to the analysis of car-driving data at LESCOT, where he is supervised by the cognitive psychologist Doctor Thierry Bellet. His work is funded by the Region Rhône-Alpes, through the "Transport, Territories and Society" Cluster. He received another grant from Region Rhône-Alpes, called Explo'RA Doc, that allowed him to stay four months at PATH.

This last grant was the source of funding allowing us to actually realise this collaboration between PATH and LESCOT.

1.3 Goals of this collaboration

The goals for this collaboration are multiple, with different points of view. There are short-term and long-term goals for both laboratories PATH and LESCOT. At the individual level, for Benoît Mathern, the purpose is both about applying his work on an actual set of data and developing international experience.

[6]By low level, we mean, analysis based on raw data.

1.3.1 Goals for PATH and LESCOT

For PATH and LESCOT, the goal of this collaboration is to tighten the current links between both laboratories. The long-term goal, kept in mind by both laboratories, is to work together on common projects. This requires submitting common project proposals, which in turn means both laboratories have to know each other well.

This leads to the mid-term goal: getting to know the each others' work, objectives and the actual way of collaborating together.

The work described in this report support this aspects, as it focuses on the short-term objective of sharing data, tools and developing a joint analysis on a set of data.

In a nutshell, the main goal of this work for PATH and LESCOT is to start sharing data, tools, and look for ways of jointly analyzing the data.The goal is to build solid foundations for future, long-term collaboration on joint projects.

1.3.2 Goals for Benoît Mathern

This collaboration takes place as part of the PhD work of Benoît Mathern. The goal here is not to develop new concepts or techniques for analysing the data. The goal is to actually apply the tools and methods, already developed in the context of his PhD. This applied work, can be compared to what has been already done at LESCOT. This will both help to validate Benoît Mathern's work and to highlight the limits or difficulties of using the tools in another context of use.

In addition, this four month stay of Benoît Mathern at PATH is also an opportunity for him, as part of the training of a PhD work, to develop a better understanding of research throughout the world. He has the opportunity to learn about cultural differences with the US. This training will help him understanding how international collaboration works, which is a precious skill for his future career.

Chapter 2

Organisation of the Work

2.1 Meetings and Organisation

The stay of Benoît Mathern at PATH has been organised with several milestones. His stay at PATH started from the beginning of October 2010 until the end of January 2011.

Before the trip, all the actors of the collaboration interacted by emails and phone calls. This has made it possible to specify the main aspects of the collaboration.

At the very beginning of his stay, Benoît Mathern worked on solving some administrative issues. Then, during the first two weeks of he learned about PATH and about the CACC experiment. On October 15, 2010 Benoît Mathern presented LESCOT's tools and LESCOT's approach to analyse data to PATH. This presentation was entitled "Using Knowledge Engineering for modeling human activity: an approach, a set of tools and a project with PATH". Unfortunately only Steve Shladover and Christopher Nowakowski could attend to this presentation.

Then Benoît Mathern focused on understanding previous results from the CACC study and then focused on the technical aspects. He developed scripts to import and post-process the data and made sure that LESCOT's tools could be used on PATH's data.

In parallel to this technical work, regular meetings were scheduled during the whole stay with Christopher Nowakowski and Steve Shladover. Every week, Benoît Mathern also had a phone meeting (with Skype software) with Prof. Alain Mille, his Ph.D. director.

Three other relevant dates are worth to mention. On November 17, 2010, Benoît Mathern participated in the CCSIP[1] workshop. This was an opportunity for him to learn about other aspects of PATH's research and to see demonstrations of several prototypes of automated vehicles developed at PATH. It was especially the occasion to see an actual demonstration of the CACC vehicle.

On December 13, 2010, Benoît Mathern attended to the seminar presenting the merging of PATH and CCIT laboratories and the plans for the future.

[1]Canada–California Strategic Innovation Partnership

On January 13, 2011, Benoît Mathern was invited as well to a seminar presenting an innovative panoramic vision system.

2.2 Licenses and Data-Privacy Issues

The first step for investigating joint data analysis is to share data and tools. However, both sharing data and sharing tools raise issues relative to licenses and to data-privacy. In the absence of any institutional agreement between PATH and LESCOT, we had to deal with those issues and find technical workarounds.

2.2.1 Issues

Data-Privacy

We focused on analysing CACC data. CACC data set contains information about 2 weeks of driving for each of the 16 drivers who took part in the experiment. This data contains some private information, such as video records where the driver can be identified, or the GPS information, for the driver's commute trips, from his home to his work and back to her home.

Such information is sensitive and cannot be spread out. According to the privacy requirements, imposed by the University's Office for the Protection of Human Subjects (OPHS), the protocol of the experiment makes sure this sensitive data are used wisely and kept safe.

The consequence is that no sensible data can be transfered to a computer outside PATH. All the data has to be kept under PATH control. This would not have been an issue by itself without another issue: licenses.

Matlab license

PATH's tools for handling data, as well as BIND framework are based on Matlab software. Matlab software requires a license to be run. And therefore PATH tools as well as BIND needs a Matlab license to be used.

PATH didn't have any license of Matlab available for Benoît Mathern during his stay. And buying one is time consuming and cumbersome. However, LESCOT did have a licenses of Matlab available. Benoît Mathern therefore installed Matlab on the laptop computer he brought from France.

At this point, we can see the outline of the issue. Benoît Mathern had access to the tools on his personal computer and can't install them on any other computer at PATH. And he had access to the data at PATH, but can't upload them on his computer according to data-privacy issues. But we needed to have the tools and the data on the same computer to make it work.[2]

[2]As we'll see in the description of the workarounds, one simple solution could have been to access the data from the network, without uploading them on Benoît Mathern's computer, but for technical reasons, due to the quantity of data and the heavy processing needs, this solution has been rejected.

BIND license

Another solution would have been to install BIND framework on one of the PATH's computer running Matlab. Such solution, would not be optimal as it means that this computer should be shared by several users. However, an other issues came here: BIND has not been released yet by INRETS–IFSTTAR. INRETS has studied the possibility to release BIND under a free license–which would have solved this particular problem–but finally decided not to release the software yet.

Without license for BIND or intellectual property agreement between PATH and IFSTTAR, this solution could not be explored.

No Intellectual Property Agreement

Both this licenses and data-privacy issues are recurrent when several institutions need to share data and software. One of the difficulties of this collaboration is that no agreement has been previously defined between PATH and LESCOT. An agreement between IFSTTAR and the University of California, Berkeley is being written for future collaborations.

2.2.2 Technical Workarounds

CACC data were secure on PATH computer. BIND software and LESCOT's license of Matlab were secure on Benoît Mathern's laptop.

Possible solutions

One could think that the tools could be used remotely on the data, via the network, without any need to have the data and tools on the same computer. This could have been a solution if the quantity of data was not so big (tens of gigabytes of data for each driver) and if the amount of processing was not so huge (hours of processing for a single trip).

Accessing the data through the network would have strongly increased the processing time. At the same time, it would have used a lot of processing ressources on Benoît Mathern's computer, which would have make his computer hardly usable.[3]

The solution we choose allowed us to dedicate one of PATH's computers on processing the data, without need of a Matlab license or of disclosing BIND source code.

Matlab Compiler

Matlab Compiler is a product of the Matlab suite. It allows to package a program or a script based on Matlab, and produces a program that can be run without installing Matlab. This way, a program written with Matlab can be used on any computer, without any need for a Matlab license.

[3]We want to highlight here that some features of BIND framework runs only on Windows platforms, but the laptop used was an Apple Macbook running Mac OS. This issue was solved by installing Matlab and BIND in a virtual machine (via Parallels software).

LESCOT owns a license of Matlab Compiler, which is installed on a dedicated computer at IFSTTAR.

We could therefore develop and test programs using BIND on some test data. Then compile those programs with Matlab Compiler and finally actually test and run the programs on the actual data set.

Description of the workflow

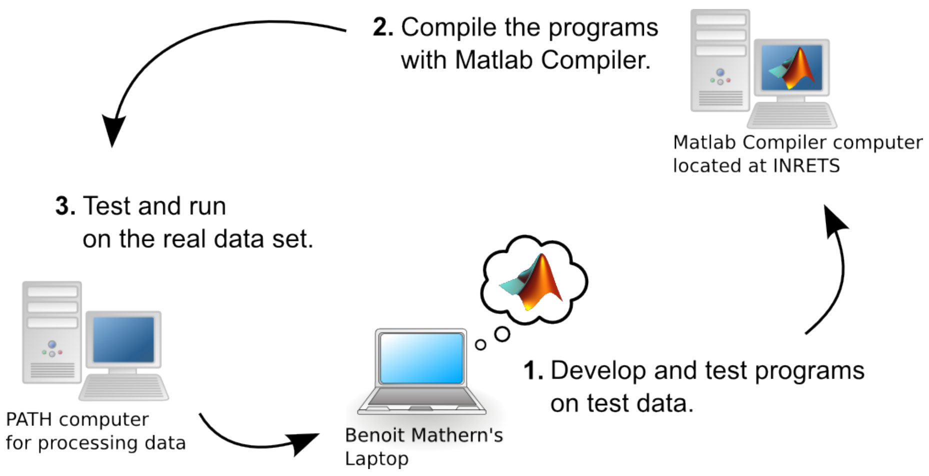

2. Compile the programs with Matlab Compiler.
Matlab Compiler computer located at INRETS
3. Test and run on the real data set.
PATH computer for processing data
Benoit Mathern's Laptop
1. Develop and test programs on test data.

Figure 2.1: The 3 steps of the workflow solving data privacy, licenses and technical constraints: **1.** develop and test programs on test data, **2.** compile the programs with Matlab Compiler, **3.** test and run on the real data set.

The actual workflow for using BIND framework on PATH's data were in 3 steps (as described in 2.1.

1. **Develop and test programs on test data.** First, we needed to develop some programs with BIND (for converting PATH's data to BIND format, or for post-processing data, etc.). For this purpose, we worked with the license of Matlab installed in Benoît Mathern's laptop computer. We could test the program on test data.
2. **Compile the programs.** Once the program was working on test data, we sent[4] the source code of the program through the Internet to the computer at LESCOT on which Matlab Compiler is installed.
3. **Test and run on the real data set.** We installed the compiled version of the script on a dedicated computer at PATH. On this computer, we uploaded as well a copy of the driver's data we wanted to analyse.

[4]We used the Subversion code-versioning system.

Then we could run the programs on the actual data set. Some problems were likely to occur (the actual data set contains a broad variety of strange situations that could not be tested with the test data). In order to solve the problems, we would come back to the script on the laptop computer running Matlab, change the program and run the whole workflow again.

Obviously, the workflow would have been much easier if we could have developed, tested and run the programs on only one computer. This solution came out to work and solve the issues of licenses and data privacy. Without these issues, there would have been only one step for doing all instead of three and we would have been more efficient.

It is good to know that such solutions exist. However, for future collaboration, we could be much more efficient with an agreement between PATH and LESCOT for sharing the tools and sharing the data.

Chapter 3

Adapting LESCOT's Tools to CACC data: Technical Aspects

The first step to share data and tools is to make sure that the data format is compatible with the tools. This chapter is about this aspect of the work, and explains what technically had to be done to load the CACC data in LESCOT's tools.

The entry point of LESCOT's tool is BIND. Therefore, CACC data have to be first converted into BIND-Trip format. This will be explained in section 3.1.

Then, because of the different way the post-processing of data is organised in PATH tools and in BIND (the first ones exports and save data in separate files, the second one save the post-processing into the same database as the raw data), the post-processing had to be reprocessed with BIND. We will also explain how to export data from BIND to ABSTRACT. All these post-processing issues will be adressed in section 3.2.

Then for the necessity of the CACC study, where multiple trips had to be compared, and considering the fact that ABSTRACT and BIND are currently more focused on single-trip analysis, we will show in section 3.3 how to use BIND and ABSTRACT in a way that allows multiple-trip analysis and trip comparisons.

Last, we will see in section 3.4 how this whole process can be generalized to future and yet unknown sets of data.

3.1 Converting to BIND-Trip Format

At PATH as well as at LESCOT, the car-driving data is based on the concept of trip. One trip is a set of data with a beginning and an end in which the driver was driving from point A to point B. One trip contains continuous or asynchronous data, videos, and meta-information about the data. What we call continuous data is a set of data that is collected throughout the whole trip, at a certain frequency. What we call asynchronous data is a set of data that is collected on an asynchronous basis, in other words, triggered by some external events. Meta-information is information about the trip itself or information about the way the data is storred. For example, it contains the name of the experiment, the id of the trip, information about the experimental conditions, as well as the organisation and the type of data that are stored in the trip.

Both PATH and LESCOT use this same high level logic to describe the data. However, differences appear on a technical level. Both PATH and LESCOT use home made tools to access the data. PATH uses Matlab scripts to load and post process the data, while LESCOT uses a framework, BIND-framework, which is also based on Matlab. A first proof of concept of data conversion from PATH format to BIND-trip format had been realised in 2009 at INRETS by Damien Sornette and Arnaud Bonnard. This proof of concept allowed us to make sure the two data format were compatible.

We will now see in detail the way data is stored at PATH and the way data is stored in BIND format. Then we'll explain how we convert the data files from PATH-trip format to BIND-trip format. Last, we'll see how to convert the video files to make them compatible with the way videos are stored in BIND-trip format.

For a description of how to use the data-conversion script, please refer to the annex A.2.

3.1.1 PATH-Trip Format

During this work, we only used CACC data. However, the other experiments done at PATH follow the same logic. We will therefore talk about PATH-format as long as we can generalise. When the format is specific to CACC data, we will refer to it as CACC-format.

Trip files are stored in a structured directory. Each trip has its own directory. And each directory is divided in a number of directories representing each a 2 minutes portion of data. These 2 minutes portions of data are stored in several 2-minutes-length data files and 2-minutes-length video files. In the CACC experiment, for each 2 minutes, 3 data files and 2 video files were created. This describes the way the data is actually stored, but not the way the data is logically accessed. Christopher Nowakowski wrote a Matlab software that take care of loading, and pre-processing a whole trip in a more convenient way.

This software (the `load_trip()` function) loads the full set of continuous and asynchronous data in Matlab. It organises this data and builds relevant meta-data in the form of a Matlab *structure*. This structure has one "meta" field, containing meta-information about the trip (name of the study, driver id, date, trip id, etc.). The other fields contain the collected data. In Matlab, structures are self describing containers of data. Therefore, the organisation of the Matlab structure reflects the organisation of the data (see Table 3.1).

There is no software provided by PATH to visualize the videos. They are just stored as 2-minute length video files.

3.1.2 BIND-Trip Format

At LESCOT, the data are stored in BIND-trips. A BIND-trip is stored as a database in a SQLite file. SQLite format allows to have a SQL database stored in a file without the difficulty of installing a database server.

Table 3.1: A description of the Matlab structure used by PATH to describe a CACC trip.

Field	**Description**
meta	meta-informations on the trip
ts	data about the timestamp
veh	data about the vehicle
gps	data about the gps
acc	data about the ACC
comm	data about the communication

Table 3.2: A description of the tables of a BIND-trip database (case of the CACC data). The tables with the prefix “Meta” are always present in a BIND-trip. The tables with the prefix “data_”, “event_” or “situation_” are specific to the experiment. Here, the raw CACC data only have “data_” tables.

Table name	**Description**
MetaDataVariables	Variables stored in the data tables
MetaDatas	Data tables
MetaEventVariables	Variables stored in the event tables
MetaEvents	Event tables
MetaParticipantDatas	Informations about the subject
MetaSituationVariables	Variables stored in the situation tables
MetaSituations	Situation tables
MetaTripDatas	Informations about the trip
MetaTripVideos	Information about the video files
data_ts	data about the timestamp
data_veh	data about the vehicle
data_gps	data about the gps
data_acc	data about the ACC
data_comm	data about the communication

BIND-trips have a generic set of tables containing the meta-information, and describing the structure of the database (tables with the prefix "Meta"). Another set of specific tables (with the prefix "data_", "event_" or "situation_"), contains the actual data collected or post-processed (see Table 3.2). The structure of these tables depends on the experiment.

In BIND-format, a strong semantic has been defined for differentiating the different types of data. Continuous data (data collected throughout the whole trip, at a certain frequency) is referred in BIND as "data". Asynchronous data (data that are collected on an asynchronous basis, triggered by some external events) are referred in BIND as "events" or "situations". An *event* occurs at a certain time stamp. *Situations* have a duration, represented by a beginning and an end.

A BIND-trip contains a link to the different video files of the experiment. The information about the offset between the video and the data is saved in the BIND-trip. BIND can handle any number of video. Logically, one view is represented by one video file for the whole trip. As a consequence, in the case of CACC data, there are two videos: one for the front view and one for the quadravision.

3.1.3 Trip-Format Comparison

Table 3.3 shows a comparison of PATH-trip format and BIND-trip format.

PATH-format uses flat files to store data, every time data needs to be loaded, all the flat files have to be read in order to build the Matlab trip structure in memory. Then the data can be processed and the results are exported in a separate flat file. PATH-format doesn't handle videos.

BIND-format uses a SQLite database to store the data. Therefore, accessing the data does not require to load all the files, you can just directly access the data you need. When data is processed, the results can be stored in the database, that is to say in the same SQLite file, which make them available for later access or reuse.

3.1.4 Converting the Data

The logical structure of the data in PATH-trips and BIND-trip is similar on most aspects. Both PATH tools and BIND use Matlab. The data conversion is quite straightforward.

We use PATH script to load the logical structure of the PATH-trip in Matlab. Then, we parse the trip structure loaded in memory to build the equivalent BIND structures:

1. Parse the "meta" field of the PATH-trip to fill the "MetaParticipantDatas" and "MetaTripDatas" tables of the BIND-trip.

2. Parse the other fields of the PATH-trip to:

    - create the corresponding "data_" tables, and fill the "MetaDatas" and "MetaDataVariables" tables;
    - fill the "data_" tables with the corresponding data of the trip.

3. Create links to the video files and calculate the offset between the data and the video.

Table 3.3: Comparison of the PATH-trip format and the BIND-trip format.

	PATH-trip	**BIND-trip**
Storage	flat files	database in a SQLite file
Querying langage	Matlab	Matlab, through BIND framework
Data access	loads everything in memory	fetch what is needed on demand
Data representation	continuous data	continuous data, referred as *data*, asynchronous data, referred as *events* or *situations*.
Data types	numeric or text	numeric or text
Videos	2-min length video files for each view	one video file for the whole trip for each view
Meta-information	defined in the “meta field”	defined in the “MetaParticipantDatas” and “MetaTripDatas” tables.
Structure	implicitly defined with the name of the field	explicitly defined (in the meta tables: “MetaDatas”, “MetaDataVariables”. . .)

The conversion script is generic. As long as the Matlab structure produced by PATH’s scripts has the same structure, the conversion should work with very little modification:

- A modification of the meta field structure needs to be taken into account in the conversion script
- The conversion script assumes that all the data are numeric (except from one field corresponding to the timecode). Therefore a modification of the script would be needed to deal with textual data.

3.1.5 Converting the Videos

The videos have to be converted as PATH stores them as 2-minutes-length files and BIND requires to have whole-trip-length videos[1]. As a result, the 2-minutes-lengths videos related to one view have to be merged together in order to build one video file.

This problem was more difficult to handle than expected: some video files would have less frames (missing images), some video files would have too many frames (images which were also recorded in the next video file), some video files would be corrupted (unable to be read), some video files would be missing.

Based on the previous proof of concept made by Arnaud Bonnard and Damien Sornette, we loaded the information about every single 2-minutes-length video file. Then, if the file had too many frames, overlapping frames were removed. Then if there

[1]It is technically possible to load as many 2-minutes-length video files as you want in BIND (with as many corresponding offset). But it would not make much sense from a user’s point of view. One video file in BIND logically refer to one view (that is to say one camera, or quadravision system).

were missing frames, additional blank frames were added. If the file was missing or corrupted, a 2-minutes-length blank video would replace it.

Then, from this temporary set of clean 2-minutes-length video files, we merge all the files in one. Last, we compress the resulting video file with a compression setting that is compatible with BIND video access.

Once the trip is converted into the BIND-trip format, we can use BIND to visualize the data, as we will explain in chapter 4. We can also use BIND, as a data-analysis framework to post-process the data and export subsets of the data for other tools.

3.2 Post-Processing Data in BIND

Once the data is available in the BIND-trip format, we use BIND as a framework for preparing and post-processing the data.

Typically, in the CACC data set, we want to know when the driver activates or deactivates the system. And when the driver deactivates the system, we want to know how the driver actually deactivated the system (by turning it off, by pressing the brake pedal, etc.). This information is not available directly in the data and we have to combine information from different fields of the BIND-trip.

Then, we want to export part of the data available to other programs. In our context, we want to make the data available for ABSTRACT.

For a description of how to use the post-processing and export script, please refer to the annex A.3.

3.2.1 Post-processing

With BIND, post-processing is meant to be stored into the BIND-trip in a new field or table. BIND, as a framework, offers methods to assist the developer to create those new fields or tables. The result of the post-processing is then easily accessible afterwards.

The structure of the post-processing script will be the same from one experiment to another: 1) create the meta-information corresponding to the new fields, 2) do the post-processing calculation. But the post-processing itself will always be something specific to an experiment (if not, it might be something that should be during the phase of importation of the data). In our case, we wanted to create new fields of data to give an easy access to relevant situations (when the driver activates, deactivates, changes the settings of the CACC).

The post-processing is computed by the function `caccCreateEventsOnTrip()`

Creating the Meta-Information We use BIND framework to create the new meta-information that are needed. For example, we create an event table named "ACC_events" with 4 fields: "event", "timestamp", "gap setting" and "speed setting". A "timecode" field will always be created as well.[2]

[2]In this example, the "timestamp" does not refer to the timecode field. It is an additional field refering to the GMT timestamp. We needed this additionnal information to compare the timecode of the data and the frame of the video (where this GMT timestamp is displayed).

From the meta-information, BIND creates the corresponding structures (tables, fields) that will contain the data.

Post-processing calculation When the structure that will contain the result of the post-processing is created, we can focus on the post-processing work. BIND offers functions to facilitate the post-processing. In our case, we create BIND-events from continuous data. The processing classes of BIND called EventDiscoverers are meant to produce events from continuous data.

We use several EventDiscoverers to extract several type of information. For example, with a threshold, we extract when the system is activated or deactivated. Then when the event is found, we know its timecode and we can extract more information to know the way the system is deactivated. We save all these events discovered from the data (basically, we create a record of the event with the "timecode" and "event" fields filled). Then we complete the additional fields ("timestamp", "gap setting" and "speed setting") with the values we get from the data. Theoretically and technically, it is not necessary to add those fields, as they can be fetched on the fly whenever needed. But it makes sense to save in an event table all the information relevant to this event. It duplicates some information, but it makes it easier to reuse the information. All the relevant information about the "ACC_events" are stored in the same table.

In our case, the post-processing was mainly about translating some data into BIND-events. In general, post-processing can also involve signal processing calculations or complex data aggregation. BIND offers facilitators for the signal-processing as well.

3.2.2 Export

BIND is meant to visualize and post-process the data collected during an experiment. For some specific data analysis, we want to export the data into another software. For example, we want to export some data for statistical analysis. In our case, we want to export data to get a further analysis using the knowledge engineering tool called ABSTRACT.

The export is computed in the function called `exportSituationsAndEvents2CSV()`. It takes several arguments to create a CSV file that can be read by many tools (including Excel or ABSTRACT). We specify what are the events or situations to export and if any additional information has to be added. In our example, we add some GPS information that can be used by ABSTRACT.

Then the CSV file is created and can be imported into ABSTRACT.

3.3 Multiple Trips with BIND and ABSTRACT

BIND and ABSTRACT focus on a single trip analysis. With the CACC data, we want to compare multiple trips together.

BIND For this purpose, we created a function in order to be able to work in a batch mode for the BIND conversion, post-processing and export processes. A first script called `mainBerkeley()` does the conversion, given a list of trips to convert. A second script, called `mainBerkeleyCaccAnalysis()` does the post-processing and export the CSV files to be used in ABSTRACT. Those scripts have been developed specifically for this work, but can be easily adapted and reused. They also contain exception handling and display a summary of the processing. This way, it is easy to know if an error occurred and what trips are affected.

Those scripts has been compiled in order to be used on computers where Matlab is not installed. A description of how to use the compiled version of those scripts is given in annex A.2 and A.3.

ABSTRACT In ABSTRACT, we reused previous scripts for importing multiple trips (called "traces" in ABSTRACT). We created specifically for this data analysis a new way to handle a set of *traces*. This ad hoc structure was then used to make some features of ABSTRACT available for a set of traces: the visualization of a trace over time, the transformation of a trace into another trace (based on rules) and the export of specific data in Google Earth.

3.4 What about Future Sets of Data?

We have made specific scripts to make sure that PATH data can be analysed with LESCOT tools. On one hand, these scripts contain some code which is specific to the data set we were working on. Typically, the post-processing and the name of the fields we export to ABSTRACT are specific to this experiment. On the other hand part of this work is generic and can be easily reused in the future.

Conversion scripts For future set of data, the conversion script can be easily reused. The function `load_trip()` provided by PATH has to be updated to match the new data set, but the conversion script would remain the same and work without any change. The only exception would be if some text data are stored in addition to numerical data. The current script does not automatically detect the type of the data and assumes all the data are numeric (with the only exception of the timestamp in GMT format).

Post-processing scripts As for the post-processing, the structure of the script can be reused, but obviously, the semantic of a post-processing is specific to an experiment. The actual post-processing has to be redefined for each experiment, but the current scripts can be used as a tutorial, to understand how to do it.

Multi-trip modifications The batch scripts are generic and can be reused without modification.

For convenience the data to be exported has been defined in the export script (in `mainBerkeleyCaccAnalysis()`), so this data definition has to be modified in this specific script.

The modification to handle multiple trips in ABSTRACT has been made generic. The only work to do to use this multiple-trips function on new data sets is to define the new set of traces. The set of traces is defined in the "traceSets.php" file as a PHP array.

We used the Unix `sed` command to automatically process the list of the trips from the triplist file.

Conclusion Basically, if the structure of the data built by the `load_trip()` function remains the same, any further set of data can be loaded with a minimum cost into LESCOT tools.

Part of the work, typically the analysis, is specific to an experiment, which means it is only relevant to keep the structure of the script, not its content.

For future data collection, it would be interesting to directly convert the data into BIND-trip format at the acquisition. It would make sense if PATH wants to use a database format to store the experimental data.

It would also be really interesting to change the way the video is collected. Video conversion is a very time-consuming process and dealing with 2-minute video files is not convenient during the analysis phase.

Chapter 4

How Can LESCOT Tools Help to Analyse CACC Data?

We have shown in the previous chapter how we made the data available in LESCOT tools (BIND and ABSTRACT). In this chapter, we will show how we can use these tools to analyse the CACC data.

LESCOT tools are designed to support an analysis of the driving activity. It means they are particularly suited to case study and on making it possible to understand the activity in its context. That's why BIND makes it possible to visualize and synchronise all the data and videos. ABSTRACT focuses more on the semantics of the activity. ABSTRACT is meant to understand the driving activity by translating low level information into more abstract information. This higher level information focuses on what is interesting to the analyst. ABSTRACT can be synchronised with BIND as well.

4.1 BIND

BIND is a framework which can be used by a developer to produce data-processing or data-analysis scripts, or to produce end-user applications for data-visualization.

We'll see how both aspects can be used to support the data analysis.

4.1.1 BIND as a data-processing framework

In the previous chapter, we have seen the technical aspects of how BIND can be used as a data-processing framework.

The goal of the analysis specifies what type of data-processing is needed on the data. Then a developer can use BIND to make scripts for data-processing. In our case, we focused on the detection of (C)ACC events. The post-processing needed to identify the different types of deactivation events has been specified by PATH, but the result of the post-processing was stored in separate files which were not easy to import. We used this specification to define the (C)ACC events with BIND.

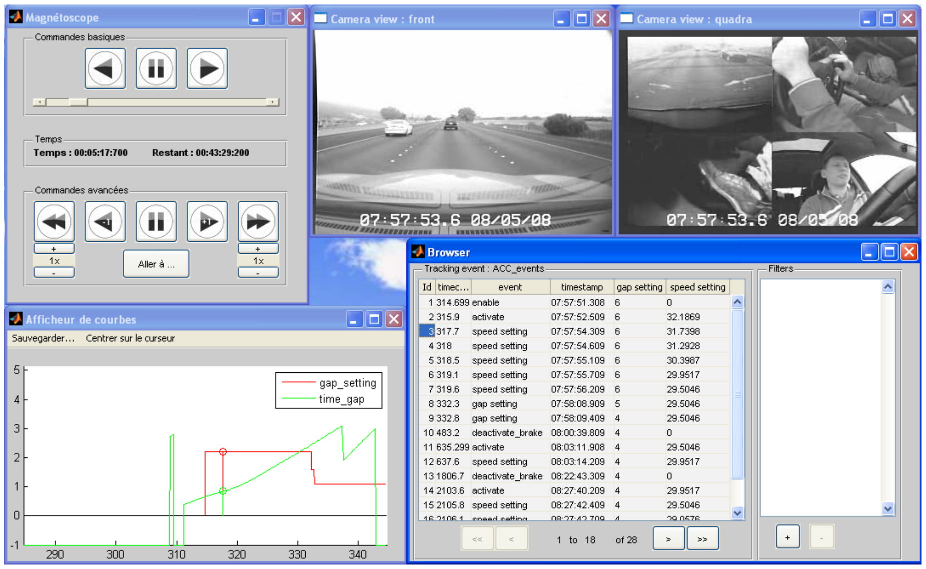
4.1.2 BIND as a data-visualization framework

BIND is also meant to support the development of data-visualization tools. In this report, we will not focus on the technical aspects of how to create such a data-visualization tool with BIND. Instead, we will only focus on one software developed with BIND, that gives an overview of what it is possible to do with BIND. We call this software BIND-*distrib* in this report.

A software like BIND-distrib is intended to be used by the analyst, that is to say a psychologist or an ergonomist trying to understand the driving activity in its context. Figure 4.1 shows the interface of BIND-distrib. BIND-distrib makes it possible to visualize the videos, a custom display of curves and events from the trip. Everything is synchronized.

This visualization is very interesting for an analyst as it makes it possible to have easy access to the curves without programming, and more important see the context of what is happening with the videos.

It is also possible to click on an event to go directly to the corresponding timecode on the videos and see the context of the event.



Temps

Temps : 00:05:17:700 Restant : 00:43:29:200

Tracking event : ACC_events

Id	timec...	event	timestamp	gap setting	speed setting
1	314.699	enable	07:57:51.308	6	0
2	315.9	activate	07:57:52.509	6	32.1869
3	317.7	speed setting	07:57:54.309	6	31.7398
4	318	speed setting	07:57:54.609	6	31.2928
5	318.5	speed setting	07:57:55.109	6	30.3987
6	319.1	speed setting	07:57:55.709	6	29.9517
7	319.6	speed setting	07:57:56.209	6	29.5046
8	332.3	gap setting	07:58:08.909	5	29.5046
9	332.8	gap setting	07:58:09.409	4	29.5046
10	483.2	deactivate_brake	08:00:39.809	4	0
11	635.299	activate	08:03:11.908	4	29.5046
12	637.6	speed setting	08:03:14.209	4	29.9517
13	1806.7	deactivate_brake	08:22:43.309	4	0
14	2103.6	activate	08:27:40.209	4	29.9517
15	2105.8	speed setting	08:27:42.409	4	29.5046
16	2106.1	speed setting	08:27:42.709	4	29.0576

<< < 1 to 18 of 28 > >>

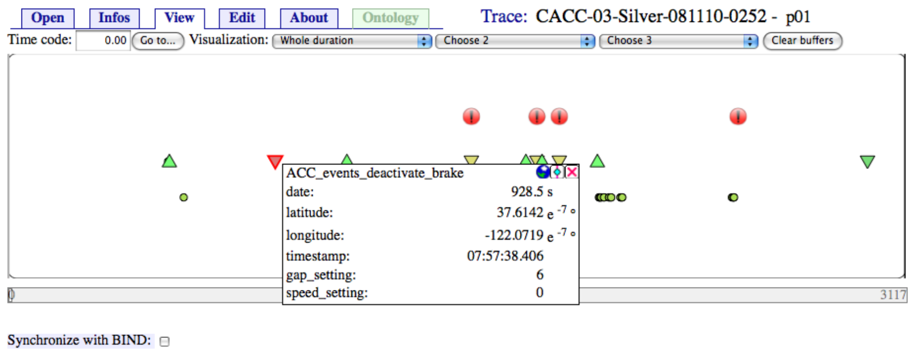
Figure 4.1: An exemple of a synchronised visualisation possible with BIND. From left to right and top to bottom: a controller, the front video, the quadravision video, a display of curves and a list of events. Everything is synchronised.

4.2 ABSTRACT

In order to go further in the understanding of the driving activity, LESCOT has developed the ABSTRACT software. ABSTRACT can display events over time, export a subset of events into Google Earth for a spatial display of the events, and translate patterns of event into other events with rules. ABSTRACT is intended to be used by the analyst (a psychologist or ergonomist), with some help from a developer for implementing transformation rules.

4.2.1 ABSTRACT visualization

ABSTRACT is a Knowledge Engineering tool. In ABSTRACT, you can display any set of events. Each event is characterised with a type, defining its semantic. The semantic is organised in an ontology. With ABSTRACT, the analyst customizes the way he or she wants to display the events, depending on their type in the ontology. This feature makes it easy for the analyst to display the information he or she needs the way that makes the most sense to him or her. This is very important as it make it possible to grasp the meaning of the data more easily. Figure 4.2 shows a screenshot of ABSTRACT. Figure 4.3 shows a screenshot of the ontology. We use Protégé software to edit the ontology.



Trace: CACC-03-Silver-081110-0252 - p01

Time code: 0.00

Visualization: Whole duration

ACC_events_deactivate_brake

date:	928.5 s
latitude:	37.6142 e -7 °
longitude:	-122.0719 e -7 °
timestamp:	07:57:38.406
gap_setting:	6
speed_setting:	0

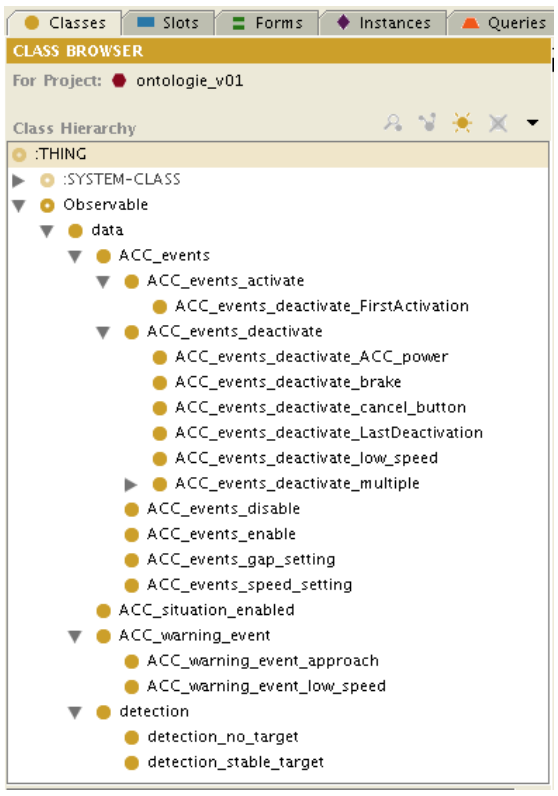
0 3117

Synchronize with BIND:

Figure 4.2: A visualization of all the events in a trip. The analyst configures how to display each type of event. When clicking on an event, ABSTRACT displays all the information available about this particular event (type, timecode and any other field related to this event). It is also possible to visualize this particular event on a map with Google Earth (as long as the GPS coordinates are available). The exclamation marks correspond to warnings. A triangle corresponds to an activation or deactivation of the system. A dot, depending on its colors, corresponds to a change in gap setting or speed setting.

We have adapted ABSTRACT software to make it possible to visualize a set of trips. For example, we can visualize all the morning commutes of driver 03 in Figure 4.4.

With ABSTRACT it is also possible to define patterns and create new events when this pattern is found. This way, it is possible to translate a sequence of event with a low



```
:THING
▶ :SYSTEM-CLASS
▼ Observable
    ▼ data
        ▼ ACC_events
            ▼ ACC_events_activate
                ACC_events_deactivate_FirstActivation
            ▼ ACC_events_deactivate
                ACC_events_deactivate_ACC_power
                ACC_events_deactivate_brake
                ACC_events_deactivate_cancel_button
                ACC_events_deactivate_LastDeactivation
                ACC_events_deactivate_low_speed
              ▶ ACC_events_deactivate_multiple
            ACC_events_disable
            ACC_events_enable
            ACC_events_gap_setting
            ACC_events_speed_setting
        ACC_situation_enabled
        ▼ ACC_warning_event
            ACC_warning_event_approach
            ACC_warning_event_low_speed
        ▼ detection
            detection_no_target
            detection_stable_target
```

Figure 4.3: Screenshot of the ontology in Protégé software.

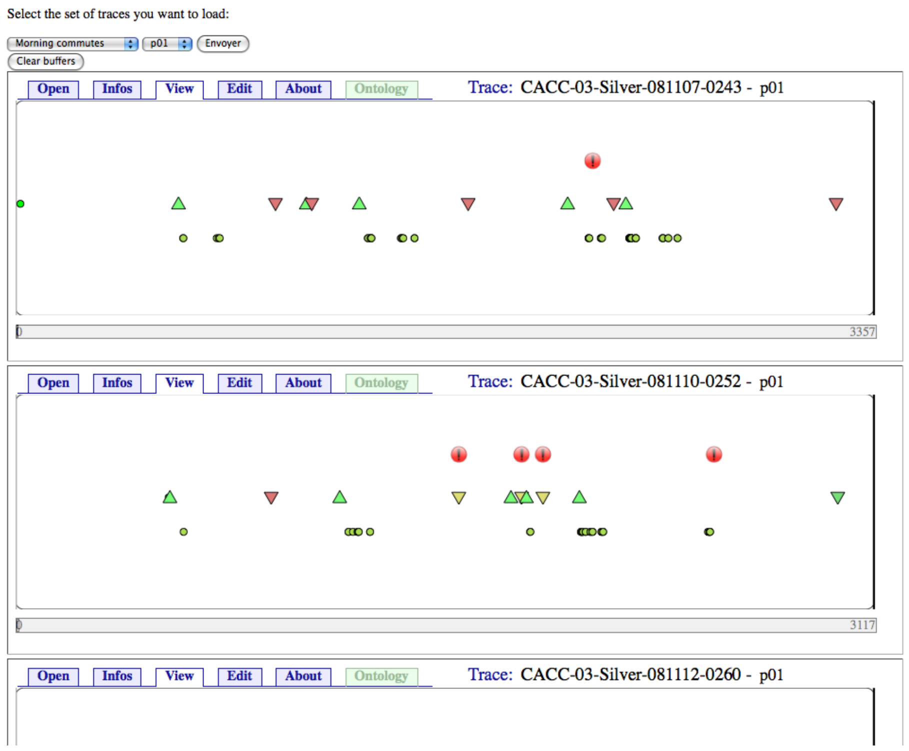

Select the set of traces you want to load:
Morning commutes
p01
Envoyer
Clear buffers
Open
Infos
View
Edit
About
Ontology
Trace: CACC-03-Silver-081107-0243 - p01
0
3357
Trace: CACC-03-Silver-081110-0252 - p01
3117
Trace: CACC-03-Silver-081112-0260 - p01

Figure 4.4: We made it possible to visualize a set of trips in parallel with ABSTRACT. Here we can see the morning commutes of the driver 03. Each timeline represents a different trip. The name of the trip is displayed on the top right corner of the timeline. On this screenshot, we can see the timelines of the "CACC-03-Silver-081107-0243" and "CACC-03-Silver-081110-0252" trips and the top of the timeline of the "CACC-03-Silver-081112-0260" trip. Each timeline displays points related to the corresponding trip. Each timeline displays the whole trip. The scale of the timeline is the percentage of time of the trip, that is to say the far left represents the beginning of the trip (0% of the time of the trip) and the far right represents the end of the trip (100% of the time of the trip). The scale is therefore relative to each trip.

```
# ACC_events_deactivate_LastDeactivation
# This transformation finds the last deactivation event of the trip

PREFIX myfn:    <java:com.ldodds.sparql.>
PREFIX kb:      <http://protege.stanford.edu/kb#>
PREFIX rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xs:      <http://www.w3.org/2001/XMLSchema#>
PREFIX fn:      <http://www.w3.org/2005/xpath-functions#>


CONSTRUCT       {

        #Add a new observe linked to n previous observes and having n new properties
        ?r1     kb:inferred     _:a     .
        _:a     a       kb:ACC_events_deactivate_LastDeactivation       ;
                kb:date ?d1     ;
                kb:latitude     ?latitude       ;
                kb:longitude    ?longitude      ;
                kb:timestamp    ?timestamp      ;
                kb:gap_setting  ?gap_setting    ;
                kb:speed_setting        ?speed_setting
}
WHERE   {

        #Obs. is not followed by an Obs of type kb:type over a certain time
        ?r1     a       kb:ACC_events_deactivate        ;
                kb:date ?d1     ;
                kb:latitude     ?latitude       ;
                kb:longitude    ?longitude      ;
                kb:timestamp    ?timestamp      ;
                kb:gap_setting  ?gap_setting    ;
                kb:speed_setting        ?speed_setting  .
        OPTIONAL {
                ?r2     a       kb:ACC_events_deactivate        ;
                        kb:date ?d2     .
                FILTER ( xs:double(?d1) < xs:double(?d2) )
        } .
        FILTER ( !bound(?r2) )
}
```

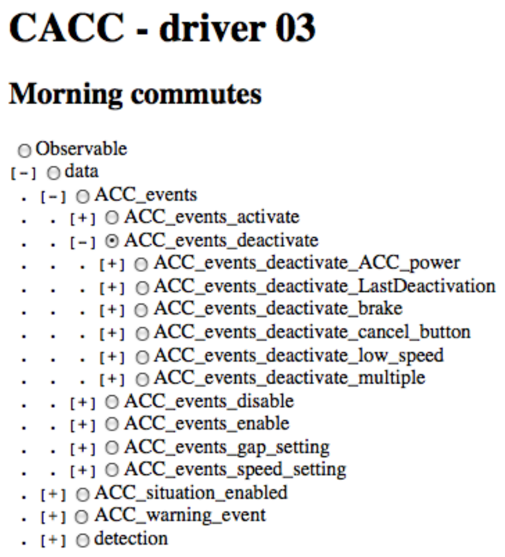
Figure 4.5: Example of a SPARQL rule used in ABSTRACT.

level semantic into a more abstract sequence of event. ABSTRACT uses the ontology to specify the semantics and keep the track of the transformations. The transformations are defined as rules. The transformation is defined in SPARQL language. Figure 4.5 shows an example of a SPARQL rule defined for the CACC experiment.

4.2.2 ABSTRACT combined with Google Earth

ABSTRACT itself focus on the visualization of events over the time. When GPS information is available, we can export the information to display the events on Google Earth. It makes it possible to visualize the events in their geographical context, which is particularly interesting to see if there is some regularity in the spatial distribution of events. For example, we want to know where the driver deactivates the (C)ACC system.

ABSTRACT offers a feature to easily export, without programming, any set of events (see figure 4.6). Indeed, you can select a type of event and export all events of this type or any of its subtypes to Google Earth.



CACC - driver 03

Morning commutes

```
 ○ Observable
[-] ○ data
 . [-] ○ ACC_events
 .  . [+] ○ ACC_events_activate
 .  . [-] ⊙ ACC_events_deactivate
 .  .  . [+] ○ ACC_events_deactivate_ACC_power
 .  .  . [+] ○ ACC_events_deactivate_LastDeactivation
 .  .  . [+] ○ ACC_events_deactivate_brake
 .  .  . [+] ○ ACC_events_deactivate_cancel_button
 .  .  . [+] ○ ACC_events_deactivate_low_speed
 .  .  . [+] ○ ACC_events_deactivate_multiple
 .  . [+] ○ ACC_events_disable
 .  . [+] ○ ACC_events_enable
 .  . [+] ○ ACC_events_gap_setting
 .  . [+] ○ ACC_events_speed_setting
 . [+] ○ ACC_situation_enabled
 . [+] ○ ACC_warning_event
 . [+] ○ detection
```

Figure 4.6: With ABSTRACT, it is easy to export all events of one type to Google Earth in order to display their geographical position. No coding skill is needed as the export is automated. The hierarchy of types is the same as in the ontology.

The user can interactively focus on more specific types of events or broader types of events. This same feature, that was initially available for a single trip has been extended for a set of trips. In figure 4.7, we can see a visualization of a set of events of type "ACC_events_deactivate_LastDeactivation" for all the morning commutes of driver 03.

4.3 A Chain of Tools as an Interactive Framework

The ABSTRACT tool was developed in collaboration between LESCOT and LIRIS laboratories. LIRIS is a laboratory in computer science with expertise in knowledge engineering aspects. ABSTRACT supports a knowledge discovery approach as described by Fayyad et al. (1996) in figure 4.8.

Actually, the whole approach of Knowledge Discovery is being adapted to the activity analysis. LESCOT and LIRIS propose to modify the classical knowledge discovery approach to make it more interactive. The goal is to integrate the knowledge of the analyst as soon as possible in the knowledge discovery process. For this purpose, the knowledge is made explicit into ABSTRACT, during the equivalent "preprocessing" and "transformation" phases. And at each step, the analyst can use his or her current

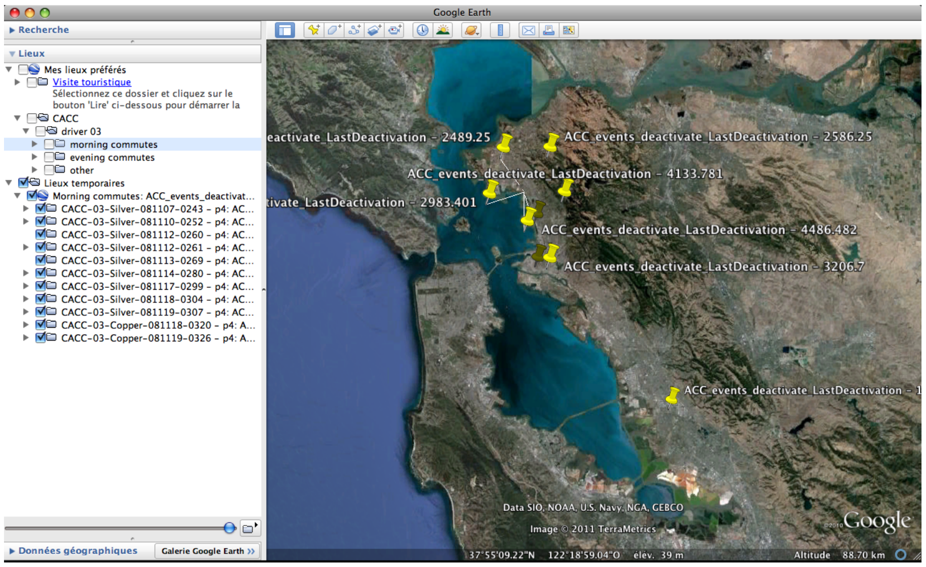

Google Earth
Recherche
Lieux
Mes lieux préférés
Visite touristique
Sélectionnez ce dossier et cliquez sur le bouton 'Lire' ci-dessous pour démarrer la
CACC
driver 03
morning commutes
evening commutes
other
Lieux temporaires
Morning commutes: ACC_events_deactivat...
CACC-03-Silver-081107-0243 - p4: AC...
CACC-03-Silver-081110-0252 - p4: AC...
CACC-03-Silver-081112-0260 - p4: AC...
CACC-03-Silver-081112-0261 - p4: AC...
CACC-03-Silver-081113-0269 - p4: AC...
CACC-03-Silver-081114-0280 - p4: AC...
CACC-03-Silver-081117-0299 - p4: AC...
CACC-03-Silver-081118-0304 - p4: AC...
CACC-03-Silver-081119-0307 - p4: AC...
CACC-03-Copper-081118-0320 - p4: A...
CACC-03-Copper-081119-0326 - p4: A...
Données géographiques
Galerie Google Earth
eactivate_LastDeactivation - 2489.25
ACC_events_deactivate_LastDeactivation - 2586.25
ACC_events_deactivate_LastDeactivation - 4133.781
tivate_LastDeactivation - 2983.401
ACC_events_deactivate_LastDeactivation - 4486.482
ACC_events_deactivate_LastDeactivation - 3206.7
ACC_events_deactivate_LastDeactivation - 1
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image © 2011 TerraMetrics
Google
37°55'09.22"N 122°18'59.04"O elev. 39 m
Altitude 88.70 km

Figure 4.7: Visualization on Google Earth of all the events of type "ACC_events_deactivate_LastDeactivation" for all the morning commutes of driver 03.

knowledge to tune the previous step. This leads to a very iterative approach with much more interaction between the analyst and the system.

ABSTRACT and BIND support this interactive cycle we are developing. BIND is meant to select and process the data, while ABSTRACT is meant to transform the data. Figure 4.9 shows how this Interactive Knowledge Discovery approach is supported by LESCOT's tools.

4.4 What about Future Sets of Data?

Each analysis is unique. The content of the analysis will always be different. However, the tools and the method we propose are suited for any analysis focusing on understanding the driving activity.

The methods described in this chapter can therefore be reused. The tools can be reused. The post-processing has to be redefined according to the new set of data. The ontology, which contains the semantics of the data analysed has to be redefined or adapted to the new goals of the analysis. New transformation rules will be created according to the needs of reformulation of the analysis.

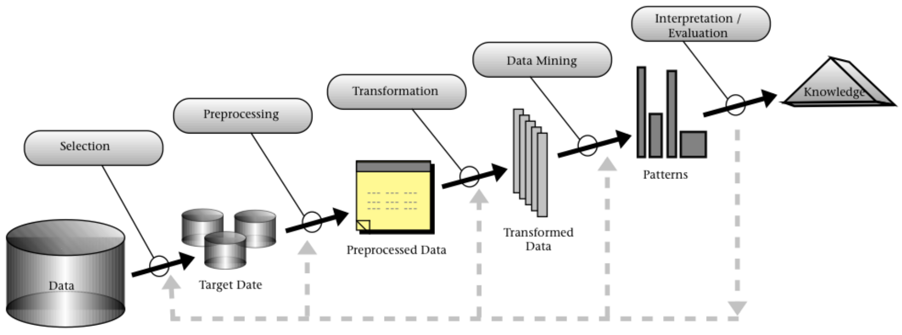


Figure 4.8: Standard Knowledge Discovery approach, from Fayyad et al. (1996). Knowledge Discovery is a succesion of several steps, from the selection, preprocessing and transformation of the data, to the data-mining, to the interpretation, by a human expert, of discovered patterns. The interpretation of these patterns builds new knowledge and helps the expert to tune the previous steps of the knowledge discovery process.

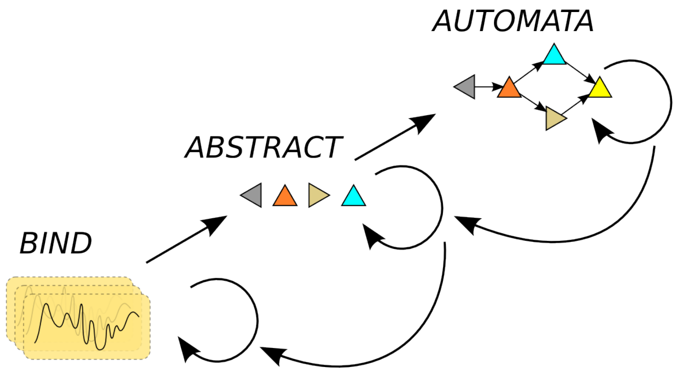


Figure 4.9: LESCOT's tools support the Interactive Knowledge Discovery approach proposed by LIRIS and LESCOT laboratories. BIND is suited for the preprocessing of the data. ABSTRACT makes it possible to transform the data and identify interesting patterns. AUTOMATA is a mining algorithm allowing to find models of the activity with an automaton formalism.

Chapter 5

Future Collaboration

The goal of this work was to tighten the current links between the PATH and LESCOT laboratories. The long-term goal is to work together on common projects. In order to support this long term objective, we have learned about each others' approaches and we have shared data, tools and ways of analyzing data.

5.1 Achievements

The realizations of this work shows LESCOT tools can be used to analyse PATH data. This work has led to:

- Develop conversion scripts to read PATH data in BIND;
- Actually convert PATH data into BIND-trip format for driver 03;
- Develop post-processing scripts in BIND to analyze CACC data;
- Actually apply this post-processing on driver 03 data;
- Visualize (C)ACC data and videos simultaneously with BIND;
- Export data for ABSTRACT from BIND;
- Develop an ontology defining a semantic for (C)ACC events;
- Actually import the trips from driver 03 in ABSTRACT;
- Visualize the (C)ACC events in ABSTRACT for the trips of driver 03;
- Define rules in ABSTRACT to identify some patterns in the (C)ACC data;
- Export interactively (with a user interface, no programming) the (C)ACC events from ABSTRACT to Google Earth.

From a more qualitative point of view, this collaboration has led to:

- Find workarounds to licences and data privacy issues to make this work possible;
- Share PATH data;
- Share the tools we use to analyse data;
- Share the methods we use to analyse data;
- Understand the differences and similarities in PATH's and LESCOT's approaches to analyse data.

This collaboration has revealed that PATH's and LESCOT's tools and approaches for analysing car-driving data are compatible. We have shown that PATH data can be analysed with LESCOT's tools, BIND and ABSTRACT. All the technical issues have been solved. The conversion scripts are generic enough to allow the future import of new PATH data into LESCOT's tools, with only minor changes.

This work has been the occasion to demonstrate from the very beginning (importing the data) to the end (analysing data into ABSTRACT) how PATH data could be analysed with LESCOT's tools.

PATH's and LESCOT's approaches have a lot in common. We post-process the data. We visualize the data. We iteratively modify the post-processing and the visualization to produce a deeper analysis. We visualize data on a map to understand their relation to the geographical context.

However, on one hand PATH goes deeper in the comparison of several trips. LESCOT's tools focus more on a single-trip analysis, while PATH's tools focus more a multi-trip analysis. On the other hand, LESCOT tools are more interactive and help an analyst to visualize the context of the data (with video or interactive visualization tools). LESCOT's tools are meant to explicitly describe the knowledge discovered in the data by an analyst (typically with the ontology and the SPARQL rules in ABSTRACT), while PATH's scripts implicitly contain this knowledge in the source code of the scripts. LESCOT's and PATH's tools both need the intervention of a developer. However, PATH's tools are entirely developed in Matlab, with very little user interfaces, which makes it difficult to use for people without an engineering background. LESCOT's tools on the other hand have much more user interfaces, in order to make the tools accessible to psychologists or ergonomists. With LESCOT's tools, a developer should work with the analyst to translate specific needs in post-processing or pattern identification into actual code.

In a nutshell, PATH's and LESCOT's approaches are complementary. PATH's scripts are designed for human factor engineers while LESCOT's tools are designed for psychologists and ergonomists.

5.2 Conclusions for future collaboration

First, for future collaboration, it will be absolutely necessary to solve the data-privacy and licensing issues. A common agreement between PATH and LESCOT has to be defined, making sure to clarify what data can be shared, what tools can be shared, and what use of the tools and of the data is acceptable.

Without any frame, we had to find technical workarounds in order to make sure LESCOT's tools remained on LESCOT's computers and PATH's data remained on PATH's computers, which made the collaboration less efficient.

Now that we have shown that we can share data and tools, it would be interesting to start a joint analysis of the same set of data. PATH would produce a human factor analysis while LESCOT would produce an activity analysis in cognitive psychology. This joint analysis would highlight the complementarity of both approaches.

Even more interesting, we could merge both analyses. For example, PATH could use LESCOT's results in the activity analysis to refine their human factor analysis. Or LESCOT could use PATH's results in the human factor analysis to get a global view of the situations that are worth studying.

A nice way to make this future collaboration effective would be to submit a joint project. The distance, the different sources of funding from one country to another and the data-privacy issues make such a collaboration difficult. But it is still possible and worthwhile.

In a short term period, PATH and LESCOT should write a common publication to make this collaboration visible to the scientific community.

This collaboration should not be taken for granted. There is a risk for it to fade away, for a human ressource reason. At PATH, the human factor or human activity analysis is a minor aspect of the laboratory research. At the moment Steve Shladover and Christopher Nowakowski are the two only following this collaboration. At LESCOT, several tools and methods for the data analysis are developed and maintained by non-permanent people. Their expertise might not be available at LESCOT in several years.

In order to prepare the future, PATH and LESCOT should maintain the contact, start preparing a legal agreement and write a joint publication. And then, when the opportunity of proposing a joint project will occur, both PATH and LESCOT will be ready to work together.

Conclusion

This work has been the occasion for PATH and LESCOT to share data and tools and to see how PATH's CACC data could actually be analysed using LESCOT's tools.

Some data-privacy and licensing issues have been raised but were solved with technical workarounds. Then, the technical issues of data-conversion and data-processing issues were solved. Data could be imported and processed in LESCOT's tools, BIND and ABSTRACT. The concept of the data analysis with BIND and ABSTRACT has been demonstrated. This made it possible to highlight the similarities and the differences between PATH and LESCOT approaches to analyse the data.

PATH approach focus on the human factor analysis while LESCOT approach focus on the activity analysis, from a cognitive science standpoint. Both approaches are complementary and this collaboration shows us how interesting it would be to go further.

In the future, PATH and LESCOT should maintain the contact and prepare a legal agreement between both laboratories. In parallel to this background work, the next step would be to propose a joint data-analysis project.

Acknowledgments.

This work was supported by Région Rhône-Alpes through the Explo'RA Doc grant and the "Transport, Territories and Society" Cluster.

Appendix A

Tutorials

A.1 “dirs.mat” file format

The “dirs.mat” file describes a list of trips. This file is used as a (default) parameter in two scripts that was developed for PATH: the data conversion script from PATH-trip to BIND-trip and the post-processing script. The “dirs.mat” file stores a Matlab variable containing a list of input and output directories, this way scripts can be run on batch mode on this selection of directories.

The “dirs.mat” file stores the `dirs` Matlab variable. The `dirs` variable is a $2 \times n$ cell-array of input and output directories (absolute path to the directories).

A.1.1 How to create the “dirs.mat” file?

Simple example of a `dirs` variable with only one trip:

```
% initialize the dirs variable
dirs = cell(2,1);
% input directory
dirs{1,1} = 'C:\CACC\Driver03\Silver\Date081106\Trip0241\';
% output directory
dirs{2,1} = 'C:\CACC\Driver03\Silver\Date081106\Trip0241\';
% save to the dirs.mat file
save('dirs','dirs');
```

Note that the output directory can be the same as the input directory, which means you’ll save the output of the script in the same directory.

You can generalise the first example to any number of trips:

```
% initialize the dirs variable
dirs = cell(2,number_of_trips);
% for each trip...
    % input directory #i
    dirs{1,i} = path_to_input_direcory[i];
    % output directory #i
    dirs{2,i} = path_to_output_direcory[i];
% save to the dirs.mat file
save('dirs','dirs');
```

A.2 Tutorial: How to convert PATH trips to BIND-trips?

A.2.1 Prerequisite

1. Install the Matlab Compiler Runtime (execute the “MCRInstaller.exe” file).
2. Copy the “Cacc2SQLiteTrip.exe” file on your hard drive. The log files generated by the execution of the script will be stored in the same directory as this executable file.
3. Install video codecs for reading the original video files and compressing the output video files (DivX codec for the conversion in BIND format). For an easy setup, you can install Windows Essentials Media Codec Pack and the DivX compression codec.
4. Install Virtual Dub on the folder “`C:\`”. The “Cacc2SQLiteTrip.exe” script will expect to find the program at the path: “`C:\vdub\vdub.exe`”.[1]

A.2.2 Execution of the script

Run the “Cacc2SQLiteTrip.exe” file. If a “dirs.mat” file is found in the working directory, this file will be used as a parameter to run the script in batch mode on all the trips in the `dirs` variable. Otherwise, a browser will appear to select the path to one trip to convert (in this case, the output directory, where the BIND-trip file will be stored is the trip directory).

All the defined trips will be processed. A “Cacc2SQLiteTrip.log” file will be created, reporting what happened during the whole conversion process.

A.2.3 Actions of the script

For each input directory (corresponding to a PATH-trip), this script:

- loads the data, converts it into a BIND-trip and save the resulting “.trip” file in the corresponding output directory.
- loads the 2 minutes-length video files, merges them and compress the result to produce two video file for the whole trip: one for the front view, one for the quadravision. The two “.avi” files are saved in the corresponding output directory.

Note that the naming of the “.trip” and “.avi” files follows this convention: “CACC-dd-Vehicle-YYMMDD-TTTT”, where “dd” is the number of the driver, “Vehicle” is the name of the vehicle, either “Copper” or “Silver”, “YYMMDD” is the date of the trip with two digits for the year followed by two digits for the month and two digits for the day, and “TTTT” is the number of the trip. For example, for the trip 320 of driver 03, the files would be named: “CACC-03-Copper-081118-0320.trip” for the trip

[1]The path to Virtual Dub is specified in hard in the Matlab function `berkeleyVideoConversion()` (line 4).

file, “CACC-03-Copper-081118-0320-front.avi” for the front video and “CACC-03-Copper-081118-0320-quadra.avi” for the quadravision video.

A.2.4 Log file

The “Cacc2SQLiteTrip.log” file will show information about the sequence of events of the execution of the script. The log file will be saved in the same directory as the “Cacc2SQLiteTrip.exe” file[2]. At the end of the log, two summaries are presented. A detailed summary shows significant information about each single trip conversion (execution time and if errors occurred). A global summary shows the global conversion time, the number of trips converted and if errors have occurred. This way, with a quick glance at the summary, you can get relevant information about what happened during the execution of the script.

A.2.5 Execution time

On the CACC experiment, for the 36 trips of the driver 03, the execution time of the conversion script took about 6.25 days[3]. This corresponds to an average of 4.2 hours per trip. About half of this time is used for the data conversion from PATH-trip to BIND-trip format and half for the video conversion.

Note that we encountered once a problem with virtualdub program during the conversion. A pop-up of virtualdub waiting for the user to intervene had blocked the conversion process for about 80 hours (during a week-end). It is recommended that somebody check regularly the conversion process. Errors in Matlab would not stop the conversion process, but errors outside Matlab (that is to say in virtualdub) can block the conversion.

A.2.6 Reusability

This script can be reused quite easily. On a new data set, or if PATH scripts for loading data changes, an update of the script should be produced (just updating with new PATH-scripts).
The script is generic and take advantage of the structure format of PATH-trips to make the conversion. There will only be an issue if the type of some data loaded is not numeric (in the CACC data, only one field was not numeric–one of the timestamp fields–, this issue was solved specifically for this particular field). The solution would be to make the conversion function (`berkeleyFolder2sqlite()`) more generic so that it can directly handle different data-types.
As for the video conversion scripts, they are specific to the way video is handled in the CACC experiment. Some coding would be necessary to adapt this part of the script to another set of videos or an other slicing principle (functions `berkeleyVideoConversion()`, `genereScriptAppend()`, `genereScriptStrip()` and `videoList()`).

[2]Any older log file will be overwritten. If you want to keep a track of what happened in your previous conversion, you should make sure that you backup the log files.

[3]The conversion time from the log was about 9.5 days (228 hours and 24 minutes), but it was due to a blocking pop-up windows which occurred during a week-end, blocking the process for about 80 hours.

A.3 Tutorial: How to post-process BIND-trips and export files to ABSTRACT?

A.3.1 Prerequisite

1. Install the Matlab Compiler Runtime (execute the "MCRInstaller.exe" file).
2. Copy the "CaccAnalysis.exe" file on your hard drive. The log files generated by the execution of the script will be stored in the same directory as this executable file.

A.3.2 Execution of the script

Run the "CaccAnalysis.exe" file. A "dirs.mat" file has to be present in the working directory, this file is used as a parameter to run the script on all the trips in the `dirs` variable.

All the defined trips will be processed. A "CaccAnalysis.log" file will be created, reporting what happened during the whole conversion process.

A.3.3 Actions of the script

For each output directory of the "dirs.mat" file (corresponding to a BIND-trip), this script:

- processes data in the BIND-trip to create new events and situations,
- processes a specific set of events and situation and exports them in a CSV file (in the same output directory as the ".trip" file) that can be imported in ABSTRACT software.

Note that the script will look for trip files respecting the naming convention used by the "Cacc2SQLiteTrip.exe" script. The naming of the output CSV file will be the name of the trip file extended with the string "4ABSTRACT.csv". For example, for the trip 320 of driver 03, the CSV file would be named "CACC-03-Copper-081118-0320.trip4ABSTRACT.csv".

A.3.4 Log file

The "CaccAnalysis.log" file will show information about the sequence of events of the execution of the script. The log file will be saved in the same directory as the "CaccAnalysis.exe" file[4]. At the end of the log, a global summary is presented. It shows the global post-processing time and if errors have occurred. This way, with a quick glance at the summary, you can get relevant information about what happened during the execution of the script.

[4]Any older log file will be overwritten. If you want to keep a track of what happened in your previous conversion, you should make sure that you backup the log files.

A.3.5 Execution time

On the CACC experiment, for the 36 trips of the driver 03, the execution time of the post-processing script took about 2.5 hours (148min). This corresponds to an average of 4.11 minutes per trip.

A.3.6 Reusability

This script focus on specific analysis. Therefore, the structure of the script can be kept for some other analysis, but it does not make any sense to keep the specific calculation relative to CACC data. The specific calculation is being made in the `caccCreateEventsOnTrip()` function. As this function computes specific information relevant to CACC data, much of this function have to be rewritten. The other functions are generic and can be reused.

A.4 Tutorial: How to use BIND-visualization plugins?

A.4.1 Prerequisite

1. Install the Matlab Compiler Runtime (execute the "MCRInstaller.exe" file).
2. Copy the "distribution.exe" file on your hard drive. The log files generated by the execution of the script will be stored in the same directory as this executable file.
3. Install video codecs for reading the original video files. For an easy setup, you can install Windows Essentials Media Codec Pack and the DivX codec.

A.4.2 Execution of the program

Run the "distribution.exe" file. This will load a user interface in which you can load BIND-trips, select visualization plugins. Then, you can visualize the data and the videos synchronously and have an easy control to move in the video/data.