ISORA-015.1

ROBOT MOTION PLANNING BASED ON WHOLE TRAJECTORY MODIFICATION

B. Baginski
Munich University of Technology, Germany
e-mail: baginski@informatik.tu-muenchen.de

ABSTRACT

We present a novel approach to motion planning for robot manipulators in *known environments.* The key *concept is to* evaluate *complete* trajectories between start and goal in the workspace and to *reshape* them incrementally. The evaluation is based on *virtual* modifications of the geometry model of the robot. An initially colliding trajectory is bended in space to be improved with respect to the evaluation.

KEYWORDS: robotics, motion planning, computational geometry

INTRODUCTION

A very important *low-level planner* for an autonomous robot system is a motion planner that enables the robot to move between arbitrary positions without colliding with work space obstacles or with itself. We address this problem for manipulators in the case of known obstacles.

A manipulate is an open kinematic chain of *n+1 links* $(0... n)$, connected with *n joints, so n* is the number of *degrees of freedom* (DOF). The first link *(base) is* fixed in the workspace. The last link carries the *tool (e.g.* gripper and load). Industrial manipulators usually have the minimum number of 6 joints. For more complex manipulation tasks, robots with more joints (called *redundant* or *hyperredundant* manipulators) are required.

The position of a joint *i, i=1...n, is* denoted with q_i. The possible values of a joint are limited in the interval $[q_{imin}, q_{imax}]$. The possible values within these intervals are assumed to be steady. The position of a robot is described by the vector $q=(q_i, \dots q_n)^T$. All possible robot positions span an n-dimensional bounded space, the *configuration space (c-space) C.* A point in c-space describes a posture of the robot in its work space. The set of all positions in C that result in a collision of any part of the robot with an obstacle are denoted with C_{coll}, all other positions are collision free and lie in C_{free}. For illustration see Fig. 1.

We only consider the case of a moving robot in a static environment (static for one task at least), thus C_{coll} is constant. Planning means to find a collision free path between a start position q_{start} and a goal position q_{goal}. A valid *solution is* any steady curve in C_{free} connecting q_{start} and q_{goal}. Planning takes place in the search space C, thus it is (at least) an n-dimensional planning problem.

Related Work

For an overview see e.g. Latombe [8] and Hwang and Ahuja [6]. A very rough classification of motion planning approaches is the following:

Explicit *or* approximative modelling of the complete c-space. All attempts to build up complete maps of the c-space fail for more than 4-5 DOF due to exponential complexity (in computational time and memory). An upper bound for these approaches are 5 DOF (see e.g. Ralli [11]). The necessary discretization has a negative effect on the planning, as it implies a lower limit for the resolution of the robot's movements.

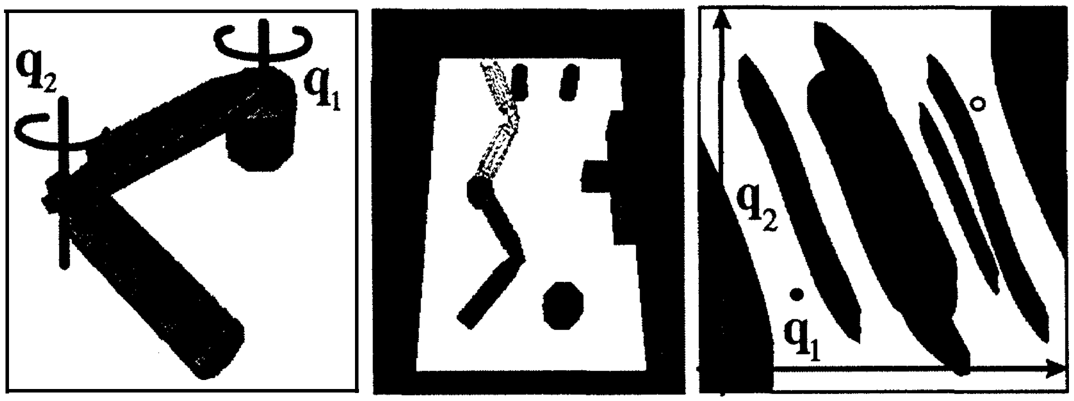

Fig. 1. An example for a two-joint robot [left]. The robot in a demonstration workspace with an example task (solid position: start, wired position: goal) [center). A map of the respective c-space (black: C_{coll}, white C_{free}) with the the same task (solid point: start, empty point: goal) (right].

Random modelling of the c-space. Another approach is to build up a randomized graph within C_{free}, see e.g. Kavraki and Latombe [7]. Subgoals are placed randomly and connected by simple local search strategies. Thus the topology of C_{free} can possibly be covered without exponential complexity. The required preprocessing may take very long for complex environments. The preprocessed c-space map gets invalid if the environment changes, and every object that is moved yields a modification.

Local Planning for positions in work space. The idea is to move from the start position towards the goal, and to pass obstacles along the way, see e.g. the potential field principle in Latombe [8]. The major drawback of these methods is to get stuck in local minima, thus requiring random escape or random exploration techniques to avoid the complexity of global search (e.g. Barraquand and Latombe [3] or Baginski [1]). Local planning for positions is powerful for simple environments. It gets expensive in complex environments, where local minima has to be escaped.

Local Planning for whole trajectories in the workspace. The local approach described here is to take a complete initial path (with collisions), and to modify it in a way to get it collision free. Therefore, a measure of *collision depth* or *badness* for a whole trajectory must be developed. Buckley [4] and Ong/Gilbert [9] use *minimal directed distances* and the *penetration growth distance* respectively, both rather expensive to calculate, and were not able to apply their evaluation to high DOF robot planning in realistic environments. Improvements of these calculations (reported by Cameron [5]) were not yet applied to path planning. Quinlan [10] needs a collision free trajectory to be reshaped under artificial forces, thus the path planning problem needs to be solved *beforehand. The* approach described in the following is a continuation of [2].

LOCAL PLANNING FOR WHOLE TRAJECTORIES

Local planning for whole trajectories can be seen from two points of view. One possibility is to look at a curve in c-space, the other possibility is to consider the volume taken by the robot's motion in the workspace.

Planning principle in the c-space. All possible trajectories (ignoring the obstacles) between q_{start} and q_{goal} can be considred as a bundle of curves, occupying the complete c-space. Some of these trajectories pass through obstacles, some are collision free. As we cannot explicitly construct a representation of the complete c-space, we cannot construct *all* trajectories as well. But we can select any one trajectory and examine it, this is a local operation, as we just examine a one-dimensional subspace of the c-space. We now modify the trajectory - again a local operation - to a neighboring trajectory in the bundle of all possible trajectories, in away that the new trajectory is *better* than the old one. This requires a *measure of badness* to evaluate the trajectory.

The evaluation function has to fulfill several conditions. It has to be unique and it should be strictly monotone (at least locally) to make any planning possible. We do some trial modifications of the trajectory and take a better one if it is found. If there is no local minimum in the evaluation function, the trajectory is *pushed* out of the c-space-obstacles into the free c-space. We start with an initial trajectory, and within the whole planning process, start and goal are actually connected. We just remove the ***badness*** by gradually reshaping the trajectory, until the whole trajectory is *good*, i.e. free of collisions. The success depends on the evaluation function. But looking at the c-space with its counterintuitively shaped obstacles - even in the two dimensional case - no useful evaluation function comes to mind. This changes, if we look at the . . .

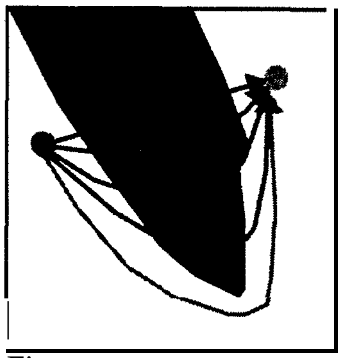
Fig. 2. Planning for a whole trajectory seen in the c-space. The trajectory is modified until it is free of collision.

Planning principle in the work space. q_{start} *and* q_{goal} *describe two* postures of the robot in work space. Any trajectory between these postures sweeps out a certain volume. If the trajectory is colliding, a part of the volume is occupied by the obstacles, Modifying the trajectory to make it *better* now gets a more obvious meaning. If the trajectory can be reshaped in a way that it intersects less with the workspace obstacles, it is most certainly closer to a feasible trajectory than before. So the planning process in the workspace is to *push the* swept volume of the whole trajectory *out* of the obstacles into the free space, see Fig. 3. The evaluation function thus needs to measure the *degree of intersection,*

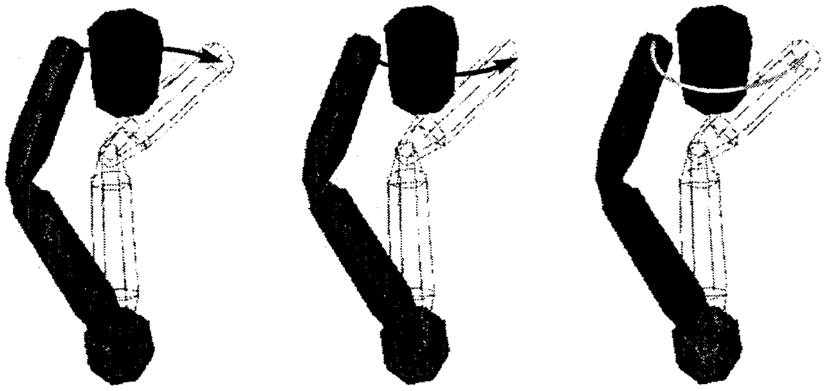
Fig. 3. Modification of the trajectory as it appears in the workspace (same example as in Fig. 2).

IMPLEMENTATION

Based on the ideas described above, we implemented a prototype planner that is able to do local planning for whole trajectories.

Shrink Measure

Evacuating positions. A manipulator is modeled as a chain of bodies (the links), connected with joints. To assign a unique shrink measure to a position, we check the bodies of the chain individually for collision with the environment, beginning at the base of the robot. If all links are free of collision, the shrink measure of this position is set to 1. If a link *i* collides (and the algorithm finds the *first* colliding link), its *size is reduced* with respect to the origin of its local coordinate system, located on the joint axis of the link and on the surface of the previous link. A precondition for this shrinking process to yield a unique result is that the link is convex (or at least *star-shaped* with respect to its local origin, the shrink center).

The shrink factor s_i of link ***i*** is defined as the largest value between O and *1* that allows the link to remain free of collisions with the environment. It is calculated approximately with a depth-limited bisection, checking for collision with $s_i=0.5$ at the beginning (we know that the link collides for $s_i=1.0$ and that it does not collide for $s_i=0.0$, as the previous link is free of collision). If it collides, we check again for $s_i=0.25$, else we check $s_i=0.75$ and so on. We limit the number of tests to a user specified precision.

To calculate the shrink measure $s(q)$ for the whole robot model in a certain posture q, each link of the robot is assigned a partial interval between 0 and 1, these parts are ordered in the order of the kinematic chain and together they cover the interval between O and 1. Now the local shrink factor s_i of the shrunk link is projected in its partial interval. Only the first colliding link of a robot is taken into account for the shrink measure calculation. All outer links are ignored, they can be interpreted as shrunk to zero size. The resulting measure is unique and in the interval [0,11. The calculation of the shrink measure is illustrated in Fig, 4.

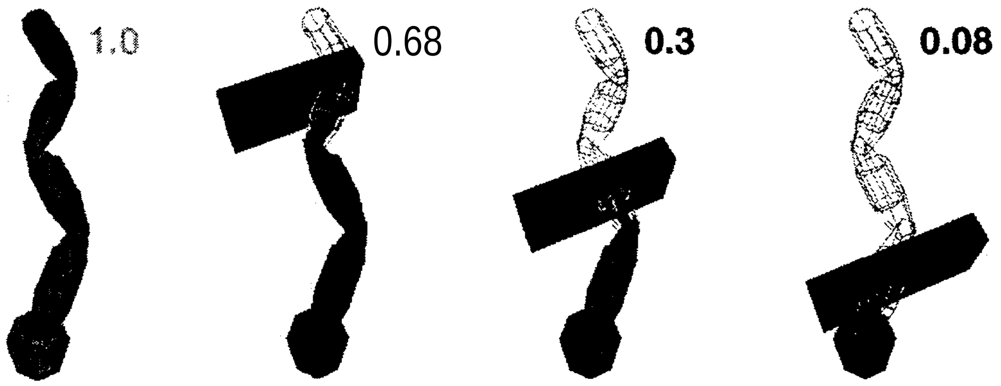

Fig. 4. Illustration of the shrink measure calculation. The size of the first colliding body in the kinematic chain is reduced until it does not collide with the obstacle. The overall shrink measure is *calculated by* mapping the link's shrink factor into the interval $[0,1]$ for the complete manipulator.

Evaluating trajectories. We define the minimum shrink measure along a path as the shrink measure of the whole path. For the calculation, the path is discretized to a sequence of single positions. This allows a very cheap computation of the shrink measure, as the collision detection and the bisection can be executed for all discretization points simultaneously. If a link collides for several of the discretization points, all of the collision free discretization points need no further consideration, as we seek just the smallest value. This exclusion continues within the bisection, thus reducing the number of required collision detections drastically.

Planning Algorithm

Trajectory modelling. To simplify the handling of a trajectory, we model it as a sequence of connected linear segments. The supporting points are the connection points of the linear segments.

Initialization. Given a start and a goal position, the linear connection is discretized and examined by calculating the shrink value for each discretization point. This is just done once, resulting in a 'shrink profile' of the linear connection between q_{start} and q_{goal}. If no collision occurs, a solution is found. If there are collisions, initial supporting points are placed at the positions of minimal shrink measure within each collision interval, and in the center of collision free intervals, if there was more than **one** collision interval detected. All resulting path segments are attributed with their shrink measure, i.e. the minimum value of all their discretization points.

Iterative modification. The main loop of the algorithm tries to move the supporting points adjacent to the segment(s) with the minimal shrink value. To move the supporting point *away* from the trajectory, we calculate a hyperplane in the c-space, orthogonal to the linear connection between the two adjacent nodes. The hyperplane is build in the i lower dimensions of the c-space only, as a movement of the upper joints will not improve the placement of the colliding link.

The restriction to orthogonal directions in an *i*-dimensional subspace is a limitation for the spatial adaption of the trajectory, but it still leaves *i-1* base vectors and their respective reverse vectors as test directions. The step length is calculated as a quarter of the linear distance between the two adjacent nodes, a heuristic that creates steps in relation to the 'granularity' of the trajectory that is modeled through the supporting points.

The test positions are evaluated by recalculating the shrink measure for the two adjacent linear segments, and the best possible position is taken as the new position for the supporting point. We demand that the shrink measure of at least one of the adjacent line segments is increased. If no point can be found, the step size is cut by half, this may even be repeated. The maximum number of tests executed to improve one supporting point is *6(i-1)*.

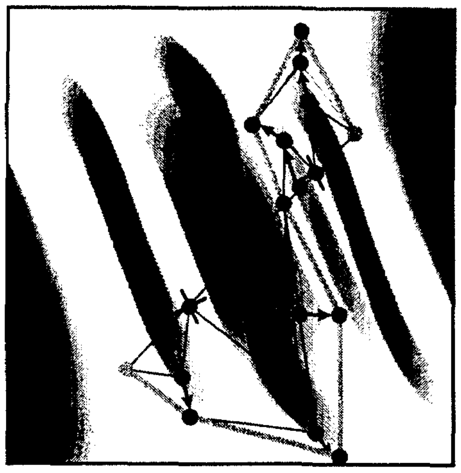
Fig. 5. **Visualization** of planning for the task from Fig. 1. The brightness of the c-space obstacles reflect the local shrink measure. As darker a region is, as smaller the robot has to be to not collide. The algorithm moves the supporting points out of the obstacles, inserts new ones if necessary and removes nodes if possible.

Insertion of additional supporting points. If, within one iteration, the linear segment with the minimum shrink value cannot be improved (neither of its adjacent supporting points were moved), a new supporting point is inserted in the center to 'allow a better adaption to the topology of the obstacles. If the distance between two supporting points falls below the discretization distance, the planner terminates with failure.

On-line optimization. To achieve trajectories with few bends, we included art on-line optimization. At the end of each iteration the direct connection between the two adjacent supporting points is evaluated for all unmoved supporting points. If the shrink measure is above the shrink measure of the two segments in between, the node is removed.

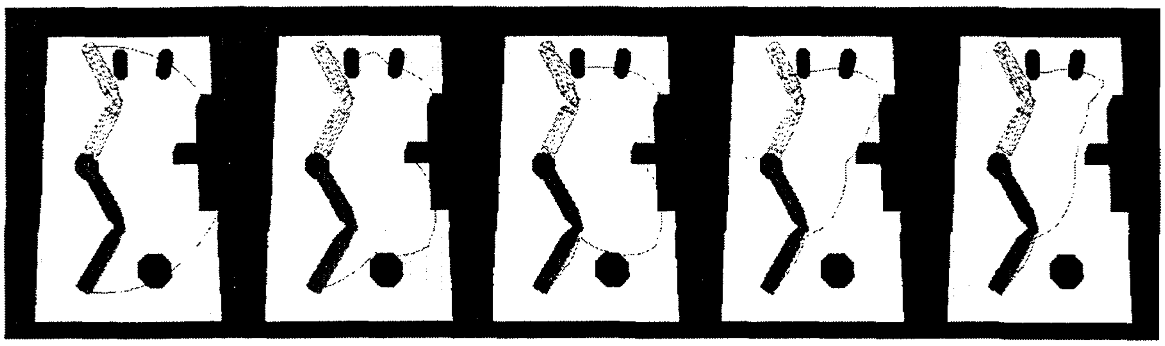
Fig. 6. The stages of planning shown in Fig. 5 visualized in the workspace.

The main loop **of** modification, insertion and optimization is iteratively repeated. The planner terminates with success, if all linear segments of the trajectory have a shrink measure of 1, else it terminates with failure. The planning process for the sample task is visualized in Fig. 5 and Fig. 6.

Some **Experimental Results**

We have used the planner in scenarios from 2 to *16 DOF*, and in general it is very successful for all kinds of different tasks. The planning time is mostly dependent on the complexity of the geometry models of the robot and the environment. All randomly created tasks in the two dimensional example of the previous chapters are solved without failure and within .1 to 4 sec. Examples for an 8 DOF and a 16 DOF robot are shown in Fig. 7. All calculation times refer to simulations on a standard HP Unix workstation,

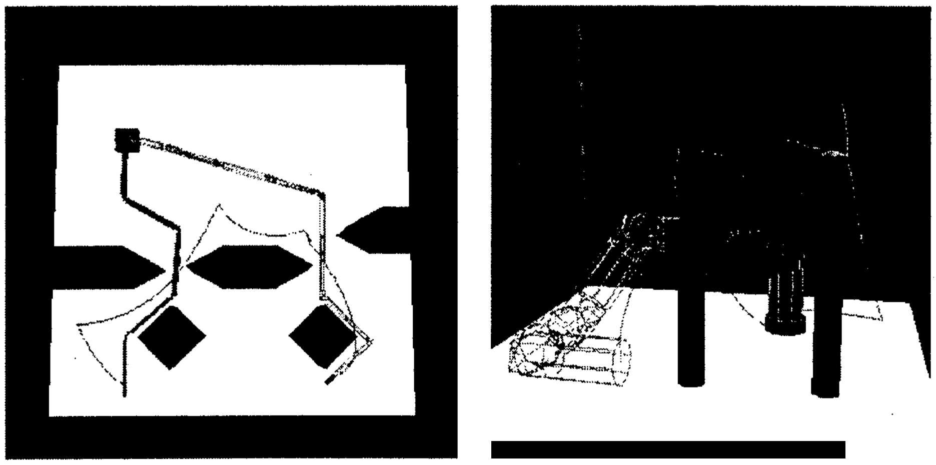

Fig. 7. This solution for an 8 DOF robot (5 rotational and 3 translational joints) is planned in less than 15 seconds. The resulting trajectory consiststs of 5 linear segments [left]. A very complex task for a 16 DOF robot. A solution is found in about 5 seconds, consisting of 7 linear segments [right].

The examples show that the planner successfully fulfills its objectives. The trajectories consist out of very few linear segment (in c-space). Even very small passages are passed with long straight motions, a unique capability for a local planner.

The success rate of our planner is very high, compared to other local planning algorithms. For robots with a higher number of DOF, especially for hyperredundant robots, the situations of failure are rare and the critical obstacle configurations are not obvious. In most cases, the failure appears not to be a consequence of the principle itself, but of the heuristics used for the step size and the step directions.

CONCLUSION

The use of a virtual value, the shrink measure, is the new quality of our concept, because it is cheap to calculate and gives us a matter to manipulate complete trajectories, that are, in the real work space, just colliding. For two fixed, collision free postures of the robot, the planning system incrementally decreases the degree of collision in between. The planning is local and of linear complexity in the number of degrees of freedom. The system draws all its information from the evaluated paths, it never tries to examine the whole high dimensional space.

REFERENCES

1. Baginski, Boris (1996a). The Z^3-Method for Fast Path Planning in Dynamic Environments. In: Proceedings of IASTED Conference on Applications of Control and Robotics. Orlando, January 1996, pp. 47-52.
2. Baginski, Boris (1996b). Local Motion Planning for Manipulators Based on Shrinking and Growing Geometry Models. In: Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, April 1996, pp. 3303-3308.
3. Barraquand, J. and Latombe, Jean-Claude (1990). A Monte-Carlo Algorthm for Path Planning with Many Degrees of Freedom. In: proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, May 1990, pp. 1712-1717.
4. Buckley, C. E. {1989). A Foundation for the „Flexible-Trajectory“ Approach to Numeric Path Planning. In: The International Journal of Robotics Research, Vol. 8, No. 3, June 1989, pp. 44-64.
5. Cameron, Stephen (1997). Enhancing GJK: Computing Minimum and Penetration Distances between Convex Polyhedra. In: Proceedings of the IEEE International Conference on Robotics and Automation, Albuquerque, April 1997, pp. 3112-3117.
6. Hwang, Yong K and Ahuja, Narendra (1992) Gross Motion Planning - A Suryey. In: ACM Computing Surveys, Volume 4, Number 23, September 1992, pp. 219-291.
7. Kavraki, Lydia and Latombe, Jean-Claude (1994). Randomized Preprocessing of Configuration Space for Fast Motion Planning. In: Proceedings of the IEEE International Conference on Robotics and Automation. San Diego, May 1994, pp. 2138-2145.
8. Latombe, Jean-Claude (1991). Robot Motion Planning. Kluver Academic publishers.
9. Ong, Chong Jin and Gilbert, Elmar G. (1994). Robot Path Planning with Penetration Growth Distance. In: Proceedings of the IEEE International Conf. on Robotics and Automation, San Diego, May 1994, pp. 2146-2152.
10. Quinlan, Sean (1994). Real-Time Modification of Collision-Free Paths. Ph.D. Thesis, Stanford University, December 1994.
11. Ralli, E. and Hirzinger, G. (1996). A Global and Resolution Complete Path Planner for up to 6 DOF Robot Manipulators. In: Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, April 1996, pp. 3295-3302.