Modular Design of Fuzzy Controller Integrating Deliberative and Reactive Strategies

Jianwei Zhang, Frank Wille and Alois Knoll
Technical Computer Science, 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. Programming 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. The design process of the fuzzy controller is demonstrated with the examples of three rule bases.

1 Introduction

This work aims at integrating sensing, planning and control so that the coupling effects of these three components can be taken into account and then the performance of the whole system can be enhanced. Planning methods which are totally separated from the sensing and control procedures normally assume that the robot motion environment is completely known. A survey of path planning methods is given in [2]. Pure geometric path planning in known environments uses a deliberative strategy. The approaches following this strategy normally have to solve problems of space division/representation, searching 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 a predescribed path.

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

- While the advantage of the deliberative strategy mainly lies in the overview of the whole reachable space, 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 must presuppose that a complete environment model is available. The main disadvantage of the reactive strategy is the so-called "dead-end" problem, because the robot does not possess the ability to read 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 known and unknown objects. In most cases in the real world, the environment is partially-known. On the off-line modelling side, with the help of CAD data and by applying a sensor fusion procedure, static information can be acquired, which represents fixed objects like walls, tables, etc. On the on-line perception side, data from a sensory system provide the controller with dynamic feedback information for detecting slightly moved objects and for avoiding unknown objects like clutters, pedestrians and other robots. Therefore it becomes an interesting problem how to design a control scheme which can fully utilise on-line sensor feedback as well as a priori knowledge.

Beyond the classical control approaches, like potential field [1], "intelligent computing methods", like neural networks and fuzzy logic are increasingly applied in sensing, modelling and robot control. Our work employs the fuzzy control approach and the modular design methodology. Actually, the principle of fuzzy control is intrinsically modular: a rule base is generated by increasingly developing each single rule, which has linguistic interpretations and its own control function. The order of these rules does not make any difference, both during the controller design and the rule evaluation. If we regard a rule base for fulfilling a certain subtask as a separate module, it is easy to understand that different rule bases can be developed independently and then integrated for realising a high-level task like collision-free motion from a given pair of start and goal position. Fuzzy control is becoming gradually as a main approach of robot sensor-based control. Applications range from the purely reactive fuzzy controller, e.g. [5], to the mixture of "behaviours" like single-goal directness and reactive collision-avoidance, e.g. [3] and [6].

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

2 The Concept for Integration

Our idea of integrating these two control strategies lies mainly in generating a set of critical points as subgoals, then using them for globally guiding the robot motion and still leaving some freedom for the plan executor to react to uncertainties.

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 since 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 the next subgoal position. The main differences between a subgoal and a final goal are as follows:

- Subgoals should be 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, assigned to the plan executor and abandoned if necessary;
- Subgoals need not be traversed exactly, while a final goal is assumed to be fixed and should be exactly reached.

2.2 Modular Fuzzy Control Scheme

Conventionally, a preplanned trajectory is executed by a feedback position controller, which uses the sample of the trajectory as the desired value and the internal position sensor as the real 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.

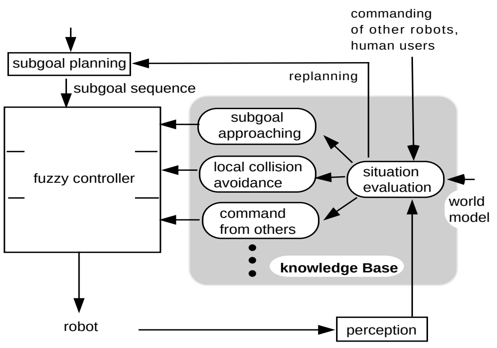

Figure 1: Integration of subgoal planning and sensor-based plan execution based on fuzzy control

Two main rule bases for driving actuators are *subgoal approaching (SA)* and *local collision-avoidance (LCA)*. Rule base *SA* is responsible for the smooth tracing of subgoal points. Rule base *LCA* should perform the subtask for avoiding unanticipated local collisions based on 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 be put into the knowledge base of the fuzzy controller. In multi-robot applications, the command of another robot, which arrives through a communication channel, can be viewed as an individual subtask and represented by a rule base. If human-robot interaction is desirable, the linguistic interference can be also defined, developed and then integrated into the knowledge base.

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

- robot-specific, i.e. a robot controller can decide by itself the importance priority of each subtask and use it to modify the influence on the outputs of these rule bases correspondingly.

- situation-dependent, i.e. the importance priorities of these subtasks are not static and cannot be preassigned; they are dynamically determined by the *situation evaluation*.

3 Subgoal Planning Issues

3.1 Planning with a Tangent-graph

The subgoal planning problem is simplified 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. They are enlarged by a constant distance r. The robot is then shrunk to its reference point. In this procedure, edges and sharp vertices of these 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 connecting the start and goal points with the shortest distance which do not intersect with the enlarged obstacles. This problem can be solved best by searching in a Tangent-graph (T-graph), a simplified V-graph, [4]. 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. 2.

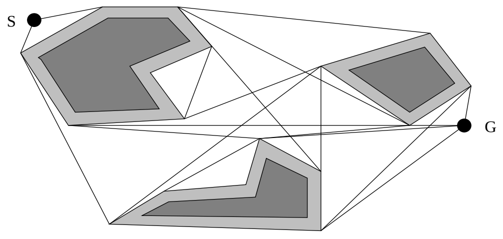

Figure 2: A T-Graph of enlarged obstacles

The A* algorithm is used to search for a global route in the T-graph since 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 A Mobile Robot System for Experiments

This concept has been implemented for a real mobile gripper system *Khepera*, Fig. 3. *Khepera* is of circular 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 microcontroller, whose instruction set is compatible with the well known MC68020. A RAM size of 128k is available for user programs.

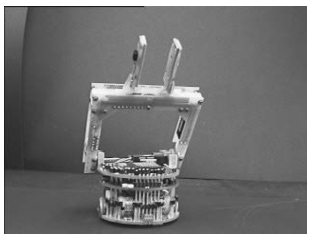

Figure 3: A mobile robot gripper system for experiments

3.3 Planning Time

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

Polys/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

The modular fuzzy controller was developed with the *TIL Shell* of *Togai Infra Logic*. The local collision avoidance as well as the subgoal approaching were realised by fuzzy *if-then* rules.

4.1 Approach of Subgoals

For briefness, we introduce the following rule base SA for tracing the path to the next subgoal. It is required to pre-calculate the two input variables *shortest distance to path* and *angle of divergence*:

- 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).
- a: The angular divergence between orientation of the path and the robot.

SA generates the following output variables:

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

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

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.

The 49 rules of subgoal approaching can be found in Appendix A. The implementation of this rule base for *Khepera* takes a computation time of 3 ms.

4.2 Local Collision Avoidance

Typically, for local collision avoidance we need to determine the value and change of five proximity sensors, e.g. infrared or ultrasonic sensors (left, half-left, front, half-right and right). The LCA rule base tries to avoid collisions with unknown or dynamic obstacles. By observing the current values and the change of the five proximity sensors LCA calculates the speed and steering angle, which is needed to avoid obstacles. The input variables are:

- *SL85*, *SL45*, *SLR0*, *SR45*, *SR85*: Current value of the proximity sensors (left, half-left, front, half-right, right).
- *dSL85*, *dSL45*, *dSLR0*, *dSR45*, *dSR85*: They represent the difference between the current and last sensor value.

LCA generates the same *Speed* and *Steer* output variables as SA. A rule set is listed in Appendix B.

4.3 Situation Evaluation

Situation evaluation generates two parameters: the importance priority K, the replanning selector *Replan*.

- K: Importance priority for the LCA rule base. Each specific situation has its importance priority assigned.
- *Replan*: Decides if a situation, which requires the path planning procedure to be invoked once again, is reached. That will be indicated by a high value in *Replan*.

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 K IS high AND *Replan* IS low

If the leftmost proximity sensor detects an obstacle, which is near and the other sensors detect no obstacle at all, then mainly perform obstacle avoidance. No replanning of the path is required.

The implementation of this rule base for *Khepera* takes a computation time of between 8 and 14 ms.

4.4 Coordinating LCA and SA

The coordination of the rule bases LCA and SA is based on the importance priority K, see also [3]. 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 function together, such a principle can be further applied. In general, for n rule bases to coordinate, n importance priorities, e.g. $K_1, K2, \ldots, K_n$ 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*.

4.5 An Algorithm for Motion Control along Subgoals

In order to integrate sensor-based maneuver as well as subgoal approaching in the motion control process, an algorithm *Motion_Control* was developed. The task of this algorithm is to guide the robot from a start position along a given subgoal sequence to the goal position with the help of the rule bases of the fuzzy controller. At the position of the next subgoal, a *finish-line* is defined to be orthogonal to the path segment. When the finish-line is crossed, the algorithm switches to the next path segment. The whole algorithm is presented as follows:

```
Algorithm Motion_Control(in SGL, in head$_0$);
Inputs: SGL   - a list of subgoals <$Q_0, Q_1, \ldots, Q_m$ >;
        head$_0$ - start heading of robot;
{  /* Initialising (set motor pid, etc.) */
   i := 0;            /* subgoal index */
   POS := $Q_0$;       /* start position of robot */
   HDG := head$_0$; /* start heading of robot */
   while i<m do {
     PSG := $Q_i$;     /* previous subgoal */
     NSG := $Q_{i+1}$;  /* next subgoal */
     i := i+1;
     /* calculate direction of current path segment
        and the finish_line at NSG */
     precalculations;
     while (finish_line was not crossed) do {
       S := read_sensor_data;
       /* calculate new position and heading by
          evaluating incremental sensor differences */
       HDG := calc_new_hdg(POS,S);
       POS := calc_new_pos(POS,S);
       /* angular divergence between robot
          and path segment heading */
       a := calc_angle(HDG, subseg_hdg);
       /* shortest distance */
       d := calc_distance(POS, PSG, subseg_hdg);
       /* evaluate rule bases */
       eval_rule_base_SE(S,K,REPLAN);
       eval_rule_base_SA(a,d,STEER$_{SA}$,SPEED$_{SA}$);
       eval_rule_base_LCA(S,STEER$_{LCA}$,SPEED$_{LCA}$);
       /* weighting */
       STEER = K * STEER$_{LCA}$ + (1-K) * STEER$_{SA}$;
       SPEED = K * SPEED$_{LCA}$ + (1-K) * SPEED$_{SA}$;
       if REPLAN then {
          replan_new_subgoal_list(POS,SGL);
          call Motion_Control(SGL, HDG);
       }
       compute_motor_velocities(STEER,SPEED);
       output_to_actuators;
     }
   }
}
```

Experiments have demonstrated the nice modular features of this concept, Fig. 4. 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. In a partially-known environment, *SA* and *LCA* are coordinated by the rule base *situation evaluation*, realise the global subgoal-guided collision-free motion.

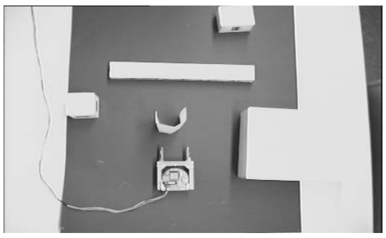

Figure 4: A test environment

5 Conclusions

A fuzzy controller is used for executing subgoal-guided motions. Fuzzy rule bases, like local collision-avoidance, can work together with the rule base for passing through subgoals, each of which with only limited amount of control rules. 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.

The modular design features enable a significant reduction of developing time, which is achieved by simple design of a single rule base, fast 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 heuristics or training. Due to the parallel characteristics of fuzzy control, these rules can be processed in real time during each control cycle. In our opinion, fuzzy control is a promising approach for realising efficient and robust robot motions.

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] Y. K. Huang and N. Ahuja. Gross motion planning - a survey. *ACM Computer Surveys*, 24(3):219–291, September 1992.

[3] S. Ishikawa. A method of autonomous mobile robot navigation by using fuzzy control. *Advanced Robotics*, 9(1):29–52, 1995.

[4] 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.

[5] F. G. Pin and Y. Watanabe. Driving a car using reflexive fuzzy behavior. *IEEE International Conference on Fuzzy Systems*, pages 1425–1430, 1993.

[6] E. H. Ruspini. Fuzzy logic-based planning and reactive control of autonomous mobile robots. In *IEEE International Conference on Fuzzy Systems*, pages 1071–1076, 1995.

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

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.*

Rules for the output variable *Steer*								
					a			
		nb	nm	n	z	p	pm	pb
	nb	pb	pb	pm	pm	p	z	n
	nm	pb	pb	pm	pm	p	z	n
	n	pb	pb	pm	p	z	n	nb
d	z	pb	pm	p	z	n	nm	nb
	p	pb	p	z	n	nm	nb	nb
	pm	p	z	n	nm	nm	nb	nb
	pb	p	z	n	nm	nm	nb	nb

Rules for the output variable *Speed*								
					a			
		nb	nm	n	z	p	pm	pb
	nb	low	low	low	high	high	vh	high
	nm	low	low	low	high	high	high	high
	n	low	low	high	high	vh	high	low
d	z	low	low	high	vh	high	low	low
	p	low	high	vh	high	high	low	low
	pm	high	high	high	high	low	low	low
	pb	high	vh	high	high	low	low	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 replanning.								
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