BEHAVIOUR LEARNING BY VISUAL GESTALT COMPLETION

Klaus-Dieter Kuhnert, Michael Krödel

University of Siegen, Institute for Real-Time-Learning Systems,
Hölderlinstrasse 3, D-57068 Siegen / Germany
kuhnert@fb12.uni-siegen.de

ABSTRACT

First, the paper presents some basic considerations of how specific behaviours are learned from complex sensor input. These thoughts are exemplified by a system that is able to learn to drive different cars in different environments exclusively by vision. The basic idea that learning is nearly always the acquisition of task-relevant probabilities is detailed and a general method how this idea may be implemented for the different subtasks is proposed. Neither explicit modelling nor soft-computing methods have been employed for this task. Our approach is primarily memory based and tries to approximate the probability density functions that rule the behaviour of task learning systems. The general paradigm of the approach is that at several levels mainly pattern matching techniques are necessary. Finally, some details of the implementation and results on learning visual autonomous driving are given.

1. BASIC IDEAS OF SYSTEM MODEL

Fixed patterns constitute a *gestalt*. This term has a quite long history in psychology and several gestalt-laws have been discovered. Basically the gestalt constitutes a mechanism to condensate different stimuli, which may be completely unrelated at a first glace, into a new entity: just the gestalt. From this formulation the reader may already suspect the relation to visual learning, but first we want to prepare the necessary ingredients of the gestalt endowing mechanism.

Some states s* of the environment of an organism are of real importance for its survival and some are not. Therefore every system that has a goal must try to protocol and evaluate the relevant states and the conditional probabilities p(s*/s#). I.e. the probability a current state s# will generate such an important state s*; possibly over a chain of other states.

These probabilities deliver the objective basis for decisions the system has to take. In most cases a state s# is not directly accessible but can be associated with a measurement pattern m# e.g. an image.

As common in control theory in the following we will use vectors for states s and measurements m.

For a rather large class of tasks with practical relevance it is nearly always possible to eliminate time dependencies by introducing new components to the state vector and thus use markov process as the description framework. It is easily seen that this description of the interaction between system and environment yields of markov random process with hidden states.

In analogy with leaning theory we call the different possible actions that have to be chosen in a state a policy p.

Fortunately for a lot of tasks it can be assumed that the policy over state space is steady and "flat" i.e. small changes in state create no catastrophic changes in the appropriate policy. At least such instable regions in state space should be restricted to few places.

By this assumption it is possible to use methods from steady function in combination with the markov process description. I.e. at least locally the discrete state measurements can be steadily approximated and the policy can be optimised by a local approximation function.

Also it is common knowledge that in organisms, due to the slow processing elements only few processing steps between recognition of a pattern and a reactive action are possible. Therefore one basic consideration in our system is to have the shortest possible connection between recognition and behaviour. Due to the same argument similar approaches have been tried with neural networks (e.g. ALVIN [1]). There the neural net was used as an approximating function for the policy and implicitly the same assumptions have been made as stated above.

To reach greater transparency of the internal system function and to acquire the ability to fine tune the systems parameters we decided to use primarily a memory based approach. An additional motivation was that in organisms reactive behaviours could also be mainly memory based (admittedly on a parallel distributed hardware). Therefore, our approach tries to implement the approximation of the state and the policy function by adapted data structures.

Learning in this framework means to prepare the appropriate data structures and of course mechanisms to fill and access them quickly but also to weight and memorize the appropriateness of behaviour at a certain situation s#.

To force flexibility into the system as little task specific knowledge as reasonable and possible should be build in.

Thus, our Approach tries to rely as less as possible on rigidly programmed models of task specific knowledge to guarantee flexibility and expandability.

Learning of mentioned conditional probabilities may be interpreted in psychological terms as completion of patterns. This way the well introduced methods of pattern recognition may be employed. To the gestalt principle which was thoroughly investigated in psychology there is fundamentally no difference between patterns distributed in space or in time. Thus, by gestalt completion also prediction is possible generating the ability to judge about unmeasured parts of the patterns and their attached states.

Also it has to be stated that within one state conditional probabilities exist which allow to recognise the state by only a part of its complete information. Thus the gestalt can be completed from the features of a situation.

2. IMPLEMENTATION OF THE BASIC IDEAS

For a specific application the general consideration sketched above can be implemented in three steps:

- identify the task relevant conditional probabilities
- approximate their distribution by discrete function
- interpolate locally to improve the precision

The first question is: it is possible by sheer pattern recognition to find the appropriate state and behaviour? Additionally, it is clear that such a system needs a *large* number of patterns (> 5000).

3. SYSTEM OVERVIEW

As an application to demonstrate the basic ideas we have chosen the task to drive a car with *unknown dynamics* on an basically *unknown track* by *visual input*. Vision is very appropriate if a non trivial relation between measurement m# and state s# should be investigated. Due to a number of effects like perspective projection, occlusion, illumination etc., the relation is very non-linear and complex.

So, the system is confronted to the situation a human is in when he starts to learn to play a computer game of car racing. In specific, in a graphical simulation the system is required to steer an autonomous car along a curvy and hilly road course with no intersections and no other vehicles respectively obstacles on the road but with the strict requirement to self-improve driving behaviour based on delayed, self-created rewards or punishments.

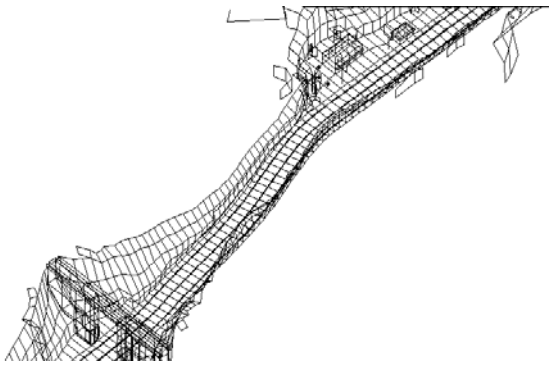

Figure 1: Wire frame model of an example track

The goal is not to follow a specific trajectory as it would be necessary in normal traffic but to drive as fast as possible. I.e. to learn a policy that optimises the longitudinal and lateral control for time optimal motion. Due to the fact that the dynamics of the vehicle are unknown standard methods from control theory are not applicable.

Figure 2: Example Image

The task may be divided into three subtasks as shown in figure 3: INTELLIGENT IMAGE PROCESSING (IIP), PATTERN MATCHING (PM) and REINFORCEMENT LEARNING (RL).

Handling a video stream in real-time urges the need to reduce the bulk of information. Because we want to build

a flexible system this reduction mechanism should not be hardwired but able to learn the task relevant features at least to a certain degree. The subsystem INTELLIGENT IMAGE PROCESSING (IIP) is responsible for processing the incoming video stream and the real-time calculation of a reduced parametric description of every single image called Abstract Complete Situation Descriptions (ACSD's).

look adequate a mechanism is needed to optimise the policy and to compute behaviours not previously recorded.

The subsystem REINFORCEMENT LEARING (RL) is responsible for determining a suitable behaviour in terms of steering commands for the current situation. This determination is based on the actual situation in conjunction with previously recorded, similar situations

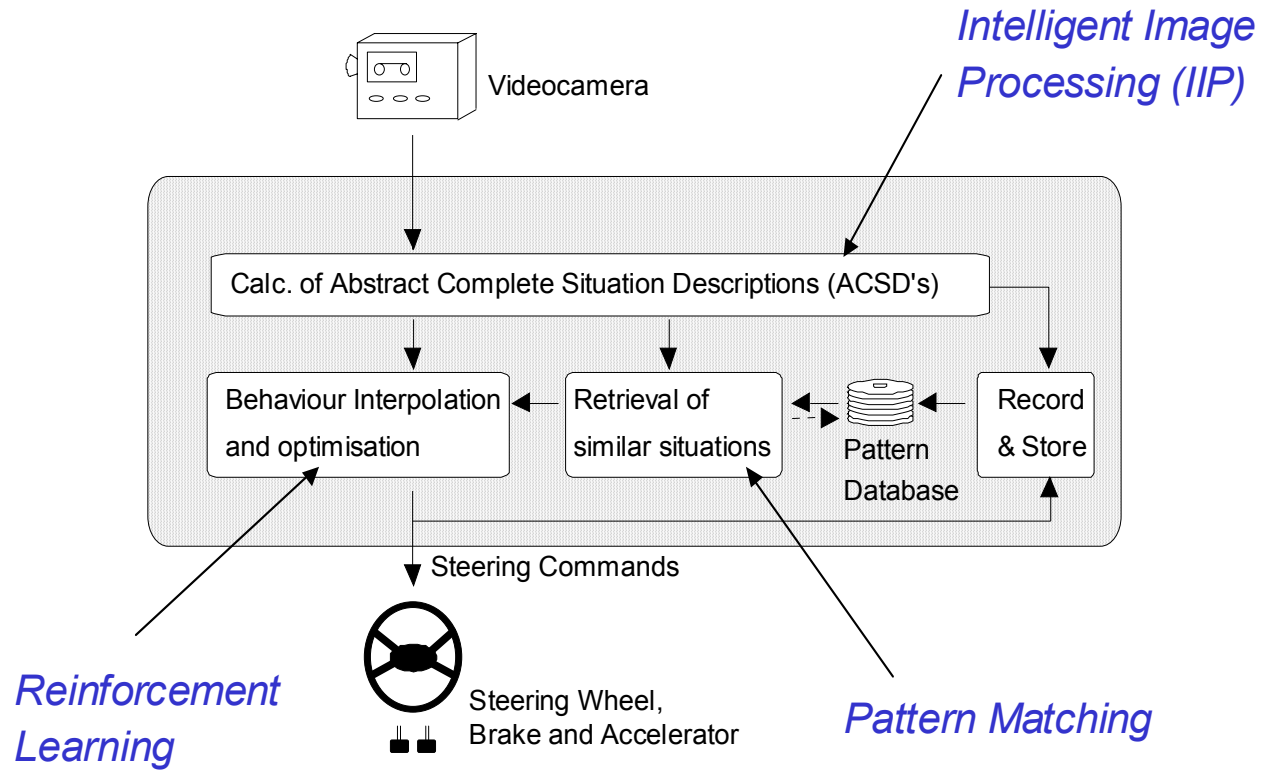

Figure 3: Structure of the system

From all the possible actions p in a situation s# at least one has to be selected for execution. Thus, the system has to learn to find the appropriate p# to a measurement m#. This task is performed by the Subsystem PATTERN MATCHING (PM). It has the task to retrieve similar measurements m in comparison to the actual measurement m# from the camera. A large number of previously recorded situations, respectively their ACSD's, is being stored in a pattern database which is being accessed by the subsystem PATTERN MATCHING in two ways: Firstly, the Subsystem PM scans the pattern database for similar situations in comparison to the actual one; secondly, it also stores the ACSD of the actual situation together with the actual steering commands for further reference.

Because not only one measurement m might be similar to the actual one (m#) and thus several behaviours might

and their issued steering commands. For such task it is crucial to rate the appropriateness of any previously issued steering command, otherwise optimisation of behaviour would not be possible. Therefore, this subsystem is also responsible for weighting the success, respectively the appropriateness of any calculated steering command and also copes with the difficulty that such appropriateness can often only be determined after a quite long time delay.

4. RELATED WORK

Up to now the visual control of systems for autonomous vehicle driving with learning components have been implemented in several ways. [1] describes a short direct connection between image processing and one soft computing learning method using a neural network. This approach provides good results but only for as long as input pictures of the scene are quite similar to the training pattern. In several variations, neural networks for this task have been used.

A completely different approach is being followed by using explicit modelling. [2] describes an early success with international attention of a vehicle system using a real-time vision system BVV2 [3]. Further developments stayed with the aspect of modelling, thus always in need for loading the system with many parameters for the modelling process.

The presented approach is a study of avoiding both neuronal networks or similar approximation methods as well as explicit models. It gathers all information purely from incoming images. It derives a situation description of the scene and develops a driving behaviour by a optimisation technique. With reference to the need for machine learning capabilities the current research follows the basic principles of Reinforcement Learning (e.g.[5]) and focuses on a way of combining such area of research with pattern recognition algorithms.

5. SUBSYSTEM IMAGE PROCESSING

A single connected camera is delivering the video stream from the environment. For each single image an abstract complete situation description (ACSD) of the situation of the vehicle has to be computed. Such conversion is basically a reduction process and in addition to the requirement of reducing the streaming video information to the relevant parts, the calculation of such ACSD's has to be very quick. Therefore, this subsystem combines traditional edge finding operators in order to locate contrast differences with a new technique of Bayes prediction in order to predict whether a contrast difference results from a road mark edge.

During the runtime of the system, a first pass scans for horizontal colour contrast differences by a conventional filter technique and memorises all locations where the colour values of two neighbouring pixels from the Single Image represent a possible candidate of a Road Mark Edge or lane border.

In a second pass, those candidates are fused into higher order objects i.e. road marks by chaining them together. The chaining is supported by the statistics which indicate the expected orientation of the road course in this location within the Single Image. If a chain can be established and fulfils minimum requirements on length and number of interconnection points in between, the whole chain is being stored as a confirmed representation of a Road Mark Edge. Otherwise, the considered candidates of Road Mark Edges are being rejected.

5.1 Usage of pre-build knowledge on road courses

As already stated the knowledge of the appearing of road mark edges or lane borders is not hardwired into the system. The knowledge of the typical shape of marking lines has first to be learned. After gathering this knowledge it can stabilize and speed up the search process.

If we assume a certain pixel to be on the road mark we want to know the probabilities of the surrounding pixel also to be on the road mark. Additionally for the search it would be advantageous to know the pixel position that lies on the road mark with the highest probability. As the reader may recognise, we just applied the basic consideration to the specific problem here.

I.e. we have to learn the conditional probability density function:

$$f(P_{x,y} \mid P_{x+i,y+j}) \qquad (1)$$

with:

$P_{x,y}$ is the event that the pixel at (x , y) is on a Road Mark Edge

$P_{x+i,y+j}$ is the event that the pixel at $(x + i , y + j)$ is on a Road Mark Edge.

This function is quite smooth in the near region of the road image but depends obviously on (x, y). It also depends on several other factors like the actual course and vehicle dynamics but these influences are assumed to be stable and representative over the learning phase.

To reach a practical implementation the function is approximated by a discretely sampled version:

$$p(P_{x,y} \mid P_{x+i,\ y+j}) \qquad (2)$$

Therefore, the Single Image is separated into Tiles and a separate function $p_{x,y}$ is estimated for each Tile. One connection between $P_{x,\ y}$ and $P_{x\ +\ i,\ \ y\ +\ j}$ is called a Chaining Vector. So, this function can not only be utilised for describing the statistical properties of Road Mark Edges but also for defining a search strategy, i.e. all the

Chaining vectors belonging to the position (x, y) can be searched in the order of their probability thus optimising the search speed. The grid size i, j and their maximum value i.e. the size of the context that is considered are the most important parameters of the method.

If we turn our argument to psychological terms the method performs a spatial gestalt completion: by knowing part of the pattern the rest of it can be derived more stable and rapidly.

During learning, in a large number of different Single Images of representative road scenes the relevant borders are marked by a human teacher. Form the labelled images the conditional probabilities according to (2) are computed. I.e the Chaining Vectors are created and stored indicating which two pixels in a certain surrounding may be connected to a Road Mark Edges with a certain probability. Since the Chaining Vectors vary with position (x,y) within the image, the conditional probabilities are computed for several discrete positions in the image called Tiles. This allows the usage of different primary interconnection directions based on the actual location within the window under concern.

For each Tile, the set of Chaining Vectors is sorted in the order of their probability, thus search can start with the most probable vector which also has the greatest chance to find an other element pixel the same road marking. Figure 4 illustrates the stored chaining vectors. Numbering is in order of decreasing probability.

With the same approach it would also be possible to compute a consistent solution for the probability p(x,y) that a single pixel is on a road marking by using the Bayes theorem. A large linear equation system would have to be solved for this purpose. It should be mentioned that such an approach has an inherent relationship to relaxation labelling [10]. Because we are not mainly interested in computing the precise probabilities the available information is primarily utilized to speed up the search. Also our approach differs in using larger (e.g. 5*5) local surroundings. Thus, our approach can be interpreted as a rough but speed optimised approximation to p(x, y).

5.2 Calculation of ACSD's

In order to describe measurements by vectors as proposed in chapter 1. the representation by lists of chained pixels is not very appropriate.

Consequently, the road borders, which are up to now represented by pixel chains, are being intersected with a fixed line grid. The resulting intersections between Chains and grid deliver a position and an angle for each intersection point. All this is stored as the Abstract Complex Situation Description (ACSD). If the grid has *i* rows and *j* columns the ACSD has (*i***j*) elements. Assuming e.g. a grid of 10 columns and 20 rows and a processing speed of 40 ms for each Single Image, the needed storage space for an ACSD protocol amounts to 40 KB per second or 2.4 MB per minute, what is quite moderate using conventional computers.

Further details on parameter selection and the detailed treatment of the ACSD intersection points are described in [6] and [7].

6. SUBSYSTEM PATTERN MATCHING

After the actual measurement m# has been condensed into a compact ACSD an appropriate behaviour has to be selected. The simplest approach is supervised learning by imitating a teacher. Thus the teacher has to show how to behave in a representative set of situations and the system protocols measurements m_i and the behaviours p_i of the teacher.

Thus, a protocol has the structure:

$$(m_1, p_1) \ldots (m_i, p_i) \ldots (m_n, p_n) \qquad (3)$$

with time as the index.

The vector (m_i, p_i) in psychology would be called a sensor-motor-gestalt . It fuses sensor pattern and action pattern into one new entity. Hence, the selection of the appropriate behaviour can simply be described by gestalt completion: if m# represents the actual measurement, exactly the behaviour p# has to be executed which is stored in the protocol together with it. I.e. do what your teacher did in this situation.

Unfortunately, identical measurement occur very seldom, especially if the input is as complex as an image. Therefore, a similarity measure has to be defined between the measurements. Such a measure was heuristically determined for the ACSD's ([7]).

The task of the Subsystem Pattern Matching consists thus in finding similar situations which have been recorded before. A similar task is to estimate the position within already driven road courses which is smoothly located in this subsystem.

Retrieval of similar situations

Given the different functional steps during the previous subsystem, the current situation is being described by its ACSD and not explicitly rebuild by a model of the vehicle or the road course.

For any comparison between different ACSD's a pattern recognition algorithm is needed. The chosen

pattern recognition algorithm used in this project is ANN [9], an advanced variant of the well known nearest neighbour algorithm. This algorithm has a O(n log m) characteristic compared to O($n \bullet m$) for the conventional nearest neighbour (with: m number of Patterns, n number of components of a pattern). The logarithmic order allows to search within approximate 10.000 Patterns with 100 components in a few milliseconds. Without such a rapid similarity recognition method a real-time realisation of our approach would not be possible. The algorithm delivers a number of similar measurements. By choosing the most similar and executing the according behaviour the car can be controlled.

7. SUBSYSTEM REINFORCEMENT LEARNING

7.1 Position estimation

As it can be seen in the results, it is possible to learn to drive with this simple approach but the resulting behaviour is in almost all cases inferior to that of the teacher. The reason consists in the similarity measure used to compare the measurements. Those measurements should be similar which require similar behaviour (the famous ethologist Nicolas Tinnbergen stated: an object is what needs a specific behaviour) but it is difficult to design such a similarity function from sparse measurements.

But even if the similarity function would be perfectly optimised the behaviour would be suboptimal because the optimal behaviour, exactly fitting to the actual measurement, was never presented by the teacher before.

Nevertheless, it is possible to drive by the simple approach because the function are obviously flat in state space.

To improve the behaviour not only the best fitting measurement should be employed but all similar measurements should be used. This way, the sensitivity to the design of the similarity function is reduced because the behaviour is based on a larger region of the state space which can include differing behaviours.

To become really independent of the more or less arbitrary design of the similarity function it is necessary to evaluate the success that is possible with a certain measurement.

As stated in chapter 1. only very few situations (and the corresponding measurements) typically have directly a value to our learning system.

Thus, the system must be able to cope with sparse and delayed rewards and punishments. On the other hand, if the system is able to assign values to each measurement the performance of the behaviour could surpass that of the teacher by goal oriented exploration. Reinforcement Learning offers one possible way, as shortly sketched in the this chapter.

Some basic evaluations of situation must exist for internal rewarding and punishment. Because we want to optimise speed, time and position along the course should be known to the internal rewarding system. Extracting the position on the course directly from the visual input proved to be quite difficult. But position provides two crucial pieces of information the system may utilise:

- the information on how the Situation Descriptions of the further road might look like (measurement expectation)

- the possibility for speed measurement as a basis for any performance determination

Initially position was done by retrieving similar ACSD's and further processing the search results. However, two major difficulties prevented any major success. Firstly, information of types of roads, e.g. left turn, look very often identical never mind at which position of the course they appear. Secondly, even ACSD's of the same left curve looks often different due to a different horizontal offset at which the same curve is driven at different times. Even several attempts in conjunction with Kalman-filter-type approaches did not deliver reliable estimation of the current position within the road course.

The second concept was not to use the ACSD's as they are but to classify them into road types (e.g. left curve, right curve, etc.). The basic idea is that most road courses have a unique profile regarding the sequence of curves including the length of the distances in between. If such profiles are being created for the current part of the road course which is being driven it can be compared to any part of previously recorded profiles. The difficulty in this approach lies in the fact that the algorithm would need to be capable to classify road images into e.g. left curves, right curves, etc. This is in sharp contrast to the overall requirement to avoid any modelling, respectively avoid teaching as much as possible.

Therefore the chosen concept is being based on comparing sequences of the *Steering Commands*. The Steering Commands are the commands to steer the vehicle to the left or right. The basic idea is that if a vehicle drives several times along the same course and therefore experiences the same sequence of curves and straight road section the sequences of Steering Command will be similar. Figure 5 shows such a sequence of lateral Steering Commands for a driven road section. The x-axis from left to right represents the time (overall approx. 3

minutes) while the y-axis represents the angle of the Steering Wheel. While the recording of the Steering Commands has been done at discrete time-intervals of 100 ms, Figure 5 shows those Steering Commands averaged over 51 entries.

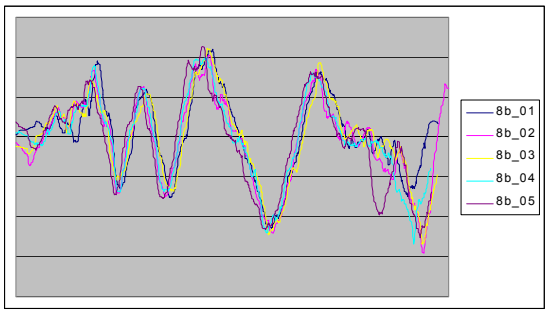

Figure 5: averaged Steering Command Protocol (x-axis: time; y-axis angle of Steering Wheel)

In general, it can be seen that the 5 displayed profiles of Steering Commands are very similar. The similarity exists regardless if the vehicle drove itself or was driven by the teacher. The small differences in details are caused due to differences of position and velocity but in all 5 cases the Steering Commands follow the same overall profile. If now only a fraction of such profile is being taken (in specific the last *n* issued Steering Command of the currently driven course) and then compared to the profiles created before, similar road parts can be identified. In conjunction with Kalman-filter-type approaches as implemented before the current position on the road can be estimated.

7.2 Reinforcement Learning concept

The base requirement for this subsystem is to provide machine learning functionality within the overall System.

In detail, the current subsystem needs to be able to optimise the quality of the Steering Commands over time. The paradigm of this research is to avoid implicit learning (e.g. by a neuronal network) as well as explicit modelling. Instead, the system of this research learns how to act in certain situations by gradual reinforcement and builds itself a set of Behavioural Patterns over time. A Behavioural Pattern (i.e. policy) is constituted by the recorded ACSD as explained before, the generated Steering Commands from that time and a factor indicating the success or failure of this combination. Thus, the System tries to learn explicitly the relation between certain states in measurement space and the reaction appropriate for the task. The system works in real-time with continuous feedback from the environment and any such Behavioural Pattern is being continuously optimised. A complete protocol of all the Behavioural Pattern generated during a round course delivers the knowledge base for further actions.

The difficulty of optimising such Behavioural Pattern stems from the fact, that the success or failure of any generated Steering Command may not be calculated immediately. Almost no situation in the environment delivers directly a response about the value of the actual behaviour. Our optimisation criterion is the time needed to drive a specific route. I.e. we try to learn to drive as fast as possible. With this criterion in mind only after a completed round or time stop an evaluation of the Behavioural Pattern used in between can be performed. Such characteristic of delayed rewarding or punishing makes learning much more difficult. Speaking in general terms, the learning methodology being used for this research can be referred to as Reinforcement Learning. More Details on the Reinforcement Learning aspect of the system, can be found in [4][8].

8. EXPERIMENTAL RESULTS

8.1 Pattern Matching

Results on Intelligent Image Processing this subsystem may be found in [6][7].

The quality of the subsystem Pattern Matching can be shown twofold. Firstly, Figure 4 shows a series of images where the first image on the top is the incoming Single Image coming from the attached video camera. All remaining images are the ones classified by the Pattern Matching algorithm as similar ones. If considered that those classified images are only some out of a pattern database with several thousand record entries which contain all various types of images it can be seen that the recognition quality is quite good.

Even if they look very similar they stem from different runs and locations.

From a pure image comparison point of view this might not be astonishing but in our case it needs to be noted that this has been achieved in addition to speed optimisation (search time for finding those 8 similar images out of a database of around 5.000 images can be done in less than 5 ms on any ordinary PC) and adaptation to road situations (the further course of the road to the left or right has higher priority on the classification of similarity than any objects on or next to the road which would distract ordinary similar-image-finding-algorithms).

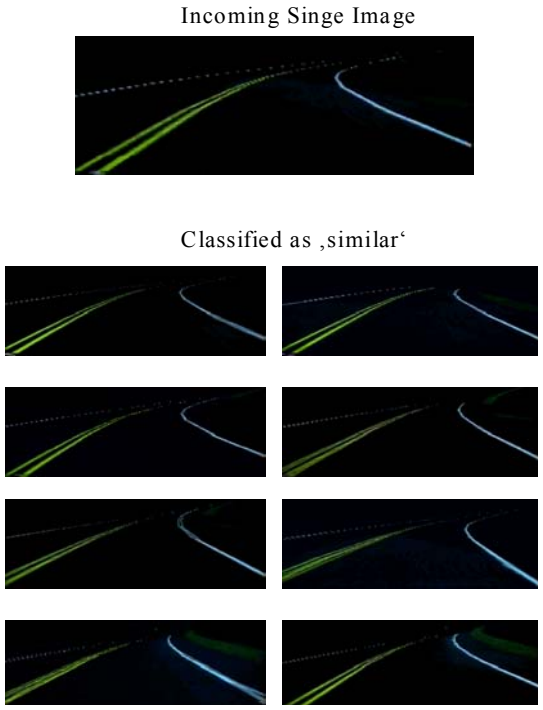

Figure 4: Result of Pattern Matching

8.2 Driving by Pattern Matching

In order to test first driving skills based on the pattern matching algorithm a pattern database with only appropriate steering commands for different situations has been build up. Such way, the Reinforcement Learning Process with it's challenge to weigh the appropriateness of issued Steering Commands is skipped at the moment but the difficulties of retrieving similar situations in comparison to the actual one and therefore first driving by pattern matching can be tested. Such way several test runs have been performed.

The tests then let the vehicle drive each time for approx. 5 minutes along a curvy and partial unknown road course. The results of such tests are listed below.

Average results of tests:

average frame frequency : 10 Hz
No. of entries in pattern database: approx. 5.400
Time of autonomous driving : 300 sec
Calculated Steering Commands: 3000 (10 per second)
Number of major errors (road course left completely): 1
Number of errors (crash barrier scratched): 4

If it is assumed that at each conducted error is the result of around 10 steering commands (the actual one and the previous 9 ones) the failure rate amounts to (5 · 10) / 3000 equals 1.6 %. Noting, that the autonomous driving is purely done on pattern matching since both Reinforcement Learning as well as intelligent calculation of Steering Commands are still outstanding, the results shown above are promising.

8.3 Reinforcement Learning

Prior to any combination of Reinforcement Learning (RL) with Pattern Matching methods, several RL methods with each time different parameter sets got studied before. In order to therefore have RL separated from the rest of the research, a virtual Race Track simulator got implemented which automatically feeds environmental input to a simulated vehicle and also receives and processes steering commands. Even though both environment and vehicle are being rebuild in a rather simple manner, the kind of interfaces to and from the RL system are very similar to the interfaces of the main research system and it's RL system.

Those interfaces include first of all a situation description, describing the situation the vehicle is currently in which is being calculated by the simulator and forwarded to the RL system. The output from the RL system are the steering commands which are being forwarded to the simulator. Based on those steering commands, a new environmental situation description is being calculated and again forwarded to the RL system. Lastly, rewards are being calculated by the simulator in parallel and forwarded to the RL system: a small punishment for simply driving along the course (usage of resources), a big punishment for bumping or scratching any obstacle and a big reward for reaching a defined goal. With those interfaces the RL system has been established according to 5.1 The final implementation of reinforcement learning is outstanding and in the future it might be interesting to be able to learn the behaviour with dynamic obstacle i.e. other cars on the track to be able to do real racing contest and not only to compare the round times.

9. SUMMARY

In the paper the actual state of a System is presented that is aimed to *LEARN* driving autonomously *different* vehicles on *different* courses exclusively from visual input.

The Subsystem INTELLIGENT IMAGE PROCESSING allows to locate the Road Mark Edges of each Single Image. A trained search algorithm, which is inspired by spatial gestalt completion allows optimal search speed and high recognition rate and consequently

efficiently converts Road Mark Edges into Abstract Complete Situation Descriptions (ACSD's).

The Subsystem PATTERN MATCHING successfully retrieves similar situations to the current one based on a pattern matching algorithm. This algorithm may be interpreted as sensor-motor gestalt completion.

The Subsystem REINFORCEMENT LEARNING is still under implementation. However, a simple approach implemented so far allows already autonomous driving on a learning-by-knowledge-transfer basis promising further positive results in the area of autonomous driving based our new paradigm.

10. REFERENCES

[1] D. A. Pommerleau, "Efficient Training of Artificial Neural Networks for Autonomous Navigation", *Neural Computation 3*, 1991

[2] E.D.Dickmanns, A.Zapp, "Autonomous High Speed Road Vehicle Guidance by Computer Vision", *Preprints of the 10th World Congress on Automatic Control*, Vol.4, International Federation of Automatic Control, Munich, Germany, July 27-31, 1987

[3] K.-D.-Kuhnert, "A Vision System for Real Time Road and Object Recognition for Vehicle Guidance", *Proc. Mobile Robots*, Oct 30-31, 1986, Cambridge, Massachusetts, Society of Photo-Optical Instrumentation Engineers, SPIE Volume 727

[4] K.-D. Kuhnert, M. Krödel, "Autonomous Driving by Pattern Matching and Reinforcement Learning", *Proc. of the IntrnationalColloquium onAutonomous and Mobile Systems*, June 25-26, 2002, Magdeburg Germany, pp26-31.

[5] R. Sutton, A. G. Barto, *Reinforcement Learning: An introduction, MIT-Press*, 2000, Cambridge (USA)

[6] M. Krödel, K.-D. Kuhnert, "Towards a Learning Autonomous Driver System", *IEEE International Conference on Industrial Electronics*, Control and Instrumentation, October 22-28, 2000, Nagoya, Japan

[7] M. Krödel, K.-D. Kuhnert, "Autonomous Driving through Intelligent Image Processing and Machine Learning", *Int'l Conference on Computational Intelligence*, October 1-3, Dortmund, Germany

[8] M. Krödel, K.-D. Kuhnert, "Pattern Matching as the Nucleus for either Autonomous Driving or Drive Assistance Systems", *IEEE Intelligent Vehicle Symposium*, June 17-21, 2002, Versailles, France.

[9] J. Langenhangen, "Nearly nearest Neighbour search with principal components", Thesis, Siegen, 2001

[10] S. W. Zucker, "Relaxation labeling: 25 years and still iterating", in *Foundations of Image Understanding*, L. S. Davis (ed.), Kluwer Academic Publ, Boston, 2001, 289 - 322.