Information Society Technologies

Project no.:	**IST-FP6-FET-16276-2**
Project full title:	**Learning to emulate perception action cycles in a driving school scenario**
Project Acronym:	**DRIVSCO**
Deliverable no:	**D7.1b**
Title of the deliverable:	**Specification of Learning Requirements (Update 1)**

Date of Delivery:	14.6. 2007
Organization name of lead contractor for this deliverable:	BCCN
Author(s):	I. Markelic, F. Wörgötter, BCCN
Participant(s):	BCCN
Work package contributing to the deliverable:	WP7
Nature:	R
Version:	2.0 (revised 10/06/2008)
Total number of pages:	21
Start date of project:	1 Feb. 2006 **Duration:** 42 months

Project Co-funded by the European Commission		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other program participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Summary: This Deliverable gives an overview of the DRIVSCO system in its different stages, training, robot operation and testing; it reviews the literature on autonomous vehicles; it treats the reviewer comments from year 2; and it provides an outline of the future objectives and tasks to be used also in the addendum to the TA for September 2007.

1 General Introduction – The DRIVSCO Concept

This deliverable is being used to provide a general introduction to the DRIVSCO project in its current state as many components come together in WP7. This general introduction will be used also for the revised TA.

Figure 1 shows the structure of the system of which many components are already operational. Three modes exist: A) Training, B) Robot Operation and C) Testing. Components which are identical are shown using the same color code. The large green block in B and C is also identical. In general, these diagrams describe a control loop, operated either by the human or the robot: Perceived sensor inputs together with an evaluation using planning will lead to an action.

During training (A) the human will perform the action. At the same time the system analyses the incoming images towards generating SVEs combined in a visual feature vector. At the moment we include curvature information as well as obstacle information into this vector (in WP7). The system also stores the performed human actions and matches the feature vector to the observed actions in the form of a perception-action mapping (Sense-Act Map). From this an interpolation is calculated for creating a dense sense-act map for data points that have not been observed.

Panel B shows the robot operation. The machine uses the incoming feature vector together with the interpolated sense-act map. These pieces are being brought together with the planner ("lookahead")[1] to create an action. Through the action of the robot its own sensor percepts will change, closing the loop.

Panel C shows how these components can be brought together in a real car arriving at a benchmarking without having to give full control to the car (which would be not allowed due to safety requirements). The human drives the car, while the system "simulates driving the car". From this there is a moment-to-moment comparison between human actions and (suggested) car actions possible and errors can be benchmarked (for details see D8.1 and D8.3). Note, in this project this is the only allowed way for system testing. It does stop short from a full closed-loop car-operation as the human will always impose a quasi steady state situation on the system, because wrong suggested car actions have no consequences on the driving!

Some short notes on the approach (more details/data will follow in the remainder of the text below):
The heuristic way of building the sense-act map has been criticized. It has also been stated that learning seems less important.

In answer to this, we note that the building of such a sense-act map is motivated by many approaches in humanoid robotics, which deal with *learning-from-demonstration* [Breazeal and Scassellati 2008; Schaal et al 2003; Dillmann 2004]. This is nowadays a prevalent learning method for human-supervised action learning and has therefore been adopted for the DRIVSCO project, too.

In humanoid robotics such a mapping needs to be achieved for a system with many degrees of freedom (robot arm and hand). In these cases plain perception-action mapping is not possible as the data- as well as the action-space has many dimensions leading to a very sparse filling of this space when using a naïve mapping approach even after prolonged experimentation. This as well as the more complex target domain of a humanoid robot (3D instead of 2D, for driving) requires generalization methods and the building of invariances for transposing actions taking learning-from-demonstration into the realm of difficult machine learning methods.

[1] The term "lookahead" is going to be used interchangeably with the word "proactive", meaning to equip the system with the ability to react to upcoming events.

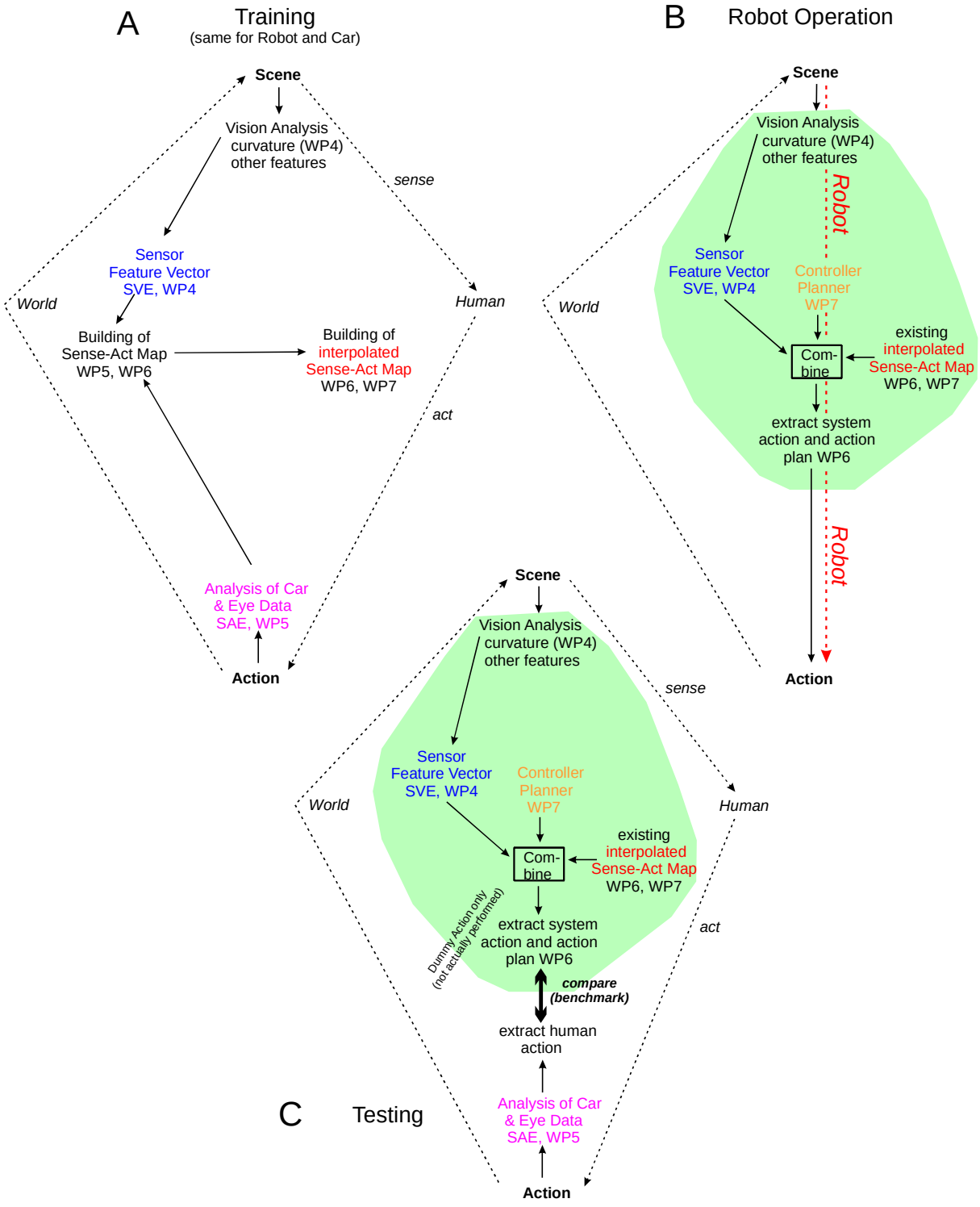

Figure 1 DRIVSCO System

In our driving scenario, things are much different. We have to deal with only 2 degrees of freedom at the output consisting of steering, and velocity. This allows densely filling the sense-act map (in the lab after about 10 rounds). Due to the small dimensionality of the target domain (2D) there is no need for more sophisticate generalization methods – at least not for the problem of road following (see below for agent architecture, etc). This may have raised the impression that the adopted method appears simplistic and heuristic. On the contrary, this method just benefits from the simple perception-action space (at least on plain roads and road following, see below for some results on real road perception-action mapping – more will be presented in September). Furthermore, it also fulfills the real-time

requirement! Below we will show results where much more sophisticated methods, that we had tested (mainly in year 1), *that show that those methods did not perform any better*. So, there has been a fair amount of work spent on the problem that at the end could not be used on the robot. Failure to report these negative results may have led to the impression of having used a simplistic, heuristic approach in an unjustified way.

2 Revision D7.1b

D7.1b has been criticized and rejected. This has largely followed with criticism of WP7 and we will use the revision of D7.1b for treating both. Hence, in the following we will carefully address all the criticisms made by the reviewers. To facilitate the reading the criticisms or comments made by the review committee are set in Courier.

The structure of this document is as follows: We start with stating the general goals of the workpackage (Section 2.1, Introduction), then we summarize previous work done in the field of automatic driving (Section 2.2, Previous Work), shortly listing their successes and shortcomings and based on that motivate our own approach (Section 2.3, Where Do We Place Ourselves With Respect to Others). We will list what work has been done in the course of the DRIVSCO project, and how it relates to the stated goals (Section 2.4, Work Done).

We will give a detailed system architecture outline (System Architecture) and finally we will present what future work we intend to do and how we plan to realize it (Future Plans, Expected Results).

2.1 Introduction

In general the objective of WP7 is to learn links between perception and action to achieve a system that is capable of proactive behaviour. We define proactive behaviour as actions that are aimed at future events. For safe driving for example, the system should be able to react to upcoming events, that will effect the own driving situation. There are two possible ways to achieve that. One would be an "extended sensor", which leads to a proactiveness by implicitly incorporating this future information into an action signal, and the other possibility is to generate explicit plans, which are sequences of actions. The first approach improves the system behaviour, yet it is limited, since only one action signal per timestep, or very few action signals, are elicited. The second approach overcomes this shortcoming. There are several benefits in endowing a system with such a "planner"[2] in addition to the capability to react to future events.

- It enables the system to cope with periods of lacking sensory input, imposing some kind of open loop control (e.g, while loosing track of the lane during an obstacle avoidance action.
- It can relieve the system from continuous sensory information processing saving these resources for other computations.
- It can help to make the system more robust/adaptive, since the action plans can in principle be used to make state predictions. These state predictions can be compared to the really encountered states and the conducted actions can be corrected accordingly if necessary.
- Typical sequences that reoccur often can be used for situation recognition.

We first looked at the initially proposed ISO learning method. It is a modern sequence learning method that was successfully applied to numerous problems [Porr et al., 2003; Trancoso, 2007; Porr, Wörgötter, 2006] and therefore seemed promising for the problem at hand. However, it was found that ISO cannot be so easily adapted to a *supervised* scenario, such as aimed at in DRIVSCO. In addition ISO would be more an example of an implicitly proactive system, as described above. Before tailoring this method to the problem, which of course would also be an option, we decided to analyze what would be the best way to tackle the problem in the first place, not restricting ourselves to a fixed course of action. Thus, all effort in this workpackage has been aimed at *supervised predictive sequence learning*. It might be, that we failed to make this clearer before, which gave rise to the impression that the course of "WP7 has shifted […] dramatically with respect to what was initially proposed", or that

[2] The word "planner" in this context is not be interpreted in the classical AI sense that usually refers to high level planning systems such as STRIPS, but rather denotes a system that can generate a sequence of short term actions.

“the learning from human driving seems to have become secondary.” But we will address this critique also further in the course of the text.

2.2 Previous work

Previous work in automated vehicle control started at the end of the 70ies. Since then heavily funded long term programs, e.g. the 800 million ECU EUREKA Prometheus project, the NavLab project from Carnegie Mellon University, CMU, the ARGO project [www.argo.ce.unipr.it/ARGO] from Universita di Parma in Italy, or the DARPA challenge [http://www.darpa.mil/grandchallenge/index.asp] funded by the US government, propelled advances in the field. Impressive results were demonstrated in 1995 by a 1600 km trip by the VaMP test vehicle (Vehicle for autonomous Mobility and Computer Vision) from Universität der Bundeswehr Munich, UBM, from Munich in Germany to Odense in Denmark and back, with speeds up to 175 km/h with only 5% human intervention. Lateral and longitudinal control were achieved based on vision sensors and “engineering methods” (comp. paragraph Physical Models).
In the same year NavLab from CMU achieved 98.2% autonomous driving on a 5000 km trip, called "No hands across America". It used neural networks for lateral control. Longitudinal, i.e. velocity control was not included, but handled by a human operator.
The ARGO project proved its potential in the “MilleMiglia in Automatico”, a journey of 2000 km over six days on the motorways of northern Italy. It achieved an average speed of 90 km/h. and the car was in automatic mode 94% of the time.

The above successes were based on the usage of very different methods and mixtures of them. One way of differentiating between them is to measure the amount of required a priori knowledge, compare Figure 2. On one extreme are the heavily model based methods, that employ either models of the physical environment, or models of the human controller and on the other extreme are all methods that try to overcome this, usually by making use of machine learning techniques.

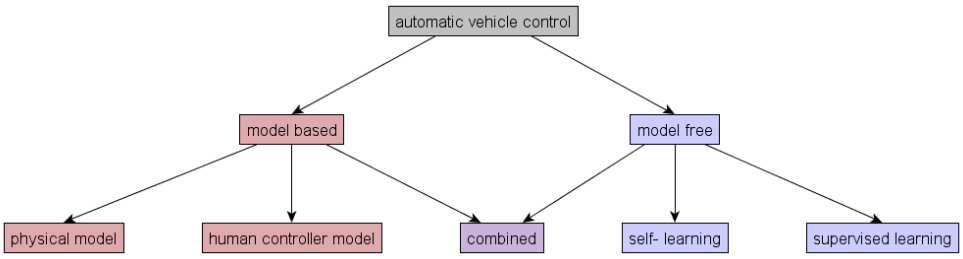

Figure 2 Visualizing the different strategies used in automatic vehicle control. The model based methods are based on a high amount of a priori knowledge (bias), whereas the model free methods try to minimize it.

In the following we will give a short overview of the different strategies and discriminate them against each other, listing their pros and cons and finally placing ourselves.

2.2.1 Model based approaches

There are two different kinds of model based approaches. One is based on physical models and the other on human models.

Physical Models
For automatic vehicle control a typical “engineering method” is to build a model of the system to be controlled and its environment. The *exact* knowledge on the parameters that determine the system dynamics results in very elegant and minimalist mathematical control laws. The main challenge is then estimating the required parameters from the sensory input. This approach has been heavily used by the group around Ernst Dieter Dickmanns. A famous example is the 4D approach to dynamic vision, where the use of spatio-temporal motion models is used as the basis for recursive state estimation [Dickmanns, 1994].

- **Pro**
    a) When all models are known a very good level of control can be achieved, grounded in the well studied and powerful Newtonian Mechanics. For examples of this strategy see [Dickmanns, 1994; 1990]
- **Con**
    a) The more complex the scene the more models and their interactions must be known. This is often not realistic. For example one would need to have a model of a truck, of a cyclist, of the road, and evtl. how all these models change with changing weather conditions. One cannot possibly consider all contingencies. It would be preferred that the system could acquire some knowledge by itself.
    b) Even if all models were known they have to be preprogrammed into the system, thus, a lot of effort is required by the system designer.

Modelling the human controller
Achieving a model of the human controller, instead of modeling the rigid objects of the environment is another example of a model based approach. The goal is to understand the human control by modelling it. The major challenge here is to arrive at control laws. Usually studies are done in simulation and therefore the necessary cues from the visual input are known. In the area of driving there is research on a) curve negotiation [Boer, 1996], b) lane changing [Salvucci & Liu, 2002] c) car following [Boer, 1999]. Most of these works are done using simulations.
Similar advantages and disadvantages as in the rigid body modeling approach apply. That is:

- **Pro**
    a) For certain well studied and verified models a very good level of approach can be achieved, when the necessary cues can be provided.
- **Con**
    a) For real applications it can be difficult to obtain good feature estimates, e.g. the interesting feature is not in the camera image.
    b) All models must be preprogrammed into the system, thus the more complex the environment the more effort is required by the system designer.

2.2.2 Model free approaches

To remedy the shortcomings of model based approaches, which is basically the necessity to provide a lot of a priori knowledge, or bias, machine learning techniques are usually employed. They can be distinguished into 1) heuristic methods and 2) inductive/knowledge extraction methods.

Heuristic Methods
Heuristic methods have shown to make good use of existing knowledge for building simple and robust controllers. They are obtained by looking at the response curve of the system output according to a certain input signal and have proven to be useful in various applications, for car driving one of the most prominent examples is ALVINN [Pomerleau, 1989]. Other examples applied to the problem of automated vehicle control were for example: Artificial Neural Networks (ANN) [Tahirovic, 2005], Pattern Matching [Krödel & Kuhnert, 2001], Artificial Evolution [Agapitos, 2007], Reinforcement Learning (RL) [Riedmiller, 2007], and Clustering Based Techniques [Vasquez et al., 2008]. All these methods have a in common that they map an output signal to an identified state.

- **Pro**
    a) A big advantage of these methods is that the relation between input cues and control output is handled automatically. Neural Networks are especially intriguing, since they can handle a great number of inputs and select implicitly the relevant ones.
    b) They generalize to a certain extend (but comp. item b) in the “Con” part) and can be robust to noise.
    c) They are well suited to realize nonlinear input-output relations.
    d) They are usually very fast.
- **Con**
    a) Concerning Neural Networks it is not easy for the human to identify which features the techniques discovered to be meaningful and what impact they have (“opaque-box

problem"). In the case of the ALVINN approach this lead to additional work aiming at solving this problem. [Pomerleau, Touretzky, 1993].

b) When learning from observation a fundamental problem is that often only positive examples can be provided. The system consequently cannot know what to do, when encountering a situation that is "bad", since it is very different from the training examples. This is a general problem of all techniques that realize a mapping between input and output signals.

c) It is difficult to include predictions of sequences, more suited for realizing closed perception-action loops.

2.2.3 Inductive/Knowledge extraction methods

Knowledge acquisition methods applied were: Case Based Reasoning [Weng & Chen, 1995], Fuzzy Logic [Salila & Lezy, 1994], and finally combinations of inductive and heuristic methods, Neuro-Fuzzy [Rusu et al., 2003].

- **Pro**
  a) The advantages of the knowledge extraction methods are that they can make hidden relationships overt and as such help to analyze the underlying control strategy. This is a very interesting property and has been applied in e.g. [Shang & Wang, 2006].
- **Con**:
  a) Traditionally such methods are used and suited in higher level knowledge, e.g. symbolic knowledge tasks. There is little research in using them for other applications.

2.2.4 Combined Approaches

The combination of model based and model free approaches is probably the most promising strategy, as some aspects are probably best described by using models, for example for motion description one can advantageously use the rigid body motion formalism. On the other hand machine learning methods can help to equip the system with more adaptability that then in return alleviates the burden of having everything to be preprogrammed into the system. A good example of a successful application is EMS-Vision (Expectation Based Multi-Focal Saccadic Vision) [Gregor et al., 2002]. This is a modular (object oriented and agent based) system extending the 4D approach with machine learning applications.

2.3 Where Do We Place Ourselves With Respect To Others

```
The selected approach should be well positioned with respect to
the state-of-the-art, including sufficient discrimination to
ALVINNS's and Dickmann's technologies for autonomous driving.
```

As seen from the above discussion, despite the amount of work that has been done in the area of automatic vehicle control and the impressive results achieved, still many problems remain unsolved. Probably the combination of both (model-based and model-free) approaches is the best strategy. But in the course of this project we focus on the model-free approaches, because: "In the long run, even autonomous learning seems to be essential since not all knowledge should have to be programmed into the system by human developers." [Dickmanns, 2002] We believe that adding a predicitive component to machine learning techniques, such that sequences of actions/states can be generated, will improve their performance as we motivated in the beginning of this document. Therefore, in DRIVSCO our goal is to build a driving system using machine learning by equipping it with lookahead, thus it can react to upcoming events. In the case of street following this would mean that it would adapt its momentary (local or reactive) steering actions towards the upcoming road trajectory, or concerning speed it would mean, that it would slow down (in a similar distance and similar way as the supervisor,) before it reaches a sharp turn. In particular we are investigating how lookahead information from the visual sensory system can be used. We want to analyze how we can use which technique for this purpose and we want to learn the necessary skills from a human supervisor. Therefore, we differ much

from the usually engineering based methods fostered by the group around E. Dickmanns[3]. We differ from other machine learning methods because we explicitly focus on adding lookahead and aim at achieving a system that generates sequences, and not only one action command per timestep, as motivated in the beginning of this document in the Introduction. That also distinguishes us from the famous ALVINN approach [Pomerleau, 1989]. In addition ALVINN was trained on synthesized images, and was used for controlling steering and not speed.

A long term goal (beyond the limits of this project) would be to arrive at a system that can interpret its current situation, because "Understanding of what is happening around oneself and what may be future actions of other subjects involved in the situation given, is a prerequisite for a desirable defensive style of driving" (E. Dickmanns in [Dickmanns 2002]).

But before such an ambitious goal can be achieved there is need to solve basic tasks. It has been found that the mentioned lack/difficulty of including lookahead information already shows in the most basic skills that such a system should be endowed with. That is following a given trajectory at a reasonable (in our case operator-like) speed. Lateral control when driving in a structured environment is relatively easy, since it can partly be treated like a reactive control loop, and the system already shows good behaviour when only relying on that strategy. Note, however, that there is strong indication that lateral control is the result of a two level control, where the basic sensory-action loop is modified by another "lookahead"- process [Donges, 1978]. In contrast, longitudinal control *requires* to take future states, e.g. a sharp turn ahead, into account. Humans do that easily. To our knowledge nobody has equipped a robot/vehicle with the ability of lateral AND longitudinal control using machine learning techniques based on visual sensory input. Research was mainly limited to simulations, or lateral, or longitudinal control, but not both, this has also been reported by [Partouche et al., 2007].

2.4 Work done

```
In year 1, a reactive controller for lane following has been
proposed. It is a simple sensory-motor mapping for a small
mobile robot that is learned from human driving.
```

Based on the premise that steering control is twofold (consisting of a reactive and a lookahead component) we started with the simpler task of learning reactive steering from human observation. For that we used an ANN (a simple multilayer perceptron trained using back-propagation), since it had proven to perform well for such applications. However, we did not use the entire image as input leaving the feature selection/extraction task to the method, but we treated the street as an obvious SVE and extracted it beforehand using conventional computer vision techniques (feature extraction). We used the right lane and from that we calculated several cues that showed relatively high correlations each (feature selection) and tested them as input parameters for the network. This way we made a compromise between not putting into the system how the incoming signals should be processed (that was learned from observing the human) but at the cost, that we solved the feature selection task for the system

The performance on training data is shown in Figure 3. It tends to oversmooth, which in curves sometimes leads to a situation where the robot gets too close to the border of the track and then cannot get back to it fast enough, which results in loosing it. It might be that this problem does not occur on wide streets that are not heavily curved, but it posed a problem to our robot, that must drive on a strongly curved and narrow track. When trying to improve performance we first tried to reduce the fitting error. We divided the steering problem into smaller problems, e.g. we trained a network only on right turns, hoping to better capture the human steering, and thus avoiding the problem of loosing the track, but as can be seen from Figure 4 it was not successful. Furthermore, we applied a more sophisticated kind of network, in this case a Radial Basis Function network. The result is shown in Figure 5. As can be seen from the image the network was overtrained. When tuning the radial basis parameter we could not get good performance either.

To learn more about the reasons for that bad performance we decided to analyze the input data further. For that purpose we constructed a matrix of the two most influential parameters against the steering

[3] Although recent publications show that this group is now bridging previous work with machine learning methods. For example in the EMS-project.

values from the human. Because of its low dimensionality and the finite values that the parameters could take it was interpretable and we found the reason why the robot was loosing track. It was due to the problem of training only on positive examples, as explained above, and which was also treated in the ALVINN approach, as explained in [Pomerleau, 1989]. The network smoothes the data and from that results a slight understeering. As such that is not problematic, but a consequence is, that the robot gets too far towards the lane boundary and from there it cannot return because these situations are critical and do not need a smooth action but fast corrective steering. When the robot once gets into this situation it has not seen before, its generated action is too low in intensity and what follows almost immediately is that the robot looses the street in its field of view. The robot cannot return, because the ANN as a reactive controller does not contain any memory.[4] In ALVINN this problem was solved by synthesizing different views from one image using image projections, and accordingly altering the steering commands. Thus, for each incoming image several training examples were produced.
In the case of the matrix (where we plotted the human steering values against the two most influential parameters) however, the solution could be acquired much simpler. It was easy to fill the uncovered state space with meaningful values. In fact it only required filling in the four extreme values in the edges of the matrix and applying a linear interpolation between the values. Of course, that could only be done, because a linear relationship between the in- and output parameters could be observed. This simple controller now works very well and its performance is shown in Figure 6. The problem of loosing track was solved! It can be seen that it is not as smooth as the ANN output, which makes a signal filtering necessary (not applied to the data shown in Figure 6), that introduces a slight delay.[5]
In this case the problem could be solved because we understood – at least roughly – the relation between in- and output data. In addition we could read it from the image when we plotted the matrix, which was only possible because of its low dimensionality. This points to a bias/variance dilemma that can be frequently found in machine learning e.g. [Weng & Chen, 2001]. Interesting would be to investigate how this could be generalized to the case of higher dimensionality. Since we are currently investigating how we can introduce speed in our system this is part of current and future work.
Summing this up, we want to enumerate the advantages of the controller:

- It works in real time.
- It has been learned from the human.
- It covers the complete state space, even if that had not been part of the training data.
- It is easy to adjust, because it is understandable.
- It can be used with a single uncalibrated single camera.

The disadvantages:

- It is not clear yet how to modify it to incorporate more input, such as speed.

Please, also note that the matrix is equivalent to the notion of a policy from Reinforcement Learning.

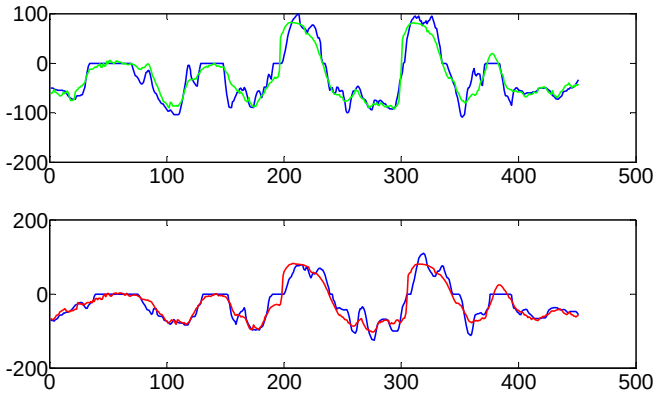

Figure 3 shows the result of generating steering data (in green and red) with a neural network, for two different test set (in blue).

[4] And even if it could return, such behaviour would be far from desirable.
[5] This delay can be eliminated by generating sequences of actions and then filtering forward. This work has been done, too. And therefore the work on learning a two-level steering agent for fixed speed has been completed.

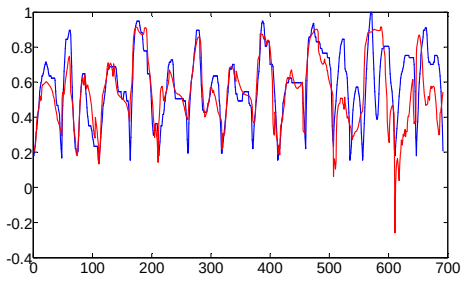

Figure 4 Generated steering data of an ANN trained on right turns only. Blue is the data generated by the human and red is the network data. The training set was based on the first 500 entries. It can be seen that the network performs badly on the test set.

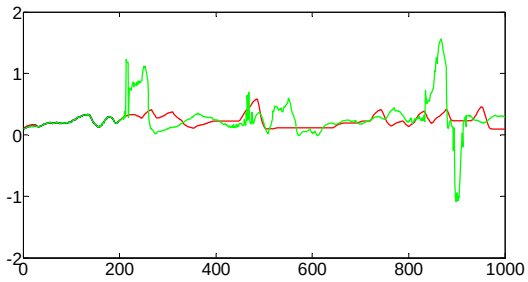

Figure 5 Radial Basis Function Network applied to steering data. The training set comprises entries from 1 to 200. The output from the network is plotted in green and the data from the human in red. It can be seen that the network is extremely overtrained.

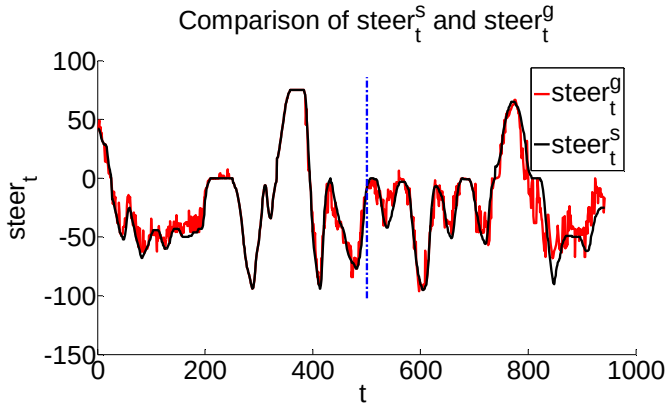

Figure 6 The black line shows the steer signal from the human driver over time and the red is the synthesized signal for the same image sequence. The data before the dashed line (from left to right) shows the performance on the training data and after it on a test data set

> These techniques […] may work well for the actual slow-moving robot, but it is not clear how the controller obtained could be used for a regular car. It is unfortunate that no attempt was made to test the approach on the dataset obtained from the test car.

It is indeed important to test the algorithms on the real car. We have started with this work and will now show first results. The right lane marker was extracted similar to the robot scenario and angles α and β were calculated as shown in Figure 7. The correlation between these angles and the human driving signal were investigated at various distances (measured in vertical pixels) forward along the road. For the case of angle α, 150 pixels in vertical direction were always kept between the red point and the green point, when moving forward along the line. For the angle β the distance was 70 pixels, as it was observed that smaller distances between the two points give better fits to driver's steering signal for this angle. Also a third angle γ was introduced, which is similar to β, but the red point always remains at the beginning of the line, while the green point moves along the line. As the whole view has a considerable up and down motion component, a correction was made to place the line start point (bottom right) always 100 pixels above the bottom of the image.

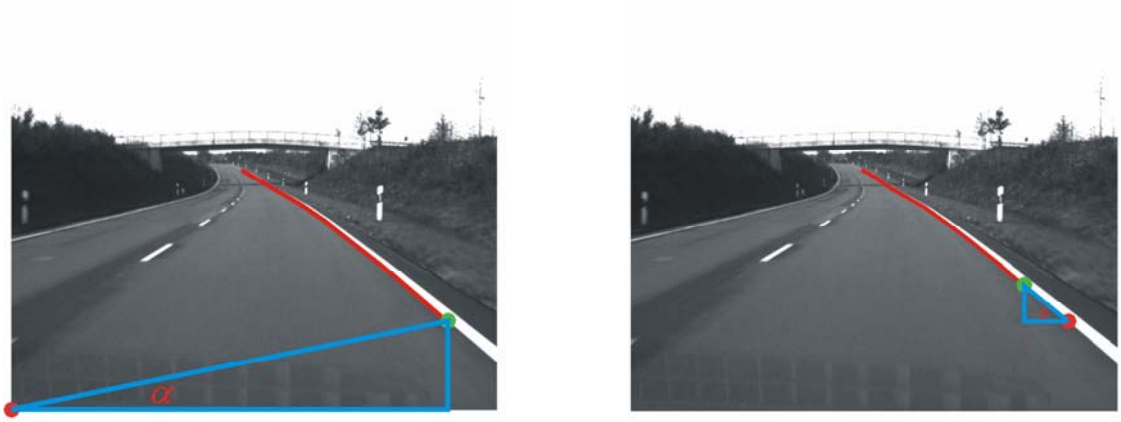

Figure 7 shows the extracted right lane (in red) from the street images and the angles α and β, used for correlating with driver's actions.

The results for the angles measured 200 and 250 pixels from the bottom of the image are given in Figure 8. The red curve is for the angle α, magenta is for β, and green is for γ. The steering signal is plotted in blue. One can see that results for 200 and 250 pixels are similar, as well as all three angles (normalized amplitude) produce similar results. Although the curve descriptors are noisy, one can observe a reasonable correlation between those descriptors and the steering signals. As this is first data, quantifications of the correlations have not yet been performed, but are on the way. Hence, this work is currently under progress.

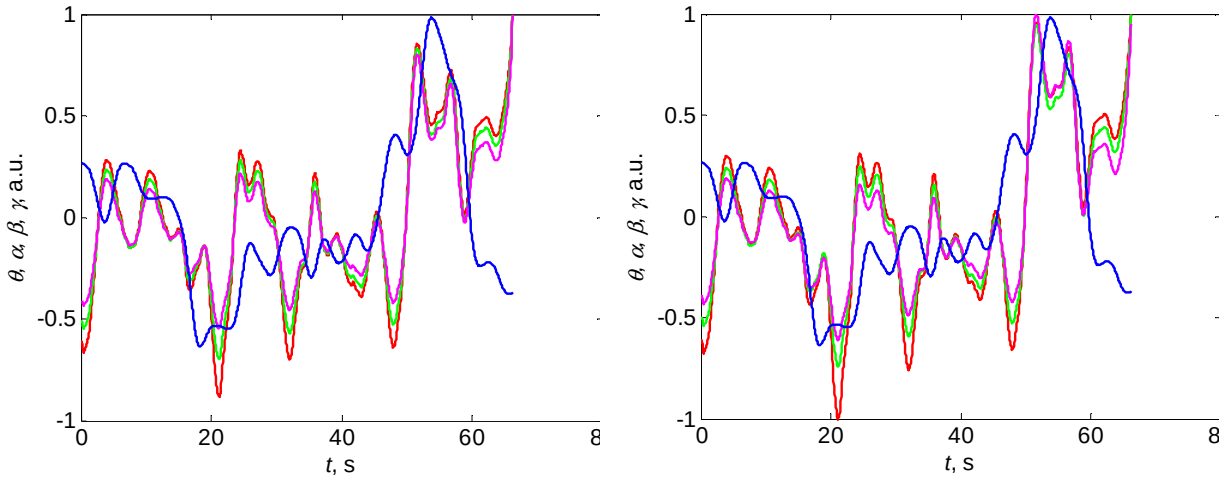

Figure 8 Line descriptors: angle α (red curve), angle β (magenta curve) and angle γ (green curve), and the steering signal (blue) with the amplitude normalized to one. The left panel is for 200 pixels up the picture, and the right panel is for the further view, 250 pixels up the image.

To equip the simple reactive controller with more lookahead based on the data from the supervisor, we proposed to build a database and use it as memory for the system. Incoming robot- or car-images can be compared to the entries of the database and thus to the actions elicited by the supervisor when exposed to a similar situation. In WP6 and 7 there are efforts under way that use the same approach for SVE-SAE linking in order to generate warning signals for the driver, based on the car data, as depicted in Figure 9. For that a number of sequences have been recorded by UMU and SDU and are being used for establishing that database and a test set. For more information please see WP6 in the PAR.

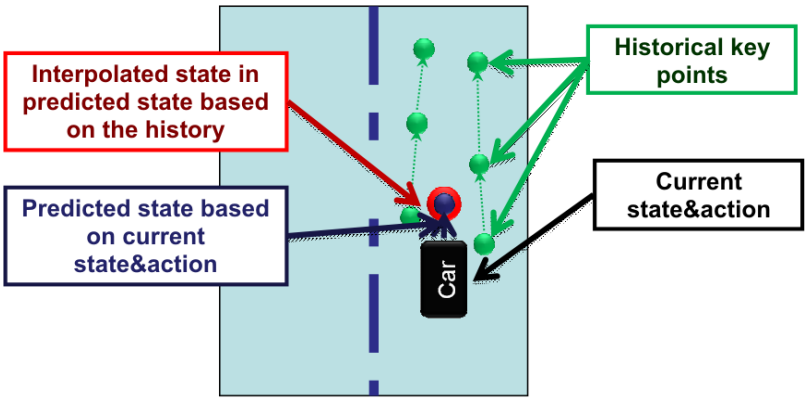

Figure 9 A warning signal is generated when the predicted state (including actions) deviates from the current state (also including actions).

> Year 2 has been dedicated to improving this simple controller. Different techniques have been tested in order to provide the controller with more lookahead.

Our next goal was to construct the second control level for steering, which should modify the short-term steering command by anticipating future situations. That was supposed to be done using visual sensory information and to achieve at a control sequence. This has been successfully finished for fixed speed (see PAR2 description of WP7). The next step, as also explained in the "Future Work" paragraph is to augment the controller using speed. This work is in progress.

> It is unclear why WP7 does not use the output of WPs4, 5 and 6 (SVEs and SAEs)

WP7 needs to deal with two aspects: A) Real-time robot control and learning from humans and B) Transfer of learning methods to the car scenario. At the beginning of the project no output from the other work packages was available, because the project had just started. Thus, effort had to be spent on data pre-processing, e.g. creating a real-time lane extraction method.
Concerning A (robot control): At the current stage of the project there is not yet real-time capable output from the main scientific work on WP 4,5, and 6 accessible for the robot set-up and robot control has to deal with this (in contrast for the car set up real-time information is already available via the HELLA lane detection system). We note, however, that the methods developed for the robot are capturing similar SVEs as those described in the PAR from WP4,6, albeit in a simpler and less complete and less robust way. Hence, WP7 and WP4,6 are rather tightly linked and discussion between the main involved groups (SDU, BCCN, VMU and KUL) are taking place to coordinate these efforts. Obviously, we have failed to make this clear in our earlier writings and failed to provide clear explanations as to this.

Concerning B (transfer to car scenario): We have now reached a state were the more complex SVE algorithms are now stable and robust (WP4,6, IMOs, tangent point, time-to-contact) and can be linked to the drivers actions (WP5) such that perception action linking has become possible (WP6), we expect that the lookahead methods introduced in WP7 can be used with some modifications on car-data, too. Real time aspects are also increasingly becoming available due to the successful work in WP1.

2.5 System Architecture

> It has to be clarified in detail how individual agents (including "street following", obstacle avoidance") and associated behaviours will be integrated into an overall consistent system behaviour that will generate supportive actions in the context of driver assistance, for instance applying existing approaches in behaviour based robotics.

In the following paragraph we will first motivate a general architecture that we then specify in detail in the subsequent paragraphs.

5.2.1 General Architecture

There has been much research on efficient robot architectures and their implementations. Starting with deliberative "sense-plan-act" systems [Chatila & Laumand, 1985] that turned out to be too slow for many real world performances to reactive "sense-act" systems [Brooks, 1985] that perform well in dynamic environments but lack the possibility to store information and plan longer strategies. The most common approach nowadays is a combination of both referred to as "hybrid architecture" [Donges, 1998], which brings together the advantages of reactive and planned behavior. However, the cost for that is the difficulty to design the interplay between processes that work on different time scales. Usually such architectures rely on several processes on different hierarchies that handle one more or less complex task each and that can communicate with each other. These processes have been given various names in the literature e.g. skills, situated modules, agents, or behaviors. Following the divide and conquer principle has proven to be useful for many years and therefore we adopt it, too. Note that the architecture does not define how to arrive at the various agents, they can be realized by hardcoding them, using machine learning techniques or realizing appropriate models for solving a certain task. In our case we call them agents and learn them from a human supervisor.

> The agent-based approach has been extensively researched on in the past. The partners do not seem to be aware of all the research done on this topic over the past 20 years, and it is not clear what the chosen direction will yield.

Our goal is to link perception and action in a way that allows for proactiveness and that we learn from a human supervisor. Therefore, we investigate how this can be realized. Using an agent based approach seems suitable *because* so much research has been done on it. It is a well known approach that we seek to *use* for achieving the stated goals. (See earlier reports and D 7.1a)

As a consequence of adopting this modular design the question of interprocess-communication (IPC) and synchronization must be clarified. Possibilities are synchronous (send-receive) or asynchronous (e.g. publish-subscribe) communication. In the latter a global timing mechanism is necessary. Since we would like all processes to work concurrently for greatest flexibility we decide for an asynchronous communication protocol that is to be realized on a real time system.

Software: Another important issue in designing robot system architectures is *reusability*. In the beginning of robotics developers built specialized systems for every robot. With the increasing number of robots, that often shared similarities, a need for reusable software arose. Such software should offer blocks general enough to be applicable to various platforms, but also specializable enough to be adaptable to a particular system. Using such software can speed up the realization of a system architecture and also the time that developers need to familiarize with a certain control software. Examples for such software systems supporting hybrid architectures are: CLARATy [Nesnas, 2001], YARP (Yet Another Robot Platform) [Metta et al., 2006], MCA (Modular Controller Architecture) [www.mca2.org], and OROCOS (Open Robot Control Software) [www.orocos.org]. There are more software systems, but we restricted ourselves to these, since they are new and actively used. We are considering using the OROCOS framework since:

1) it is under GPL (gnu public license)
2) is actively used
3) **offers explicit real time support**
4) supports modular design
5) the documentation is available
6) supports distributed computing

Disadvantages are:

1) only partly usable between different operating systems
2) communication based on and restricted to CORBA (Common Object Request Broker Architecture, short CORBA. A specification for an objectoriented middleware)

As indicated by bold letters, item three was found to be the most beneficial feature of OROCOS. The Real-Time Toolkit (RTT) middleware facilitates building real-time software.
We decided against CLARATy since it is not entirely free software and against MCA due to little documentation. YARP is not restricted to a certain operating system. Unfortunately it only supports soft real time constraints by focusing on distributed computing. This only pays off when the system to be realized is highly parallelizable.

5.2.2 Concrete System Architecture

In this paragraph we will motivate the following features of the proposed architecture:

- modular (agent based)
    a) asynchronous communication
    b) real time capability
- agents learned from human supervisor
- explicit feedback loop/efferent copy for realizing proactiveness
- maintenance of a belief/world model (for planning) in the time domain

The usage of an agent based approach was already motivated above. According to this we will now further specify the system. As shown in Figure 10 the architecture consists of entities with information flow from sensors to actuators. In addition to this usual unidirectional flow, we use a feedback mechanism which resembles the expectations of the system which can be used to make predictions in the world model and thus to reduce uncertainty in the results of the feature selection process. All boxes again contain modules that belong to the same functional class, e.g. several feature extraction algorithms. Some modules are already used and printed in black letters in the figure, whereas others not yet used or realized but could easily be added are printed in grey. The shown architecture is being implemented in an object oriented manner, which is suitable for realizing a modular approach. All boxes can be thought of as classes and the modules as instances derived (inheriting) from the class they belong to. They are being realized as threads working concurrently at their own pace.
Please note, that such a setup is state-of-the-art and not a special development from our side. It is general and augmentable enough to be implemented on a variety of systems (in our case robot and possibly car) and modules can easily be added or exchanged.
In the following we are defining the interfaces of particular modules:

Sensor Interface: The sensor interface is a process that continuously receives data from a sensor, timestamps them and puts them in a global data buffer. Therefore, the interface is the requested sensor data, e.g. image frames in case of the camera sensor interface, and output is the timestamped data frame.

Preprocessing / Feature Extraction:
In the robot case all feature extraction modules take camera frames as input. Therefore they consult the global data buffer for the camera images and grab the newest frame. After extracting the features they are designed for they write the feature values to another global data buffer. Please note, that the car is equipped with more sensors e.g. LIDAR. Although this is not yet considered here, the proposed system architecture allows a fast incorporation for further sensor modules.

State Belief / World Model:
The world model maintains two data objects containing information concerning the state of the world (environment) and the state about the system (self). In contrast to the other classes it has the ability to track knowledge in space and time. To clarify what is meant by that we give an example concerning the process containing the “self” object: It receives data from the feature extraction modules, e.g. about the systems position with respect to the street. At every instant in time it memorizes this information (forgetting very old entries), resulting in a history. It also receives the actions of the

system from the feedback loop, thus it can also build an action memory. Based on these it can realize two things. First it can evaluate incoming feature values that are always prone to noise and uncertainty and second it can make predictions about a) future actions and b) future states.
The interface to this class thus is as follows: inputs are a) extracted (image) features and b) conducted actions. Output is the updated data objects.

Higher Level System / Decision Maker:
Consulting the state belief (the data objects) of the system this module decides which of the available skills should be activated. E.g. in case of collision course with an independently moving object (IMO) it would activate the obstacle avoidance module. This is a higher level decision in the sense that for realizing obstacle avoidance the trajectory follower is necessary. But this is then handled automatically by the agent in charge, in this case the obstacle avoider. It can also schedule interrupts to quit execution of a certain agent, when preconditions for this agent are suddenly not given anymore, e.g. due to a course correction of an IMO.

Skills / Agents / Behaviors:
The activated agent obtains its input from the world model. Note that every agent knows what input is needed for its special task. It either processes this information in a direct way, or it can employ other agents for that. In each case the outcome are action signals, in our case steering and velocity commands.
Since the system is designed to work on different timescales, also the action commands are on different timescales, e.g. a reactive steering command in comparison to a predicted sequence of steering commands. Therefore, a fusion unit is necessary the task of which is to optimally combine the generated actions. The calculated control commands are again written to some dedicated buffer and time-stamped.

Act / Motor Interface:
Analogous to the sensor interface the motor interface takes the newest control commands from the particular buffer and sends them to the motors. However, in contrast to the sensor interface, it **must** do this in time intervals of fixed duration![6] In addition to this it continuously checks if the generated action commands are incoming fast enough by comparing elapsed time between two consecutively used action commands. If not, then this it is a hint that the feature extraction layer might work too slow, or that the system is moving too fast. In any case it must notify the world model about its health state, possibly resulting in a lower driving speed of the system.

[6] Note, that this is a very important constraint, since it can seriously alter the behaviour of the system if not considered! This is one of the reasons why we are going to use a dedicated real time scheduler.

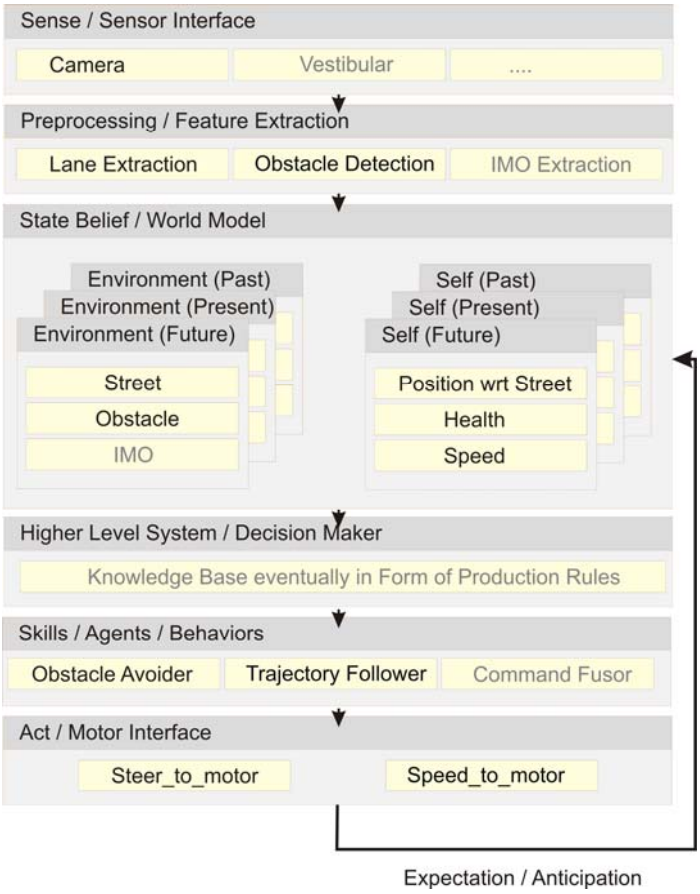

Figure 10 The proposed Robot/System Architecture. In addition to the conventional unidirectional information flow we add a feedback loop to enable proactiveness of the system. Boxes within boxes (e.g. Camera) are referred to as modules. Modules already incorporated in the current system are written in black, and those not available but eventually useful to include are written in grey.

5.2.3 Technical Realization

Our system is based on a Linux 2.6 kernel, that can be patched to achieve real time performance[7]. Several patches are available, for OROCOS the real time application interface (RTAI) [www.rtai.org] is required. We have applied it and installed the OROCOS library.
The communication between components[8] in OROCOS is realized by three mechanisms:

1) *push* for asynchronous communication, a component pushes data to others that subscribe
2) *query* for requesting data from another component
3) *send* one-way communication to send data to a server

Data is communicated in form of CORBA data objects. In Figure 11 we show the planned collaboration between modules of our architecture, by indicating which data objects are contained in the particular module and how they travel within the architecture.

[7] Note that real time performance mainly means that processes are carried out in a defined time. Not that they are necessarily fast. That of course depends on the used hardware and implemented algorithms.
[8] The term „component“ is used, because it is a basic concept in OROCOS. It is equivalent to the way we used the term “module”.

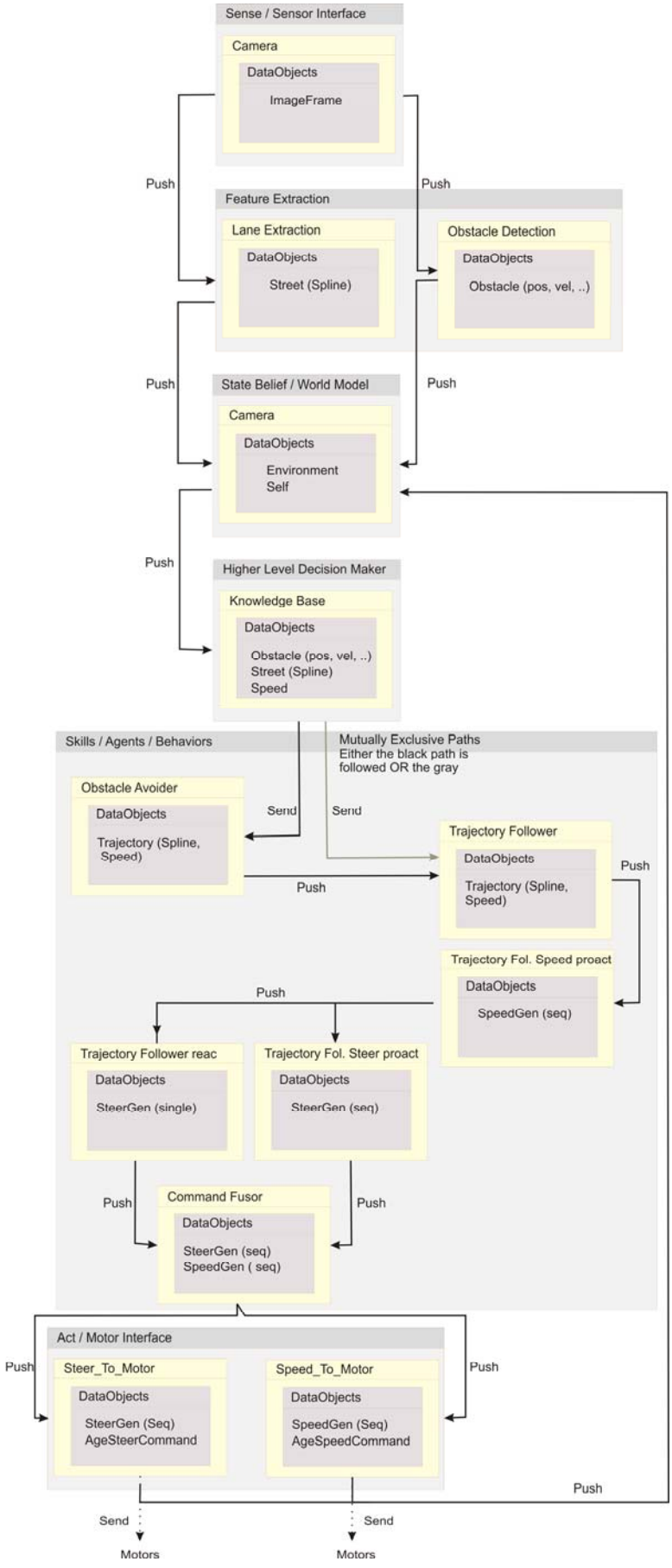

Figure 11 shows the interaction between modules of the architecture, akin to an UML collaboration diagram.

2.6 Future Plans

It is of utmost importance that the objectives, task and deliverables of WP7 be updated. The objectives must be clearly defined in a principled manner.
After abandoning ISO, the objectives, task and deliverables must be stated.

As can be inferred from above, there are six major concerns in order to achieve a proactive driving system:

1) Port to a real-time architecture using dedicated software.
2) Proceed/improve learning linking between SVEs and SAEs.
3) Improve the predictive capacities (lookahead).
4) Include more results from WP4, 5, 6 in aspects 2 and 3 above.#
5) Include night vision
6) Test the system structure on the car data

6.2.1 Objectives

Concerning 1, (specified in Task 7.1)

- Create the appropriate behavior of the system due to motor commands scheduled at the correct time intervals.
- Include multiple agents working concurrently at individual pace in real time.
- Increase reliability of system by eliminating dependencies between processes from different agents.

Concerning 2, (specified in Task 7.2)

- Finish/Elaborate the developed agents
    a) Improve the speed generating agent.
    b) Augment the steer generating agent to consider speed.
    c) Elaborate the obstacle avoidance agent.

Concerning 3, (specified in Task 7.3)

- Include predictive components in the driving as already started.
- Investigate the interdependency between current situation (and its speed) and the future road trajectory.
- Implement a mechanism which uses the future situation from image analysis to select the appropriate driving pattern from the learned one (human).
- Possibly include the aspect of predicting upcoming SVEs (hence in the image domain).

Concerning 4, (specified in Task 7.4)

- Results from WP 4, 5, 6 need to be included – if possible – in the robot system.
- SVEs need to be synchronized between car and robot applications.
- SVE-SAE mapping and prediction methods need to be synchronized between car and robot applications.

Concerning 5, (specified in Task 7.5)

- Night vision needs to be analyzed in the car and the robot scenario (note this is essentially done and there are no forbidding differences between day and night imagery. Hence, nothing remains to be done here apart from a possible implementation of system components at the end of the project in the car.)

Concerning 6, (specified in Task 7.6)

- To realize the proposed system setup from the beginning of this document (comp. Figure 1 c) we will (this process has been started, please comp. “Work done”) port working agents to the car scenario.

6.2.2 Tasks

Task 7.1 Realization of the proposed architectural framework with OROCOS. That includes:

- Task 7.1.1: Implementation of the described components and their communication pathways in OROCOS
- Task 7.1.2: Performance testing and debugging.

Note these steps will have to be continuously adapted to the outcomes from tasks 7.2 and 7.3.

Task 7.2 Improving the speed generating agent. To achieve longitudinal control the system must especially make use of the upcoming observed trajectory that we treat as an SVE. Trajectories are being learned from human driving.

- Task 7.2.1 We will investigate how the upcoming trajectory should be represented best in conjunction of the above sentence. As a first approach we can modularize the parts of the road and combine such pieces (e.g. curves, straight segments etc.) to create the required representation for a certain situation.
- Task 7.2.2 Incorporating the upcoming trajectory information into the speed generating agent. Possibly this could be achieved by also using artificial neural networks for generalization purposes.
- Task 7.2.3 Modify the steering agent using speed information to augment the steering (partly done).

Task 7.3 Elaborate the obstacle avoidance agent:

- Task 7.3.1: The obstacle avoidance agent acts like a path planner. The task is to implement the obstacle avoider by essentially an “imagined lane” is being “drawn” around the obstacle and followed. This imagined lane is being generalized from the human’s driving (partly done).
- Task 7.3.2. Using IMO information the obstacle avoider can be modified to treat also car following or approaching objects.

Task 7.4 Results from other WPs

- Task 7.4.1 Assess real-time capabilities from the methods developed in WPs 4 ,5 and 6.
- Task 7.4.2 If real-time compatible implement in the robot.
- Task 7.4.3 Synchronize SVEs and SAE-SVE proactive mapping between car and robot using similar methods if possible.

Task 7.5 Night vision

- Task 7.5.1 Implement night driving in the robot. (Note the actual robot control will not be different from day driving, hence – as the night vision system is installed and working – there is nothing to be done here anymore apart from showing that the same augmented learned controllers also work with IR illumination.)
- Task 7.5.2 Implement night warning in the car.

Task 7.6 Port working agents to car scenario

- Task 7.6.1 Re-adjust the state descriptor in accordance to the car data, as for example currently being done for the reactive controller.
- Task 7.6.2 Port working agents to the car and benchmark according to Figure 1c above.

6.2.3 Expected Results

At the end we will have a robot equipped with an extendable system architecture (which can therefore include results from other WPs) that captures the path following strategy from a human supervisor including coping with non-moving obstacles as well as being able to follow other “cars”. Specifically:

1) The system will operate in real-time using driving data extracted from human driving to emulate autonomous robot driving.
2) The system will be able to incorporate the upcoming trajectory information into the speed generating agent.

3) It will adjust the actions of the steering agent using speed information to improve performance.
4) It will be able to anticipate in a limited way future SVEs (lane structure) by predicting them from future road trajectories.
5) The system will also be able to avoid obstacles and to follow other robots using the human's behavior as a model.

2.7 References

1. Agapitos A., (2007) Evolving robust and specialized car racing skills. IEEE Congress of Evolutionary Computation 1187-1194.
2. Boer E., (1999) Car following from the driver´s perspective. Transportation Research Part F; Traffic Psychology and Behaviour, 2(4): 201-206.
3. Breazeal C., Scassellati B., (2008) Robots that imitate humans. TICS 6(11):481-487.
4. Brooks R., (1985) A robust layered control system for a mobile robot. IEEE J. Robot. Automat., 2(1):14-23.
5. Boer E.R., (1996) Tangent point oriented curve negotiation. Intelligent Vehicles Symposium, pp. 7-12.
6. Chatila R., Laumond J.-P, (1985) Position referencing and consistent world modelling for mobile robots. In Proc. ICRA, pages 138-145.
7. Dickmanns E. D., (2002) The development of machine vision for road vehicles in the last decade. IEEE Intelligent Vehicle Symposium 1:268-281.
8. Dickmanns E. D., (1994) The 4D-approach to dynamic machine vision", In: Proc. 33rd IEEE Conf. on Decision and Control 4:3770-3775.
9. Dickmanns E. et al., (1990) An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles, Transactions on Systems, Man and Cybernetics 20(6):1273-1284.
10. Dillman R., (2004) Teaching and learning of robot tasks via observation of human performace. Robotics and Autonomous Systems 47:109-116.
11. Donges E., (1978) A two-level model of driver steering behaviour. Human Factors. 20(6) 691-707.
12. Gat E., (1998) On three-layer architectures. Artifcial Intelligence and Mobile Robots, editors D. Kortenkamp, R. P. Bonnasso, and R. Murphy, AAAI Press,.
13. Gregor R. et al., (2002) EMS-vision: a perceptual system for autonomous vehicles. Intelligent Transportation Systems Conference, Detroit, 3(1):48-59.
14. http://www.darpa.mil/grandchallenge/index.asp
15. Krödel M., Kuhnert, K-D., (2001) Autonomous driving through intelligent image processing and machine learning, Int'l Conference on Computational Intelligence.
16. Lang, M., Horwood J., (1995) Which parts of the road guide steering? Nature, 377(6547):339-340.
17. Metta, G. et al., (2006) YARP: Yet Another Robot Platform. International Journal of Advanced Robotics. 3:43-48.
18. Nesnas R. et al., (2001) Toward developing reusable software components for robotic applications. Proceedings of the International Conference on Intelligent Robots and Systems (IROS).
19. Partouche D, et al. "Intelligent Speed Adaptation Using a Self-Organizing Neuro-Fuzzy Controller", Proceedings of the IEEE Intelligent Vehicles Symposium, 2007
20. Pomerleau D., (1989) ALVINN: An Autonomous Land Vehicle In a Neural Network. Advances in Neural Information Processing Systems, Morgan Kaufmann. 1:305-313.
21. Pomerleau D.A., Touretzky D.S., (1993) Analysis of feature detectors learned by a neural network autonomous system. International Conference on Intelligent Autonomous Systems (IAS-3) pp. 572-581.
22. Porr B. Wörgötter F., (2006) Fast heterosynaptic learning in a robot retrieval task inspired by the limbic system. Bisosystems 8(1-3): 294-299.
23. Porr B., Ferber C., Wörgötter F., (2003) ISO learning approximates a solution to the inverse-controller problem in an unsupervised behvioral paradigm. Neural Computation, 15(4):865-884.
24. Riedmiller M. et al., (2007) Learning to drive a real car in 20 minutes. FBIT, pp. 645-650.
25. Rusu P. et al., (2003) Behavior-based neuro-fuzzy controller for mobile robot navigation. IEEE Transactions on Instrumentation and Measurement, 52(4)1335-1340.
26. Schaal S., Ijspeert A., Billard A., (2003) Decoding, imitating and influencing the actions of other: The mechanisms of social interaction. In: Computational Approaches to Motor Learning by Imitation, 358:537-547.
27. Salila Z., Lezy P., (1994) Longitudinal control of an autonomous vehicle through a hybrid fuzzy/classical controller. WESCON/94.'Idea/Microelectronics'. Conference Record.
28. Salvucci D. Liu A., (2002) The time course of a lane change: driver control and eye-movement behaviour. Transportation Research Part F, 5, 123-132.

29. Shang T., Wang S., (2006) Knowledge acquisition and evolution methods for human driving intelligence, International Journal of Innovative Computing, Information and Control. ICIC International, 2(1):221-236.
30. Tahirovic A. et al., (2005) Longitudinal vehicle guidance using neural networks", Proceedings, IEEE International Symposium on Computational Intelligence in Robotics and Automation,
31. Trancoso J., et al., (2007) Discretization of ISO-Learning and ICO-Learning to be included into reactive neural networks for a robotics simulator, LNCS 4528:367-378.
32. Vasquez, D. et al., (2008) Long-term prediction of future motion through visual observation and learning. Autonomous Robots (submitted).
33. Weng J., Chen S., (1995) Autonomous navigation through case-based learning. Proceedings International Symposium on Computer Vision.
34. www.argo.ce.unipr.it/ARGO/
35. www.mca2.org
36. www.orocos.org
37. www.rtai.org
38. Zlochin M., Baram Y., (2001) The bias-variance dilemma of the monte carlo method", Artificial Neural Networks – ICANN pp. 141-147.