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:	**D7.1 (Update 1)**
Title of the deliverable:	**Specification of Learning Requirements (Update 1)**

Date of Delivery:	15. 11. 2007
Organization name of lead contractor for this deliverable:	BCCN
Author(s):	F. Wörgötter, BCCN, Irene Markelic, BCCN
Participant(s):	BCCN
Work package contributing to the deliverable:	WP7
Nature:	D
Version:	2.0
Total number of pages:	6
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)	

Specification of the Requirements for Structured Visual Events (SVEs) and Structured Action Events (SAEs) in the context of Learning

BCCN, Göttingen

Bernstein Center Göttingen
Bunsenstrasse 10
37073 Göttingen
15. Nov. 2007,

Table of Contents

1. Abstract

This document is the update of Deliverable D7.1. It lists which of the SVEs and SAEs specified in D7.1 have been realized and how their specification had to be adapted. Since a good software architecture is necessary allowing the agents to interact, in the second part of this document we specify requirements for such a system.

2. Introduction

In the first version of this Deliverable, we presented an outline on how we want to use intelligent agents for semi-automated driving in a structured environment. We assumed that driving can be broken down into atomic actions, that are either applicable alone or in combination. We have suggested using committee machines for learning their combinations, but this issue has not been considered any further so far.
We stated that SVEs and SAEs can be defined on different semantic levels, and that one challenge for applying machine learning techniques is to identify the most suitable level for it.
Thus, coherent SVEs and SAEs must be identified, then structured, and decided on the level on which correlations between both events should be established. We already specified several SVEs and SAEs that we considered meaningful a priori, and also the parameters describing them (e.g. see table 1 in the first version of D7.1).

Following those outlines we now realized some of the goals stated in the first version of this document. We implemented the two most basic agents for street following and obstacle avoidance. Both agents were first specified, learned and implemented and tested on the robot. For a quick start we used a very straight-forward software design only allowing for testing these two agents. To overcome this shortcoming and to adapt to a more scalable system we elaborated a software architecture that we are planning to implement in the near future.
Accordingly, the structure of this document is as follows: a) description of the two agents, b) presentation of the planned software architecture and c) requirements for future work.´

3. The Street Following Agent

This is considered to be the most basic agent of the system. Its only task is to keep the car/robot on the street driving at an appropriate speed. Of course these capabilities are to be learned from the driver. In the first version of this document our specification requirements were guided by theoretical considerations only, and we found that some of them had to be altered in the course of realizing. However, it turned out that almost all of the initially stated requirements were useful. From the visual input, we first considered lane marking signals as relevant, our position within the lane, and heading vs. lane marking signals, also we suggested the possible usage of optic flow. Only the last aspect was not incorporated.

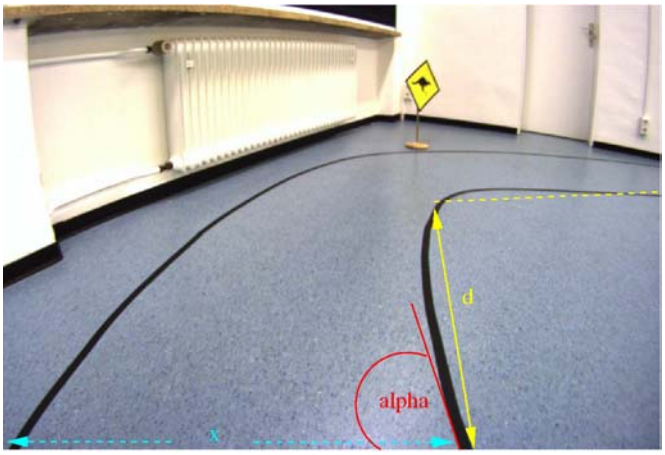

Figure 1 Extracted image Features for the Street Following Agent.

Features extracted from the camera images are displayed in Figure 1. It has been found that the angle between the first right lane marking and the x-axes of the image, *alpha*, and the x-value of the starting point of the lane signal, *x*, correlate well with the steering information of the robot.

For getting a velocity signal we measure the Euclidean distance between the starting position of the lane signal, at the bottom of the image, and the next break point, denoted *d* in the image. A break point is a point with high curvature that we get by polygonizing the lane marking, using Bezier-Splines.

Since *alpha* and *x* correlate so well with the steering, and also *alpha* and *d* with the velocity signal, we base our correlations on this level (see Fig 3). Learning reduces to surface fitting of the control signal to the two input values, as shown in Figure 2 for the steering control signal.

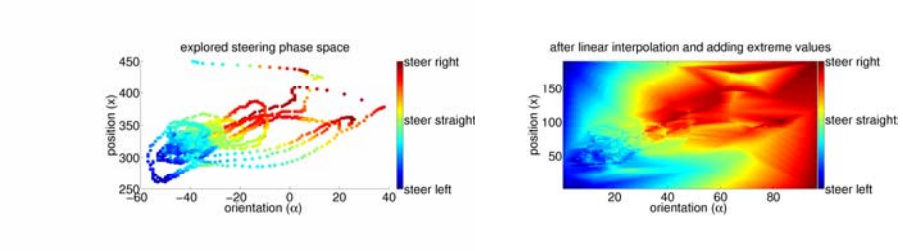

Figure 2 Left: Supervised steering signal plotted against *x* and *alpha* after the robot was remote controlled for several labs on the parcours (1500 datapoints). Colors denote the strength of the respective steering signals. Right: The same steering surface after linear interpolation.

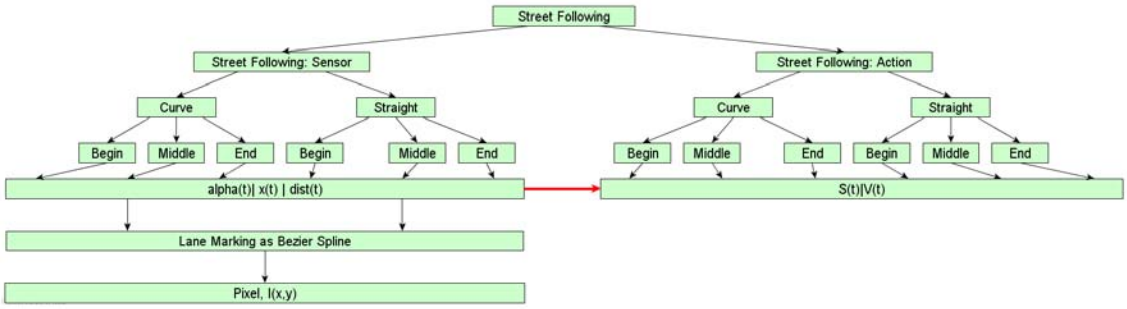

Figure 3 The level-hierarchy for the street following agent on which correlations proved to be best, marked in red.

4. The Obstacle Avoidance Agent

This agent is more complicated for two reasons, first: it possibly needs more information than visible in an image at a time, and second: it must not only avoid obstacles but also return to the trajectory given by the street, thus it must be combined with the street follower.

In a dynamical system, or a potential field approach, this is solved elegantly by making the next position of the robot a function of features of the trajectory and the obstacle [**Errore. L'origine riferimento non è stata trovata.**].

Another frequently adopted technique is occupancy grids. A grid of the world, either in two or three dimensions, is put up, were each cell is either free or containing an obstacle. A path planner then

calculates an optimal path through this world, by employing some cost function.

Both approaches require knowledge on the position of the obstacle and the street. But when turning away from an obstacle the robot looses this information, i.e. it lives in an only partially observable world, which makes the maintenance of some kind of world model, mandatory to serve as memory.

To this point we experimented with three kinds of obstacle avoidance behaviors: The first one, a), is inspired by the potential field approach as mentioned above. The second one, b), assumes a stereotypic behavior of a human when avoiding obstacles and the third one, c), is a type of path planning. For all approaches the position of the obstacle, the distance and direction with reference to the robot are needed. Also we prepared for taking velocity and heading direction of a moving obstacle into account.

The approaches differ in the way they are combined with the street following agent. In a) they are added to each other, i.e. their control output is summed, and in b) and c) they are executed consecutively. The first approach seems to be the most promising, since it scales best.

In b) we extracted stereotypic trajectories in the control data space from the human and parameterized them. The obstacle avoider carries out a more or less preprogrammed chain of actions mimicking the driver's behavior and also guiding the vehicle back on the track so that it can continue with the street following behavior. Here, the system's response to environmental changes is very restricted while being in the avoidance state, e.g. it is almost impossible to handle two or three objects at the same time, without training the system for every single situation.

In c) we bended the detected street lane such as it would lead around the obstacle like an elastic rubber band. Although this approach is very attractive it proved to be difficult to control.

However, as said in the beginning of this paragraph, all approaches are in danger of failing, when, after an evading movement the street is out of sight, thus concluding it can be said, that one important issue to be tackled next is how to store world knowledge.

5. Software Architecture

A sophisticated software architecture is the core of a well working robot system. Milestones in robot architectures are: a) sense-plan-act, b) behavior based, and c) hybrid architectures [**Errore. L'origine riferimento non è stata trovata.**]. Prevalent features of a) are: unidirectional information flow and usage of sensor information for constructing a world model that is used for planning. This capability for high level reasoning is the biggest advantage of the system. At the same time it is a drawback, because it proved to be so time consuming that sps-agents often could only perform open loop control. Closed loop control could be achieved by the behavior based paradigm, but at the cost of eliminating the planning unit. The attempt of combining the advantages of both approaches while avoiding their shortcomings lead to hybrid architectures. In its most known form, the three three-layer architecture, it comprises three layers, with a fast reactive one on the lowest level, a slower planning unit on top and an intermediate layer, called the sequencer, combining them [**Errore. L'origine riferimento non è stata trovata.**].

The robot architecture for our system should yield the following requirements:

- have a planning and a reactive unit, according to the three-layer architecture. Consequently there is need for asynchronicity, which means a time server must be employed and a particular communication model chosen.
- allow for easily adjustable software, thus no recoding of higher or lower programme levels becomes necessary when changing single parts of the system. This requires a modular system

design. Also, since processing of visual data is expensive, it should be permitted to run particular modules on special machines, requiring distributed programming.

6. Requirements

Concluding from the foregoing paragraphs, future work of the project will be devoted to the following topics:

a. Identification of a suitable world model and its maintenance.
b. Implementation of a scalable software architecture.
c. Adding more agents to the system, e.g. street sign detectors.
d. Investigating how to best combine agents.
e. Towards Proactivity

As a first step of tackling a) the usage of odometry is planned. Since the robot system does not possess GPS sensors, odometry is a simple way for getting the robot´s position relative to some goal. If this should not suffice more sophisticated techniques e.g. map building can be used.

Concerning b) we propose the usage of an event driven architecture, which can be highly distributed and loosely coupled. There are numerous open source software packages available, such as: CLARAty, CIRCA, JAUS, PLAYER/STAGE, FIPA, OROCOS and OSCAR. First it should be examined if any of those prove practical for the system according to the following needs: Does it support real time support? Does it support event driven design, i.e. does it allow messaging? Is it supporting distributedness of the system? Is it an active and well documented project?
The course of action for realizing c) and d) will be discussed in the upcoming project meeting in Granada.

The last issue e) refers to an important goal of the project, i.e. the ability to foresee certain situations coming up and proposing appropriate action chains. The idea here is to construct a predictor agent who is in charge of this. A first version of the predictor based on the street following agent is planned to be implemented soon. To this point the decision which steering and velocity signal is issued, depends on the extracted features as explained above. Thus, for each image, exactly one control vector is generated. The goal is to augment this to a sequence of vectors forecasted by the predictor agent. This can be done the following way:
For each image a control vector is generated. Therefore it is possible to store all the subsequent control vectors of an image, such that a database is generated, containing all actions that were conducted, after the occurrence of a certain visual input. Now, for any given visual input, the predictor agent can consult this database, and compare the current image, or rather the input features, with those, that have been stored. If there is already a very similar item in the database, the predictor agent returns the next n stored control actions assigned to that particular input vector. This has already been tested on raw data, and yielded promising results.

7. References

[1] Fajen, B. R., Warren, W. H. A Dynamical Model of Visually-Guided Steering, Obstacle Avoidance, and Route Selection, International Journal of Computer Vision, 2003, 54(1-2), p. 13-34.

[2] Müller, J. P., Architectures and Applications of Intelligent Agents: A Survey: The Knowledge Engineering Review, 1999, Vol. 13:4, p. 353-380
[3] Gat, E., On Three-Layer Architectures, editors, Artificial Intelligence and Mobile Robots, 1997, Edited by David Kortenkamp, R. Peter Bonasso and Robin Murphy.