From Numerical Interpolation to Constructing Intelligent Behaviours

Jianwei Zhang and Alois Knoll

Faculty of Technology, University of Bielefeld,
33501 Bielefeld, Germany

Abstract. In this paper we propose a general framework for describing and constructing sensor-based behaviours. We point out that a complex high-level task can be realised by a set of modular, cooperating behaviours. Each of these behaviours can be decomposed into local control actions which can be interpreted using linguistic "IF-THEN" rules. Several sample applications in real-world robotic systems are presented: a mobile gripper system equipped with proximity sensors and a two arm system with force/torque sensors. For controllers without sensor-action models, this framework can be universally applied after properly selecting the system inputs. Furthermore, "common sense" knowledge can be integrated and the control parameters can be rapidly adapted through incremental learning.

1 Introduction

To realise high-level robot tasks, the conventional robot control architecture employs the so-called SMPA (*Sensing-Modelling-Planning-Action*) strategy, which follows a strict sequential order of planning and execution of elementary operations. However, problems occur with such a control architecture: a). Algorithms for modelling and planning may be highly complex; b). The time delay from perception to action is usually long due to the computational burden; c). A system based on such an architecture is not fault tolerant. Therefore, a lot of recent work on robot control aims at finding efficient sensor-based solutions to reduce the temporal delay between perception and action.

Brooks' subsumption architecture [2, 11] essentially consisted of combining a set of parallel reactive behaviours without building complete world models. The main problems with this architecture are: a). Task-directed symbolic goals are difficult to be integrated in the behaviours (thus only insect-like "intelligence" can be emulated); b). The hard switch between different behaviours is unnatural.

We use the concept of behaviour in the context of realising tasks specified on a high-level. A behaviour is a control module which directly or indirectly uses the current perceptual information for achieving an explicit goal, i.e. the collective output of behaviours implements the task. Usually, each individual behaviour can be modularly developed and tested. If not only mobile robots but also robot arm systems are discussed, *sensor-based skills* instead of behaviours are employed to describe the basis control modules from the point of view of control engineers.

We give some examples of behaviours that are directly related to control:

Robot motion: *collision-avoidance, goal-direction, constant speed control, object-tracking, force control, following motion commands in natural language, etc.*

Robot vision: *visually guided location, active vision: track, saccade, camera coordination, etc*[1].

Like conventional process control, perception-action cycle can be implemented with either the model-based or the connectionist methodologies. Model-based approaches must specify explicit sensor-robot system models. Typical applications are calibrated methods for hand-eye coordination and the artificial potential-field for collision-avoidance. However, they suffer from the following problems: a). They are not adaptable to varying environments; b). They cannot be built incrementally or modularly; c). They cannot be interpreted symbolically. In one word, they are not really the way humans would do. Connectionist approaches use expert knowledge or learning to acquire the characteristics of the sensor-action system. Recently, such approaches are applied to the sensor-based control of robots as well as in classical process control as the so-called "computational intelligence" becomes a rapidly growing research area. Applications of artificial neural networks [1, 7, 4, 6] demonstrate the intelligent characteristics such as self-organising, adaptation and distributed processing, but the "block-box" structure stays as an obstacle for integrating symbolicist approaches which represent the other important part of human intelligence. Fuzzy control also finds applications in behaviour implementation [5, 8, 18], but these controllers are mainly realised with human desgin instead of self-adaptation.

In the following sections, we first discuss the important issues related to the implementation of sensor-based behaviours. Then we introduce the B-spline model and show how a behaviour and a local control rule can be specified and adapted based on this model. In this way, the advantages of neural networks and fuzzy systems can be combined. Two examples are given to further illustrate the construction process of a behaviour. The feasibility and advantages of the proposed method are demonstrated using two sample applications: a). motion for screwing with two robot arms; b). motion control of a mobile gripper system.

2 Issues in Realisation of Behaviours

2.1 Sensor Data for Control

In order to develop a robust on-line robot controller, external and internal sensor data should be applied directly in each control cycle instead of building and updating the world model. If sensor data is coupled with motion control in a simple form, the robot can decide its reaction in time. The idea of "*situatedness*"

[1] More behaviours for a eye-head system can be found in [3].

by Brooks [11] is comparable to this concept. By "*bounded rationality*" Simon [10] summarised the principle that humans often use only incomplete or imprecise knowledge for problem-solving.

Sensor data needed for direct integration in robot control possess the following properties:

- They are relative. These data are mainly derived from the external sensor measurements and their derivatives or the differences between the sensor values and the internal model. Such a variable value is not related to the robot or sensor alone, but to the interaction between the robot and its environment.
- They are local. Normally, only part of the environment, which is directly involved in the current robot motion, is perceived by the sensor system. Each sensor measurement represents one aspect of the object's features. No time-costly sensor fusion is performed (sensor data fusion is therefore transferred to task fusion).
- They are task-oriented. Modelling and interpretation of the sensor data depend on the control tasks. Only the control-relevant data are selected, preprocessed and represented.

2.2 Types of Controller Inputs

The complexity of the controller depends mainly on the dimension of the input space, i.e. the number of variables which influence the control action. Generally, these input variables can be classified into the following types:

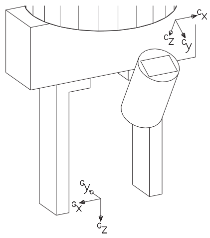
- Direct sensor readings. These are normally one or multi-dimensional signals which can be processed relatively fast. The representation level of information is low. An example in Fig. 1(a) shown proximity sensors in the mobile robot. Another example is the six-Dimensional force/torque sensors mounted on the gripper of a robot arm, Fig. 1(b).

(a) *Khepera*, equipped with infra-red sensors.

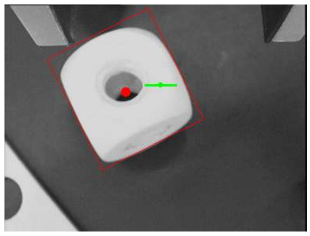
(b) A 6D force/torque sensor mounted on the robot gripper.

Fig. 1. Two examples of sensors whose output is directly fed to controller.

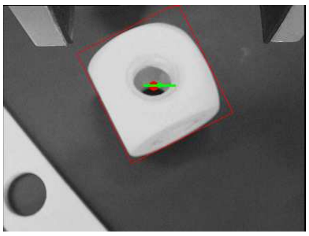
– Feature description variables. They are extracted from signals and images and represent information of medium to high level. Fig. 2(a) shows the configuration of a hand-camera used for our experiment. Such a "self-viewing" configuration enables the camera to have two gripper fingers in its view. The extracted features can be the relative distance from the TCP (*Tool Centre Point*) to the centre of object to be grasped and the relative angle between the orientations of the gripper and the parallel grasping edges. As shown in [13], the projected *principal components* as features can be also grouped in this category.



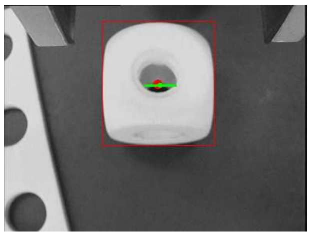
(a) "Self-viewing" hand-camera



(b) The start position



(c) Improved position



(d) The optimal position for grasping

Fig. 2. Visually guided grasping with a robot gripper.

– Combinations of planning and sensory information. Such a combination is particularly important when *purposive* instead of *reactive* behaviours are to be developed. The planning level assigns the symbolic information such as subgoals of tasks or geometric subgoals for collision-free paths. The sensory information is mainly the robot state estimation through the fusion of internal sensors. The difference between a subgoal and the current state may be taken as input variables. Fig. 3 depicts two variables applied to decide on the control action to keep a pre-planned path: d: the shortest distance between the robot and the pre-planned path segment, and α: the angular divergence between orientation of the path and the robot.

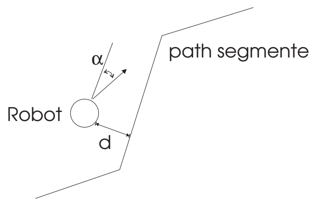


Fig. 3. Combination of planning and sensory information as inputs.

2.3 Hierarchical Decomposition of Behaviour Based on Rules

For many perception-action processes, no explicit mathematical models are available. One strategy of intelligent control is to partition the input space with into overlapping hyper-blocks (*lattice-based*) and to study the local control action for them. A control rule describes the relation between one typical configuration in the input space and the control action. There are good reasons for making such a perception-action mapping $Input\_Space \rightarrow Action$ symbolically interpretable instead of a black-box:

- Linguistic modeling provides a way of transferring skill from human experts to robots;
- Analysis and validation of the controller development;
- Supervision of the learning process.

In order to make robot behaviours emerge and adapt to new environments, we suggest a modularisation of behaviours and using a coordinator to determine the interactions between them. The decomposability can be described with the following hierarchical conception:

A local control rule determines a correct control action for a subclass of the input space.

An elementary behaviour has an explicit goal and is implemented by a set of cooperating atomic rules.

The behaviour arbiter coordinates multiple simultanuosly active behaviours to achieve a high-level task and is realised by a set of meta-rules.

The following section will demonstrate that B-spline basis functions can be used to partition the input space, which can then be interpreted with linguistic terms. The features of the B-spline model provide a suitable framework to describe local control rules, to aggregate multiple rules for constructing behaviours and to blend cooperating behaviours to carry out a predescribed task.

3 B-Spline Models for Constructing Behaviours

3.1 B-Splines Basis Functions

Although B-splines have been mainly used in off-line modelling, we have shown that they may constitute a suitable model for describing sensor-based behaviours.

B-spline basis functions are naturally defined convex function hulls which can be best interpreted as linguistic labels. The synthesis of a smooth curve with basis functions can easily be associated with the blending of local control actions. These points are the main motivation for our work on utilising B-splines to design behaviours.

In our previous work we compared the basis functions of periodical *Non-Uniform B-Splines* (NUBS) with a fuzzy controller. In this paper, we also follow the usage of this type of NUBS basis functions.

Assume x is a general input variable of a control system which is defined on the universe of discourse $[x_0, x_m]$. Given a sequence of ordered parameters (knots): $(x_0, x_1, x_2, \ldots, x_m)$, the $i-$th normalised B-spline basis function (B-function) $X_{i,k}$ of order k is defined recursively, see Fig. 4. More details are presented in [14].

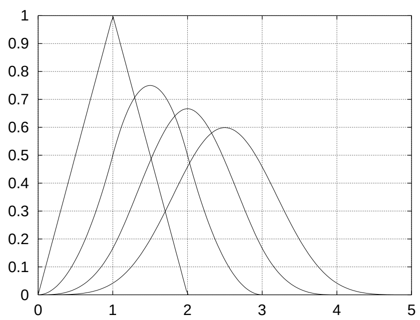


Fig. 4. Non-uniform B-functions of order 2, 3, 4, 5 defined for linguistic terms.

3.2 Local Rules and Their Aggregation

To determine the local control action of a MISO system with n inputs $x_1, x_2, \ldots, x_n$, which are viewed as *linguistic variables*, if we use

- periodical B-spline basis functions interpreted as linguistic terms like "small", "medium", "large" which do not possess crisp boundaries[2];
- singletons to specify local control values;

a local control action can be described in the following form:

IF $(x_1 \text{ is } X^1_{i_1,k_1})$ and $(x_2 \text{ is } X^2_{i_2,k_2})$ and $\ldots$ and $(x_n \text{ is } X^n_{i_n,k_n})$
THEN y is $Y_{i_1 i_2 \ldots i_n}$,

where

[2] In the CMAC network [4] they can be named as active units defined on the overlapping receptive fields of a sensor.

- x_j: the j-th input ($j = 1, \ldots, n$),
- k_j: the order of the B-spline basis functions used for x_j,
- $X^j_{i_j,k_j}$: the i-th linguistic term of x_j defined by B-spline basis functions,
- $i_j = 0, \ldots, m_j$, representing how fine the j-th input is fuzzy partitioned,
- $Y_{i_1 i_2 \ldots i_n}$: the control vertex (deBoor points) of *Rule*$(i_1, i_2, \ldots, i_n)$.

The aggregation of all the local control rules can be represented as:

$$y = \frac{\sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} (Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j))}{\sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j)} \tag{1}$$

$$= \sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} (Y_{i_1,\ldots,i_n} \prod_{j=1}^{n} X^j_{i_j,k_j}(x_j)) \tag{2}$$

This is called a *general NUBS hypersurface*. Building a behaviour can be viewed as the process of shaping the control surface. In CAD applications, the criterion for defining the "ideal" surface can be the visual appearance or some measures like length, curvature, energy, etc. For control applications, they should optimise certain cost functions, e.g. the action-value in the Q-learning paradigm.

We tested a large amount of non-linear functions from low to rather high dimensions, [15]. It was shown that based on the B-spline model, any non-linear MISO functions can be approximated. This feature provides the basis of using this model for modelling general perception-action behaviours.

3.3 Steps for Constructing a Behaviour

The steps for developing a behaviour with B-spline models can be summarised as follows:

1. Select inputs.
2. Select the order of the B-functions for each input variable.
3. Determine the knots for partitioning each input variable.
4. Compute the virtual and real linguistic terms for all inputs.
5. Initialise the control vertices for the output.
6. Learn the control vertices.
7. If the results are satisfied, terminate.
8. Modify the knots for inputs, go to 4;
   or Refine the granularity and use more training data, go to 3;
   or Increase the order of B-functions, go to 3;
   or Delete certain inputs and/or add new ones, go to 2.

In step 3, it is very important to know how the knots should be distributed over the input space. An intuitive answer is to put the knots where the output has its extrema. If such information is available, e.g. by approximating an analytically representable function, we can apply this principle to select the knots.

If the output of a control system is unknown, the knots may first be equally distributed and then adapted with an approach similar to the optimisation of a self-organising neural network.

The control vertices can be initialised with the approximate *a priori* values, e.g. the experience data from experts if available. Otherwise they can be set to zero.

3.4 Adaptation of a Behaviour

Adaptation of a controller is usually transformed into an optimisation process, which often suffers from the problem of runnning into a local instead of a global minimum if numerous parameters affect the cost function in a non-linear, unpredictable manner. If the modification of a single parameter only results in a local change of the control surface the learning speed will increase significantly. We show that the B-spline model possesses this property.

Assume $\{(X,y_d)\}$ is a set of training data, where

- $X = (x_1, x_2, \ldots, x_n)$: is the input vector, and
- y_d : the desired output for X .

The Mean-Sqare-Error is defined as:

$$E = \frac{1}{2}(y_r - y_d)^2, \tag{3}$$

where y_r is the current output value during training.

The parameters to be found arethe local control actions $Y_{i_1,i_2,\ldots,i_n}$, which make the error in (3) as small as possible, i.e.

$$E = \frac{1}{2}(y_r - y_d)^2 \equiv \text{ MIN.} \tag{4}$$

Each control vertex $y_{i_1,\ldots,i_n}$ can be modified by using the gradient descent method:

$$\Delta Y_{i_1,\ldots,i_n} = -\epsilon \frac{\partial E}{\partial Y_{i_1,\ldots,i_n}} = -\epsilon(y_r - y_d)\prod_{j=1}^{n} X^j_{i_j,k_j}(x^j), \qquad 0 < \epsilon \leq 1 \tag{5}$$

This learning function can be classified as a back-propagation method. The only special feature of using B-spline basis function is that the gradient descent method can guarantee that the learning algorithm converges to the global minimum of the error function since the second partial differentiation with respect to $Y_{i_1,i_2,\ldots,i_n}$ is constant:

$$\frac{\partial^2 E}{\partial^2 Y_{i_1,\ldots,i_n}} \;=\; -\,\epsilon(\prod_{j=1}^{n} N^j_{i_j,k_j}(x_j))^2 \;\geq 0 \tag{6}$$

This means that the error function (3) is convex in the space $Y_{i_1,i_2,\ldots,i_m}$ and therefore possesses only one (global) minimum.

In the following, we show how to build elementary behaviours using the adapttion method through a one-deimensional example. Consider a control system with one sensor input and one output of action. Assume that the output should react to the sensor data like a $\sin(2\pi x^2)$ function. The process of adaptation is shown in Fig. 5. The set of symbolic rules interpreting the controller behaviour can be extracted as follows:

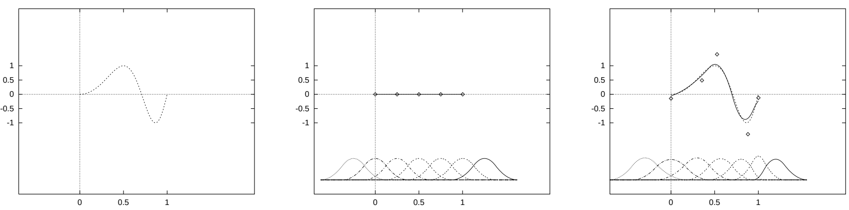
IF S_Reading IS zero THEN Action IS zero

IF S_Reading IS small THEN Action IS positive_middle

IF S_Reading IS medium THEN Action IS positive_big

IF S_Reading IS large THEN Action IS negative_big

IF S_Reading IS maximum THEN Action IS zero



(a) The function $\sin(2\pi x^2)$ (b) The initial output (c) After adaptation

Fig. 5. Mapping the sensor readings into the action values emulating the function $y = \sin(2\pi x^2)$. The B-spline basis functions in (b) and (c) defined on the interval $[0, 1]$ represent the linguistic terms "zero", "small", "medium", "large", "maximum" (from left to right). The values of the diamond-points represent the linguistic terms of the control action, "zero", "positive_medium", "positive_large", "negative_large", "zero".

3.5 Rapid Reinforcement Learning

In unsupervised learning, it is usually possible to define an "evaluation function" if the desired data of the output are unknown. Such an evaluation function should describe how "good" the current system state $((x_1, x_2, \ldots, x_n), y)$ is. For each input vector, an output is generated. With this output, the system transits to another state. The new state is compared with the old one; an adaptation is performed if necessary.

Assume the evaluation function, denoted by $F(\cdot)$, results in a bigger value for a better state, i.e. for two states A and B, if A is better than B, then $F(A) \geq F(B)$. The adaptation of the control vertices can be performed with a similar representation as in supervised learning. Assume that the desired state is A_d. The change of control vertices can be written as:

$$\Delta Y_{i_1,\ldots,i_n} = S \;.\; \epsilon \;.\; |F(B) - F(A_d)| \;.\; \prod_{j=1}^{n} X_{i_j,k_j}(x_j). \tag{7}$$

where $S = sign(F(A) - F(B)) * sign(F(B) - F(A_d)) * sign(y)$ represents the correct direction to modify the control vertex. For more details see [16].

3.6 Situations

Even for the same task, there may exist different evaluation functions in different situations. We use the following example of mobile robots to discuss the situations which the robot can possibly face, Fig. 6.

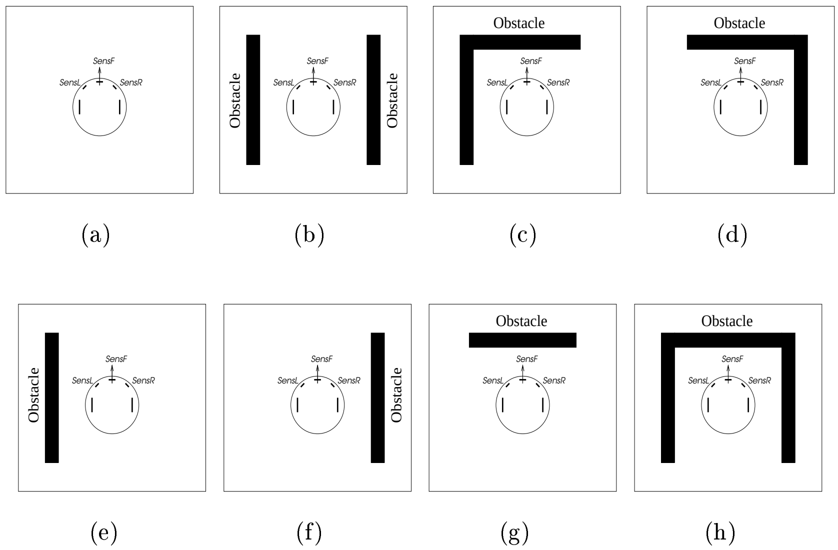


Fig. 6. Possible situations and actions (a): free space, straightforward; (b): in a corridor, straightforward; (c): turn right; (d): turn left; (e): turn right; (f): turn left; (g): turn left; (h): turn left.

In the situations of Fig. 6(b), 6(e), 6(f), the robot should try to keep the difference of *SensL* and *SensR* as small as possible. For the cases shown in Fig. 6(c), 6(d), the robot should try to minimise the sum of all three sensors *SensL*, *SensV* and *SensR*. Fig. 6(g) and 6(h) illustrate two cases, for which no reasonable evaluation function can be found, the robot can simply turn left.

The evaluation function F can be summarised as follows:

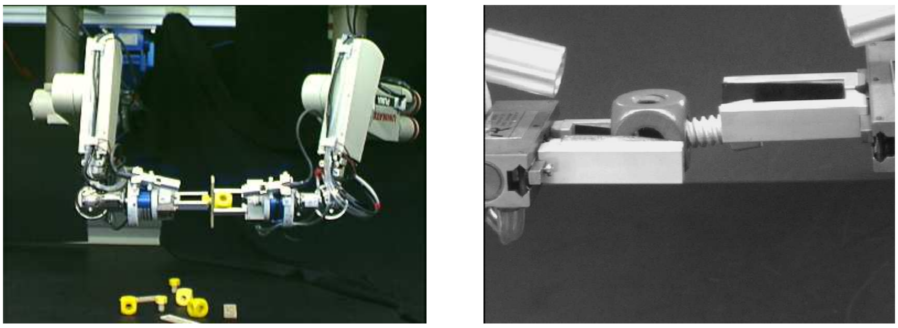
- $F(SensL, SensF, SensR) = -(SensL + SensF + SensR)$, if $SensF$ is big (Fig. 6(c) and 6(d)).
- $F(SensL, SensF, SensR) = -|SensL - SensR|$, if $SensF$ is small, $SensL$ or $SensR$ is not zero (Fig. 6(b), 6(e) and 6(f)).
- $F(SensL, SensF, SensR) = 0$, otherwise (Fig. 6(a)).
- in cases of Fig. 6(g) and 6(h): simply turn left.

4 Sensor-Based Screwing Operation

4.1 Screwing Control Problem

Among assembly operations, insertion and screwing are important for investigating sensor-based control methods, [9]. In order to enhance the flexibility of a robotic system, approaches are necessary which make it possible to control a general-purpose hand/gripper based on sensor inputs. Only with sensors can the diverse uncertainties occurring during different screwing operations be detected and correctly handled.

The problem of the screwing of a bolt into a nut originates from our collaborative project which aims at the assembly of aggregates built from the wooden elements of a toy construction set. The "elevator control" of a toy aircraft was selected as one aggregate to be built, Fig. 7.



(a) Two cooperating manipulators (b) Screwing operation

Fig. 7. The experimental set-up for fixtureless assembly.

For a general purpose arm/gripper system, uncertainties like imprecise grasping, slippage of the part in the hand and vibration must be taken into account. Without using sensors, such an operation can fail under each of these uncertainties. Therefore, sensor-based compensation motions become necessary. The resulting forces in the normal and orientation directions should be minimised and stable. Additionally, to guarantee a successful *screwing-in* phase, a constant force in the approach direction should be exerted.

4.2 Behaviours for a Successful Screwing Operation

The whole screwing skill needs the following behaviours:

Compensation in the approach-direction

Compensation in the normal- and orientation-directions

Screwing

The input information is provided by the force feedback during the motion. In the screwing operation, instead of absolute forces, the deviations of the real forces from the desired ones are used as the input variables, which can be restricted to $\pm 2N$ in our application. The linguistic terms and their definition intervals are specified. At each of both ends of the input range $[-2N, +2N]$, two *virtual linguistic terms* are added to maintain the smooth controllability at the end of the interval [12]. If B-spline basis functions of order three are used, the generated linguistic terms can be seen in Fig. 8, where A_0 and A_9 are virtual linguistic terms, A_1 to A_8 are e.g. $HighNegForceError$, $LowNegForceError$, etc.

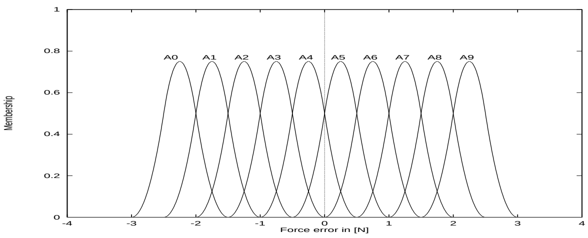


Fig. 8. The basis function for the inputs within the effective range $\pm 2N$.

Linguistic terms of the output variables are defined by control vertices. They can be specified approximately if data for the control process are available, or initialised as zero if there is no *a priori* knowledge. A sample rule is:

IF the deviation from the desired force is very high
THEN the arm should move back in a big stretch

More details of the learning approach can be found in [17].

4.3 Experimental Results

We give an example of screwing with large positioning deviation of the bolt. Fig. 9 illustrates the control curves to compensate the force in the Y-axis i) by approximate initialisation using expert knowledge; ii) after some intermediate learning steps; and iii) after sufficient learning steps, which enables optimal force control for this task.

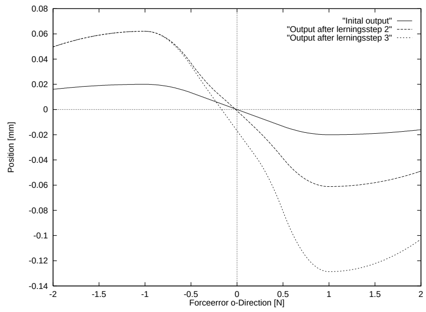


Fig. 9. The control curves during learning.

5 Purposive and Reactive Behaviours of Mobile Robots

B-Spline experiments were also carried out with the mobile robot system shown in Fig. 1(a). Controllers were tested for behaviours like keeping the pre-planned path, avoiding collisions with unknown obstacles, following human instructions and the coordination of them. In the following, we briefly describe three basic modules.

5.1 Approach of Subgoals (SA)

First, we introduce the the behaviour "Subgoal Approaching", which generates the appropriate speed and steering angle to be able to follow the current path segment to the next subgoal. Subgoals can be planned under the given representation of the environment. This module requires the pre-calculation of two variables *shortest distance to path* d and *angle of divergence* α (Fig. 3). The output variables are the robot's forward speed (*Speed*) and steering angle (*Steer*).

By discriminating the relations between the robot's current position and the path segment into the following classes:

- Completely off the path on the left side;
- Far away on the left side;
- Slightly left of the path;
- Almost on the path;
- Slightly right of the path;
- Far away on the right side;
- Completely off the path on the right side,

Rules for path tracking to the next subgoal can be either developed based on heuristic experiments or learned through real practice.

A typical rule of this module looks like this:

IF	The robot is located slightly to the left of the path, but its orientation is almost on the path
THEN	It will steer slightly to the right by applying a high speed

Fig. 10 shows an example of tracking a sequence of pre-planned path segments.

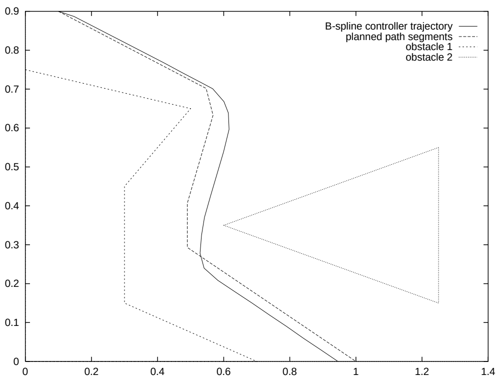


Fig. 10. Trajectory of the controller using the rule base "Subgoal Approaching"

5.2 Local Collision Avoidance (LCA)

The behaviour LCA is assigned to the task to avoid collisions with unknown or moving obstacles. By observing the current values of the proximity sensors, LCA calculates the speed and steering angle, which is required to avoid obstacles. Control rules can be extracted by

- either by modelling the human experiences coping with the following situations: "dead end", "obstacle from right", "obstacle from left", "obstacle ahead", "obstacle from half-left/right", "no obstacle nearby", or
- "learning by doing" with the unsupervised adaptation method.

5.3 Situation Evaluation (SE)

The behaviour "Situation Evaluation" uses the current sensors as input and generates the importance priority K as output. The rule base calculates K for all possible situations.

The output variable K is defined for the importance priority of the LCA rule base. Each specific situation is assigned its importance priority.

K	Situations
Very_Low:	no obstacle avoidance, subgoal approach only
Low:	slightly doing obstacle avoidance, mainly subgoal approach
High:	mainly obstacle avoidance, slightly trying to approach subgoal
Very_High:	obstacle avoidance has priority, subgoal approach is irrelevant

A typical control rule of this module looks like this:

IF The leftmost proximity sensor detects an obstacle which is close, and the other sensors detect no obstacle at all,
THEN Steer halfway to the right at low speed. Mainly perform obstacle avoidance.

5.4 Blending Behaviours

Behaviours can be blended analogously to the blending of single control rules. A *arbiter* with a meta-rule can be described as:

IF situation_evaluation IS for_B_i THEN apply Behaviour B_i

As an example, the coordination of the rule bases LCA and SA is based on the importance priority K. By denoting the *Speed* and *Steer* parameters of both rule bases as $Speed_{SA}$, $Steer_{SA}$ for subgoal approach and $Speed_{LCA}$ and $Steer_{LCA}$ for local collision avoidance, the effective *Speed* and *Steer* becomes:

$$Speed = Speed_{LCA} \cdot K + Speed_{SA} \cdot (1 - K),$$

$$Steer = Steer_{LCA} \cdot K + Steer_{SA} \cdot (1 - K).$$

If more than two rule bases work together, the principle can be further applied. In general, for s rule bases to coordinate, s importance priorities, e.g. $K_1, K2, \ldots, K_s$ should be set. By classifying different situations, the dynamic decision for these parameters can be formulated with control rules and then integrated into the situation evaluation.

Fig. 11 shows two examples of the on-line collision-avoidance as well as the goal approaching behaviour. The starting condition of this scenario is that the robot is originally moving along a straight line towards a goal from bottom to top. In the left figure, the trajectories 1, 2, 3, 4 correspond to the cases of an unanticipated object moving at 10, 25, 50 and 90% of the robot's maximal velocity. Trajectory 4 is a straight course since the robot detects that its path is again free of objects. The right figure shows the object moving towards the robot. Curves 1, 2, 3, 4, 5 correspond to the robot trajectory when the moving object moves head on to the robot or with a deviation of 20, 40, 60 and 80 degrees.

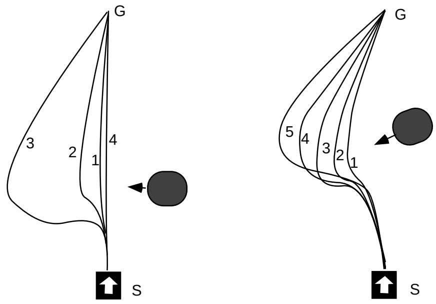


Fig. 11. Approaching a goal while avoiding an unanticipated object.

6 Discussion

We showed that sensor-based behaviours can be incrementally constructed based on a B-spline model. One level up, multiple behaviours can be also coordinated and blended just as multiple single rules. The approach possesses good interpretability, adaptability and generality if the dimension of the input space is limited.

Several advantages resulting from the approach are:

- Knowledge encoding by transforming numerical data to symbolic representation. As a result, huge amount of data is compressed with the "IF-THEN" structure. If the model of the input/output relation is not available, this compression is quite compact. The proposed model can serve as a bridge between numeric input/output data and symbolic control rules.
- Incremental methodology results in the transparency of the behaviour building process. The modular partition of a behaviour in local control rules is actually the reason for rapid convergence of learning. This property benefits from the appropriately selected cost or error function as well as the local influence of control vertices on the whole control surface.
- The combined design/learning methodology. What must be done in the design phase is quite simple: select input variables, determine the granularity of partitioning the input space and some approximate output values if they are available. This ability to integrate human knowledge can be viewed as one distinctive feature of the approach.
- Smooth output. If a B-spline basis function of order k is used, the output is $(k-2)$-times continuously differentiable.

With our approach, the perception-action cycle is finally represented in form of "IF-THEN" rules with optimised parameters. No complex programming and control expertise are needed. Fine-tuning of the main controller parameters can be done on-line and automatically. The method of combining design and learning can be applied to robot systems for acquiring a wide range of sensor-based skills.

References

1. J. S. Albus. A new approach to manipulator control: The Cerebellar Model Articulation Contorller (CMACS). *Transactions of ASME, Journal of Dynamic Systems Measurement and Control*, 97:220–227, 1975.
2. R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, RA-2:14–23, April 1986.
3. M. J. Daily and D. W. Payton. Behaviour-based control for an eye-head system. *SPIE Vol. 1825 - Intelligent Robots and Computer Vision XI*, pages 722–732, 1992.
4. W. T. Miller. Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on System, Man and Cybernetics*, 19:825–831, 1989.
5. F. G. Pin, H. Watanabe, J. Symon, and R. Pattay. Autonomous navigation of a mobile robot using custom-designed VLSI chips and boards. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 123–128, 1992.
6. H. Ritter. Parametrized self-organising maps for vision learning tasks. In *ICANN Proceedings*, 1994.
7. H. Ritter and K. Schulten. Topology conserving mappings for learning motor tasks. In *AIP Conf. Proc. 151, Neural Networks for Computing*, pages 376–380, 1986.
8. A. Saffiotti, E. H. Ruspini, and K. Konolige. Blending reactivity and goal-directness in a fuzzy controller. *IEEE International Conference on Fuzzy Systems*, pages 134–139, 1993.
9. K. Selke and P. Pugh. Sensor-guided generic assembly. In *Proceedings of the 6th International Conference on Robot Vision and Sensory Controls, Paris*, pages 11–19, 1986.
10. H. A. Simon. *The Sciences of the Artificial.* MIT Press, Cambridge, MA, 1969.
11. L. Steels and R. Brooks (editors). *The Artificial Life Route to Artificial Intelligence: building Embodied, Situated Agents.* Lawrance Erlbaum Associates Publishers, 1995.
12. J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models. In *IEEE International Conference on Fuzzy Systems*, 1996.
13. J. Zhang and A. Knoll. Constructing fuzzy controllers for multivariate problems using statistical indices. In *International Conference on Fuzzy Systems, Alaska*, 1998.
14. J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models - principles and applications. *International Journal of Intelligent Systems (forthcoming)*, 13(2/3):257–286, February/March 1998.
15. J. Zhang and K. V. Le. Naturally defined membership functions for fuzzy logic systems and a comparison with conventional set functions. In *European Congress on Intelligent Techniques and Soft Computing, Aachen*, 1997.
16. J. Zhang, K. V. Le, and A. Knoll. Unsupervised learning of control spaces based on B-spline models. *Proceedings of IEEE International Conference on Fuzzy Systems*, Barcelona, 1997.
17. J. Zhang, Y. v. Collani, and A. Knoll. On-line learning of sensor-based control for acquiring assembly skills. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.
18. J. Zhang, F. Wille, and A. Knoll. Modular design of fuzzy controller integrating deliberative and reactive strategies. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, 1996.