In "Fuzzy logic techniques for autonomous vehicle navigation", edited by A. Saffiotti and D. Driankov, Springer, 1999.

Integrating deliberative and reactive strategies via fuzzy modular control

Jianwei Zhang and Alois Knoll

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

Abstract. This work presents the concept and realisation of the integration of deliberative and reactive strategies for controlling mobile robot systems. General motion control at the task-level in a partially known environment is divided into two consecutive steps: subgoal planning and subgoal-guided plan execution. A modular fuzzy control scheme is proposed, which allows independent development and flexible integration of different rule bases, each for fulfilling a certain subtask. An on-line situation evaluator assigns each rule base a weight according to the importance of the subtask. In this way, during motion between subgoals, the robot does not move along a statically planned trajectory but under the control of a sensor-based scheme guided by subgoals.

1 Introduction

This work aims at integrating sensing, path planning and control so that the coupling effects of these three components can be taken into account in order to enhance the performance of the whole system. Planning methods which are totally separated from sensing and control normally assume that the robot environment is completely known. Surveys of path planning methods are given in [3]. Pure geometric path planning in known environments constitutes a deliberative strategy. The approaches following this strategy normally have to solve problems of space division/representation, search and complexity analysis of planning algorithms. By contrast, the reactive strategy regards path planning as a local feedback control problem. The task of local motion control is to determine the motion parameters for driving all the actuators by evaluating the up-to-date sensor information as well as how well a predescribed path is followed.

If we compare the deliberative strategy with the reactive one, the following conclusions can be drawn:

- The advantage of the deliberative strategy is mainly its global planning ability for the whole environment, while the reactive strategy lends itself to taking into account dynamic aspects of the environment and applying sensor data directly to determine the robot path.

– However, methods of the deliberative strategy require that a complete environment model be available. The main disadvantage of the reactive strategy is the problem of the robot running into dead ends because it has no road maps and thus behaves rather "short-sightedly".

Integration of deliberative and reactive strategies will contribute to the solution of robot motion control in a mixed environment with both known and unknown objects, i.e. the robot's environment is only partially known. Off-line modelling can be realised by using CAD data and by applying sensor fusion procedures so that static information can be acquired which represents fixed objects like walls, tables, etc. In the on-line perception phase, data from a sensory system provide the controller with dynamic feedback information for avoiding unknown moving objects like pedestrians and other robots. Therefore, it becomes an interesting problem how to design a control scheme that can fully utilise on-line sensor feedback as well as *a priori* knowledge.

Beyond the numerical approaches, i.e. potential field [1], "intelligent computing methods" like neural networks and fuzzy logic are increasingly applied in sensing, modelling and robot control. In real environments, exact sensor data as well as obstacle models are hard to acquire. Usually a feasible, approximate solution has high priority than an optimal, computation-intensive and noise-sensitive solution. In sensor-based robotics, a domain where human-beings do much better than robots, modelling and then imitating human behaviours are especially meaningful. Fuzzy logic control provides an appropriate tool for these purposes. Recently, applications of the fuzzy control range from purely reactive fuzzy controllers, e.g. [6,10], to the mixture of "behaviours" like single-goal directness and reactive collision-avoidance, e.g. [7,4]. Our work employs the fuzzy control approach and the modular design methodology. A general architecture for integration of planning and execution [2] is based on a supervisor which is situation-driven. Our system fits into this architecture. By using fuzzy meta-fules, similar concept is summarised in [7] as *context-dependent blending*. In our work, we propose the method of generating subgoals for collision-free motion, and then implementing the subgoal approaching as an elementary rule base. An on-line *situation evaluator* determines its priority as well as priorities of other elementary rule bases. Integration of path planning and sensor-based control is realised by designing elementary rule bases and correctly blending them.

This paper is organised as follows: section 2 describes plan subgoals and the selection of sensor data, then introduces the basic idea of the modular fuzzy control scheme. Planning issues for mobile robot systems are discussed in section 3. Section 4 describes the design and implementation of the fuzzy controller and its integration in a control algorithm for subgoal-guided motion. Section 5 draws some conclusions.

2 The concept for integration

Our idea of integrating the deliberative and reactive control strategies lies mainly in generating a set of critical points as subgoals, then using them for globally guiding the robot motion by still leaving some freedom for the plan executor to react to uncertainties about the dynamics of the environment, the precision of environmental data and areas about which nothing or little is known.

2.1 Introduction of subgoals

For applications in a partially known, dynamic environment, the plan executor does not need a detailed geometric path like an interpolated spline curve provided by a planner because some of the path positions may have to be modified anyway due to the unknown static and dynamic obstacles. What is most useful for the on-line motion control is a set of subgoals, e.g. where the robot has to change its direction relatively sharply in order to arrive at the next subgoal position. The main differences between a subgoal and a final goal are as follows:

- Subgoals are much easier to reach than a final goal;
- The robot should usually move continuously through a subgoal point while it should stop at a final point;
- A subgoal can be flexibly generated, communicated to the plan executor and abandoned if necessary;
- Subgoals need not be traversed exactly, where as a final goal is assumed to be fixed and should be reached exactly.

2.2 Sensor data

In order to develop a robust and flexible on-line robot controller, external and internal sensor data should be applied directly in each control cycle instead of being used for building and updating the world model. If sensor data is coupled with motion control in a simple form, the robot can determine its reaction in time. The word "*situatedness*" used by Brooks [9] develops a similar idea. Simon [8] summarised with the concept of "*bounded rationality*" the principle that human-beings often use only incomplete or imprecise knowledge for problem-solving.

Sensor data needed for direct integration into motion control possess the following features:

- Relativity. 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 is not related to the robot or sensor alone, but to the interaction between the robot and its environment.

– Locality. 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 thus transformed to task fusion).
– Task-orientation. Modelling and interpretation of the sensor data depend on the control tasks. Only the control-relevant data are selected, preprocessed and represented.

2.3 The modular fuzzy control scheme

Conventionally, a pre-planned trajectory is executed by a feedback position controller, which utilises the sample of the trajectory as the desired value and the internal position sensor as the true value. With such a controller the data from the external sensors for acquiring *en route* information cannot be integrated into the controller. To solve this problem, we propose the following control structure for realising subgoal-guided, sensor-based robot motions, Fig. 1.

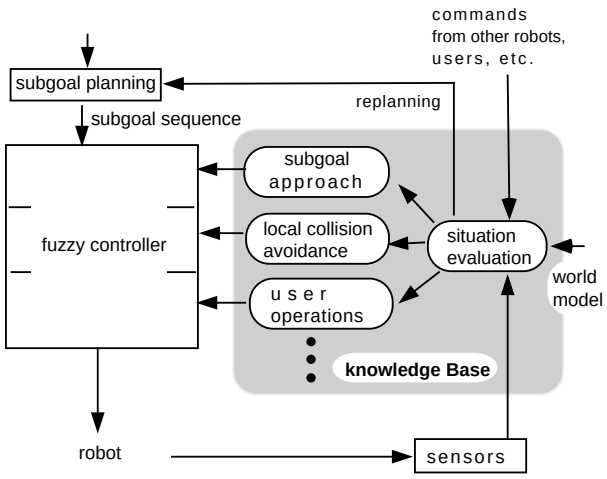


Fig. 1. Integration of subgoal planning and sensor-based plan execution.

Two main rule bases for driving actuators are *subgoal approach (SA)* and *local collision-avoidance (LCA)*. Rule base *SA* is responsible for the smooth traversing of subgoal points. Rule base *LCA* should perform the subtask of avoiding unanticipated local collisions based on external sensor data. In section 4, the ideas and design procedures of rule bases *SA* and *LCA* will be presented in more detail.

Parallel to the rule bases SA and LCA, further modules can be independently designed in the form of rule bases, each for a specified subtask, and they can be added to the knowledge base of the fuzzy controller. In multi-robot applications, the commands of other robots, which arrive through a communication channel, can be viewed as an individual subtask also represented by a rule base. An example scenario is shown in Fig. 2.

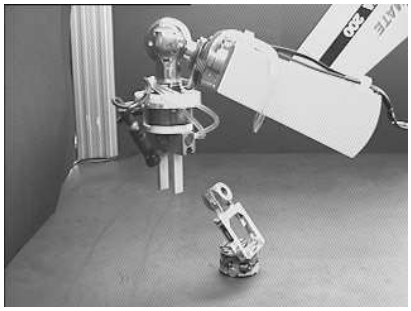


Fig. 2. In order to hand over an object to a manipulator, the mobile robot must receive coordinating commands from other robots or the central supervision system.

If human-robot interaction is desirable, the linguistic interaction can be also defined in rule form and integrated into the knowledge base. For example, a human operator can use spoken instructions, a joystick or special keys of the keyboard to take over the control of the robot. In this case, priority should be assigned to each rule base.

To resolve potential conflicts between the output values, the coordination of the different rule bases becomes very important. Generally, criteria of such a coordinating action are

- robot-specific, i.e. a robot controller can decide by itself the priority of each subtask and use it to modify the influence on the outputs of these rule bases correspondingly;
- situation-dependent, i.e. the priorities of these subtasks are not static and cannot be assigned in advance; they are dynamically determined by the *situation evaluation*.

2.4 The experimental system

The modular fuzzy control scheme has been implemented on a real mobile robot system *Khepera*. *Khepera* is of cylindrical shape with a diameter of 52mm. Additional modules can be mounted on the top of *Khepera*, e.g. a gripper and a vision module. The environment is observed by eight IR sensors (six on the front and two on the back). *Khepera* uses a Motorola MC68331

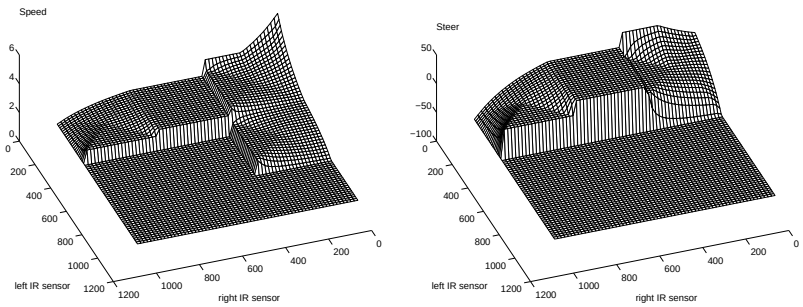
micro-controller, whose instruction set is compatible with the well known MC68020. A RAM size of 128k is available for user programs.

The sensitivity of the IR sensors varies for different objects and is limited to 5cm. The directly controlled variables are the velocities of the robot's left and right wheel, which are denoted by v_l and v_r respectively. In order to test robot independent control programs, the robot's forward speed (*Speed*) and steering angle (*Steer*) are translated to v_l and v_r.

Since the proximity sensors as well as the controller outputs *Speed* and *Steer* are imprecise, it does not make sense to develop complicated, exact algorithms to use the sensor data for world modelling and to control the robot motion with a high resolution. If the control of a mobile robot is compared with the behaviour of a human, it is easily understood that fuzzy logic rules emulating the human decision-making process with "IF-THEN" rules can be applied in the design of such a robot controller.

2.5 Fuzzy logic for sensor-based motion control

The variables of the sensors as well as the control variables can be viewed as *linguistic variables*, such as *Sensor_Left, Sensor_Right, Speed, Steer*. Each linguistic variable can be covered with overlapping *linguistic terms* specified by fuzzy sets, like NB (*negative big*), NM (*negative middle*), NS (*negative small*), Z (*zero*), P (*positive*), PM (*positive middle*) and PB (*positive big*).

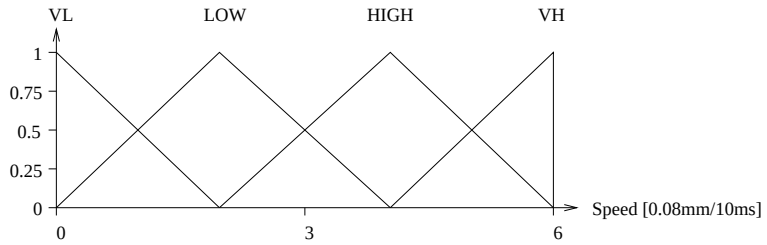


(a) The forward speed. (b) The steering angle.

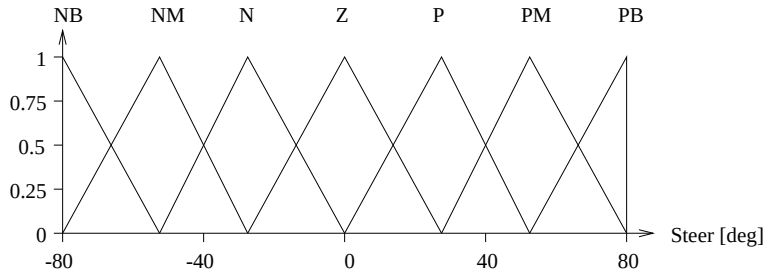
Fig. 3. Mapping sensor data to the control output.

In general, the perception-action relation is a multiple to multiple mapping function. To visualise such a relation with a three-dimensional graphic, we use the following simplified example. Fig. 3 illustrates the procedure of mapping sensor space to control space. This is an example for tracking the

contour of an object, in which the mappings of the input variables *Sensor_Left* and *Sensor_Right* to the output variables *Speed* and *Steer* are shown. The *Speed* output will be assigned a high value when both IR sensors supply a low input (no obstacle in vicinity), and a low value otherwise (Fig. 3(a)). Fig. 3(b) shows the dependency between the IR sensors and the steering angle. *Steer* will be negative if the right IR sensor reading is high and positive if the sensor reading is low, but only if the left IR sensor detects no obstacle on the left side at the same time. In this way the robot follows the contour of an object in clockwise direction. The linguistic terms of the input and output variables can be specified with fuzzy sets using triangles, trapezoids, B-spline basis functions[1], or they can be just selected as a crisp value (fuzzy-singleton). The linguistic terms of variables *Speed* and *Steer* in Fig. 4 are defined by triangular fuzzy sets.



(a) The forward speed.



(b) The steering angle.

Fig. 4. Linguistic terms of the two output variables.

[1] We introduced an approach to model fuzzy sets with B-spline basis functions in [11].

3 Subgoal planning issues

3.1 Planning with a Tangent-graph

If the robot has approximately circular or square shape, the subgoal planning problem can be reduced to a 2-D case by representing the robot as a disc with radius r. Since the dynamic characteristics of the environment make the exact computation of subgoals unnecessary, only a rather conservative approximation of the environment is employed.

Obstacles are assumed to be described as polygons. The obstacle data can be acquired by the data of a building, interactive modelling by CAD system or later by automatic modelling by laser scanner and vision system. These obstacles are enlarged by a constant distance r, where r can be the sum of the robot radius and a safety distance. The robot can be then shrunk to its reference point. In this procedure, edges and sharp vertices of the polygons are extended by r and the intersection points are computed as the new vertices of the enlarged obstacles. After that, planning subgoals consists of finding a sequence of straight lines which do not intersect with the enlarged obstacles and which connect the start and goal points with the shortest distance. This problem can be solved best by searching in a Tangent-graph (T-graph), a simplified V-graph, [5]. The number of arcs in a T-graph is considerably reduced by eliminating non-convex edges and non-tangential lines from the corresponding V-graph. An example of a T-graph is shown in Fig. 5.

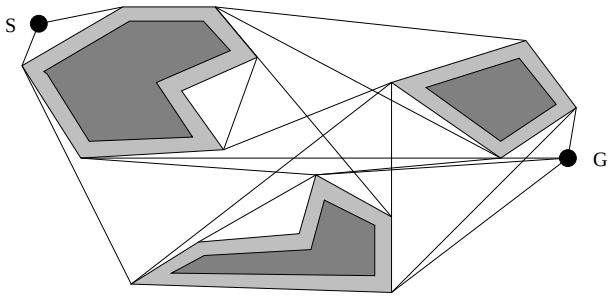


Fig. 5. A T-Graph of enlarged obstacles.

The A* algorithm is used to search for a global route in the T-graph because it can find the shortest path if such a path exists. The nodes of the shortest path from a start position S to a goal position G are a sequence of vertices of the enlarged obstacles. They are viewed as the subgoals for guiding the global direction of the robot motion and can be represented as a sequence $<Q_0, Q_1, \ldots, Q_m>$.

3.2 Planning time

Theoretically, the overall computation complexity of the pre-calculation for constructing a T-graph is on the order of $O(n^2 \log n)$, where n is the number of vertices of the polygons. In the case of the *Khepera* robot using highly optimised fixed point arithmetic instead floating point, we achieve nearly half the speed of a Sparc 5 workstation. The following table shows the time for the calculation of V-graph, T-graph and subgoals of three example environments:

Polygons / Edges	4 / 33	3 / 13	1 / 4
init T-graph	116 ms	21 ms	18 ms
construct V-graph	16241 ms	591 ms	22 ms
construct T-graph	289 ms	44 ms	5 ms
Subgoal plan	538 ms	284 ms	30 ms

4 Design of a fuzzy controller for plan-execution

This section discusses the design of three typical rule bases by using heuristics for classes of situations. The tasks of these rule bases are:

1. Approach subgoals supplied by the planner;
2. Avoid local collisions by evaluating sensor data;
3. Evaluate situations to coordinate the tasks 1 and 2.

The Mamdani-type controller is employed in the design since it allows us to convert heuristics directly into control algorithms.

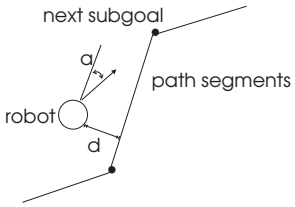


Fig. 6. Combination of planning and internal sensory information as inputs.

4.1 Approaching subgoals

The planning level assigns geometric subgoals for collision-free paths. The geometric distance between a subgoal and the current state, which is estimated by evaluating internal position sensors in the wheels, are be taken as the information for control. We use two variables d and a that are applied to decide on the control action to keep a pre-planned path (Fig. 6):

– Variable *d*: The shortest distance between the robot and the path segment connecting the previous subgoal and the next one (in the following called *path* for short). This linguistic variable d is represented with the following linguistic terms, each of which is defined by the fuzzy sets shown in Fig. 7(a):

  - NB: far off the path to the left;
  - NM: not too far off the path to the left;
  - N: slightly off the path to the left;
  - Z: almost on the path;
  - P: slightly off the path to the right;
  - PM: not too far off the path to the right;
  - PB: far off the path to the right.

– Variable *a*: The angular divergence between orientation of the path and the robot. The following linguistic terms are used (Fig. 7(b)):

  - NB: driving in the opposite direction, slightly to the left;
  - NM: direction is totally off the path to the left;
  - N: direction is slightly off the path to the left;
  - Z: almost on the path segment;
  - P: direction is slightly off the path to the right;
  - PM: direction is totally off the path to the right;
  - PB: driving in the opposite direction, slightly to the right.

SA generates the following output variables:

– *Speed*: The speed of the robot.
– *Steer*: The steering angle, based on the current direction of movement.

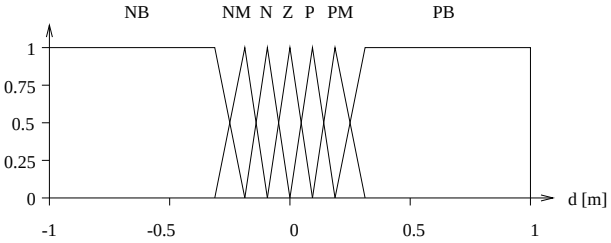
By classifying "situations" of the robot's current position to the path segment to be tracked, rules for path tracking to the next subgoal can be developed. In appendix A, 49 rules for "subgoal approaching" are listed. It is the task of the main control program to verify the current robot position and to switch to the next path segment.

A typical fuzzy rule of this module looks like this:

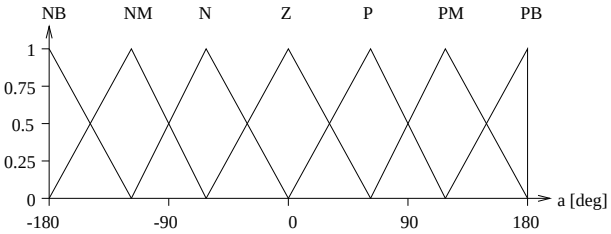
IF (d IS N) AND (a IS Z) **THEN** (*Speed* IS HIGH) AND (*Steer* IS P),

which is the fuzzy logic representation of the following heuristic rule: "If the robot is located slightly to the left of the path but its orientation is almost on the path, then it must steer slightly to the right at a high speed."

Fig. 8 shows an example of the trajectory, realised by the fuzzy controller, to track a sequence of pre-planned path segments. The processing of the 49 rules for *Khepera* takes only 3 ms.



(a) The shortest distance from path.



(b) The angular divergence.

Fig. 7. Linguistic terms of two inputs for "subgoal approach".

4.2 Local collision avoidance

Typically, for local collision avoidance we need to determine the value of five proximity sensors, e.g. infrared or ultrasonic sensors (left, half-left, front, half-right and right) if we approximately view the control as a Markov decision process, i.e. the control action is merely determined by the current sensor value and not by the historical sensor readings. The LCA rule base tries to avoid collisions with unknown or dynamic obstacles. By observing the current values of the five proximity sensors, LCA calculates the speed and steering angle, which is needed to avoid obstacles. The input variables are[2]:*SL85 SL85*, *SL45*, *SLR0*, *SR45*, *SR85*, the current value of the proximity sensors. The four linguistic terms are based on triangular membership functions, which have different distances from each other because of the non-linearity of these sensors.

- VL: no obstacle in sight;
- LOW: an obstacle is far away;
- HIGH: an obstacle is close;

[2] They are referring to the IR sensors arranged at different angles, e.g. "sensor on the left at angle 85°".

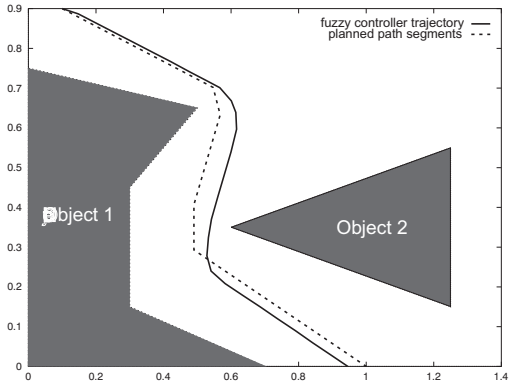


Fig. 8. Trajectory of the controller using the rule base "subgoal approach".

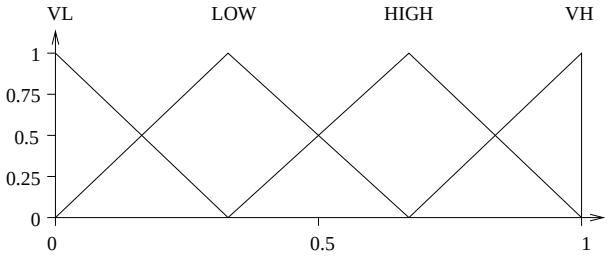
– VH: almost in touch with an obstacle.

The fuzzy rules can be extracted by modelling human experience when coping with the following situations: "dead end", "obstacle from right", "obstacle from left", "obstacle ahead", "obstacle from half-left/right", "no obstacle nearby".

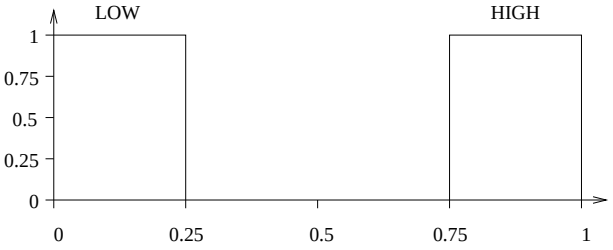
4.3 Situation evaluation

The rule base "situation evaluation" of the *Khepera* robot uses the "near-sighted" proximity sensors as input and generates two output variables: the priority K and the re-planning selector *Replan*. The rule base calculates the priority of each module for all possible situations.

– K: the priority for the LCA rule base, normalised in [0,1]. Each specific situation is assigned its priority (Fig. 9 (a)):
  - VL: no obstacle avoidance, subgoal approach only;
  - LOW: put little emphasis on obstacle avoidance, mainly try to approach subgoal;
  - HIGH: mainly obstacle avoidance, slightly try to approach subgoal;
  - VH: obstacle avoidance has priority, subgoal approach is irrelevant.
– *Replan*: decide if a "no-way-out" situation, which requires the path planning procedure to be invoked once again, is reached. That will be indicated by a high value in *Replan* (normalised in [0,1]) (Fig. 9 (b)). A typical case is that the next subgoal is occupied by an obstacle. In this situation, the robot can only be freed by inhibiting the next subgoal and planning a new subgoal sequence.
  - LOW: no re-planning required;
  - HIGH: re-planning required.



(a) *Priority.*



(b) *Re-planning.*

Fig. 9. Linguistic terms for two state variables.

The fused rule bases for local collision avoidance and situation evaluation are listed in appendix B.

A typical fuzzy rule of this module looks like this:

IF (*SL85* IS HIGH) AND (*SL45* IS VL) AND (*SLR0* IS VL) AND (*SR45* IS VL) AND (*SR85* IS VL) **THEN** (*Speed* IS LOW) AND (*Steer* IS PM) AND (*K* IS HIGH) AND (*Replan* IS LOW)

The interpretation of the above rule is:

"If the leftmost proximity sensor detects an obstacle which is near, and the other sensors detect no obstacle at all, then steer halfway to the right at low speed. Mainly perform obstacle avoidance. No re-planning required."

4.4 Coordinating LCA and SA

Rule bases can be blended analogously to the blending of single control rules. A meta-rule can be described as:

"IF situation_evaluation IS for_$RuleBase_i$ THEN apply $RuleBase_i$"

The coordination of the rule bases LCA and SA is based on the priority K, see also [4]. The value of K is determined by heuristic rules (see Appendix B), which are tuned slightly in the experiment to guarantee collision-free motions as well as correct subgoal approaching. 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:

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

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

In general, the *situation evaluation* considers the sensor information and provides each rule base with an individual weight. This modular concept can be extended by adding further control modules which are implemented by fuzzy rules. Each new subtask receives its rule base which will be added in the knowledge base of the robot controller. For n rule bases to coordinate, n priorities, e.g. $K_1, K_2, \ldots, K_n$ should be set. By classifying different situations, the dynamic decision for these parameters can be formulated with fuzzy rules and then integrated in the *situation evaluation*. The linear relationship between the *priority* variable K_i $(i = 1, \ldots, n)$ and the m defuzzified control variables of the n rule bases, $y_{i,j}$ $(j = 1, \ldots, m)$ can be expressed as follows:

$$Y_i = y_{1,i} \cdot K_1 + y_{2,i} \cdot K_2 + \ldots + y_{n,i} \cdot K_n \; with \; i = 1, \ldots, m$$

where Y_i represents the direct value of the i-th control variable. In normal cases, all modular rule bases possess the same control variables. *Situation evaluation* will provide values for special control variables for which no weighting is necessary.

4.5 Implementation

The flow chart of the robot control program is shown in Fig. 10. Experiments have demonstrated the nice modular features of this concept, Fig. 11. The rule base *SA* alone works well for realising its subgoal approaching subtask in a completely known environment. As expected, the test in a completely unknown environment with the rule base *LCA* shows that collisions with obstacles can be avoided, but the robot can possibly move into a dead-end or a cycle. In a partially known environment, *SA* and *LCA* are coordinated by the rule base *situation evaluation* and realise the global subgoal-guided collision-free motion. In this way, during motion between subgoals, the robot does not move along a statically planned trajectory but under the control of a subgoal-guided, sensor-based controller. On-line sensor data can be evaluated to detect local collisions and the motion control is adapted to the dynamic environment.

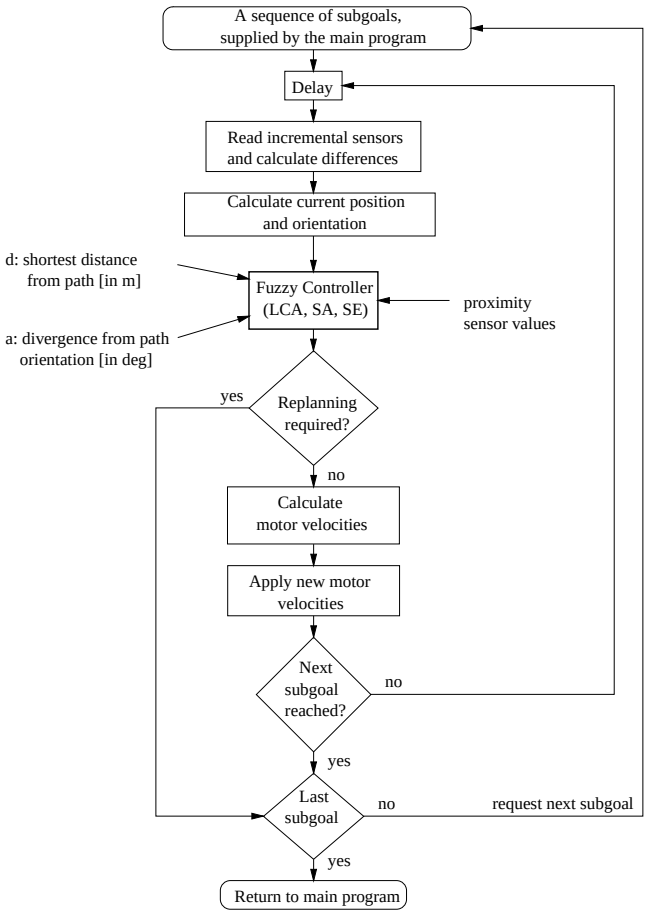


Fig. 10. Flow chart for integrating "subgoal planning" and "local collision avoidance".

Fig. 12 shows the subgoal approach in a completely known environment. The robot follows a pre-planned subgoal segments and keeps adequate distance at the two vertices (here two subgoals) with the help of the local control.

Fig. 13 shows the same environment as in Fig. 12 with an unknown obstacle, which blocks the robot's movement to the first subgoal. The robot drives around the side where most free space is available and at the same time moves to the second subgoal because, after avoiding the obstacle, the first subgoal has already been passed by.

In Fig. 14 the robot is on its direct path to the goal from bottom to top. In Fig. 14(a), a dynamic obstacle is crossing the robot's path. The trajectories 1, 2, 3 and 4 correspond the test cases, in which an unanticipated obstacle moves with 10%, 25%, 50% and 90% of the robot's maximum speed. Trajectory 4

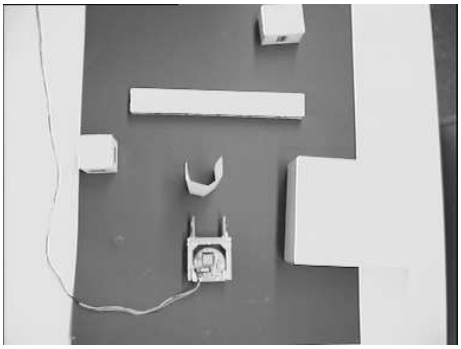


Fig. 11. A test environment.

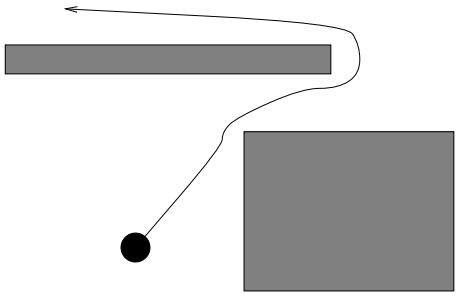


Fig. 12. Subgoal approaching in a known environment.

becomes straight again since the obstacle has already crossed the path before the robot arrives. Curves 1, 2, 3, 4, 5 in Fig. 14(b) 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.

5 Conclusions

The fuzzy control scheme is used for executing subgoal-guided motions. Fuzzy rule bases, e.g. for local collision-avoidance, can work together with the rule base for passing through subgoals, each of which with only a limited number of control rules. The main advantages of using fuzzy control for mobile robots can be summarised as follows:

Modularity. Fuzzy control is intrinsically modular: a rule base is generated by elaborating each single rule, which has a linguistic interpretation and its own control function. The order of these rules is does not make a difference, both during controller design and rule evaluation. If we regard a rule base performing a certain subtask as a separate module, it is easy

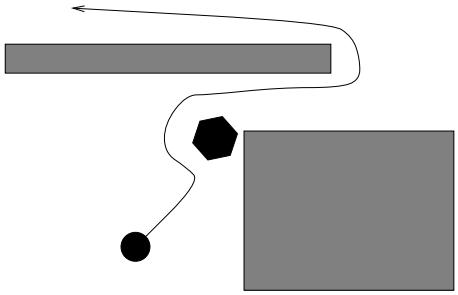


Fig. 13. Subgoal approaching with an unknown obstacle.

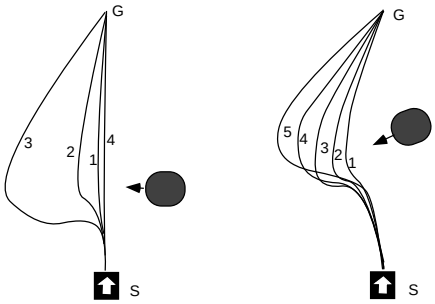


Fig. 14. Collision-avoidance with a dynamic obstacle.

to understand that different rule bases can be developed independently and then evaluated together for realising a high-level task.

Efficiency. The modular design enables a significant reduction of development time, which is achieved by simple design of a single rule base, rapid prototyping and efficient debugging. Further fuzzy rule bases, such as for dealing with the commands from other robots or a human user, can be separately developed by using either heuristics or training. Thanks to the simple computation and the possibility of parallel processing of fuzzy rules, fuzzy controllers can run in real-time with moderate computing power requirements.

Transparency. Since the sensor-based robot control strategy takes advantages of the heuristics of human experience, the control procedure is still transparent, which is an important property of the intelligent control philosophy. The transparent mapping from input space to action contributes to solving: a) the skill transfer from human experts to robots; b) the analysis and validation of the controller development; c) supervision of the learning process.

Low-cost. The fuzzy control concept utilises the sensor data qualitatively rather than quantitatively. Imprecise data like infra-read sensor readings and low-resolution gray-level camera images can be applied efficiently.

Based on simple, intuitive control rules, tracking of a pre-planned path is not very exact. However, precise path tracking in a partially known environment is not necessary at all.

Adaptability. By selecting a special type of fuzzy controller, e.g. the B-spline type and designing appropriate learning algorithms, the robot controller can be self-optimised or even totally trained automatically.

The limitations of the applied approach are:

- The real-time subgoal planning is only possible for a robot whose shape is approximately circular or square like. If obstacles in the robot's environment are not sparsely distributed and rotation of the robot is needed to cross some tight passages, the two-dimensional T-graph will no longer be adequate for subgoal planning. In such cases, the three-dimensional configuration space will be needed. Unfortunately, fast computation of subgoals realised in this work cannot be guaranteed in general with the current on-board computers.
- To correctly develop fuzzy rules, all possible situations should be taken into account. On-line tuning of membership functions needs a lot of "trial-and-error" procedures. Automatic evolution of fuzzy control rules and on-line self-tuning of membership functions based on B-splines can contribute to solving the problem (which is our current work [13,14]).
- The number of proximity sensors applied in our experiment is small. If multiple sensors and/or vision systems are used, the "curse of dimensionality" will occur since the number of complete rules in a fuzzy controller grows exponentially with the number of controller inputs. Automatic input and feature selection becomes a necessity. Another focus of our current work is to combine techniques of dimension reduction like "principal component analysis", "output-related features", to make the use of fuzzy control approach for complex sensor patterns possible and easier, [12].

Acknowledgement

The authors wish to thank Frank Wille for implementing the fuzzy rule bases and experimenting with the *Khepera* robot. We would also like to acknowledge the valuable comments of the anonymous reviewers that greatly helped us to improve the final version of the paper.

References

1. J. Barraquand, B. Lanlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on System, Man and Cybernetics*, 22(2):224–241, 1992.
2. R. Chatila. Deliberation and reactivity in autonomous mobile robots. *Journal of Robotics and Autonomous Systems*, pages 197–211, 1995.

3. Y. K. Huang and N. Ahuja. Gross motion planning – a survey. *ACM Computer Surveys*, 24(3):219–291, September 1992.
4. S. Ishikawa. A method of autonomous mobile robot navigation by using fuzzy control. *Advanced Robotics*, 9(1):29–52, 1995.
5. Y.-H. Liu and S. Arimoto. Proposal of tangent graph and extended tangent graph for path planning of mobile robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991.
6. F. G. Pin and Y. Watanabe. Driving a car using reflexive fuzzy behavior. *IEEE International Conference on Fuzzy Systems*, pages 1425–1430, 1993.
7. 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.
8. H. A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.
9. L. Steels and R. Brooks (editors). *The Artificial Life Route to Artificial Intelligence: building Embodied, Situated Agents*. Lawrance Erlbaum Associates Publishers, 1995.
10. H. Surmann, J. Huser, and L. Peters. A fuzzy system for indoor mobile robot navigation. In *IEEE International Conference on Fuzzy Systems*, pages 83–86, 1995.
11. J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models - principles and applications. *International Journal of Intelligent Systems*, 13(2/3):257–285, Feb./Mar. 1998.
12. J. Zhang and A. Knoll. Situated neuro-fuzzy control for vision-based robot localisation. *Robotics and Autonomous Systems*, (to appear) 1999.
13. J. Zhang, K. V. Le, and A. Knoll. Unsupervised learning of control spaces based on B-spline models. In *Proceedings of IEEE International Conference on Fuzzy Systems, Barcelona*, 1997.
14. J. Zhang and V. Schwert. Rapid learning of sensor-based behaviours of mobile robots based on B-spline fuzzy controllers. In *Proceedings of the IEEE International Conference on Fuzzy Systems, Anchorage*, 1998.

A Appendix A - Rule Base SA (*Subgoal Approach*)

Rules for tracing the path and approach the next subgoal, with

a = *angle between the orientation of the robot and the planned path segment*, and

d = *shortest distance between path and robot.*

Input		Output	
d	a	*Steer*	*speed*
Completely off the path on the left side			
NB	NB	PB	LOW
NB	NM	PB	LOW
NB	N	PM	LOW
NB	Z	PM	HIGH
NB	P	P	HIGH
NB	PM	Z	VH
NB	PB	N	HIGH
Far away on the left side			
NM	NB	PB	LOW
NM	NM	PB	LOW
NM	N	PM	LOW
NM	Z	PM	HIGH
NM	P	P	HIGH
NM	PM	Z	HIGH
NM	PB	N	HIGH
Slightly left of the path			
N	NB	PB	LOW
N	NM	PB	LOW
N	N	PM	HIGH
N	Z	P	HIGH
N	P	Z	VH
N	PM	N	HIGH
N	PB	NB	LOW
Almost on the path			
Z	NB	PB	LOW
Z	NM	PM	LOW
Z	N	P	HIGH
Z	Z	Z	VH
Z	P	N	HIGH
Z	PM	NM	LOW
Z	PB	NB	LOW
Slightly right of the path			
P	NB	PB	LOW
P	NM	P	HIGH
P	N	Z	VH
P	Z	N	HIGH
P	P	NM	HIGH
P	PM	NB	LOW
P	PB	NB	LOW
Far away on the right side			
PM	NB	P	HIGH
PM	NM	Z	HIGH
PM	N	N	HIGH
PM	Z	NM	HIGH
PM	P	NM	LOW
PM	PM	NB	LOW
PM	PB	NB	LOW
Completely off the path on the right side			
PB	NB	P	HIGH
PB	NM	Z	VH
PB	N	N	HIGH
PB	Z	NM	HIGH
PB	P	NM	LOW
PB	PM	NB	LOW
PB	PB	NB	LOW

B Appendix B - Rule Base LCA (*Local Collision Avoidance*) and SE (*Situation Evaluation*)

Input					LCA output		SE output	
SL85	SL45	SLR0	SR45	SR85	Sp.	St.	K	Repl.
Dead end situation. Requires re-planning.								
VH	VH	-	VH	VH	VL	Z	VH	HIGH
HIGH	VH	VH	VH	VH	VL	Z	VH	HIGH
VH	HIGH	VH	VH	VH	VL	Z	VH	HIGH
VH	VH	VH	HIGH	VH	VL	Z	VH	HIGH
VH	VH	VH	VH	HIGH	VL	Z	VH	HIGH
HIGH	HIGH	VH	VH	VH	VL	Z	VH	HIGH
VH	HIGH	HIGH	VH	VH	VL	Z	VH	HIGH
VH	VH	HIGH	HIGH	VH	VL	Z	VH	HIGH
VH	VH	VH	HIGH	HIGH	VL	Z	VH	HIGH
Collision avoidance in free space - Obstacle from right								
VL	VL	VL	VL	LOW	HIGH	N	LOW	LOW
VL	VL	VL	LOW	LOW	LOW	NM	LOW	LOW
VL	VL	LOW	LOW	LOW	LOW	NB	HIGH	LOW
VL	LOW	LOW	LOW	LOW	LOW	NB	HIGH	LOW
VL	VL	VL	VL	HIGH	LOW	NM	HIGH	LOW
VL	VL	VL	LOW	HIGH	VL	NB	HIGH	LOW
VL	VL	LOW	LOW	HIGH	VL	NB	VH	LOW
VL	VL	VL	HIGH	HIGH	VL	NB	VH	LOW
VL	VL	HIGH	HIGH	HIGH	VL	NB	VH	LOW
VL	VL	VL	VL	VH	VL	NB	VH	LOW
VL	VL	VL	LOW	VH	VL	NB	VH	LOW
VL	VL	VL	HIGH	VH	VL	NB	VH	LOW
VL	VL	LOW	HIGH	VH	VL	NB	VH	LOW
VL	LOW	HIGH	HIGH	VH	VL	NB	VH	LOW
VL	VL	VL	VH	VH	VL	NB	VH	LOW
VL	VL	LOW	VH	VH	VL	NB	VH	LOW
VL	VL	VH	VH	VH	VL	NB	VH	LOW
VL	LOW	VH	VH	VH	VL	NB	VH	LOW
LOW	HIGH	VH	VH	VH	VL	NB	VH	LOW
Collision avoidance in free space - Obstacle from left								
LOW	VL	VL	VL	VL	HIGH	P	LOW	LOW
LOW	LOW	VL	VL	VL	LOW	PM	LOW	LOW
LOW	LOW	LOW	VL	VL	LOW	PB	HIGH	LOW
LOW	LOW	LOW	LOW	VL	LOW	PB	HIGH	LOW
HIGH	VL	VL	VL	VL	LOW	PM	HIGH	LOW
HIGH	LOW	VL	VL	VL	VL	PB	HIGH	LOW
HIGH	LOW	LOW	VL	VL	VL	PB	VH	LOW
HIGH	HIGH	VL	VL	VL	VL	PB	VH	LOW
HIGH	HIGH	HIGH	VL	VL	VL	PB	VH	LOW
VH	VL	VL	VL	VL	VL	PB	VH	LOW
VH	LOW	VL	VL	VL	VL	PB	VH	LOW
VH	HIGH	VL	VL	VL	VL	PB	VH	LOW
VH	HIGH	LOW	VL	VL	VL	PB	VH	LOW
VH	HIGH	HIGH	LOW	VL	VL	PB	VH	LOW
VH	VH	VL	VL	VL	VL	PB	VH	LOW
VH	VH	LOW	VL	VL	VL	PB	VH	LOW
VH	VH	VH	VL	VL	VL	PB	VH	LOW
VH	VH	VH	LOW	VL	VL	PB	VH	LOW
VH	VH	VH	HIGH	LOW	VL	PB	VH	LOW
Avoiding direct collision with obstacle ahead								
VL	VL	LOW	VL	VL	LOW	Z	HIGH	LOW
VL	VL	HIGH	VL	VL	VL	Z	VH	LOW
VL	VL	VH	VL	VL	VL	PB	VH	LOW
VL	LOW	HIGH	LOW	VL	LOW	Z	VH	LOW
VL	HIGH	HIGH	HIGH	VL	VL	Z	VH	LOW
VL	HIGH	VH	HIGH	VL	VL	PB	VH	LOW
VL	VH	VH	VH	VL	VL	PB	VH	LOW
Avoiding direct collision with obstacle from half-left/right								
VL	LOW	HIGH	VL	VL	LOW	PM	VH	LOW
VL	LOW	VH	VL	VL	VL	PB	VH	LOW
VL	LOW	LOW	VL	VL	LOW	PM	HIGH	LOW
VL	HIGH	VH	VL	VL	VL	PB	VH	LOW
VL	VH	HIGH	VL	VL	VL	PB	VH	LOW
VL	VH	VH	VL	VL	VL	PB	VH	LOW
LOW	HIGH	HIGH	LOW	VL	VL	PB	VH	LOW
HIGH	VH	HIGH	VL	VL	VL	PB	VH	LOW
HIGH	VH	VH	LOW	VL	VL	PB	VH	LOW
HIGH	VH	VH	HIGH	VL	VL	PB	VH	LOW
HIGH	VH	VH	HIGH	LOW	VL	PB	VH	LOW
VL	VL	HIGH	LOW	VL	LOW	NM	VH	LOW
VL	VL	VH	LOW	VL	VL	NB	VH	LOW
VL	VL	LOW	LOW	VL	LOW	NM	HIGH	LOW
VL	VL	VH	HIGH	VL	VL	NB	VH	LOW
VL	VL	HIGH	VH	VL	VL	NB	VH	LOW
VL	VL	VH	VH	VL	VL	NB	VH	LOW
VL	LOW	HIGH	HIGH	LOW	VL	NB	VH	LOW
VL	VL	HIGH	VH	HIGH	VL	NB	VH	LOW
VL	LOW	VH	VH	HIGH	VL	NB	VH	LOW
VL	HIGH	VH	VH	HIGH	VL	NB	VH	LOW
LOW	HIGH	VH	VH	HIGH	VL	NB	VH	LOW
No obstacle in vicinity								
VL	VL	VL	VL	VL	VH	Z	VL	LOW