Robotica (1999) volume 17, pp. 365–371. Printed in the United Kingdom © 1999 Cambridge University Press

Motion planning in crowded planar environments

Yoav Lasovsky and Leo Joskowicz

Institute of Computer Science, The Hebrew University, Jerusalem 91904 (Israel)
E-mail: josko@cs.huji.ac.il www:http://www.cs.huji.ac.il/~josko

(Received in Final Form: February 10, 1999)

SUMMARY
We present a new algorithm for fine motion planning in geometrically complex situations. Geometrically complex situations have complex robot and environment geometry, crowded environments, narrow passages and tight fits. They require complex robot motions with coupled degrees of freedom. The algorithm constructs a path by incrementally building a graph of linearized convex configuration space cells and solving a series of linear optimization problems with varying objective functions. Its advantages are that it better exploits the local geometry of narrow passages in configuration space, and that its complexity does not significantly increase as the clearance of narrow passages decreases. We demonstrate the algorithm on examples which other planners could not solve.

KEYWORDS: Motion planning; Crowded environments; Configuration space; Robot motions.

1. INTRODUCTION

Effectively planning the motion of objects in geometrically complex environments is of great practical importance for many tasks and domains. Geometrically complex environments are charaterized by complex robot and obstacle geometry forming a crowded environment requiring fine, coupled motions to navigate through narrow openings and tight passages (Figure 1). In robotics, this is necessary for maneuvering robots through cluttered environments. In manufacturing, it is necessary to determine manipulation and assembly operations of parts into tight spaces, such as mounting the oil filter in a car engine. In computer graphics, it is necessary to realistically navigate, animate, and manipulate complex scenes, such as a virtual refinery, without having to manually specify many key frames.[1] Many geometrically complex problems in other applications, such as molecular biology[2] and medicine[3,4] also require effective motion planning algorithms.

This paper presents a novel algorithm, for motion planning in geometrically complex planar environments.[5] The algorithm, which is an extension of the EXTRACT algorithm,[3] plans the robot motion by incrementally building a graph of linearized convex configuration space cells encoding the contact constraints between the robot and the fixed obstacles. Path search proceeds in $A*$ fashion by moving to neighboring configuration space cells and solving a series of linear optimization problems with varying objective functions. This method, which better reflects the local geometry of narrow passages, is potentially more effective than randomized search or collision detection. It is also resolution sound and complete. We demonstrate the algorithm on several tight-fit examples which other planners could not solve.

2. PREVIOUS WORK

Global motion planning strategies, which construct a cell partition or a road-map of the configuration space and then search it for a path, are impractical for geometrically complex situations because of the high cost of computing the full configuration space. The complexity cannot be easily reduced by approximating or abstracting the configuration space because of the tight fit and narrow clearances between objects. Techniques such as planning in low-dimensional configuration space projections[6] or exploiting the object's geometric regularities[7] are not applicable. Local motion planning strategies directly search for a path, performing the necessary geometric computations that guarantee non-interference as the search progresses. Such strategies depend on the efficiency of the geometric computations and the effectiveness of the search strategy.

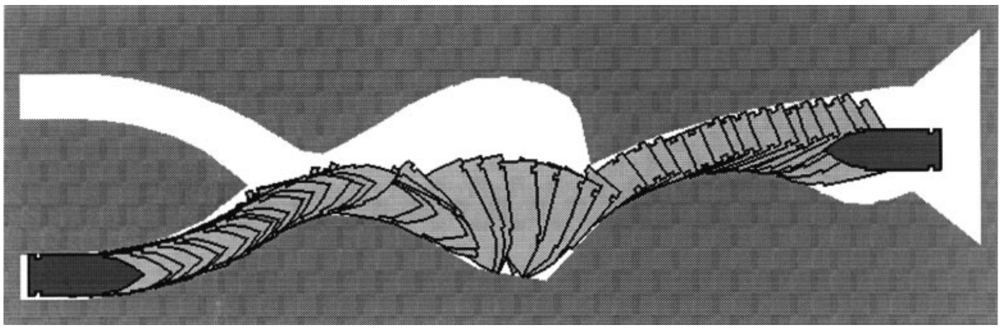
Fig. 1. Tight fit bullet moving on a feeder track.

Existing local search methods emphasize search effectiveness and are only applicable to problems with simple to moderately complex geometry.[8–12] The chance of randomly finding a free configuration is very small, and hierarchical collision detection[13,14] are ineffective in tight fit situations.

Several recent planners explicitly address geometrically complex situations. Benchetrit's algorithm[13] hierarchically searches and decomposes configuration space cells in $A*$ fashion, using a Manhattan distance metric in the evaluation function. To test for free cells and configurations, the planner relies on a collision detection algorithm that uses a hierarchical bounding box representation and local transformations. The algorithm can solving problem with many geometric features, but has difficulties in solving tight fit situations (Section 5.1).

Chang and Li's[15,16] algorithm is based on randomized potential field path planning. It uses adaptive multiresolution search and a fast collision detection algorithm. To escape local minima, it performs a preset number of random walks followed by gradient motion. It relies on user-defined constrained volumes to limit the search space. The algorithm is demonstrated on testing the accessibility of a part in a complex arrangement of pipes, and on inserting a part in a turbine. It took hours to produce results on a two-processor SGI Reality Engine and occasionally failed altogether due to its probabilistic nature.

Hsu et al.[9,10] use randomized networks that limit the space explored for new interference-free configurations to a small area around the current configuration. The space is explored incrementally and more efficiently. The notion of *expensive* configuration space is defined as a space in which it is improbable to randomly find free configurations. In such cases, the algorithm requires manual identification of intermediate collision-free configurations.

Joskowicz and Taylor's EXTRACT algorithm[3] handles spatial geometrically complex, tight fit insertion problems, where fine, complex, coupled six-degree of freedom motions in a preferred direction are necessary. The algorithm avoids collision detection altogether, using instead local, linearized configuration space constraints derived from the object shapes to compute successive interference-free motion steps in a preferred direction. It has been tested on complex examples consisting of objects with up to 10,000 facets and clearances of 2mm toan accuracy of 0.25mm. The algorithm has limited search capabilities, requiring a preferred direction and producing quasi-monotone paths. The algorithm presented in this paper extends the search capabilities of this approach.

Faverjon and Tournassoud[17] use linearized configuration space constraints and quadratic optimization to plan motions of cooperating mnipulators. The main limitation of their algorithm is the search, since the algorithms plans motions locally, without keeping track of explored configuration space regions. This prevents systematic backtracking and detours which place the robot locally further from the target.

3. SOLUTION OVERVIEW

We have developed an algorithm that searches a path by incrementally building a graph of adjacent convex free configuration space cells. Each cell is defined by a set of linearized constraints, constructed according to a robot/obstacles features proximity pairing. The configuration space cells are more complex than the cells used in the 2^n-cube cells, but there are much fewer of them. Each cell is characterized by a canonical, non-redundant set of constraints, which is in most cases of constant size. The planes bounding the cells obtained by linearizing the contact constraints are used to identify adjacent cells. The normals to the cells boundary planes are used as objective functions in a linear optimization problem, whose solution yields the next motion step. The algorithm uses the $A*$ search strategy and can solve search problems requiring significant backtracking.

4. PATH PLANNING WITH LINEARIZED CONFIGURATION SPACE CELLS

This section presents the details of the algorithm. Section 4.1 describes the representation of the robot and the free working space. Section 4.2 introduces the notation and problem formulation. Section 4.3 defines feature proximity and pairing. Section 4.4 describes the construction of a single configuration space cell. Section 4.5 describes constraint redundancy elimination, and section 4.6 describes the incremental cell construction strategy. Sections 4.7 and 4.8 describe the search strategy and cost function.

4.1 *Object and workspace representation*

We represent the robot by a series of sampled points on its boundary, and the free space between obstacles as the union of overlapping trapezoids. Each point in free space belongs to two exactly trapezoids, and no trapezoid is contained in any other. We also require that trapezoids having a common intersection or sharing an edge should be of comparable size. This subdivision greatly simplifies the process of robot vertices migration from one configuration space cell to another. We assume that a good subdivision is given as an input to the algorithm. A simple method for producing a subdivision with overlaps is to create two different subdivisions[18] and then merging them.

4.2 *Problem formulation*

The workspace obstacles are defined with respect to a fixed global coordinate frame. The robot geometry is defined with respect to a local coordinate system. The *configuration* of a robot is defined by three degrees of freedom, two translations, $\bar{p}=(x, y)$ and a rotation θ. The transformation $F(\bar{p}, \theta)$ maps points in robot coordinates to points in global coordinates for robot configuration $(\bar{p}, \theta)$. The set of all configurations forms a configuration space C.

Let $\bar{r}$ be a robot point whose coordinates are with respect to the robot coordinate frame. The position $\bar{v}$ of a robot point $\bar{r}$ in configuration $(\bar{p}, \theta)$ with respect to the global coordinate frame is:

$$\bar{v}=F(\bar{p}, \theta)\bar{r}=Rot(\theta)\bar{r}+\bar{p}$$

where $Rot(\theta)$ is the rotation operator specifying the robot orientation with respect to the global coordinate frame. Let R and S be the sets of points describing the robot and the obstacles. Let $H(x)$ be a function that describes the shape of the obstacles such that $H(x) \leq 0$ for every point x in the Euclidean plane that is not inside any of the obstacles. A robot point $\bar{r}$ in configuration $(\bar{p}, \theta)$ is interference-free iff:

$$H(F(\bar{p}, \theta)\bar{r}) \leq 0$$

This condition, formulated over all robot points and all obstacles, defines the robot's *free configuration constraints* which must hold for the robot to not interfere with any of the obstacles. This set of free robot configurations is called the free configuration space:

$$C_{free} = \{(\bar{p}, \theta) \mid H(F(\bar{p}, \theta) \cdot \bar{r}) \leq 0;\ \forall \bar{r} \in R\}$$

4.3 *Proximity pairing*

The set of constraints defining the configuration space cells are formulated with contact constraints.[19] In our case, the single contact constraint is a moving vertex-fixed edge constraint. Free space is defined by at most $O(nm)$ contact constraints, where n is the number of robot vertices and m is the number of obstacles' edges. Computing the whole set at once is impractical for very complex obstacles. Instead, we iteratively construct configuration space cells whose complexity is at most $O(n)$ based on proximity relations.

Each robot vertex lies in two overlapping trapezoids for a given robot configuration. This defines the proximity relation of each robot vertex. The match is used throughout the algorithm as the identifying "name" for the configuration space cells. For each pair of free workspace region and robot vertex, a set of four constraints is formulated – one for every edge of the quadrilateral. This set of constraints define the robot configurations in which the robot vertex remains inside the quadrilateral. The set of $4n$ constraints defines the configuration space cell such that no robot vertex leaves its enclosing region.

4.4 *Linearized configuration space cells*

Given the list of n pairwise matchings between robot vertices and their containing regions, the $4n$ contact constraints define a three-dimensional configuration space cell. For every robot vertex $\bar{r}$, four constraints are produced, one with every edge h_i of the work space quadrilateral region. Let the equation of an edge h_i be $\bar{a}_i \bar{x} + c_i$.

Substituting $\bar{r}$ in position $\bar{v}$ when the robot is in configuration $(\bar{p}, \theta)$ yields:

$$\bar{a}_i(F(\bar{p}, \theta) \cdot \bar{r}) + c_i \leq 0 \tag{1}$$

This set of constraints is non-empty if there is no interference between the robot and the obstacles, and each robot vertex remains inside its corresponding workspace free region. Using the standard small motion approximation this set becomes a set of linear constraints. Let $\bar{v}$ be the position of vertex $\bar{r}$, when the robot is in configuration $(\bar{p}_0, \theta_0)$. The position $\bar{v}'$ of $\bar{r}$ after a small relative motion $(\bar{\epsilon}, \alpha)$ is given by:

$$\bar{v}' = Rot(\alpha) \cdot \bar{v} + \bar{\epsilon}$$

Since $(\bar{\epsilon}, \alpha)$ is a small motion, it can be approximated as:

$$v'_x = v_x - \alpha v_y + \epsilon_x$$

$$v'_y = \alpha v_x + v_y + \epsilon_y$$

Substituting the approximated new configuration in the edge equation yields:

$$a_1(v_x - \alpha v + \epsilon_x) + a_2(\alpha v_x + v_y + \epsilon_y) + c \leq 0 \tag{2}$$

Rearranging by the small motion parameters yields:

$$a_1 \epsilon_x + a_2 \epsilon_y + \alpha(a_2 v_x - a_1 v_y) + (a_1 v_x + a_2 v_y + c) \leq 0 \tag{3}$$

The result is a set of linear constraints in the new motion parameters $(\bar{\epsilon}, \alpha)$ which are related to the configuration space parameters by the relation:

$$F(\bar{p}, \theta) = F(\bar{\epsilon}, \alpha) F(\bar{p}_0, \theta_0)$$

This set of linear constraints approximates the configuration space cell that describes all proximity preserving robot motions. It is convex and singly connected, which guarantees interference-free motions up to the approximation between any two configurations inside the cell.

4.5 *Canonical representation*

The set of configuration space constraints in Equation 3 is highly redundant because all contacts cannot be realized simultaneously, and most contacts cannot be realized at all. Geometrically, this corresponds to intersecting a large set of half planes. The intersection forms a convex polygon in the plane. The number of segments in the intersection polygon is typically much smaller than the number of half planes in the set. The half planes not appearing in the intersection are redundant. To reduce storage space and running time, we remove redundant constraints using the Convex Hull Method (CHM),[20] which yields a canonical form in which all equations are in solved form and the inequalities represent a full dimensional polyhedral set free of redundancies. The canonical form is unique and allows simple comparison between two constraint sets.

4.6 *Incremental cell construction and motion*

To search for a path, the algorithm incrementally constructs configuration space cells, starting from the cell defined by the initial robot configuration. To move within the cell, a linear optimization problem is formulated with the constraint set and an objective function defined by a given motion direction. The solution yields a small motion vetor that keeps the proximity relations and avoids interferences. The resulting motion vector is applied to the current robot configuration yielding a new one.

The linear programming problem at the k^{th} iteration is formulated using constraints in the form of Equation 3, over all robot vertices and their corresponding edges of the containing trapezoids:

maximize $T_k(\epsilon_x^k, \epsilon_y^k, \alpha^k)$

subject to:

$$a_1 \epsilon_x^k + a_2 \epsilon_y^k + \alpha^k(a_2 v_x - a_1 v_y) + (a_1 v_x + a_2 v_y + c) \leq 0$$

$$|\alpha^k| \leq \alpha_{max}$$

where $T_k(\epsilon_x^k, \epsilon_y^k, \alpha^k)$ is the linear objective function which set by the global search method (see section 4.7). The small motion approximation assumes small rotations in every motion step, therefore the $|\alpha^k| \leq \alpha_{max}$ constraint limiting the size of the rotation in the step is added. It further subdivides the configuration space, limiting each cell to a certain range of angles. For tight fits, this additional subdivision does not increase significantly the number of configuration space cells.

After a small motion step, the new configuration is still within the same configuration space cell. One or more of the robot vertices placed in the new configuration lie on the edges of their containing cells. This is due to the property of linear programming by which if a solution is found, it is always optimal and lies on a vertex of the convex polyhedron defined by the set of constraints. Identifying the edges or vertices on which the robot vertices now lie can be difficult due to numerical errors. It might also require many small steps to cross a cell's boundary, since only a few vertices are migrated in each step. We thus opted for the overlapping free working space subdivision.

Migration of the robot vertices between adjacent overlapping regions is straightforward, since every robot vertex always resides in two workspace regions. After each motion step, each vertex is associated to the other region it resides in. The migrated vertices are situated in the middle of the new containing region, not on its edges, as it would be without using overlapping workspace regions. The algorithm performs membership queries only for the four neighbors of the trapezoidal region containing the vertex. Updating the vertices-regions match list takes linear time in the number of robot vertices. After every re-matching process, the constraints set for the next step is also generated in linear time.

4.7 Search strategy

To determine which cell to construct and explore next, we use a variant of the $A*$ algorithm. Starting from the cell containing the initial configuration, the search strategy incrementally explores the configuration space structure. As the search progresses, the algorithm constructs a directed acyclic graph whose nodes are configuration space cells and whose edges are cells adjacencies.

The algorithm builds and explores cells until the cell containing the goal configuration is reached. When the goal is reached, the algorithm tracks back the full motion path from start to goal configurations by traversing the graph nodes until the start cell is reached. When the goal configuration is not reached and all cell neighbors have been explored, the goal configuration cannot be reached and the algorithm terminates with a failure message.

To advance from one configuration space cell to any of its immediate neighbors, a list of normals to the cell's bounding planes is maintained. Each normal is assigned a value by which the normals list is sorted. We chose to give the normal a value which is its scalar product with the normalized global start-to-goal vector. The best normal in the list (not already used, largest value) is set to be the objective function in the current cell's linear programming problem. Solving the linear programming problem yields a small motion step towards one of the immediate neighbors of the current cell. Exploring all normals in the list as possible directions for motion guarantees the completeness of the algorithm, up to the approximation.

4.8 Heuristic cost function

We use an Euclidean distance-based cost function which is defined on the nodes as the $A*$ algorithm cost-function. The simplest method to estimate distances between nodes is to set a representing configuration for every configuration space cell and measure the distance between them.

Unlike the cells in a 2^n-cubed decomposition method [13], the cells in our algorithm are not uniform in shape. Picking a representing configuration that is close to the cell's boundary can reduce the efficiency of the search algorithm. Nevertheless, there is no simple way to compute a good representing configuration, and so we have chosen to sue the first configuration the robot reaches inside the cell as the representing configuration for computing the cell's value. The standard Euclidean distance is used for the cost function:

$$f^*(n_c,n_s,n_g) = h(n_c,n_s) + g^*(n_c,n_g)$$

where q_c is the current cell's representing configuration, q_s, q_g are the start and goal configurations and h, g^* are Euclidean distance functions. The * stands for approximated distance measure. Note that the path distance so far, $h(q_c,q_s)$ is computed accurately by storing the value and passing it onward to the next node visited. The regular Euclidean distance function is used as the optimistic distance measure from current configuration to the goal.

5. IMPLEMENTATION AND EXPERIMENTS

We have implemented the algorithm and have tested it on several challenging examples (Figures 1–4). All tests were run on an SGI O_2 workstation with a 180Mhz R5000 processor and 192MB RAM. Table I shows the results. The minimal clearance is the relative size of the smallest free space passage relative to the robot's diameter. It includes the number of search graph nodes explored through planning (opened nodes), the number of nodes listed as potential path configurations (closed nodes) and the number of configurations in the final path. CPU running time is measured in seconds.

The results indicate that very few unnecessary cells of the configuration space are constructed, meaning that the search is very effective. The fish example presents a hard search problem, in which an exhaustive search of configurations space was made, exploring many of cells on the way. Our experiments validate our initial intuition, which was to exploit the geometrical structure of the space to reduce the number of cells (or sampling points) constructed during the search. The numbers are much smaller than, for instance, in using the approximated cell decomposition method[13] or randomized planning.[9,10] The computation time of a single motion step is proportional to the number of robot points, but does not depend on the obstacles complexity, and therefore is low comparing to full collision detection test, even with hierarchical bounding box data structures.

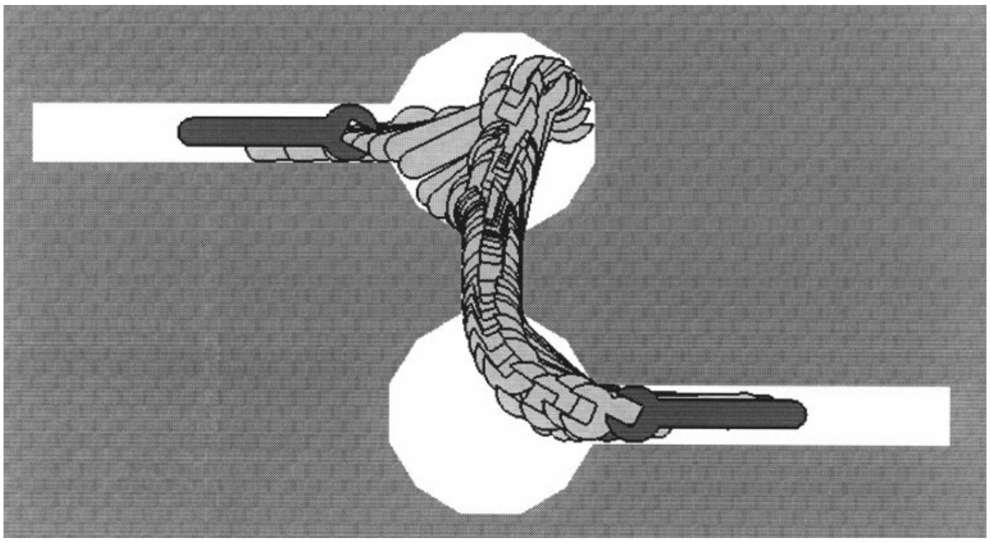

Fig. 2. Wrench in a narrow pipe.

5.1 Comparison with Benchetrit's planner

Very few existing motion planners have specifically emphasized geometrically complex tight fit planning. Comparing performance with existing planners is only of interest if there is a reason to believe that the planner would be able to solve such problems. Existing planners might solve hard many degrees-of-freedom problems, but would have difficulties solving tight fit problems. Most published papers do not provide sufficient details or an interesting planar example to test or program.

In this section, we present a performance comparison between our planner and the planner developed by U. Benchetrit.[13] The algorithm is based on the approximated cell decomposition approach, in which the configuration space is partitioned by a 2^n tree. The approximated cell decomposition approach has the important advantage of

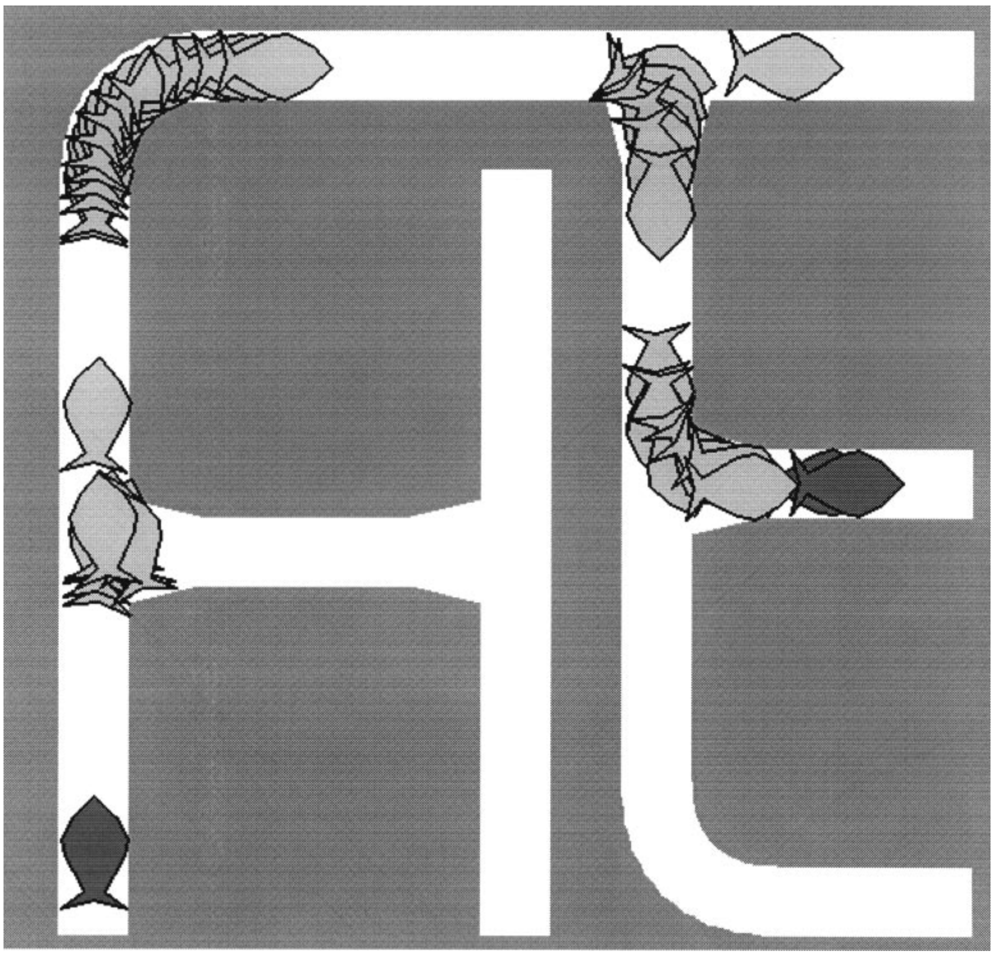

Fig. 3. A hard 'fish' search problem.

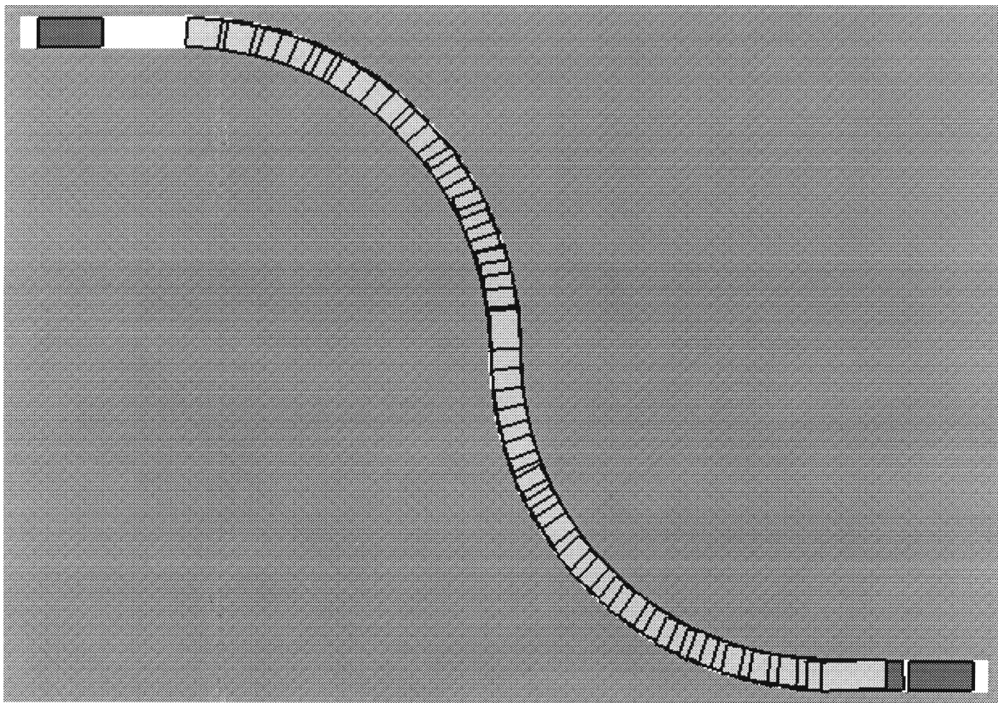

Fig. 4. A very tight fit cart-on-a-rail.

being both simple and general. Benchetrit has tested his algorithm on three dimensional problems containing thousands of geometric features. His algorithm is not designed to efficiently solve tight fit problems, but being general algorithm, is capable of solving many kinds of planning problems.

Figure 4 shoes a tight fit planar cart-on-a-rail example. We ran Benchetrit's algorithm on this example and on the feeder track example shown in Figure 1. The results for the cart-on-a-rail example are shown Table II. The table clearly shows the advantage of our program in tight fit planning. In the larger clearance case, our program opened half the number of cells opened by Benchetrit's planner. In the smaller clearance case, our program's advantage is even greater, opening about eight times less cells. The running times match the number of cells opened by the program. In addition, Benchetrit's planner was tested on the feeder example, but was not able to solve it after more than 20 hours.

Table I. Results for the examples in Figures 1–4.

Example name	wrench	feeder	rail	fish
# of obst. edges	46	102	106	74
# of robot points	103	26	18	12
min. clearance	5%	3%	3%	5%
opened nodes	2091	775	1161	372252
closed nodes	1406	487	1032	360914
steps in path	756	426	426	124
run time (secs.)	366	13	28	8900

Table II. Comparative results of running Benchetrit's planner (A) and our planner (B) on the example in Figure 4.

	Planner A		Planner B	
clearance	6%	2%	6%	2%
opened nodes	1933	4207	809	508
closed nodes	641	1461	399	378
steps in path	453	959	351	366
run time(sec)	114	332	17	10
max. tree span	7	8	–	–

6. CONCLUSIONS

Motion planning in geometrically complex environments is an important category of problems for which no practical satisfactory solution has been found yet. This paper presents a step towards that goal. Building on the theoretical basis and methods described in reference 3, a practical motion planner was developed, with improved search capabilities. The algorithm, is an incremental cell decomposition algorithm that builds a graph of linearized convex cells. The novel characteristics of the algorithm are: (a) free workspace representation as overlapping convex regions (b) canonical configuration space cell representation with no redundant constraints (c) search strategy based on the geometry of the cell, and (d) use of linear programming to move inside and between cells. A key feature of the algorithm is that, unlike all other planers, its complexity does not significantly increase as the clearance of narrow passages decreases. A three degrees of freedom version of the algorithm was successfully implemented and tested. The results clearly show the advantages of the suggested algorithm: reduced number of configuration space cells, small number of unnecessary steps, and no use of collision detection tests.

Several extensions are contemplated for the future. First and foremost, more extensive testing on more difficult examples and quantitative comparison with the performance of existing implementations (e.g. reference 10). Second, improvements in several aspects of the algorithm, such as including the moving edge fixed vertex contact constraints, using adaptive cost function, and precomputing intermediate interference-free configurations. Third, a three-dimensional version of the algorithm. The core planning algorithm remains the same, since the linearization carries over to three dimensions, as demonstrated in reference 3. Issues of cell exploration and workspace subdivision, and comparison with reference 21 require further investigation.

Acknowledgements
We thank Dr. Uri Benchetrit for letting us use his programs and examples and Michel Berkelaar for his help with the linear programming solver.

References

1. Y. Koga, K. Kondo, J. Kuffner and J-C. Latcombe, “Planning Motions with Intentions” *Proceedings of SIGGRAPH’94* (1994) pp. 35–48.
2. P. Finn, D. Halperin, L. Kavraki, J.C. Latombe, R. Motwani, C. Shelton and S. Venkat, “Geometric Manipulation of Flexible Ligands” *Proceedings of the ACM Workshop on Applied Computational Geometry* (ACM Press, 1996) pp. 244–252.
3. L. Joskowicz and R.H. Taylor, “Interference-Free Insertion of a Solid Body into a Cavity: An Algorithm and a Medical Application”, *Int. J. Robotics Research* **15**(3), 211–229 (1996).
4. A. Schweikard, R. Tombropoulos, L. Kavraki, J. Adler and J-C. Latombe, “Treatment Planning for a Radiosurgical System with General Kinematics” *Proceedings of the IEEE International Conference on Robotics and Automation* (IEEE Press, 1994) pp. 1720–1727.
5. Y. Lasovsky, *Motion Planning in Crowded Planar Environments* (MSc. Thesis, The Hebrew University, 1988).
6. S.J. Buckley, “Fast Motion Planning for Multiple Moving Robots”, *Proceedings of IEEE International Conference on Robotics and Automation* (IEEE Press, 1990) pp. 633–639.
7. A. Giraud and D. Sidobre, “A Heuristic Motion Planner Using Contact for Assembly” *Proceedings of IEEE International Conference on Robotics and Automation* (IEEE Press, 1987) pp. 524–530.
8. B. Donald, “A Search Algorithm for Motion Planning with Six Degrees of Freedom” *Artificial Intelligence* **31**, 295–354 (1987).
9. D. Hsu, J-C. Latombe and R. Motwani, “Path Planning in Expansive Configuration Spaces” *Proceedings of the IEEE International Conference on Robotics and Automation* (IEEE Press, 1997) pp. 125–131.
10. D. Hsu, L. Kavraki, J-C. Latombe, R. Motwani and S. Sorkin, “On Finding Narrow Passages with Probabilistic Roadmap Planners” *Preprints of the 3rd Workshop on Algorithmic Foundations of Robotics* (1998) pp. 161–166.
11. L. Kavraki, P. Svestka, J-C. Latombe and M. Overmars, “Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces” *Proceedings of the IEEE Transactions on Robotics and Automation* (1996) **Vol. 12**(4), pp. 123–145.
12. J. Barraquand, L. Kavraki, J-C. Latombe, T-Y Li, R. Motwani and P. Raghavan “A Random Sampling Scheme for Robot Path Planning” *Int. J. Robotics Research* **16**(6), 759–774 (1997).
13. U. Benchetrit, *Robot Motion Planning* (PhD Thesis, Dept. of Mechanical Engineering, Technion – Israel Institute of Technology, 1996).
14. M. Ponamgi, D. Manocha and M. Lin, “Incremental Algorithms for Collision Detection Between Solid Models”, *Proceedings of the 3rd Symposium on Solid Modeling and Applications* (1995) pp. 293–304.
15. H. Chang and T-T. Li, “Assembly Maintainability Study with Motion Planning”, *Proceedings of IEEE International Conference on Robotics and Automation* (IEEE PRess, 1995) pp. 1345–1351.
16. T-Y. Li and H. Chang, “Design for Maintenance by Constrained Motion Planning”, *Proceedings of the International Conference on Robotics and Automation* (IEEE Press, 1996) pp. 334–360.
17. B. Faverjon and P. Tournassoud, “A Local Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom”, *Proceedings of the IEEE Conference on Robotics and Automation* (1987) pp. 324–330.
18. M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry* (Springer, Berlin, 1997).
19. J-C. Latombe, *Robot Motion Planning* (Kluwer Academic Publisher, Dordtrecht, 1991).
20. T. Huynh, L. Joskowicz, C. Lassez and J-L. Lassez, “Practical Tools for Reasoning about Linear Constraints” *Fundamenta Informaticae* **XV**, 357—380 (1991).
21. N. Amato, O.B. Bayazit, L.K. Dale, C. Jones and D. Vallejo, “OBPRM: An Obstacle-Based PRM for 3D Workspaces”, *Preprints of the 3rd Workshop on Algorithmic Foundations of Robotics*, Houston, Texas (1998) pp. 320–332.