in *Proceedings of the Seventh International Conference on Advanced Robotics*, Sant Feliu de Guixols, Spain, 1995, pp. 63-70.

Hybrid Motion Control by Integrating Deliberative and Reactive Strategies

Jianwei Zhang, Alois Knoll
Faculty of Technology, University of Bielefeld,
33501 Bielefeld, Germany
phone: ++49-521-106-5325; fax: ++49-521-106-2962;
e-mail: zhang(knoll)@techfak.uni-bielefeld.de

Abstract— *This paper first discusses two strategies for controlling robot motions: planning under pre-described models, and reactive local control by evaluating sensor information. A hybrid control concept is then introduced to integrate both the deliberative and the reactive strategies so that a priori knowledge can be efficiently utilised and on-line sensor data integrated. Based on subgoals for guiding global motion directions, this concept can be decomposed into three components: subgoal planning, subgoal interpolation and subgoal-guided plan execution. Means of realising these components are briefly described. Simulation with examples for both manipulators and mobile robots demonstrates the feasibility of this concept.*

I. Introduction

This work aims at combining advantages of both deliberative and reactive strategies to realise the task-level programming of robot motions. Since the end of the 1970's, computer scientists, control engineers and mathematicians have begun to investigate the problem of autonomous robot motions. This whole theme, including off-line planning of geometric paths for an environment model as well as on-line generation of continuous motion parameters for each individual actuator, can be generally summarised as the motion control problem. An important part, path planning in completely known environments, has been thoroughly discussed and a series of approaches have been proposed. Overviews of this research area are given in [12], [7], and [4]. In [12] path planning is defined as the "mover's problem" or "FINDPATH problem" and further discussed from the viewpoint of computer graphics. Latombe's book [7] gives an introduction to this research area and the detailed mathematical formulation of diverse topics and different solutions. Huang and Ahuja [4] summarises work in path planning for mobile robots, manipulators and multiple robots. Nevertheless, several problems related to real robot applications, such as sensor-based solutions and on-line dynamic generation of trajectories, are not fully covered.

Pure geometric path planning in known environments is a deliberative strategy. This theme area relates closely with problems of space division and representation, search strategies and complexity analysis. Path planning in high-dimensional spaces needs relatively intensive computations based on topological and/or geometrical representations. As a basic concept, configuration space (c-space) [8] is introduced to describe the product space of the subspaces constructed by each degree of freedom of a robot. Generally speaking, the advantages of the deliberative strategy can be summarised as follows:

- A connectivity network serves as a map for robot motion so that the geometric optimal path can be planned based on the overview of the totally reachable space.
- Based on a general-purpose map, collision-free paths can be found between any pair of given points.

However, this strategy possesses the following unavoidable disadvantages:

- A complete environment model must be available - a demand which cannot be fulfilled in most real applications.
- The fine division of the c-space causes an exponential growth of collision-free cells, while the inflexibility of the possible path modification strategy also increases.
- External sensors cannot be integrated during the execution of a path plan.

In contrast to the deliberative strategy, the reactive strategy generally regards path planning as a local feedback control problem, for which a real-time solution should be found. A local environment model and up-to-date sensor data are utilised as input information. Aspects of the sensory and control systems in the real world, like model incompleteness, data unexectness and execution failure, must be taken into account. The task of local motion control is to determine the motion parameters for driving all the actuators by evaluating the up-to-date local information as well as a pre-described path. Two important methods falling under the reactive strategy are potential field and fuzzy control. Based on the concept of artificial potential field, algorithms for collision-avoidance for both manipulators and mobile robots have been developed, e.g. [1] and [5]. Fuzzy control, which is based on theories of fuzzy sets, linguistic variables and generalised modus ponens, shows itself increasingly as a promising tool for local motion control of robots, [3], [13] and [9]. The main advantages of the reactive strategy are:

- Dynamic aspects of the environment can be considered since a structural modelling of obstacles becomes unnecessary and sensor data can be directly applied to determine the robot path.
- Approaches with this control strategy are relatively fast, since in most cases only limited local information is processed.

Its main disadvantages are:

- The so-called "dead-lock" problem can occur since the robot doesn't possess map-reading ability and moves rather "short-sightedly".
- The optimal solution of the robot motion can be achieved only with difficulty since the lack of global information can easily lead to inefficient motions.

Practical application of a robot system in the real world demands not only efficient usage of prior information about the environment but also an on-line suboptimal solution with the ability to modify trajectory and integrate sensors. Brooks has proposed the "subsumption" control architecture consisting of parallel distributed components and applied it to the mobile robot control, [2]. However, task-level programming cannot be fully realised with such an architecture. Quinlan and Khatib [10] proposed the *Elastic Bands* approach for integrating path planning and control. An elastic band is generated by a planner and can be deformed in real time to avoid moving obstacles. This work describes only the implementation for a mobile robot.

In the next section, a hybrid concept for motion control is introduced and the basic ideas of this hybrid control strategy are described. Sections III, IV and V explains procedures for realising three components of this concept, respectively, subgoal planning, subgoal interpolation and subgoal-guided plan execution. Section VI summarises the work and discusses the future work.

II. A Concept for Integrating Path Planning and Trajectory Generation

A. Subgoal-Based Motion Control Scheme

A hybrid architecture describing our integrated concept for robot motion control is shown in Fig. 1. An elementary motion command like "*move from position* **s** *to position* **g**" is input. The task of the off-line module *subgoal planning* is to generate a sequence of critical points as subgoals among the static obstacles. Some properties of subgoals are explained in II.B, and the generation of subgoals for a mobile robot described in III. To realise robot motion along the subgoals, the motion environment can be further classified into two types: completely- or incompletely-modelled environment according to whether unmodelled objects appear during *en route* motion.

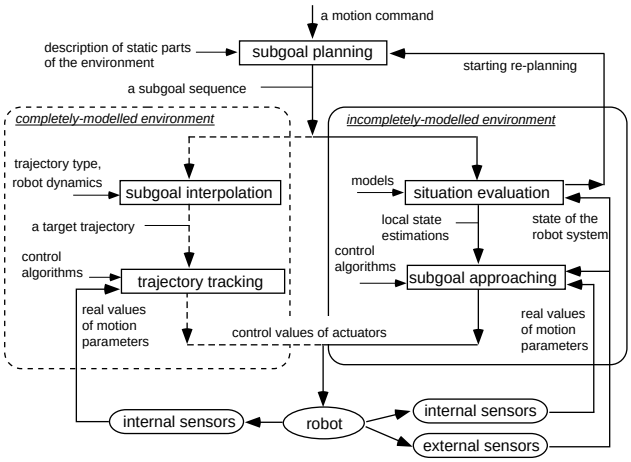

Fig. 1. Integrated consideration of motion control problem

In a completely-modelled environment, a trajectory specified with motion parameters like velocity, acceleration, etc. can be generated with high quality since it can be directly applied to drive actuators. Given a trajectory type and robot dynamics, the task of *subgoal interpolation* is to compute trajectories fulfilling certain criteria, such as smooth and time-optimal. The trajectory generated is regarded as a target trajectory which is then tracked by comparing with the real-values of the motion parameters.

During on-line control, not only internal but also external sensors are evaluated. *Situation evaluation* compares the model data and the on-line dynamic sensor data, and provides information on whether a subgoal is realisable or new subgoals should be replanned. To approach realisable subgoals, the local neighbouring subgoal data and several system variables for general system estimations are input into *subgoal approaching*. Based on a real-time control algorithm, motion parameters are generated for driving actuators.

B. Introduction of Subgoals

For applications in a dynamic environment, the motion executor does not need exact geometric paths provided by the planner since some of the path positions may have to be modified due to dynamic obstacles and the imprecise modelling of some static objects. What is most useful for the on-line motion execution is not a detailed geometric path but a set of critical points, e.g. where a robot has to change its direction relatively sharply in order to arrive at the next subgoal position. These critical points are called subgoals. The idea of generating subgoals is to use them for globally guiding the robot motion and still leaving some freedom for the plan executor to react on uncertainties. Subgoals should fulfill the following two conditions:

- Subgoals are collision-free positions;
- A straight line connecting a pair of adjacent subgoals should have no intersections with the stationary obstacles.

The execution of an exact motion plan often cannot be adhered to. In fact, due to the uncertain properties of the environment and robot execution, it does not make much sense to force the robot to move to each planned position exactly. Therefore, the degree of approach of the robot to a subgoal $\mathbf{q}$ can be defined by a fuzzy measure instead of an exact numerical measure. See Fig. 2 with the following linguistic terms: *AF: approaching and far, AN: approaching and near, Z: subgoal reached, L: leaving.*

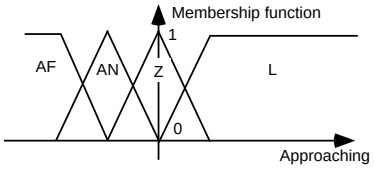

Fig. 2. Description of variable *Approaching* with fuzzy sets

The main differences between a subgoal and a final goal of robot motions are the following:

- A subgoal should be much easier to reach than a final goal;
- The robot usually moves continuously through a subgoal point while it stops at a final point;
- A subgoal can be flexibly generated and must not be traversed exactly, while a final goal is assumed to be fixed and should be exactly reached;
- A subgoal can be abandoned.

The number of subgoals is determined by how many subgoals should at least be generated to avoid collisions. If the free-space is large and has a simple structure (meaning there are only a few obstacles restricting the robot motion), then only a small number of subgoals will be generated. Otherwise, if the robot has to avoid many obstacles or move within very narrow spaces, a large number of subgoals will be created.

III. Planning Subgoals for the Existing Models

Given the *a priori* obstacle and robot model, a global map for subgoal planning can be built up. In this map, a search method can be applied to find a route for guiding the global motion direction. Initial subgoals are generated along this route, leading to a subgoal sequence for further smoothing and adaptation in local areas, see Fig. 3. Such a planning process has been used for both mobile robots and articular robot arms, briefly described in the following two subsections.

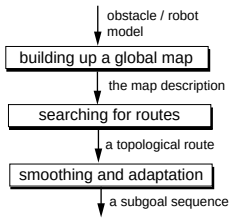

Fig. 3. Procedures for planning subgoals

A. Planning Based on a Tangent-Graph

The subgoal planning problem for mobile robots is simplified to a 2-D case by representing the robot as a disc with radius r. Although 3-D subgoal planning is theoretically solvable, it is computationally expensive and thus unsuitable for on-line replanning. The dynamic characteristics of the environment also make the exact computation of subgoals unnecessary. Obstacles are assumed to be described as polygons. They are enlarged by a constant distance r. The robot is then reduced 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 the enlarged obstacles. This problem can be best solved by searching a Tangent-graph (T-graph), a simplified V-graph, see [15]. 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. 4. The dotted lines are the arcs of the T-graph.

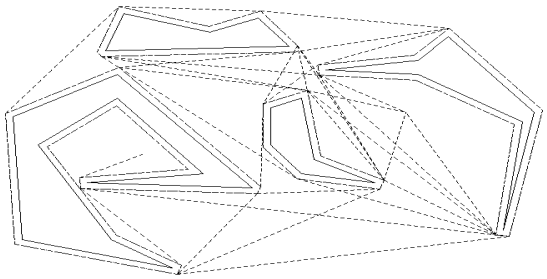

Fig. 4. A T-Graph of enlarged obstacles

The A*-algorithm is used to search for a global route in the T-graph since it can find the shortest path if a path exists. The nodes of the found shortest path from a start position $\mathbf{s}$ to a goal position $\mathbf{g}$ 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:

$$< \mathbf{q}_0 = \mathbf{s}, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m = \mathbf{g} > .$$

B. Planning Based on C-Nets for Robot Arms

To build up a global map for a robot arm, the free-space of the c-space is divided according to its topological structure. Such a topological division for a robot with multiple rotational joints is complex due to the transformation of obstacles from Cartesian space to c-space. An algorithm for this has been developed in [16], with which the free-space can be divided into finite *empty-blocks*. If each *empty-block* is regarded as a node, the neighbourhood of two nodes can be checked by whether they possess a common boundary. A *Characteristic Net* (C-net) can be constructed by linking all the neighbouring nodes.

A simple example with a two-dimensional robot arm illustrates the principle of the algorithm. A rod-chain consists of rod 1 and rod 2 which can rotate on a plane. Their lengths r_1, r_2 are assumed equal. Their rotation ranges are $\Theta_1 = \Theta_2 = [0, 2\pi)$. In Fig. 5, this robot model and two disc-obstacles are depicted. The left figure of Fig. 6 shows the division of the c-space $\Theta_1 \times \Theta_2$. The original obstacles are transformed into c-space obstacles (the black regions). The vertical segments represent the division lines, B_{nm} the *empty-blocks* after the division. The C-net at the right of Fig. 6 explains clearly the topological structure of the free-space.

Fig. 5. A rod-chain and obstacles in the work space

Given a start and a goal position, a sequence of *empty-blocks* is searched in the C-net and is called a route. To determine geometrical subgoals along a route, it is possible that certain new subgoals should be generated. An approach is proposed which works like pulling a rubber-band, Fig. 7.

The subgoal generation process in another example with the same robot model but a more complex environment is shown in Fig. 8. This algorithm is also implemented for the three-dimensional c-space (for the gross motion) of the PUMA-type robot. See Fig. 9 for an example.

Fig. 6. Division of the c-space and the C-net

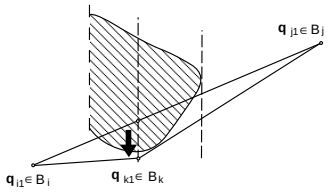

Fig. 7. Generation of new subgoals in local area

C. Subgoal Adaptation

Subgoals generated by the above processes are regarded as initial subgoal points. They can be further adjusted to adapt in the local areas as shown in Fig. 10. There are many factors which influence the determination of the subgoal locations. Such a decision-making process is non-linear. Fortunately, many of these factors can only be described linguistically in heuristics. Several examples are discussed in the following:

Fig. 8. Subgoal generation process in a 2D c-space

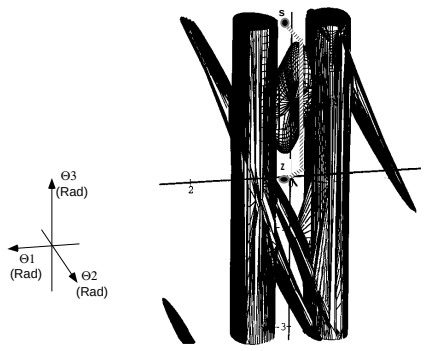

Fig. 9. A 3D example of planned subgoals in the c-space of a PUMA-type manipulator in an environment with three convex obstacles

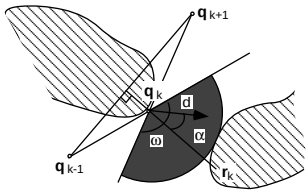

Fig. 10. Fan-shaped area for adaptation of subgoals

- Uncertainty of obstacles. One heuristic could be: "if uncertainty concerning the obstacle is big, then push the subgoal away from it by a large distance."
- Local geometry. One heuristic could be: "in a corner with a high curvature, the subgoal should be a fair distance away from the corner of the obstacle."
- Free space. The heuristic could be: "How far a subgoal can be pushed away is decided by the available free space around it."
- Local path segment. One rule would be: "if the path goes from a short segment to a long one, the subgoal should be pushed closer to the long segment side."

An example of the adapted subgoals is shown in Fig. 11.

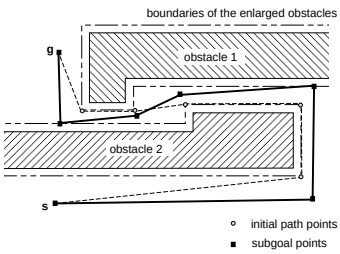

Fig. 11. The generated subgoals after the local adjustment

IV. Trajectory Generation in a Completely-Modelled Environment

Since distances between subgoals are usually different, *Non-Uniform-B-Splines* (subsequently called B-splines for brevity) are the most suitable model for the interpolation task, [16]. Based on a set of basis functions, control points (or *de Boor* points) are computed from the data points to be interpolated. The basis functions and these control points specify a unique smooth curve.

A. Problem Representation

Assume that m data points $\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_m$ should be interpolated. The following well-ordered parameter values are defined for the basis functions of the B-Splines of order k:

$$t_0 < t_1 < \ldots < t_m < \ldots < t_{m+k}$$

These parameter values are called knots. The normalised basis functions are denoted as $N_{j,k}$ (see [16] for their computation). A B-spline curve of order k can be constructed by blending a set of control points with the basis functions:

$$\mathbf{q}(t) = \sum_{j=0}^{m} \mathbf{v}_j N_{j,k}(t), \quad t \in [t_{k-1}, t_{m+1}] \tag{1}$$

(1) is a piecewise defined polynomial of degree $k-1$. It describes the curve course in each coordinate of the trajectory in the n-dimensional c-space over parameter t.

The control points $\mathbf{v}_j$ are normally not identical with the data points for the interpolation, but can be determined by solving an equation system. For details see [16].

B. Dynamic Constraints

The boundary of the minimal motion time of a sub-trajectory $\mathbf{q}_j^i(t)$ is determined by the robot motion capacity represented by the dynamic parameters of all actuators of each DOF. For DOF i, these constraints can be represented as follows:

$$|\dot{q}_j^i(t)| \leq \dot{q}_{max}^i \tag{2}$$

$$|\ddot{q}_j^i(t)| \leq \ddot{q}_{max}^i \tag{3}$$

$$|u_j^i(t)| \leq u_{max}^i \tag{4}$$

where i ($i = 1, \ldots, n$) is the DOF index, j ($j = 1, \ldots, m$) represents the index of the sub-trajectory, and u^i is the torque of DOF i.

C. Implementation and Examples

To generate a smooth trajectory through a subgoal sequence, the parameters of a spline curve, which can

be interpreted as the travel time between the neighbouring subgoals, are initialised. This trajectory can be further optimised by adjusting these parameters according to different criteria like motion time, consumed energy, etc. (Fig. 12). An extra module is employed to detect explicitly the possible local minima and local collisions. The principle for detecting local minima is described in [16]. If local collisions are detected and some points should be adjusted, a module "*subgoal modification*" which works also with the "rubber-band-pulling" principle is called. In this case, the trajectory generation process will be repeated again. The computation cost depends on the precision needed to reach optimality. Therefore, a trade-off between optimality and computation time should be made.

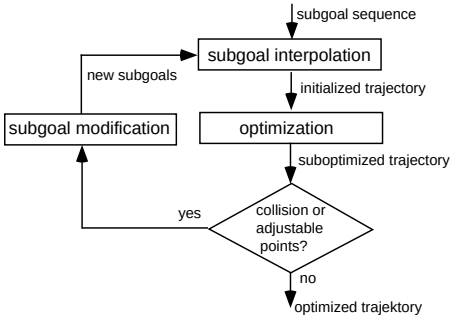

Fig. 12. Steps for trajectory generation and modification

Fig. 13 shows an optimisation example for the problem described in Fig. 5 in the c-space. Fig. 14 depicts the position, velocity and acceleration profiles of both joints.

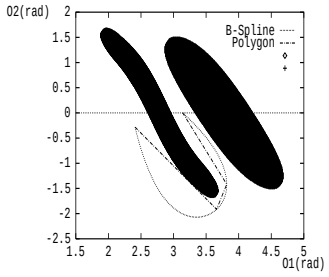

Fig. 13. Optimised trajectory through the subgoals in the c-space

V. Subgoal-Guided Reaction in Incompletely-Modelled Environments

A. A Fuzzy Control Concept

The conventional execution of trajectories is performed by a feedback position controller, which uses the planned trajectory as the desired value and the internal position sensor to feed back the real value. Here, the data from the external sensors for perceiving *en route* information cannot be integrated into

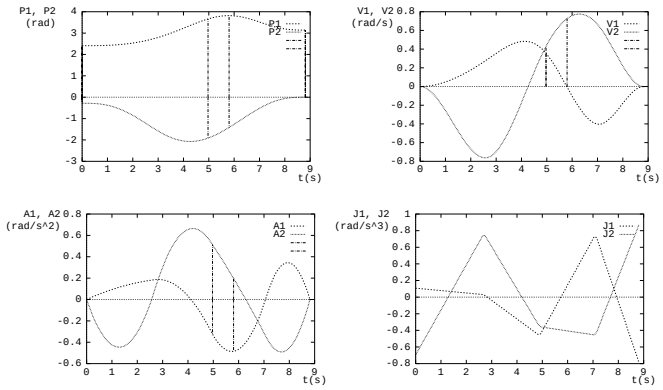

Fig. 14. Trajectory profiles of two joints passing through subgoals

the controller. To solve this problem, we propose the following control structure for the execution of subgoal-guided motions (Fig. 15).

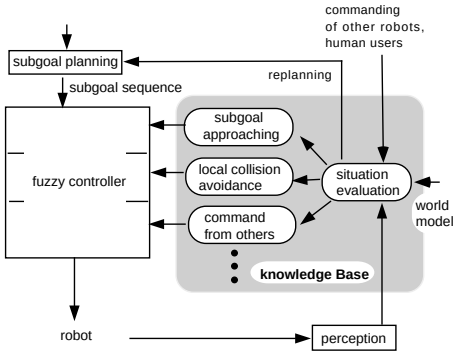

Fig. 15. Strategy for sensor-based subgoal approaching in incompletely-modelled environments

This fuzzy controller uses subgoals from *static planning* and *dynamic planning* and the robot states from the *situation evaluation* as inputs. Based on the pre-described knowledge of the environment, the expected sensor values can be computed on-line. By comparing the expected and measured distance values, the sensor data are evaluated, and several intermediate variables, called *fuzzy states*, are used to describe the results of the *situation evaluation*. Two more rule bases are: *subgoals approaching (SA)* and *local collision avoidance (LCA)*. The rule base SA is responsible for the smooth motion through subgoals. The rule base LCA is employed for avoiding unanticipated local collisions based on the intermediate states from *situation evaluation*. The rule bases "SA" and "LCA" are mainly developed by modelling heuristics. See [14].

B. Subgoals Approaching

Assume that the following sequence of m subgoals is planned:

$SS =< \mathbf{q}_0, \mathbf{q}_1, ...\mathbf{q}_k, \mathbf{q}_{k+1}, ..., \mathbf{q}_m >$

This is a discrete ordered set. Originally, the ex-

ecution of a trajectory was a control problem concerning the arbitrary combination of arbitrarily many subgoals. It would be very difficult to use the subgoal sequence directly to construct the control space due to the variable and possibly very large dimension. Fortunately, such a control problem can be reduced to a treatable dimension through the following model conversion process (Fig. 16).

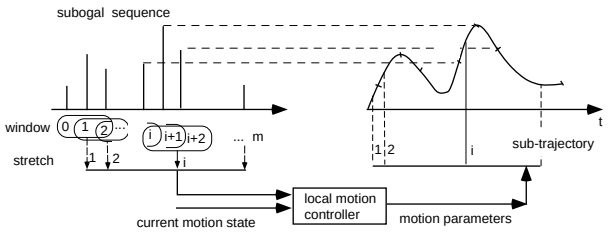

Fig. 16. Principle of dimension reduction for executing a subgoal sequence

Imagine the robot is moving between the $(k-1)$-th and k-th subgoal. The stretch from the $(k-1)$-th subgoal to the current robot position represents the latest motion history, the stretch from the current position to the k-th subgoal represents the next motion, and the stretch from k-th to $(k+1)$-th subgoal predicts the future action of the motion after that. Letting k vary from 1 to m, the whole motion process along the subgoal sequence SS is divided into m consecutive stretches: $1, 2, ..k, ..., m$. There are m overlapping local "windows": $(0, 1, 2)$, $(1, 2, 3)$, ..., $(k-1, k, k+1)$, ..., $(m-2, m-1, m)$, $(m-1, m, m)$. They are constructed to be used as inputs for controlling the motions within the stretches $1, 2, ..., m$, respectively. The total trajectory through all subgoals is formed consecutively by sub-trajectory 1, 2, ..., k, which makes up the time history after integration of the output of the fuzzy controller with inputs of window 1, window 2, ..., window k.

In fact, the idea of this model conversion is inspired by the principle of the B-spline interpolation. Like splines with higher orders, the motion prediction could be extended to more than two subgoals, which would make the control action more accurate, but would increase the dimension of the control space.

C. Implementation Examples

Fig. 17 shows two examples of the on-line collision-avoidance behaviour of a mobile robot. The starting condition of this scenario is that the robot is originally moving along a straight course 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 free of objects again. 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 frontally to the robot or with a deviation of 20, 40, 60, and 80 degrees.

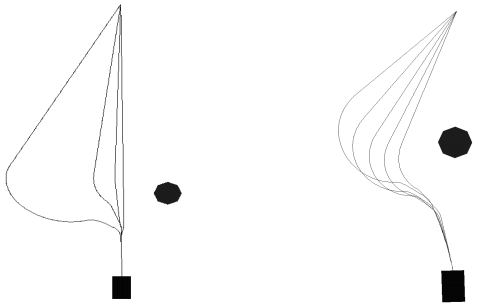

Fig. 17. Avoiding an unanticipated object

Fig. 18 shows the trajectory of the robot in an environment with static and dynamic obstacles.

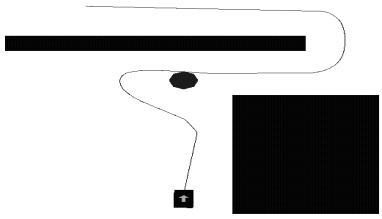

Fig. 18. Motion among static obstacles and an unanticipated object

VI. Conclusions and Discussions

A. Evaluation of This Concept

In this hybrid control concept, the problem of path planning and plan execution is solved hierarchically. In the subgoal planning level, only the connectivity structure of the free-space is interesting. With the help of the T-graph method for mobile robots and C-net method for a robot arm, a global route consisting of subgoals can be efficiently found by neglecting some geometrical details. In the subgoal interpolation process, smooth trajectories can be efficiently generated and modified. They can be further optimised according to different cost functions. With the fuzzy control approach, the off-line modelled environment data and the on-line perceived sensor data can be fully utilised. The local motion controller implemented with fuzzy rules demonstrates a robust solution for the motion execution in partly-known environments. On-line collisions can be avoided by the reactive controller in real-time.

In this way, the efficency of a planner can be enhanced. Through planning gross subgoals, only the

data which are really demanded for motion execution are generated. The planning time can be then reduced. By introducing the fuzzy representation, subgoals can be generated flexibly and adapted to local environments. In the plan execution, the robustness of the motion controller can be enhanced since it possesses a certain freedom to react to the dynamic changes of the environment with help from the sensor information.

Simulation results show that trajectories generated by splines generally possess smoother profiles than the one generated by a fuzzy controller, and the planning results can be directly sent to conventional robot controllers. The reason is that the information for subgoal interpolation with splines is global and complete. Additionally, the interpolation and optimisation algorithms are easily extendable to any high-dimensional c-spaces. However, the ability to integrate sensor data and the real-time property distinguish fuzzy control as an important tool for realising this concept in an incompletely-modelled environment.

B. Future Work

In this work, planning, optimisation and fuzzy control have been integratively considered. It would be interesting to investigate to integrate other intelligent computing methods. Apparently, computation in state space as well as fuzzy control only cover part of the control approaches which emulate the human thinking and behaviour models. Research results in several neighbouring areas, such as parallel computing and neural networks, can also be useful for the robot motion control problem.

In order to accelerate the computation- and memory-intensive processes of the subgoal planning, massive parallel connective computer architecture can be applied. The free-space representation can be divided into finite parts. Similarly the trajectory computation can be performed in each single subspace of the c-space. For subgoal computation, a distributed representation of the c-space and a distributed computation can be designed so that an optimal solution can be realised in real-time.

The cooperation of multiple fuzzy behaviours was shown with the example of "subgoal approaching" and "local collision-avoidance". An attractive extension of this concept would be the investigation of the collision-free and efficient blending of several behaviours, such as subgoal approaching, local collision-avoidance, following another robot, fine manipulation, etc. which can be separately developed and optimised. A general solution could be achieved with help of a non-linear model. Neural networks could be a suitable tool to model, train and optimise such a blending function.

References

[1] Barraquand, J., Lanlois, B. and Latombe, J.-C. "Numerical potential field techniques for robot path planning," *IEEE Transactions on System, Man and Cybernetics*, 22(2):224–241, March/April 1992.

[2] Brooks, R. A. "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, RA-2:14–23, April 1986.

[3] Congdon, C. and Huber, M. et al. "CARMEL Versus FLAKEY: A comparison of two winners," *AI Magazine*, pages 49–56, Spring, 1993.

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

[5] Khatib, O. "Real-time obstacle for manipulators and mobile robots," *The International Journal on Robotics Research*, 5(1):90–98, Spring 1986.

[6] Koren, Y. and Borenstein, J. "Potential field methods and their inherent limitations for mobile robot navigation," *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, pp. 1398-1404, 1991

[7] Latombe, J.-C. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[8] Lozano-Pérez, T. "Spatial planning: A configuration space approach," *IEEE Transaction on Computers*, C-32(2):108–120, February 1983.

[9] Pin, F. G., Watanabe, H., Symon, J., and Pattay, R. S., "Autonomous Navigation of a Mobile Robot Using Custom-Designed Qualitative Reasoning VLSI Chips and Boards," *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, pp. 123-128, 1992.

[10] Quinlan, S. and Khatib, O. "Elastic bands: Connecting path planning and control," In *Proceedings of IEEE Int. Conference on Robotics and Automation,*" pages 802-807, Atlanta, 1993.

[11] Saffiotti, A., Ruspini, E. H. and Konolige, K., "Blending Reactivity and Goal-Directedness in a Fuzzy Controller," *Proceedings of IEEE International Conference on Fuzzy Systems*, pp. 134-139, 1993.

[12] Whitesides, S. H. "Computational geometry and motion planning," *Computational Geometry, edited by G. T. Tourssaint*, pages 377–427, 1985.

[13] Zhang, J. and Bohner, P. "A Fuzzy Control Approach for Executing Subgoal-Guided Motion of a Mobile Robot in a Partly-Known Environment," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 545-550, 1993.

[14] Zhang, J. and Raczkowsky, J. "Robust subgoal planning and motion execution for robots in fuzzy environments," *Proceedings of Internaltional Conferece on Intelligent Robots and Systems (IROS), Munich*, pages 447–453, 1994.

[15] Liu, Y.-H., and Arimoto S. "Proposal of Tangent Graph and Extended Tangent Graph for Path Planning of Mobile Robots," *Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento*, pp. 312-317, 1991.

[16] Zhang, J. *An Integrated Approach for Efficient Planning and Execution of Robot Motions*. Infix, Sankt Augustin, 1994.