Dream-like simulation abilities for automated cars

DREAMS4CARS

Grant Agreement No. 731593

Deliverable:	D1.3 – System Architecture (Release 2)
Dissemination level:	PU – Public
Delivery date:	24/12/2018
Status:	Final

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731593

Deliverable Title	System Architecture (Release 2)		
WP number and title	WP1 Application domain requirements, Architecture and Product Quality Assurance		
Lead Editor	Sean Anderson, USFD		
Contributors	Mauro Da Lio, UNITN		
	Henrik Svensson, HIS		
	David Windridge, MU		
	Rafael Math, DFKI		
	Elmar Berghofer, DFKI		
	Andrea Saroldi, CRF		
Creation Date	16/11/2018	**Version number**	1.3
Deliverable Due Date	31/12/2018	**Actual Delivery Date**	24/12/2018
Nature of deliverable	x	R - Report	
		DEM – Demonstrator, pilot, prototype, plan designs	
		DEC – Websites, patents filing, press&media actions	
		O – Other – Software, technical diagram	
Dissemination Level/ Audience	x	PU – Public, fully open	
		CO - Confidential, restricted under conditions set out in MGA	
		CI – Classified, information as referred to in Commission Decision 2001/844/EC	

Version	**Date**	**Modified by**	**Comments**
0.1	16/11/2018	Mauro Da Lio	Definition of contents and section 1.
0.2	19/11/2018	Mauro Da Lio	Sections 2.1 and 2.2
0.3	21/11/2018	Mauro Da Lio	Directions to edit sections for collaborators
0.4	06/12/2018	Andrea Saroldi	Overall review
0.5	06/12/2018	Sean Anderson	Added sections 3.2.3, 3.3.1, 3.3.2
0.6	06/12/2018	Sean Anderson	Added section 3.2.2 from HIS
0.7	11/12/2018	Henrik Svensson	Minor additions and corrections for 3.1 and 3.2.2 and added description of LRM by MU.
0.8	13/12/2018	Sean Anderson	Minor integration edits
0.9	14/12/2018	Mauro Da Lio	Section 3.2.1. General revision.

1.0	14/12/2018	Sean Anderson	Section 3.2.3. General revision.
1.1	20/12/2018	Sean Anderson	Edits suggested by reviewers.
1.2	23/12/2018	Mauro Da Lio	Final version
1.3	24/12/2108	Hermann Heich	Final layout/corrections

Definitions, acronyms and abbreviations

Abbreviation	**Meaning**
CDZ	Convergence-divergence zones (in Damasio's dorsal stream architecture)
Euro NCAP	European New Car Assessment Programme
HWC	Highway Code
MSPRT	Multi-hypothesis Sequential Probability Ratio Test algorithm
WTA	Winner Takes All algorithm

Executive Summary

This document is the final version of the Dreams4Cars system architecture. It describes the architecture of the learning/simulation system (the offline dreaming machinery) and the architecture of the agent (the runtime artificial driver agent).

This document completes D1.1 (version 1 of the system architecture) illustrating the final configuration but without repeating (as much as possible) concepts that did not change or have already been illustrated. In particular, considerations concerning traditional design practices and the *expected benefits* of the Dreams4Cars architecture have been given in D1.1 in both the executive summary and section 1, and are here summarized at the end of section 2.

This document gives a system-level description that is meant to be an introduction to the Dreams4Cars approach. Details concerning implementations are given in the public deliverable D2.3 for what concerns the agent and in the upcoming public deliverable D3.3 for what concerns the dreaming machinery[1].

The agent uses a biologically inspired architecture with 5 loops (Section 2):

1. The action priming loop (dorsal stream), which converts sensory data into activation patterns of the motor cortex, encoding the many instantaneous action possibilities in parallel;
2. The frontal cortex loop which implements a logical reasoning system that can steer the agent behaviour via the biasing of action selection;
3. The action selection loop (basal ganglia), which implements robust and bias-able selection of one action;
4. The cerebellar loop, which learns forward and inverse models of the vehicle dynamics;
5. The motor output loop which converts the selected actions into vehicle specific commands via learned inverse vehicle dynamics models.

Simulations and learning occur offline in different places and ways (Section 3):

a) Episodic simulations occur via two different mechanisms:
    - The convergence-divergence zones (CDZ) organization of the dorsal stream learns compact representations of events that may be used to instantiate a first form of episodic simulations.
    - Symbol recombination via Genetic Algorithms produce a second form of episodic simulations that complements the first.
b) Learning from episodes involves the running of detailed simulations in the OpenDS environment. The value of action sequences, generated by a Logical Reasoning Module that embeds the Highway code, are learned at the time of action selection via a software module that implements Reinforcement Learning.
c) Embodied simulations occur as follows:
    - The cerebellar loop learns forward and inverse models of the vehicle dynamics.
    - These are used to instantiate embodied simulations for low-level and tactical level control and for the synthesis of the dorsal stream loops that compute the salience function.
d) Learning from embodied simulations is carried out via direct and indirect optimal control (that learns the salience function for tactical level manoeuvres).

[1] D2.3 and D3.3 also include updates to the implementation plans that were formulated in D1.1 (test plans updates are given in the public deliverable D5.2).

Table of Contents

List of Diagrams

1 System Architecture

The Dreams4Cars system is composed of 3 different environments (Figure 1). *This has not changed since release 1* of this deliverable; hence the purpose of the three environments is only briefly reminded.

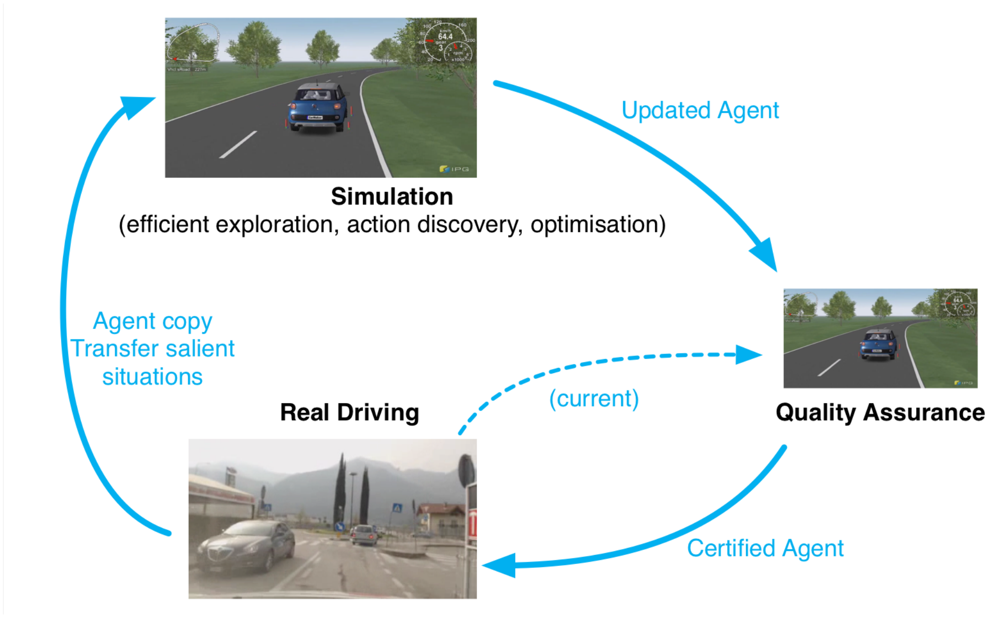

Figure 1: Dreams4Cars system Architecture.

1. The Real Driving environment corresponds to the human "wake" state. In this environment the Co-driver[2] agent operates the vehicles and collects information for creating episodic and embodied simulations.
2. The Simulation environment corresponds to the human "dream" state. In this environment (a copy of) the agent operates in a virtual world where fictitious situations are *self-generated*. Optimization of the agent sensorimotor abilities via two different dreaming machineries termed "episodic" and "embodied" simulations have been developed and occurs here in various forms.
3. The Quality Assurance environment is an auxiliary environment where a copy of the optimized agent (output from point 2) is tested against a library of test cases; the performance of the agent is assessed with several types of metrics (possibly including the Euro NCAP scenarios); the progress in agent abilities is monitored.

The current release of this deliverable (D1.3, System Architecture, release 2) reports the final results of WP1.2 (system architecture of both the runtime and offline components). It describes the final architectures of the agent in itself and of the dreaming mechanisms. Considerations concerning traditional design practices and the *expected benefits* of the Dreams4Cars architecture have been given in D1.1 in both the executive summary and section 1 and are not repeated here except for a summary at end of section 2. More information concerning the implementations of the agent and of the simulation system are given in the public deliverables D2.3 and D3.3.

A project video that gives a general idea of the agent (tailored for communication, not for scientific dissemination) is given at: https://youtu.be/0mylY6_ZDJY

[2] The agent driving the vehicles in this project is termed "Co-driver agent" or shortly "Co-driver".

2 Final agent architecture

The final version of the agent architecture is shown in Figure 2.

Compared to the initial architecture the main innovation is *a clearer definition of the action biasing mechanisms* (in the frontal cortex loop). Furthermore, the *internal organization* of the already existing loops (dorsal stream, cerebellum and basal ganglia) *has been better specialized* and specified.

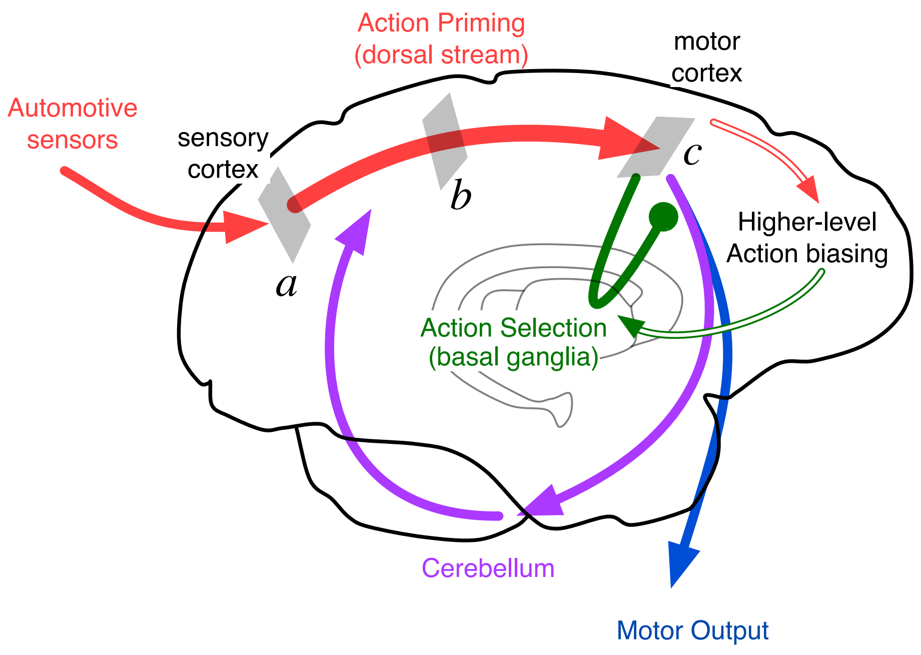

Figure 2: Final agent sensorimotor architecture.

It has to be mentioned that some loops (dorsal stream, cerebellum and frontal cortex) operate in slightly different way in the offline versus online mode, as explained below.

2.1 The dorsal stream and the motor cortex

In online operation the purpose of the dorsal stream is detecting (all) affordable actions and creating action plans. The output of the dorsal stream takes the form of active and inhibited regions (see examples in Figure 3) of a structure shown in Figure 2, label *c*, corresponding to the biological “motor cortex”. A two-dimensional array is used for the purpose, where the two dimensions correspond to abstractions of the longitudinal and lateral control (longitudinal jerk and curvature rate).

The dorsal stream may be implemented either algorithmically or with neural networks or with hybrid implementations that combine simple neural network building blocks with algorithmic wrappers (Dreams4Cars use this latter, e.g., [1]). The training of the dorsal stream is based on optimal control, using the learned vehicle forward models, via episodic simulation processes described in the next section.

As the main online use of the dorsal stream is converting sensory data (from the available automotive sensors) into active/inhibited regions of the motor cortex, the dorsal stream may be trained to operate with any set of sensors and hence the architecture is sensor agnostic as long as sufficient sensor data is produced.

Furthermore, in the online use the dorsal stream may also operate in conjunction with the sensory anticipation produced by the cerebellum as if it were just another source of data: simulated. This may have several uses [2], [3], among which filtering of perceptual data and switching between different learned forward/inverse models (D2.3 section 3).

Beside the online use of the dorsal stream, there is another use offline, for creation of episodic simulations that will be described in the section 3.

2.1.1 Modular organization

The dorsal stream may be conveniently organized in modular structures.

- First *excitatory* and *inhibitory* circuits may be realized in parallel and separately (Figure 3, left: the three affordable lanes create three humps of activity encoding the longitudinal/lateral control of the three actions; right: obstacles create inhibited regions cancelling actions that might result in collisions).
- Second, for both excitatory and inhibitory circuits, *re-use of elementary modules* is possible. For example, the algorithm/network that computes the peak of activity for goal *b*, may be re-used for *a* and *c*. Similarly, a module that creates inhibitions for one obstacle may be re-used for all obstacle and, within the same obstacles for the individual future positions of its predicted trajectory (e.g., [1] or D2.3 section 2.1.4.2).

Modularity contributes significantly to the safety, interpretability and predictability of the dorsal stream when it is implemented with neural networks. Building blocks may be tested singly, their output can be interpreted and their collective behaviour will ensure collision free behaviours once they have been proved to be collision free individually [1].

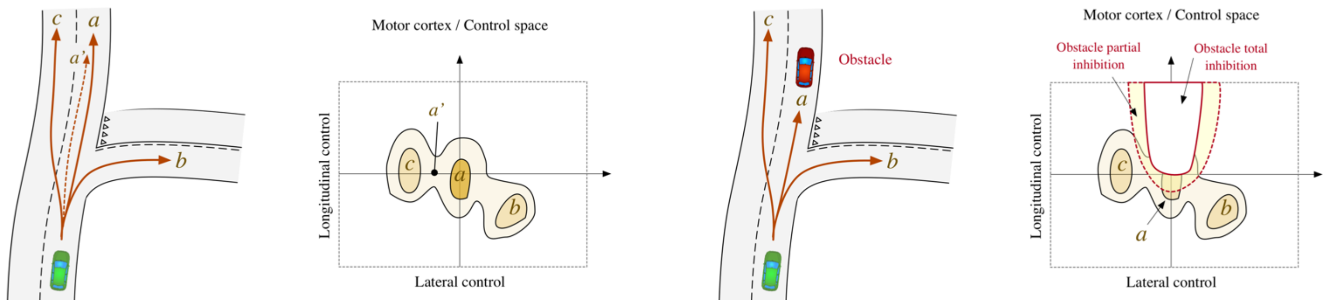
c
a
a'
b
Motor cortex / Control space
Longitudinal control
Lateral control
a'
c
a
b
c
a
Obstacle
b
Motor cortex / Control space
Obstacle partial inhibition
Obstacle total inhibition
Longitudinal control
c
b
a
Lateral control

Figure 3: Dorsal stream function. Left: the three affordable lanes create three humps of activity encoding the longitudinal/lateral control of the three actions; actions that allows to remain in the road but not exactly in the lanes have less salience (shaded area) so that they are selected only as a last resource. Right: obstacles create inhibited regions cancelling actions that will likely produce collisions; close encounters are partially inhibited so that they are chosen only as a last resource.

2.2 Action biasing and action selection

It is worth stressing that the goal of the dorsal stream is not trajectory planning in the traditional sense (creating one optimal trajectory). Instead the goal is *detecting affordances and creating a value map* (the motor cortex – let us call it "*mc*") for the selection of the longitudinal/lateral control (that when iterated produces a trajectory).

The actual selection of action is postponed, and this allows to combine action possibilities with higher-level directives and biases that may derive from a hierarchy of intentions; either derived from programmed traffic rules or from learned action sequences as follows.

For example, let us assume that the intention of turning right at *b* is instantiated at a higher level. Hence the yellow-marked lane in Figure 4 is highly desirable. The activation pattern corresponding to this intention is created (Figure 4, right) and is normalized (let us call it a biasing gain matrix "*B*"). It identifies all controls that comply with goal *b*. Let the "strength" of the intention be represented by a scalar (possibly learnable) weight w_b. The actual action selection is then carried out by weighting the motor cortex according to the shape function given by the biasing matrix *B* and the intensity given by weight w_b. Roughly speaking (albeit the actual algorithm is more complex) the decision is not carried out on the motor cortex in itself, but on a modified motor cortex *mc'* as follows:

$$mc' = mc * (w_b B)$$

where "*" stands for the elementwise product. This way higher-level intentions (such as sequences of actions) may *steer* the agent behaviour.

- However, they are not instantiated exclusively and there always is the possibility to carry out another decision should the preferred choice be too difficult.
- Furthermore, high-level intentions operate in a sandbox, where only safe actions created in "*mc*" can be biased and then selected; this way, any error in the high-level intentions is rejected by the lower level control as will be shown below.

For example, the highest-value action in Figure 3, left is *a* (remain in the current lane). Without any bias this would be chosen. Should the higher-level intention of the agent be taking exit *b*, the biasing matrix *B* may be used to artificially increase the importance of option *b*, eventually leading to the choice of *b* if w_b Is large enough. The weight w_b may vary and hence, while the biasing matrix *B* specifies which controls corresponds to "*b*", the weight specifies how strong the preference for "*b*" is (should the original motor cortex salience of "*b*" in Figure 3, left, be very small –for example because it is a very difficult manoeuvre– it may happen that *b* is not selected anyway).

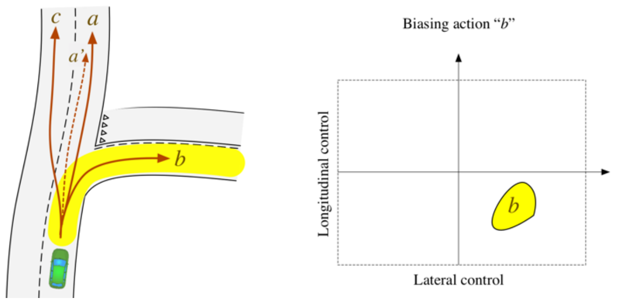

Figure 4: Biasing action *b*. A normalized copy of the motor cortex with the salience map of action *b* is created and used to weight the motor cortex itself. The weight may vary and hence, while the biasing matrix specifies which controls corresponds to "*b*", the weight specifies how strong the preference for "*b*" is (should the motor cortex salience of "*b*" be very small –for example because it is a very difficult manoeuvre– it may happen that *b* is not selected anyway).

This way it is possible to steer the instantaneous selection of actions including choosing low value actions that are necessary for an action sequence that achieves higher rewards in the longer-term (for example a safe overtake). For this, in principle, the weights w_b may be learned via reinforcement learning [4], hence learning when and how intentions and action sequences are safe.

In any case, since no action is instantiated exclusively, a strong bias *B* would be necessary if the curve were significantly more difficult than driving straight. Furthermore, if the curve did not exist or turned out to be blocked, whatever the bias strength it would not be chosen, because the activity at *b* in the main motor cortex would be zero (so mc' has no selectable peak at b).

Hence, once the low-level behaviour is safe, *steering for longer term-strategies are automatically safe* because no unsafe action (inhibited in the main motor cortex) may ever be selected.

If new action sequences are formulated (for example the highway code rules are edited), they turn out into new action sequences creating new biasing matrices. So, the agent may in principle incorporate new strategies into the same unifying action-selection mechanism.

The action selection algorithm of choice is the Multi Hypothesis Sequential Probability Ratio Test (MSPRT) [5]. The MSPRT algorithm accumulates evidence for competing actions (hypotheses) over time, and outputs the action corresponding to the accumulated evidence that first crosses some specified threshold (the threshold is a hyper-parameter that requires tuning to trade-off speed and accuracy). Compared to a winner-take-all (WTA) action selection algorithm that does not accumulate evidence over time, the MSPRT is more robust to noise. However, MSPRT does tend to add latency to action selection compared to a WTA method. It is believed that the basal ganglia implement such a kind of decision mechanism [6].

Details on the implementation of the biasing mechanism and the MPSRT algorithm are given in D2.3.

2.3 Learning forward/inverse models (the cerebellar loop)

In the online operation the cerebellar loop collects copies of the motor commands and sensory data that are (among the others) consequences of the execution of the motor commands. This way, both forward models and inverse models may be trained [7].

The particular interest for Dreams4Cars is to learn forward models for the vehicle dynamics to be used as building blocks for embodied simulations, such as the discovery of motor sequences to achieve short-term goals (D3.3, [8], [9]) and the training of the dorsal stream (computing salience according to the real vehicle dynamics).

Another goal is learning inverse models to be used for vehicle control, which allow to convert the abstract control parameters (longitudinal jerk and curvature rate) encoded in the motor cortex space into the actuator commands of a specific vehicle. This way adaptation to vehicles with different dynamics or to different environments may be achieved with switching between appropriate inverse models.

A final, inline use of the forward model is to produce sensory anticipations that have several applications as mentioned in section 2.1.

2.4 Vehicle control

The control approach used in Dreams4Cars is based on the execution of kinematic trajectories via the use of inverse models (see D2.3).

Figure 5 shows the control scheme. The motor plans instantiated in the dorsal stream are encoded in the abstract longitudinal/lateral control space of the motor cortex (longitudinal jerk and curvature rate). They are converted into the command for the actuators of a specific vehicle by means of inverse models of the vehicle dynamics. Updates to motor plans are continually produced by the Co-driver.

Any inaccuracy of the inverse model (together with environmental disturbances) may in turn cause deviations. The stability of the control loop of Figure 5 has thus been studied in [10], concluding that there are very large margins of stability even if the inverse model were not finely tuned/adapted.

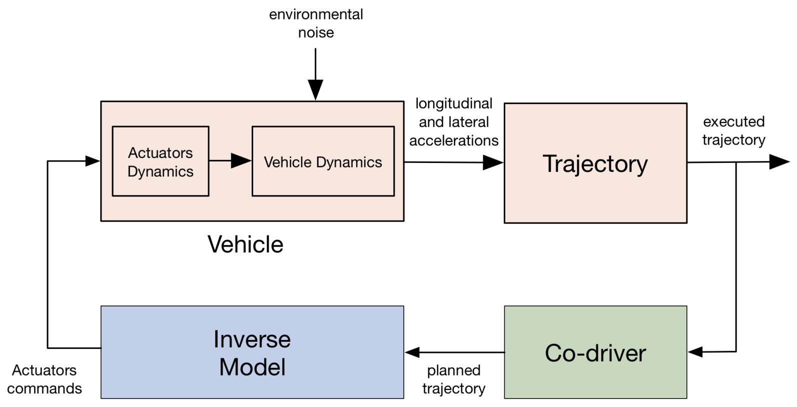

Figure 5: Inverse model approach for vehicle control.

2.5 Summary of the expected benefits of the Dreams4Cars Agent Architecture

The expected benefits of the above architecture lie in the largely *autonomous discovery* of salient situations and *self-reconfiguration* of sensorimotor control, with increase of the number of scenarios that can be studied and reduction of human coding needs. Furthermore, the chosen network-of-network architecture produces robust *safe emergent behaviours* (emerging from simple verifiable building blocks). Finally, the agent is better suited for *lifelong learning*, as novel situations may trigger the dreamlike learning process.

The advantages may be listed as follows:

1) The agent learns models of the vehicle dynamics and can use the models for chassis and tactical level control optimizations. This means adapting to different vehicles dynamics and (possibly on the fly) to different environments and operating conditions; it also means predicting vehicle response for sensory anticipation and condition monitoring/failure detection. It also means using learned models within embodied simulations to discover (via direct/indirect optimal control) the value of short-term and tactical level actions (learning action salience based on learnt vehicle/environment).
2) The agent learns compact representations of episodes, and can use them to create new episodes, from which to learn higher-level strategies via reinforcement learning.
3) The agent may incorporate symbolic rules (in particular the rules of the highway codes) via biasing mechanism that act on top of a safe sandbox composed of the low-level motor control substrate. Where schematic-legal manoeuvres are no longer possible the agent would still evaluate physically feasible trajectories as a last resource.
4) The network-of-network architecture may be trained for the individual components separately.
5) The modular architecture of the dorsal stream significantly contributes to the safety verification: individual modules (especially thanks to the fact that the motor cortex is interpretable) may be tested in isolation and once they are deemed to be safe, their collective superposition will also be safe (such property is more difficult to grant for non-structured neural networks and for end-to-end collectively trained networks).
6) The creation of fictitious situations (both low-level embodied simulations and the high-level episodic simulations) is *largely automatized*. Where in the traditional approaches, simulations are human directed (which means that every possible situation have to be conceived and programmed in the simulation by human designers), here simulations are automatically created, going beyond simple re-sampling of recorded events, and hence increasing the number of cases that can be examined and used for training.
7) The agent can self-reconfigure with large degree of autonomy. While in traditional approaches, once a critical situation is detected and diagnosed the software has to be re-coded by a human driver, with the Dreams4Cars approach neural network components are further trained on newly discovered situations (or new vehicle dynamics and environments) requiring less human designer intervention.
8) Novel situations where the agent underperforms (not necessarily accidents) become focal points for starting new (lifelong) learning processes, with reduced human supervision needs; whereas malfunctions must be manually studied in details with the traditional approach.

3 Simulation system

The simulation system is the body of methods and algorithms that allow the implementation of the offline (dream state) simulations and learning (see the confidential deliverables D3.1 and D3.2 and the next public deliverable D3.3).

3.1 Episodic and embodied simulations

Dreams4Cars uses a recent idea concerning the existence of *two* distinct biological imagery mechanisms: *episodic simulations* and *embodied simulations* [11]. The two forms differ with respect to neural mechanics and brain locations, and –functionally– with respect to content detail and time-scale:

1) Episodic simulations rely on *neocortical/hippocampal* systems. They generally focus on events at higher abstraction and occurring on longer time scales. For example, imagining that a pedestrian standing on the sidewalk might suddenly cross the road is an episodic simulation.
2) Embodied simulations rely on *cerebellar* loops. They include more details in terms of movement, perception and work on shorter time scales. For example, predicting the vehicle response to a particular steer swerve is an embodied simulation.

3.2 Episodic simulations

There are two distinct mechanisms for episodic simulations. One, more closely inspired to the biology, works better for simple episodes as well as for episodes related to the modelling of other road users' behaviours, and is well suited for creating extraordinary situations. Another, more abstract, works better for more complex episodes allowing the recombination of many different items, but is less suited for creating extraordinary situations.

3.2.1 Episodic simulation originating in convergent-divergent dorsal stream

The dorsal stream, has a slightly different implementation between online and offline use (Figure 6). For the latter in particular, it has a convergent-divergent structure that follows the organization proposed by Damasio [12] (for the online use it does not need). Hence, sensory data collected in "*a*" (Figure 2) are initially compressed into a reduced size tensor "*b*" (we speak here of a deep neural network implementation), from which they are expanded to the motor cortex "*c*". Backwards (offline use) signalling is postulated by Damasio, which means that information may also flow backwards from "*c*" to "*b*" and, in particular, from "*b*" to "*a*".

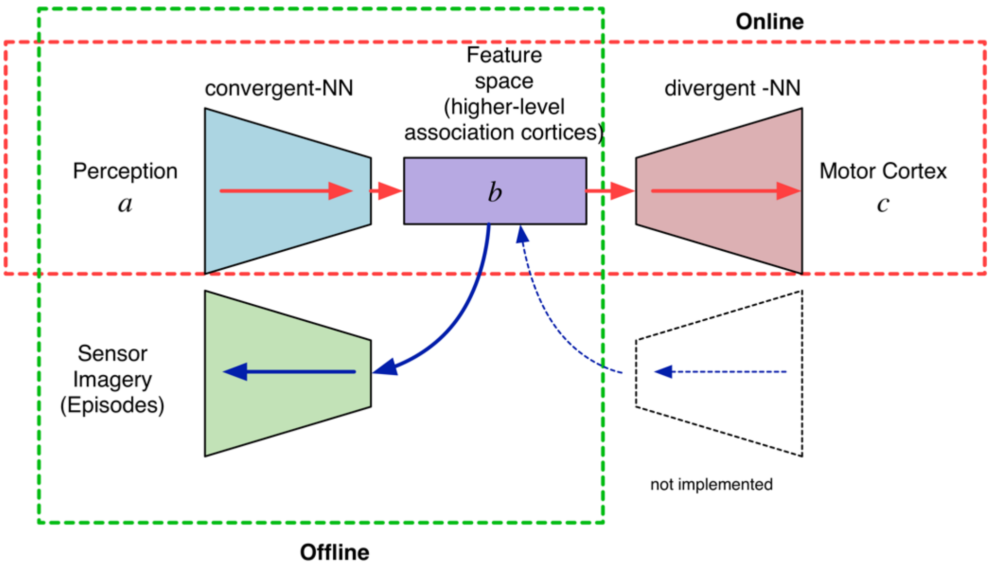

Figure 6: Convergence-divergence zones implementation of the dorsal stream for episodic simulations.

The branches *a-b* (convergent), *b-c* (divergent) and *b-a* (divergent) are implemented with neural networks in Dreams4cars and are exploited for creation of episodes. The process exploits the fact that, the convergence-divergence zones force the categorization of sensor-motor data in concepts at "*b*" (actually across the whole CDZs). Hence, during wake time, compact representations of events are learned at "*b*". Note that the convergence-divergence zones *a-b-c* resemble the structure of an auto encoder (AE) except the output in the motor cortex (see example [13]). This way the compact representation "*b*" is optimized for what is relevant for actions (rather than being optimized to reproduce the input).

Sensor imagery, and the creation of episodes, is thus obtained by sampling the feature space "*b"* and running backwards the branch *b-a*. With careful organization of the neural networks, novel and even extraordinary events may be created. Episodes created in this way may be used as components of simulations as described in the next section (for example creating a car that behaves in an unusual way). The episodes may also be used as part of embodied simulation described in section 3.3 (for example learning an obstacle avoidance manoeuvre for a car behaving in the unusual way just created).

3.2.2 Higher-level episodic simulation via symbol re-combinations

The basic function of episodic simulation via symbol recombination allows to create different types of episodic simulations of traffic situations that an autonomous car controller (i.e. the Co-driver in this project) can learn new behaviours from.

The system for the episodic simulation via symbol re-combination consists of three main parts: a) the episodic generation mechanism creating novel scenarios, b) a logical reasoning module (LRM), c) a tool-chain that transforms a scenario-description into the OpenDS simulation files with the car simulator OpenDS.

Episodic generator

The mechanism for creating the dream scenario, henceforth referred to as the *episodic generator*, is based on the notions of *recombination and mutation* as used in *genetic algorithms*. The motivation of using techniques inspired by genetic algorithms for the *episodic generator* is that the basic operators of selection, recombination and mutation are a flexible means for controlling the amount of change, the complexity, and the diversity of the dreams. The aspects of the traffic situation that can be recombined and modified to novel traffic situations are described in the Section "Dreams in OpenDS".

The logical reasoning module

The LRM (see public deliverable D2.3) is a module reasoning in relation to the legal road rules, i.e., the Highway code (HWC). The subsumption Perception Action hierarchy embodied within the LRM, implements the symbolic (i.e. high-level representational) component of the Dreams4Cars system, which is responsible for high-level scene interpretation/annotation and for introducing legal biasing in intention (section 2.2). The module operates on an *ad hoc* internal (schematic) symbolic representation of the traffic situation and returns action sequences for longer term goals[3] (e.g. the transitions required to carry out an overtake manoeuvre).

The LRM subsumption framework is constrained to have the capability to act *reversibly*, that is to say, in a *generative* manner via reverse PA logical-variable instantiation, such that hallucinated high-level legal road configurations are spontaneously generated alongside the corresponding legal intentionality in order to instantiate the offline dreaming process. The latter is an instance of *top down exploratory PA motor babbling*, in which theorem proving-via-resolution is applied to random instantiations of logical variables in order to establish scenarios consistent with the legal road protocols.

[3] These are converted into biasing matrices as shown in section as shown in section 2.2 and their priority will be defined by weights that are learned as explained in section 3.1.4.

Dreams in OpenDS

The simulation of the episodic dream scenarios is realized in the open source driving simulator OpenDS developed by DFKI. The interface for connecting the generated scenarios with OpenDS is a xml-scheme which through a tool-chain (described in full in confidential deliverable 4.3) generates the physical simulation.

The OpenDS xml file includes the following road elements, which consequently can be recombined into novel traffic scenarios for the Co-driver:

- Geometry elements that describe the shape of the road.
- Number, types, geometries and other attributes of lanes.
- Traffic directives (signs, speed limits etc.).
- Traffic elements (vehicles and their behaviours).
- Pedestrians and descriptions of their behaviour.

The xml-scheme has provided a flexible way of realizing dream scenarios and can easily be extended as new features may be added to the OpenDS simulator for more complex and diverse dream scenarios.

The dream process

The dream process consists of the following steps (excluding the learning of new behaviours described in Section 3.2.3).

(1) the episodic generation mechanism and the LRM creates a dream scenario of previous traffic situations (episodes or other road users' behaviours created with section 3.2.1 may be added here),
(2) the scenario is encoded as an xml-file,
(3) a tool-chain transforms the xml-description into an a physically realistic simulation of the dream scenario in the vehicle dynamics simulator OpenDS
(4) the Co-driver controls the vehicle in the OpenDS simulation, the Co-driver logs inputs/outputs and a logical reasoning module derives semantic information about the traffic situation at different levels of abstraction (called semantic annotations) from the Co-driver,
(5) the log is analysed and the analysis generates input to the episodic generation mechanism and the process re-starts.

Thus, the dream starts by having a particular type of traffic situation consisting of a particular set of objects, road structure, traffic density *et cetera*, which is then transformed into new dreams based on the Co-driver performance and progression of learning (described in the following sections).

3.2.3 Learning Action Selection from Episodes

The main goal of episodic simulation is learning of high-level strategies (what is better/safer to do in a given context). This is carried out by learning the biases w_b (as function of the context) for different action sequences generated by the LRM.

For an organism to ensure its survival, or for an automated vehicle to drive without crashing, one key computational problem that it must solve, is the clean and decisive selection of which action it is going to perform next. This is a difficult problem to solve in real-world situations because the environment is a noisy and largely unpredictable place. Noise in sensory input data could lead to the selection of an inappropriate action, or it could lead to the agent switching actions constantly, never persisting with one action to its completion.

In the vertebrate brain, it is believed that a group of sub-cortical structures known as the basal ganglia process action selection. It is this neural system that solves the problems outlined above. It has been shown that the function of the basal ganglia can be approximated by a decision-making algorithm; the multi-hypothesis sequential probability ratio test (MSPRT) (see D2.3). This algorithm has been shown (under certain assumptions) to perform optimally, it chooses the action for which there is the most evidence in the shortest time possible.

Short-term and long-term "actions"

A separate but related question is what constitutes an action? The definition of "action", in the context of driving, means different things at different levels of description. For example, at a high level, an action might be a manoeuvre; lane follow; overtake; make right turn, etc. At a lower level, an action is what control inputs need to be made to follow the desired path. What separates these two levels of actions is the timescale over which they are performed. The higher level is evaluated over timescales of the order of seconds to tens of seconds, whereas the lower level is evaluated over much shorter timescales. This seems to be the way that the vertebrate brain is organised with recurrent loops through the basal ganglia and cortical areas, which deal with many different levels of information processing.

Thus, we think that the problem of action selection in driverless cars may be best tackled hierarchically: a higher level selects which manoeuvre to perform, and a low-level selection mechanism selects how best to perform that manoeuvre. The Dreams4Cars agent therefore includes action selection at these two hierarchical levels, with one MSPRT deciding which manoeuvre to make over relatively long timescales, and second MSPRT deciding which control commands to implement on the very next time-step (Figure 7).

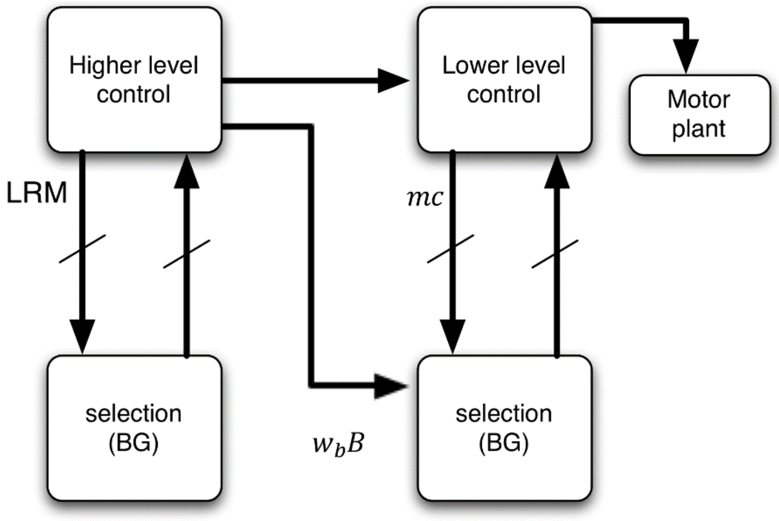

Figure 7: Action selection occurs on two level. A first selection occurs at higher-levels where one (or a few) actions proposed by the LRM are given higher weights w_b. Weights and bias matrices w_bB are then passed to the lower-level to bias the motor cortex mc.

The higher level of action selection interacts directly with the logical reasoning module (LRM). Put simply, the LRM encodes the rules of the road: what is allowed or not allowed manoeuvre.

The LRM first vetoes any control signals that would lead to car entering a road position that violates the highway code. Vetoes are translated in very low weights w_b (so that manoeuvres that violate the traffic rules are saved only as a last resource to avoid collisions).

Reinforcement Learning.

After vetoing what remains is a list of legally permitted manoeuvres, with no information on how best to choose between them. One remaining question is thus how does the high-level action selection loop choose between options that look all identically desirable? For example, on the motorway the agent is allowed to follow the slow-moving car in front, but it is also allowed to change lane and speed up to the speed limit. If both are legal, how might a vehicle choose which action is best to perform? Namely how does the agent discover convenient and safe action sequences?

In the vertebrate brain, the answer to this question is “Reinforcement Learning” (RL): the process by which actions become more likely to be performed again in future, if they led to rewarding consequences. There is a great deal of theoretical work on RL dating back half a century or more, but it is only recently, that machine learning has developed to such a state that we are able to begin to use RL on large-scale real-world problems.

The vertebrate brain has been shown to perform RL on connections between the cortex and one of the major input nuclei of the basal ganglia known as the "striatum". The connections between the cortex and the striatum are either made stronger or weaker depending on the consequences of the preceding actions. Thus, these cortical-striatal connections encode some measure of the desirability of performing each particular action given the context in which the agent finds itself. Since this process is occurring at the input of the action selection mechanism, we can model this process by placing a Reinforcement Learning neural network on the input to the high level MSPRT. This RL network learns the relative desirability (w_b) of each of the manoeuvres that have been identified as being legal by the Logical Reasoning module, and then the high level MSPRT effectively prioritizes the most desirable manoeuvres.

3.3 Embodied simulations

Embodied simulations are well suited for learning low-level chassis control and tactical-level control (namely the value of different manoeuvres). It is via embodied simulations that, for example, an agent can learn which evasive manoeuvre is best in which context (where the context is the learned forward model, which differ from one vehicle to another and from one environmental condition to another).

3.3.1 Learning forward/inverse models

This section deals with the learning of forward models using artificial Neural Networks. These (simple) networks are intended to implement sensory prediction caused by motor commands.

Inspiration may be found in paper [12] and, in particular, from the architecture of Cerebellar Adaptive Filters, and the way these filters can be trained. Despite these models are inspired from the biological organization of the "cerebellar chips", it turns out that there are several analogies with traditional Control Theory approaches, in particular, the weights of the filter may represent the impulse response of the modelled dynamic plant; the process of learning is somewhat equivalent to the estimation of the Finite Impulse Response (FIR) of a moving average (MA) process.

As newer samples of training data become available (the agent motor commands and resulting sensory effect are continuously collected) the filters may be re-trained offline (i.e., at "sleep" state), reaching good estimates of: a) the underlying dynamical process, b) its uncertainty and c) the sensory noise.

Finally, consistent with the overall philosophy of the Dreams4Cars, we do not use a brute-force big-data approach. Instead we embed into the networks some form of a-priori knowledge about the characteristics of the plant. This "pre-wired" knowledge may be as little as specifying which are the causal input and output, which variable is the derivate of which other, or which structure the equations modelled by the network should approximately have. These ideas are discussed in paper [9]. Providing such prior knowledge greatly enhances the robustness and efficiency of the learning process without significantly diminishing the network modelling power (e.g., discovering that acceleration is the derivative of the velocity in a noisy environment would require lot of data, whereas this kind of relation can be wired into the network structure beforehand with no side effect) [8].

3.3.2 Learning from embodied simulations via Direct/Indirect Optimal control

Embodied simulations, do not necessarily need the complete vehicle dynamics simulation environment (as it is for episodic simulations). Indeed, the learned forward models may be used for simulations themselves, to try and optimize sequences of commands to achieve particular goals. Embodied simulations can thus be used without the complete OpenDS environment (and run much faster); they can be to some extent combined with simple episodic simulations of the CDZ type (section 3.1.2); for example, learning evasive manoeuvres in response the behaviour of another agent that is generated by episodic simulation.

Learning of this type is equivalent to the solution of Optimal Control problems based on the learned dynamics. Both direct and indirect Optimal Control problems are formulated and solved. Indirect problems are used on the Calculus of Variations and the Pontryagin principle. Direct problems are formulated as learning problems where a neural network learns the control input sequence that, given the context and vehicle initial state, optimizes the vehicle trajectory with the actual vehicle dynamics. Properly formulated loss functions define the optimality criteria. The result of optimal control solution (the value for potential control choices) is mapped onto the lateral/longitudinal control map ultimately learning the salience function of the dorsal stream (Figure 3).

4 Bibliographical References

[1] M. Da Lio, A. Plebe, e D. Bortoluzzi, «On Reliable Neural Network Sensorimotor Control in Autonomous Vehicles», *IEEE Trans. Intell. Transp. Syst.*, pag. submitted.

[2] R. Grush, «The emulation theory of representation: motor control, imagery, and perception.», *Behav. Brain Sci.*, vol. 27, n. 3, pagg. 377–96; discussion 396–442, giu. 2004.

[3] G. Hesslow, «The current status of the simulation theory of cognition.», *Brain Res.*, vol. 1428, pagg. 71–9, gen. 2012.

[4] V. Mnih *et al.*, «Human-level control through deep reinforcement learning», *Nature*, vol. 518, n. 7540, pagg. 529–533, feb. 2015.

[5] C. W. Baum e V. V. Veeravalli, «A sequential procedure for multihypothesis testing», *IEEE Trans. Inf. Theory*, vol. 40, n. 6, pagg. 1994–2007, nov. 1994.

[6] R. Bogacz e K. Gurney, «The basal ganglia and cortex implement optimal decision making between alternative actions.», *Neural Comput.*, vol. 19, n. 2, pagg. 442–477, feb. 2007.

[7] J. Porrill, P. Dean, e S. R. Anderson, «Adaptive filters and internal models: Multilevel description of cerebellar function», *Neural Netw.*, vol. 47, pagg. 134–149, 2013.

[8] S. James, A. Sean, e D. L. Mauro, «Longitudinal Vehicle Dynamics: A Comparison of Linear State-space, Nonlinear Physical and Neural Network Models», *Be Submitt.*

[9] M. Da Lio, D. Bortoluzzi, e R. P. Gastone Pietro, «Modeling Vehicle Dynamics with Neural Networks», *Veh. Syst. Dyn.*

[10] R. Donà, G. P. Rosati Papini, M. Da Lio, e L. Zaccarian, «On the Robustness and Stability of Hierarchical Vehicle Lateral Control with Inverse/Forward Dynamics Quasi-Cancellation», *IEEE Trans. Intell. Transp. Syst.*

[11] H. Svensson e S. Thill, «Beyond bodily anticipation: Internal simulations in social interaction», *Cogn. Syst. Res.*, vol. 40, pagg. 161–171, 2016.

[12] K. Meyer e A. Damasio, «Convergence and divergence in a neural architecture for recognition and memory», *Trends Neurosci.*, vol. 32, n. 7, pagg. 376–382, lug. 2009.

[13] M. Da Lio *et al.*, «Exploiting Dream-Like Simulation Mechanisms to Develop Safer Agents for Automated Driving», in *IEEE 20th International Conference on Intelligent Transportation Systems*, Yokohama, Japan, 16-19/10 2017 submitted.