HAVEit

Highly automated vehicles for intelligent transport

7th Framework programme
ICT-2007.6.1
ICT for intelligent vehicles and mobility services
Grant agreement no.: 212154

The future of driving.

Deliverable D31.1 Co-driver command vector available (1^{st} version)

Version number	Version 1.0
Dissemination level	PU
Lead contractor	Continental Automotive GmbH
Due date	30.04.2009
Date of preparation	03.05.2009

Authors

Dr. Nashashibi, Fawzi

Dr. Benenson, Rodrigo

Mr. Lopez-Resende, Paulo

Dr. Glaser, Sébastien

Project Manager

Prof. Alfred Hoess

Project Co-ordinator

Dr. Reiner Hoeger

Continental Automotive GmbH, STA EG

Siemensstrasse 12

93055 Regensburg

Germany

Phone	+49 941 790 3673
Fax	+49 941 790 13 3673
e-mail	reiner.hoeger@continental-corporation.com

Table of contents

Table of contents iii
List of Figures iv
Executive summary 1
1 Introduction 2
2 Expected inputs/outputs of the co-pilot system 4
3 The co-pilot’s architecture 7
3.1 Coarse planning - the strategy level 7
3.1.1 LCPC Algorithm-1 9
3.1.2 LCPC Algorithm-2 9
3.1.3 DLR algorithm 11
3.1.4 Fusion of the LCPC1-LCPC2-DLR output 11
3.2 Fine planning - the motion planner (PMP) 11
4 Results 15
4.1 Implementation and integration 15
4.2 Example outputs 15
5 Conclusion 18
References 19
Annex 1 Abbreviations 20

List of Figures

Figure 1: HAVEit system architecture 3
Figure 2: Coarse planning (manoeuvre level); for instance, next manoeuvre to execute is to "change lane to the left and accelerate" 7
Figure 3: Strategy decision grid (a) and strategy manoeuvre tree (b) 8
Figure 4: The nine possible manoeuvres, a cost value is assigned to each cell (manoeuvre). 10
Figure 5: DLR's decisional tree 11
Figure 6: According to the given driving situation, the best manoeuvre is decided accounting the trajectories rating using the grid and tree Valentials. The appropriate refined trajectory is then planned (here, overtaking and acceleration on the left) 12
Figure 7: Construction of the sequence of states in time 13
Figure 8: Green square (ego-vehicle), blue squares (vehicles-obstacles), coloured dots are the successive trajectory positions from red (zero speed) to green (high speed), orange is medium speed. Three scenarios: change left and accelerate (a), change left and hold (b), change left and stop (c) 14
Figure 9: Closed loop simulation of a HAVEit scenario using SMPLab/SILAB tool 15
Figure 10: Co-pilot planned trajectories for 4 different scenarios (use-cases) 17
Figure 11: Visualization tool developed to illustrate HAVEit scenarios using geo-localized data 17

Executive summary

The overall objective of the HAVEit project is to develop technical systems and solutions that improve automotive safety and efficiency using an adapted automated driving on roads. INRIA, LCPC and DLR contribute to the overall objective by developing the Co-pilot system in collaboration with the following partners: ICCS as data fusion provider and IBEO as a sensor provider.

The Co-Pilot (WP3100 in the HAVEit project structure) is fundamentally intended to support the driver by identifying the current driving situation and providing a recommendation of the action to be done next. The coming action is a manoeuvre that has to be executed by the driver or by the vehicle controllers in a highly automated mode. There is also an evaluation whether the present situation can be mastered by the technical system or not.

Hence, the Co-pilot is a piece of software that integrates several algorithms computing the safety envelope and the coming motion vectors.

This document summarizes the co-pilot's architecture of the WP3100, its functioning and its operational scheme and interactions with the other sub-systems of SP3000. A technical description of the co-pilot system is also included as well as the algorithms used to generate safe and optimized trajectories.

The selected trajectory is then used by the *CommandAndValidate* sub-system (WP3300) in order to generate the command vector used by the vehicle controller to realize the feasible trajectory.

1 Introduction

The intention of this report is to describe the co-pilot system that generates the command vector that has to be used by the vehicle controllers and actuators in order to execute a computed path.

General objective of the Co-pilot:

- From the results of the multi-sensor based fusion module, the co-pilot algorithms will identify the type of situation the vehicle is in and generate the best drive vector to handle the situation. If an emergency situation occurs, this vector will have a risk value attached to it.
- In practice, the system will determine the optimal strategy to achieve the current driving objective accounting the driver state and the driving context.
- Strategy determination is based upon the analysis of the current driving situation (based on both environment sensor information and future estimation). The assessment of the danger level of the current driving situation may lead the system to select a contingency strategy for the sake of safety. "Strategy" means a set of feasible optimal manoeuvres and trajectories to be realized by the vehicle or the driver. Those are described in terms of vector commands and a geometric geo-positioning in space.
- The optimality of the trajectories should be understood in terms of:
  - trajectories lengths
  - computation costs
  - optimal driving comfort
  - fuel consumption
  - safety margins
  - …

The co-pilot will always generate a path. The trajectory will be effectively executed according to the Mode Selection and Arbitration Unit (MSU) decision and through the driver interface (Figure 1).

These algorithms will be evaluated on the joint system demonstrator (FASCar) developed by DLR and referenced in HAVEit as the WP4100.

In the following section (section 2) we will provide a list of expected and implemented inputs / outputs of our co-pilot system.

In section 3 of this document the overall system architecture is described as well as each of the co-pilot sub-systems in a separate subsection. In section 4 we will present the results obtained following the integration of the system to the joint development and simulation platform “SMPLab” developed by DLR. This document ends with the conclusion in section 5.

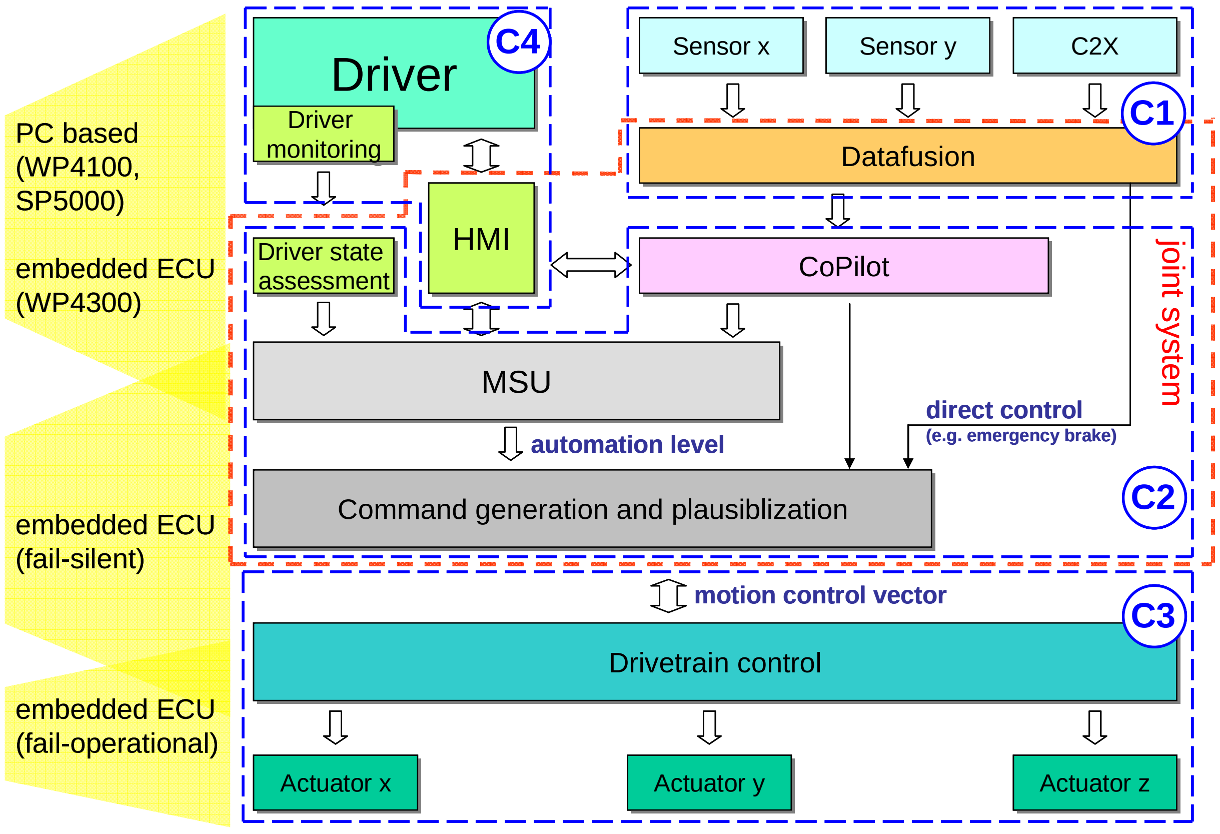


Figure 1: HAVEit system architecture

2 Expected inputs/outputs of the co-pilot system

The Co-pilot system needs sensors for the environment perception (vehicles, pedestrians, lane markings, traffic signs, obstacles, etc.) as well as for the driver state assessment (drowsiness, attentiveness, etc.). These sensors will be mounted in or outside of the vehicle.

We will not describe the sensors or the data fusion system developed in HAVEit since they are subject of respective corresponding deliverables (for descriptions of the perception system, please refer to deliverables D11.2, D51.1, D52.1, D53.1 and D54.1).

The following table shows the list of interactors involved with the co-pilot system[1].

Input	1. **Data coming from the data fusion system** Involved interactors: I11 / I12 • I11: Ego Vehicle information o Timestamp o Kinematic state information; Variance of the estimated values o Position relative to the point the trajectory was calculated. Geodetic position o Lane assignment o Standstill detection o Sensors that provided the data for extracting the information. • I12: Perception Model o Objects ▪ Track information: ID and track lifetime ▪ Position and kinematic state estimation; Variance of the estimated values ▪ Object classification and dimensions: Width, height, objects type ▪ Obstacle probability ▪ Lane assignment ▪ Number of detected obstacles ▪ Sensors that provided the data for extracting the information for each object. o Lanes ▪ Position: Lateral and longitudinal position relative to the ego vehicle ▪ Curvature and curvature rate ▪ Width ▪ Lane type (Soft, Emergency, Hard etc.) ▪ Left and right lane marking type ▪ Observed as traversable; Observer from map ▪ Lane reliability o Road information ▪ Road type (unknown, highway, rural, urban) ▪ Road type additional (none, ramp, additional map-based information)

[1] For more detailed description of the interactors please refer to the architecture deliverable D12.1.

	▪ Gradient ▪ Warning signs ▪ Sensors that provided the data for extracting the information for each object. o Intersection information ▪ Intersection type ▪ Distance to intersection ▪ Traffic signal state ▪ Sensors that provided the data for extracting the information for each object. o Trip information ▪ Time ▪ Trip duration 2. **Data from the Driver Interface system** Involved Interactors: I10: Driver's primary task command o Position of the acceleration pedal o Steering wheel angle o Steering wheel rate o Steering wheel torque o Vehicle speed o Yaw rate o Longitudinal acceleration o Lateral acceleration o Lateral speed
Output	**A list of ranked geometric trajectories:** Each trajectory is a set of geo-referenced and locally referenced 2D positions of the vehicle. It is also described by a set of basic controls (linear speed, steering or yaw angle). Involved Interactors : I7 / I8 / I20 / I 22 / I 23 / I30 • I7: Relevant targets and key environment information o Application specific • I8: Ranking of trajectories o ranking of each trajectory o cost (and sub costs) associated to each trajectory • I20: Vehicle positioning, current executed trajectory and trajectory limits o current local position : lateral position o current local position : heading control o current trajectory o lateral boundary of the trajectory • I22: Limits of the free space • I23: Limits of the health horizon • I30: Set of ranked trajectories with descriptions o maneuver description o longitudinal position in the lane

	o lateral position with respect to the lane o time position o length of the trajectories o ranking of each trajectory o cost (and sub-costs) o lateral position of the lane o longitudinal position of the lane o desired velocity o desired lateral position in the lane

3 The co-pilot's architecture

The co-pilot is an advanced path planner that aims at constantly elaborating an "optimal" trajectory for the vehicle. Optimality is to be understood as a compromise that takes into account several needs and constraints: comfort, fuel consumption, respect of driving rules, safety, short path, fast path, etc. The main output of the co-pilot sub-system (C2.1 in Figure 1) is a trajectory that is described in terms of a set of points and positions, endowed with the intended vehicle dynamics.

In order to achieve a strong cooperation with the driver, irrespective of the automation level, the co-pilot process is achieved using two main functionalities:

- *The definition of a driving strategy* at a high level, which is described using a manoeuvre language. Three different approaches are evaluated and then fused in order to give a more reliable result.
- *The definition of trajectory* at a lower level: this function uses the previously selected manoeuvre to define a reduced possibility field of the trajectory.

3.1 Coarse planning - the strategy level

The *definition of a driving strategy at a high level* sub module is based on fast and simple algorithms that evaluate the possibility of performing several predefined manoeuvres. The aim of this high level is to quickly eliminate a part of the search space, thus reducing the calculation time of the *definition of trajectory* at low level. It will also allow a high level communication towards the driver, in the form of a *manoeuvre grid* or a *manoeuvre tree.*

Nine manoeuvre cases can be identified by combining following basic actions: three in longitudinal direction (accelerating, decelerating and staying at constant speed) and three in lateral direction (staying in the same lane, going to the left and to the right). Some examples of these manoeuvres are: "stay in the same lane, decelerate", "change to the left lane, accelerate (see Figure 2)", etc.

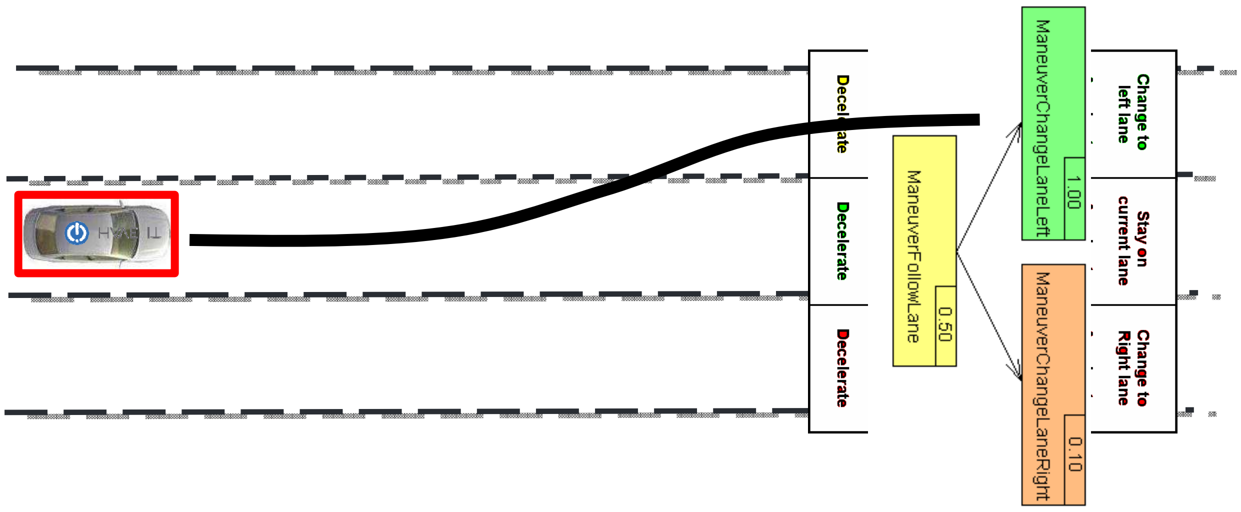


Figure 2: Coarse planning (manoeuvre level); for instance, next manoeuvre to execute is to "change lane to the left and accelerate".

To these basic manoeuvres, an Emergency Brake manoeuvre is added, corresponding to a full brake till standstill, and a Safe State manoeuvre, corresponding to a user- or system-initiated slow-down on a dedicated lane. This gives a total of eleven manoeuvres.

Each manoeuvre gets a performance indication which is called *Valential.* What is meant by performance is a global qualitative and quantitative evaluation of the manoeuvre accounting given criteria. The Valential is used to directly discard certain areas in the solution space and to give a clear overview of the situation in the Human Machine Interface.

There are two ways to represent these manoeuvres: the *grid* which is a 3x3 matrix, giving 9 cases for the 9 basic manoeuvres plus 1 case for the emergency manoeuvre and 1 for the safe state manoeuvre. Each case is coloured according to its Valential (red to orange to yellow to green for increasing Valentials). The *tree* representation gives the current situation, and it visualizes the possible actions with their Valential as the branches of this tree. The same color code is used.

Both representations are fully equivalent and are both delivered to the client modules.

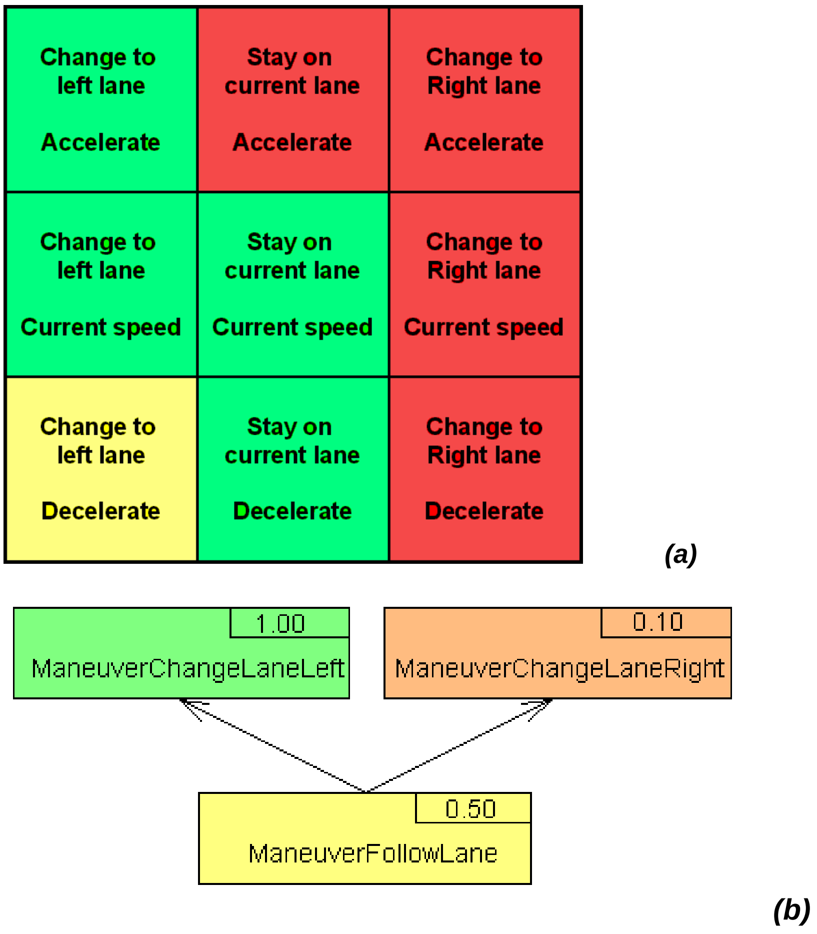

Change to left lane Accelerate
Stay on current lane Accelerate
Change to Right lane Accelerate
Change to left lane Current speed
Stay on current lane Current speed
Change to Right lane Current speed
Change to left lane Decelerate
Stay on current lane Decelerate
Change to Right lane Decelerate
(a)
1.00
ManeuverChangeLaneLeft
0.10
ManeuverChangeLaneRight
0.50
ManeuverFollowLane
(b)

Figure 3: Strategy decision grid (a) and strategy manoeuvre tree (b)

For the strategy level two algorithms are delivered by LCPC: one is based on a risk concept and the other is based on a multiple performance indicator concept. A third algorithm which is based on fuzzy rules is delivered by DLR. The three algorithms run in parallel leading to robustness and redundancy in this safety relevant module. Their three Valentials are fused to give one Valential for each manoeuvre to be passed to the *"definition of trajectory"* sub-module and other client modules.

3.1.1 LCPC Algorithm-1

A first manoeuvre evaluation algorithm is created by LCPC. It is based on the evaluation of risk in a vehicle lane. It evaluates the risk related to the speed and the relative distance to each vehicle in the lane. The method is extended on adjacent lanes by using a virtual vehicle set at the same curvilinear position but on the adjacent lane. A common approach in risk theory is to define the risk related to an event using two criteria: the probability that the event occurs and the gravity of the events occurring. These two concepts are briefly discussed.

In our manoeuvre selection, we assume the worst event is a collision. We calculate the gravity of the possible collision, using the Equivalent Energetic Speed (now EES). EES corresponds to the deformation energy of a damaged vehicle during a collision given their respective speed and mass. It is directly linked with the damage done to the human in the vehicle. After calculating the EES, the value of the gravity is directly found by a look-up table, which is constant for a given application.

The probability is deduced from the analysis of indicators as inter-vehicular-time, time-to-collision and reaction distance. A first parameter is the Time-To-Collision (TTC). Hayward [D12.1] defined TTC as: *The time required for two vehicles to collide if they continue at their present speed and on the same path*. We use the two extreme values to determine a probability of collision of 1 (for a TTC of 10*s*) and 0 (for a TTC below 1*s*). Between these values the probability is linear with respect to the TTC. The TTC itself is not sufficient to describe the risk related to the situation: for instance, when two vehicles are close to each other, with the same speed, the TTC is large, but if the inter-vehicular distance is small, the situation could be dangerous. In order to take into account this kind of situation, we enhance this definition with the consideration of the inter-vehicle time. The inter-vehicle time and probability of collision are linked by a look-up table which is tuned for the application.

In the two previous sections, we have described the computation of the gravity index and of two probabilities. The risk of a certain manoeuvre is found as the sum of the products of the gravity index and the respective probabilities.

The risk that is calculated in this algorithm gives a performance indication or Valential on each of the eleven manoeuvres.

3.1.2 LCPC Algorithm-2

A second algorithm made by LCPC evaluates the important performance indicators, other than the risk, for the different manoeuvres, such as speed, comfort and fuel consumption. This algorithm will create sub-manoeuvres with different longitudinal target speeds and different lateral target positions but belonging to the same parent-manoeuvre. It will evaluate each of them, in order to have a profound understanding of the corresponding parent-manoeuvre. The most probable sub-manoeuvre of the detected objects are predicted and projected on the detected lanes, giving a rough understanding of their movements.

In a next step the performance (Valential) or alternatively, the cost of each proposed sub-manoeuvre of the ego-vehicle is calculated. This cost is a more complex than the risk described in the first algorithm. It integrates other such as speed, estimated comfort, con-sumption, and road code offences, amongst other possible ones.

The *definition of risk cost* is based on the same measurements as described in the first LCPC algorithm. Here, the risk cost takes into account possible instability from slipping if road friction information is available.

The Speed cost penalizes slow sub-manoeuvres. It is calculated as the difference of the distance which could be reached at legal speed limits and the distance actually reached within the time period of the suggested sub-manoeuvres.

The Comfort cost is calculated as the quadratic sum of all variations in acceleration proposed by the sub-manoeuvre.

The consumption cost is deducted by a simple formula on the acceleration and speed profile of the trajectories. Together with the *Comfort cost* it will give a priority to gentle sub-manoeuvres.

Road code offence cost integrates penalties for speeding and for crossing full road marks. Driving on the left lane on a high way does not lead to penalties by law but in the algorithms it gets small offence costs inviting the pilot to choose the right lane when possible.

The *wish cost* promotes sub-manoeuvres that are in line with the human driver wish, directly expressed, e.g. by the activation of the blinkers, or indirectly, e.g. by the position of the gas pedal.

The *total cost* of sub-manoeuvres is the weighted sum of each partial cost. The weights used set the character of the Co-pilot. The *total costs of a manoeuvre* is obtained as the average *total cost* of its corresponding sub-manoeuvres and translates directly in a Valential for each manoeuvre, and is outputted by this algorithm.

Here is a summary of those costs to be computed for each grid cell of the strategy decision grid:

RISK_COST	=	GravityOfCollision x ProbabilityOfCollision
SPEED_COST	=	DistanceAtLegalSpeedLimit – RealDistance
COMFORT_COST	=	Sum of X and Y jerks during trajectory
OFFENCE_COST	=	Penalties for right overtaking, excessive speed
WISH_COST	=	Cost of not following driver wishes: e.g. indicators,...
CONSUMPTION_COST	=	Combination of X accelerations and speed
TOTAL_CELL_COST	**=**	**Weighted sum of all PARTIAL COSTS**

Finally, as each cell is representing a specific manoeuvre, those costs enable a ranking of all possible 9 manoeuvres (Figure 4).

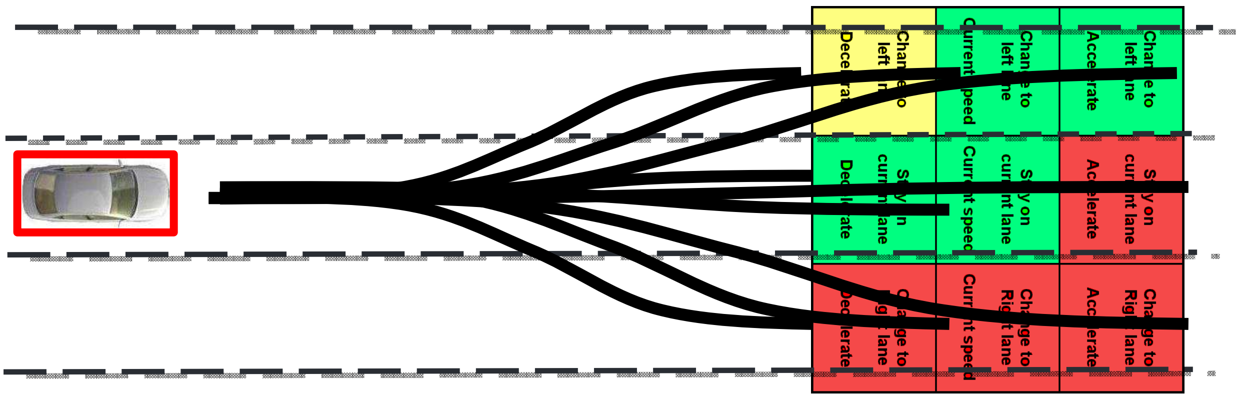


Figure 4: The nine possible manoeuvres, a cost value is assigned to each cell (manoeuvre).

3.1.3 DLR algorithm

The algorithm developed by DLR uses a very different approach but delivers the same output structure as the LCPC algorithms. Indeed, the basic idea is to generate a decision of the manoeuvre to be executed by the vehicle. The decisional system is based on a fuzzy logic system which outputs are the next manoeuvre to be executed as well as the corresponding *valential.* The set of successive manoeuvres forms a manoeuvre tree. Each node is a manoeuvre-valential couple and each arc is a pointer to a next possible manoeuvre. In Figure 5 the yellow box represents the current manoeuvre, the green and red boxes represent the next possible manoeuvres with their corresponding valentials. Here the manoeuvre to be executed is the one with the higher valential which states that the vehicle should overtake by changing the lane on the left.

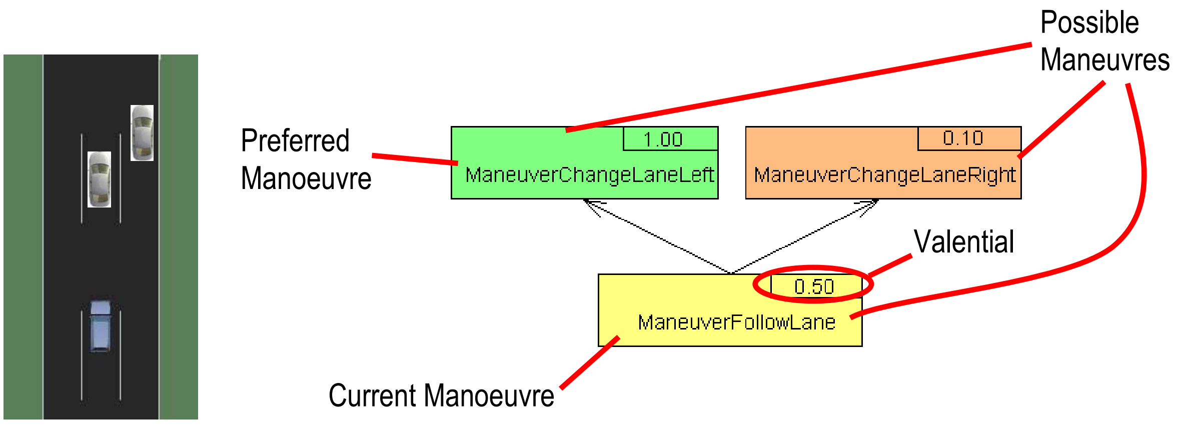


Figure 5: DLR's decisional tree

3.1.4 Fusion of the LCPC1-LCPC2-DLR output

The two LCPC algorithms and the DLR algorithm run in parallel and at the same frequency. The output of these three algorithms is fused by a voting module into a single output. The very different nature of these three algorithms creates true robustness and reliability to this safety relevant component.

As there is a unique translation between the tree and the grid format, the fusion module output can be given in both representations.

3.2 Fine planning - the motion planner (PMP)

The *“definition of trajectory”* sub-module generates a detailed spatio-temporal description of the optimal trajectory to be executed. It describes and evaluates the manoeuvres proposed by the *“definition of driving strategy”* sub module in greater detail, choosing the best manoeuvres first, until a predefined calculation time span (which is linked to a safe reaction time) elapses. The output of this sub-module is amongst others is included in interactor *“I30 Set of trajectories with descriptions”,* ranked from best to worst by interactor *“I8 Priorization of trajectories”.*

In practice, once a coarse plan has been defined a specific motion has to be defined for the vehicle. This motion needs to define the state of the vehicle for the future time instants. The sequence of desired states in time is called *a trajectory* (Figure 6).

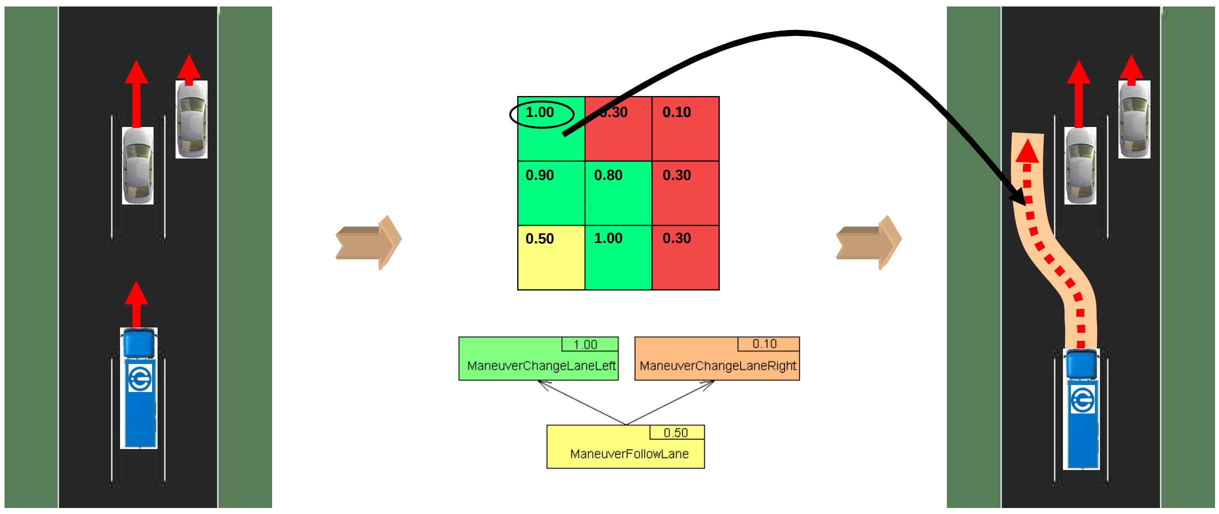


Figure 6: According to the given driving situation, the best manoeuvre is decided accounting the trajectories rating using the grid and tree Valentials. The appropriate refined trajectory is then planned (here, overtaking and acceleration on the left).

In order to provide guarantees of the safe motion of the vehicle, when computing the trajectory the vehicle has to correctly consider its own limitations and the future movement of the other vehicles.

The approach taken in HAVEit follows the works of [Fraichard2007], [Petti2007] and [Benenson2008].

Partial planning: Since the vehicle has a limited visibility its plans can only reach a limited horizon. Since a wall (traffic jam, road blockage, etc...) could exist on the frontier of the unobserved areas all trajectories are required to stop before reaching the end of the visibility region. When the observed region is updated, the trajectory is also updated. Because of the partial nature of the provided trajectory, we call this approach *Partial Motion Planning* [Petti2007].

Safety: In order to ensure that the trajectory is feasible by the vehicle, trajectories generation strategy is based in a search in the commands space. Given an initial vehicle state (position, speed, steering), we search the set of commands that will allow to reach at best the goal. The model used to integrate the effects of a sequence of commands takes into account the saturation of the vehicle in steering and acceleration [Petti2007].

Also, for any given state of the partial trajectory it is verified that the vehicle is capable of stopping without colliding. By doing so, we ensure that at any time the solution available will not actively collide the vehicle. In order to provide this guarantee it is necessary to use a conservative prediction of the vehicle's surroundings [Benenson2008].

Simplifications: Directly using a full search on a discrete commands space, using a continuous curvature distance metric to reach a specific goal and doing brute force collisions checking has been shown to provide satisfactory results [Petti2007, Benenson2008b]. However, the HAVEit project presents specific needs, and the previous works need some adaptation.

First, driving in HAVEit is modelled as operations on lanes. The goals and obstacles are defined as presence on lanes. This provides a coarser (faster) discretisation for collision checking and simplifies the distance metric to goal (how far are we from reaching the centre of the desired lane?).

Secondly, instead of searching a trajectory that avoids the obstacles and reaches the goal as best as possible by any means, a simplified approach is used: the trajectory goes straight towards the desired lane and stops if any obstacle is present. The circumvention of obstacles is prohibited, this responsibility is delegated to the strategy level which will decide the sequence of lane changes required to circumvent an obstacle.

These simplifications allow a more efficient implementation, in code size, memory usage and computation time.

Algorithm overview: In this simplified version the construction of the trajectory is equivalent to a direct “greedy search” of the nearest safe child state node (see Figure 7). Starting from the initial vehicle state, a finite set of commands is used to integrate the vehicle model to create a set of child nodes. For each node it is verified if the vehicle can or cannot safely stop. The safe node that is nearest to the desired goal is selected for the next iteration.

The output of the algorithm is then the sequence of states from the initial state to the current state (which is a state in the future).

In the current implementation each state is described by a vector composed of

[time, x position, y position, orientation, steering angle, speed]. Having a sequence of positions in time allows deducing the speed element, but we add this redundant element to ease the control stage.

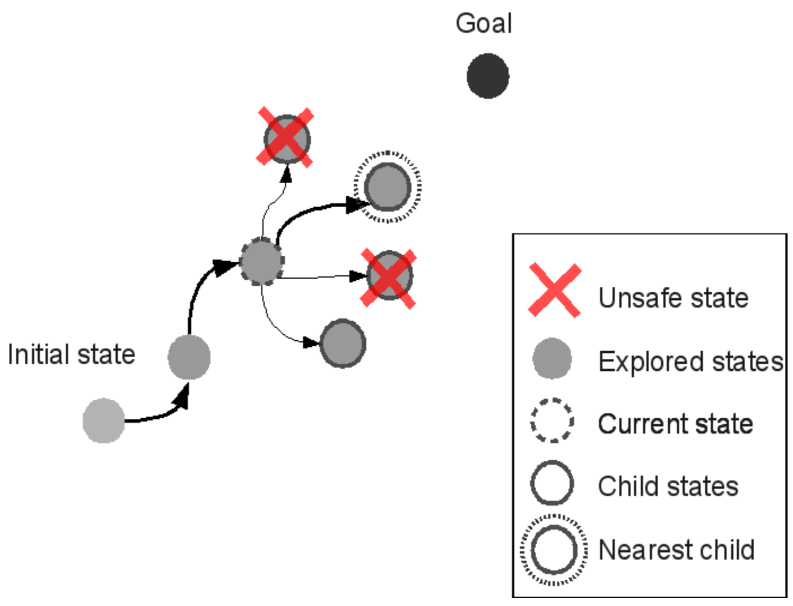


Figure 7: Construction of the sequence of states in time

Assisted driving: It is important to notice that the safety guarantees of the generated trajectory are *only valid in automated mode.* When the car is in a manual mode there is no certainty of where it is going to move, no certainty that it will respect the defined trajectory. Also, the transition between assisted and highly automated mode is fairly challenging from a safety perspective.

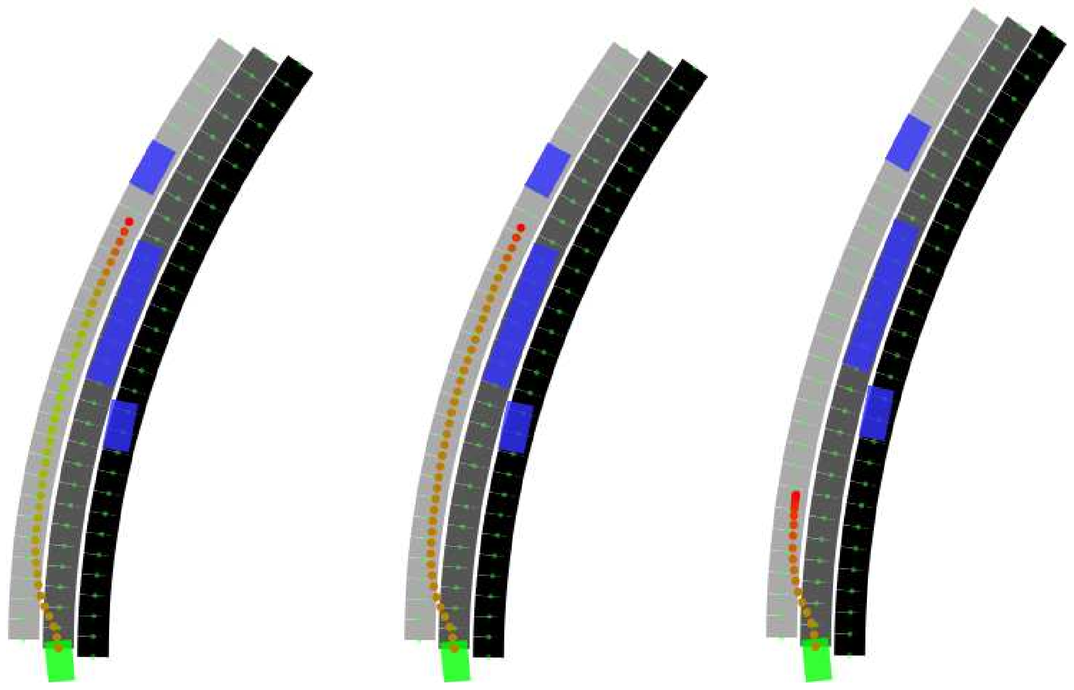


Figure 8: Green square (ego-vehicle), blue squares (vehicles-obstacles), coloured dots are the successive trajectory positions from red (zero speed) to green (high speed), orange is medium speed. Three scenarios: change left and accelerate (a), change left and hold (b), change left and stop (c).

4 Results

4.1 Implementation and integration

The described algorithm was implemented in pure C with static memory allocation. This C program was interfaced with a C++ visualization interface developed by INRIA for HAVEit simulation purposes (see Figure 8 above), and with the C++ HAVEit joint simulation framework called **SMPLab/SILAB** developed by DLR and WIVW (see Figure 9 below).

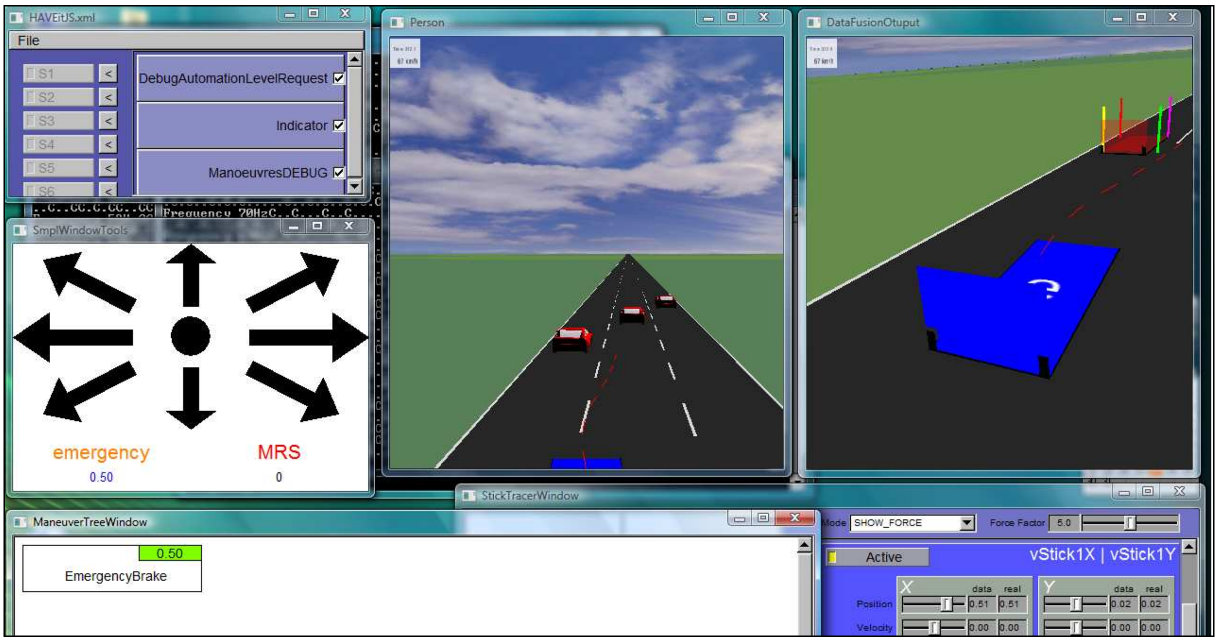


Figure 9: Closed loop simulation of a HAVEit scenario using SMPLab/SILAB tool

These C++ interfaces allow us to test the trajectory planning in specific scenarios and also the full co-pilot system in a closed simulation loop.

When running on a 2 GHz computer the simplified implementation of the fine planner provides full trajectories in a few tens of milliseconds, enough for real time operation.

The inputs and outputs used during static tests or dynamic simulations are strictly the same as described in the interactors document.

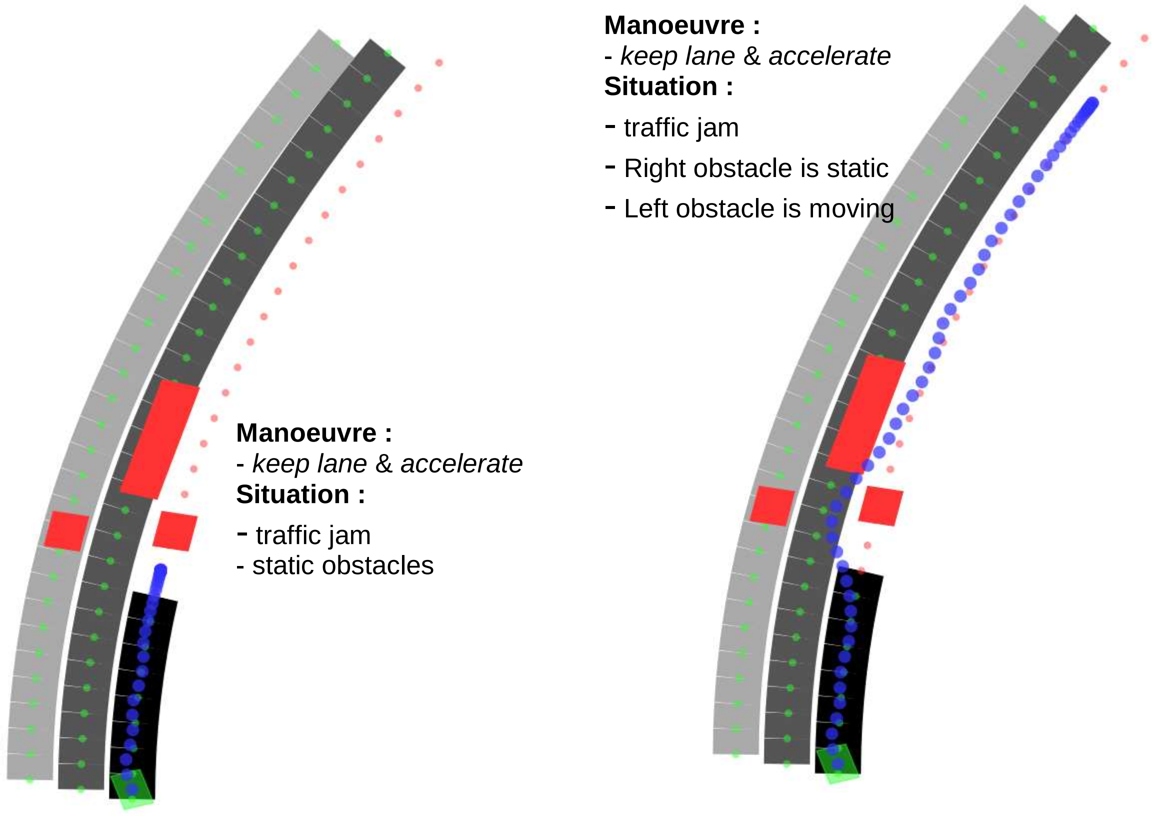
4.2 Example outputs

In Figure 10, we present an example of output of the simplified Partial Motion Planning method for several scenarios. For different manoeuvres, the planning method provides different trajectories.

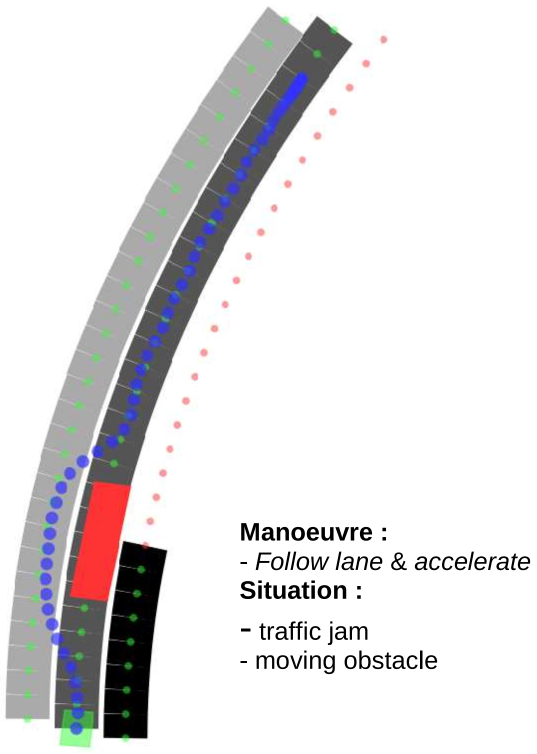
Figure 8 presents a screenshot of the employed closed loop simulation environment, where the manoeuvres and trajectories are computed in real time.

Results on some HAVEit use-cases:

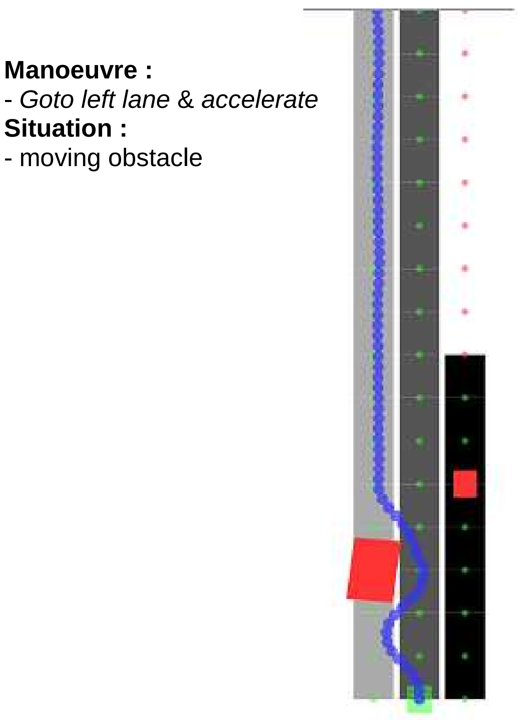
In Figure 10, in red are the static or mobile obstacles, in green is the ego vehicle and in blue are the computed successive positions of the ego vehicle planned trajectory accounting a selected manoeuvre (best ranked according to the strategy coarse level). In Figures 10a to 10c, the order from the manoeuvre level was to keep the current lane and accelerate. Because of the static obstacles (traffic jam) in Figure 10a the PMP planner decided to stop since no overtaking was possible. In Figure 10b we changed one parameter: the middle obstacle was moving. This allowed the planner to find a possible overtaking solution instead of stopping the HAVEit vehicle. In Figure 10c, the static obstacle on the lane obliged the ego vehicle to perform overtaking before returning to the lane. Figure 10d is quite a tricky situation illustrating a possible malfunctioning of the strategy manoeuvre level. Here, the decision is to go to the left lane and accelerate in spite of the presence of an obstacle. The planner was forced to find a trajectory performing the corresponding manoeuvre; this explains the hazardous beginning of the trajectory and its latter complexity!


Manoeuvre :
- *keep lane & accelerate*
Situation :
- traffic jam
- static obstacles
Manoeuvre :
- *keep lane & accelerate*
Situation :
- traffic jam
- Right obstacle is static
- Left obstacle is moving

(a) **(b)**



(c)



(d)

Figure 10: Co-pilot planned trajectories for 4 different scenarios (use-cases)

Figure 11 shows another visualization tool used to illustrate the trajectory planning and execution. It is based on a Google Earth for its ability to overlap geo-localized 2D maps. In this figure the lane markings are in cyan, the obstacles are dynamic obstacles while the vehicle shows the HAVEit vehicle (ego vehicle). The planned trajectory is in yellow; its geometric shape is projected on the ground.

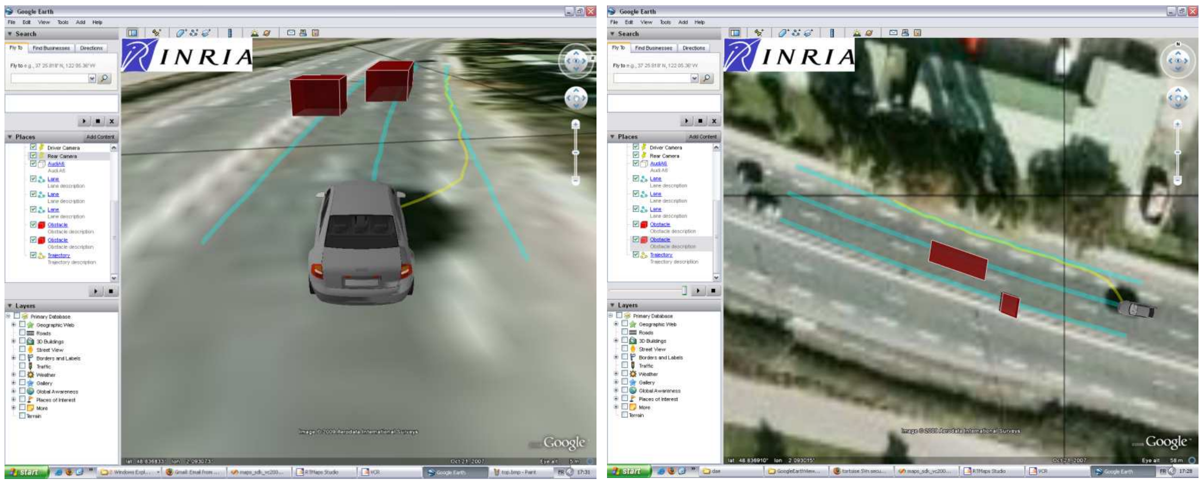


Figure 11: Visualization tool developed to illustrate HAVEit scenarios using geo-localized data

5 Conclusion

The main objective of WP3100 during the project’s first reporting year is to design, develop and integrate a reliable co-pilot capable of assisting the driver in his driving task on open roads. The prototype developed by the work package partners has been integrated in the development platform for to the WP4100 demonstrator (FASCar of DLR).

This deliverable describes the co-pilot system architecture and especially the motion planner that generates the trajectories and vector command to be executed by the controller.

The system is a two-stage algorithm. The first stage called the “manoeuvre level” aims at deciding the type of the next manoeuvre that should be performed in order to reach the global goal or objective. Given the manoeuvre, the second algorithm called the “trajectory level” is in fact a motion planning algorithm that aims at generating the trajectory to be followed by the vehicle in order to execute the advised manoeuvre. The generated trajectory is a geometric shape expressed in terms of a set of time-stamped states. Each state specifies the position, the speed and the angle of the vehicle at a given time. This is the path that is sent to the *CommandAndValidation* component (WP3300) which calculates the command vector sent to the controller for execution.

In order to facilitate its integration in the FASCar demonstrator, the system was implemented on a dedicated prototyping platform provided by DLR. Several driving scenarios and use-cases have been successfully tested using simulated realistic data, proving the validity of the concept.

The next step is the integration of the system in the real FASCar vehicle and the execution of the scenarios using real data coming from the data fusion subsystem.

References

[D11.2]	D11.2 Specification, HAVEit deliverable, 2009
[D12.1]	D12.1 Architecture, HAVEit deliverable, 2009
[Petti2007]	S. Petti, “Safe navigation within dynamic environments: a partial motion planning approach,” Ph.D. dissertation, Ecole des Mines de Paris, 2007.
[Fraichard2007]	T. Fraichard, “A short paper about motion safety,” in Proceedings of the IEEE International Conference on Robotics and Automation, 2007. http://hal.inria.fr/inria-00134467
[Benenson2008]	R. Benenson, T. Fraichard and M. Parent, "Achievable safety of driverless ground vehicles", in Proceedings of the IEEE International Conference on Control, Automation, Robotics and Vision, 2008. http://hal.inria.fr/inria-00294750
[Benenson2008b]	R. Benenson, “Perception for driverless vehicles: design and implementation,” Ph.D. dissertation, École des Mines de Paris, 2008.

Annex 1 Abbreviations

CAF	Continental Automotive S.A.S., France
CAG	Continental Automotive GmbH
CoP	Co-Pilot
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DM	Driver Monitoring
ECU	Electronic Control Unit
GPS	Global Positioning System
HAVEit	Highly Automated Vehicles for intelligent transport
HMI	Human Machine Interface
HW	Hardware
IBEO	Ibeo Automobile Sensor GmbH
ICCS	Institute of Communication and. Computer Systems
INRIA	Institut Nationale de Recherche en Informatique et Automatique
LCPC	Laboratoire Central des Ponts et Chaussées
MSU	Mode Selection Unit
PMP	Partial Motion Planning
SW	Software
TTC	Time To Collision
WP	Work Package within HAVEit