Robotica (2006) volume 24, pp. 539–548. © 2006 Cambridge University Press
doi:10.1017/S0263574705002262 Printed in the United Kingdom

A local-based method for manipulators path planning, using sub-goals resulting from a local graph

S. Zeghloul*, C. Helguera** and G. Ramirez*

(Received in Final Form: September 10, 2005. First published online: July 7, 2006)

SUMMARY
This paper addresses the path planning problem for manipulators. The problem of path planning in robotics can be defined as follows: To find a collision free trajectory from an initial configuration to a goal configuration. In this paper a collision-free path planner for manipulators, based on a local constraints method, is proposed. In this approach the task is described by a minimization problem under geometric constraints. The anti-collision constraints are mapped as linear constraints in the configuration space and they are not included in the function to minimize. Also, the task to achieve is defined as a combination of two displacements. The first displacement brings the robot towards to the goal configuration, while the second one allows the robot to avoid the local minima. This formulation solves many of classical problems found in local methods. However, when the robot acts in some heavy cluttered environments, a zig-zaging phenomenon could appear. To solve this situation, a graph based on the local environment of the robot is constructed. On this graph, an A* search is performed, in order to find a dead-lock free position that can be used as a sub-goal in the optimization process. This path-planner has been implemented within SMAR, a CAD-Robotics system developed at our laboratory. Tests in heavy cluttered environments were successfully performed.

KEYWORDS: Manipulators; Local minima; Collision avoidance; Path planning; Dead-lock; Zig-zaging phenomenon.

1. BACKGROUND
The development of technologies for autonomous robots raises many different and important problems. One of these problems is the collision-free path planning for manipulators, which has a primary importance for automatic programming of robots. In robotics, this problem can be stated as finding a trajectory from an initial to a goal configuration while avoiding the obstacles placed in the environment.

* Laboratoire de Mécanique des Solides, UMR CNRS 6610, Université de Poitiers, SP2MI, Bd. Pierre et Marie Curie, BP 179, 86962 Futuroscope de Chasseneuil (France)
** Instituto Tecnológico y de Estudios Superiores de Monterrey – Campus Laguna, Paseo del Tecnológico No. 751, Col. Ampliación la Rosita, Torreón, Coah. CP. 27250 (Mexico)
Corresponding author: G. Ramirez, E-mail: ramirez@lms.univ-poitiers.fr

There exist a large number of methods which attempt to solve the path planning problem, however, there are very few effective approaches for solving these problems for manipulators with high number of degrees of freedom working in cluttered workspaces. Latombe[1] made a good resume of theses approaches. Despite many differences, these approaches can be classified in two general classes: the global and the local methods.

The global methods need a complete description of the free space where the robot can move safely. In global planners, the robot is represented by a point evolving in the configuration free space, whose coordinates are the components of the configuration vector. In order to obtain the configuration free space the global methods needs an obstacle transformation from their cartesian representation into the robot's configuration space (C-space). Also, the geometric planning is performed before the path execution. Then, the planner produces a path from the initial configuration to the final configuration (if there exists one) which avoids the obstacles. Some examples of this approach are the cell decomposition proposed by Lozano-Pérez,[2,3] the well-known octree method,[4,5] the generalized cones,[6] the voronoi diagram,[7,8] the visibility graphs,[9] etc.

These methods are computationally expensive and need a large space of memory so they are not well adapted for on-line applications. However, these planners have proved to be successful when the number of degrees of freedom considered is small (3 or 4 d.o.f.), as they always find a collision free path, if there exists one.

In the other hand, the local methods do not need a prior knowledge of the entire environment. They are sensor-based methods, in order to build a representation of the environment in which the robot moves. For a large family of robotics problems involving manipulators with high number of degrees of freedom, these planners are very efficient. Developed by Khatib,[10] the Artificial Potential Field approach is the most popular local method. In this method the goal position attracts the robot towards it, while obstacles act as repulsive forces. For each iteration, the artificial force is regarded as the most promising direction of motion, and path generation proceeds along this direction by some increment. There exist many different approaches inspired in this idea, as the superquadric potentials,[11] the Newtonian potentials,[12] the heat transfer model potentials,[13] the constraints method,[14] among others.

The artificial potential methods are well adapted for real-time applications, as well as for path-planning in robotic work-cells with high number of degrees of freedom.

However, the addition of attractive and repulsive potentials can lead to the appearance of the major problem in local methods: the presence of local minima in the potential function.

As we pointed out above, local minima remain an important cause of inefficiency of the local methods. Also, the global methods are computationally expensive and not well adapted for on-line applications. In order to solve these problems, many researches mixed these two techniques. The global techniques (node construction, cell division, etc . . .) are generally used in a preprocessing stage in order to construct a graph and find an initial path, and then a local technique is used to move between the adjacent nodes or cells in the graph. We can cite the mixed method proposed by Faverjon,[15,16] Warren,[17] or the approaches based on the randomized path planner (RPP)[18–20] or the probabilistic roadmaps (PRM).[21,22]

In the mixed methods, building the graph is computationally expensive, but once developed off-line, it can be usually queried effectively with any start and goal configurations. Nevertheless, when path planning involves many different workspaces and robots, these techniques seem unappropriates. We propose a local planner which uses the graph-searching techniques on-line, avoiding the off-line preprocessing and solving the local minima problems.

In this approach, the path calculation is only based on a local view of the environment, and the dynamic properties of the manipulator are ignored. These considerations transform the physical motion planning problem into a purely geometrical path planning problem. The motions of the manipulator are only constrained by the obstacles and by the manipulator itself (joints limits, velocity limits, etc.).

The approach that we propose is first based on the constraints method developed by Faverjon,[14] and it is the continuation of the planner that we previously presented.[23] The method can be divided in two complementary modules: the path planning module and the unblocking module. In the path planning module a free-collision path is found using the constraints method, using a particular task description. The second module is activated when the manipulator get caught in a dead-lock situation, the unblocking algorithm found a sub-goal in order to escape from the blockage.

2. PATH PLANNING METHOD

The proposed collision-free path planner is based in the local method developed by Faverjon.[14] As mentioned above, planning safe trajectories means finding a path that brings the manipulator from an initial configuration $\boldsymbol{q}_i$ to the final configuration $\boldsymbol{q}_f$, avoiding collisions. In this paragraph, the local method proposed in reference [14] for trajectory planning, while avoiding collision with obstacles in workspace, is presented. This planner is installed within SMAR CAD-Robotics system.[24]

2.1. Constraints method

This approach is a substitute of the potential field approach. The main idea of the method is to separate the achievement of the task from the constraints of anti-collision between the moving objects and the environment. The task is described by the minimization of the deviation of the trajectory from the line, in the C-space, defined by the current configuration and the final configuration, plus eventually some geometric constraints. As opposed to classical potential field method, collision avoidance is represented by simple linear constraints in the C-space and is not included in the function to minimize.

2.1.1. Task description. Classically, local methods are based on an iterative procedure where the variation of the configuration vector is found by using the following formula:

$$q_i = q_{i-1} + dq_i \tag{1}$$

where dq_i is the variation of the configuration vector at the iteration i. It is calculated by using the artificial potential methods or according to the constraints method. In our case, the task is defined by the final configuration q_f to reach, as follows:

$$\tau(q) = q - q_f \tag{2}$$

where q represents the current configuration vector of the robot.

From this definition, the path planning problem can be stated as the following minimization problem:

$$\text{minimize } f = \frac{1}{2}\|\dot{\tau}(q) - \underline{\dot{\tau}}\|^2 \tag{3}$$
$$\text{with } \underline{\dot{\tau}} = -\frac{\tau(q)}{\max_{i=1,\ldots,n} |\tau(q)^i|}$$

where $\underline{\dot{\tau}}$ is the desired value for $\dot{\tau}(q)$ in order to obtain a straight line to the goal in the C-space. At each step this parameter is recalculated without saving in memory the deviations caused by the obstacles.

2.1.2. The velocity damper. In order to construct the geometric constraints of the problem, Faverjon defines the anti-collision constraints. They are translated on geometric terms in the C-space.

The anti-collision constraint, also called *velocity damper*, assures that the distance's variation has to be bigger than a value which avoids collisions between the objects. If d is the minimal distance between the objects, then d has to remain bigger than a security distance d_s. Thus the authors show that there cannot be any collision if there is imposed:

$$\dot{d} \geq -\xi \frac{d - d_s}{d_i - d_s} \; pour \; d < d_i \tag{4}$$

where d_i is the influence distance from which a constraint is included in the calculation process, $\boldsymbol{\xi}$ represents a positivce coefficient in order to adapt the convergence, $\dot{d}$ is the time derivative of the distance which defines the anti-collision constraint. It is shown by the authors[14] that the distance variation can be obtaiend from to the configuration parameter variation by using $\dot{d} = (J'n \mid dq)$, J being the Jacobean matrix calculated at the points of minimal distance, n is a unit vector along the segment that connects these points and | means the inner product.

The inequality (4) can be translated into a simple linear constraint on configuration parameters' increments dq between time t and $t + dt$ as follows:

$$J^t n \,|\, dq \geq -\xi \frac{d - d_s}{d_i - d_s} dt. \tag{5}$$

2.2. *Proposed method*

The constraints method mentioned above finds a trajectory in a large number of cases. However, the local description of the C-space does not always allow finding a path up to the goal. This method acts, as the original potential fields approach, as a fast gradient-descent optimization procedure, and has a limited ability to deal with the local minima of the function in cluttered environments. Thus, we propose to solve these dead-lock phenomena through an analysis of the local geometry of the environment.

Generally, a dead-lock situation can be detected when the active geometrical constraints remain the same through several iterations, so in order to avoid this problem, it would be sufficient to determine a movement that tend to change from one set of constraints to another set. According to this idea, the displacement dq will be defined as follows:

- if $d > d_i$, none of the constraints are active and the trajectory is a straight line to the goal. In this case, the displacement is given by:

$$dq = \frac{q - q_f}{\max_{j=1,\ldots,nddl} \|q - q_f\|} \tag{6}$$

- if $d < d_i$, one or several constraints are active. The displacement is obtained by combining two displacements dq_{goal} and dq_{block}, according to the following relation:

$$dq = \sum_{k=1}^{N} \alpha^k dq_g^k + (1 - \alpha^k)\, dq_l^k \quad \text{with} \quad \alpha^k = \frac{d^k - d_s}{d_i - d_s} \tag{7}$$

where N is the number of active constraints, dq_g^k is the displacement calculated according to (6), dq_l^k is the displacement that allows to change from one set of constraints to another, α^k is a weighting coefficient for the two calculated displacements according to the distance d^k for the constraint k.

If the distance d^k is close to the security distance, priority is given to the movement that enables the robot to change the constraint, whereas if d^k is close to the distance of influence, priority is given to the movement which would bring the robot closer to the final configuration. The displacement dq_l^k is obtained using an inverse differential model:

$$dq_l^k = J^t (J J^t)^{-1} \cdot dx_i^k \tag{8}$$

In order to change the active constraints the movement dx_i must be defined. It is this displacement that will allows the manipulator moves towards another geometrical constraint (face, edge, vertex).

Considering the Figure 1, the local methods are classically in a blockedstate. The minimum distance between the objects

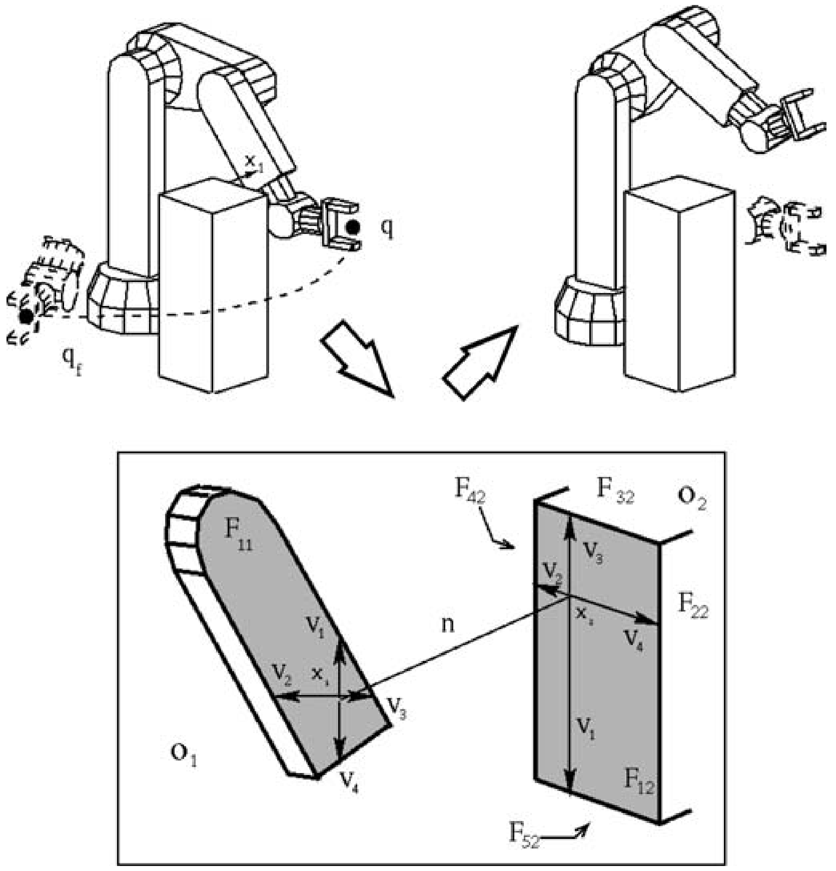

Fig. 1. Second displacement choice.

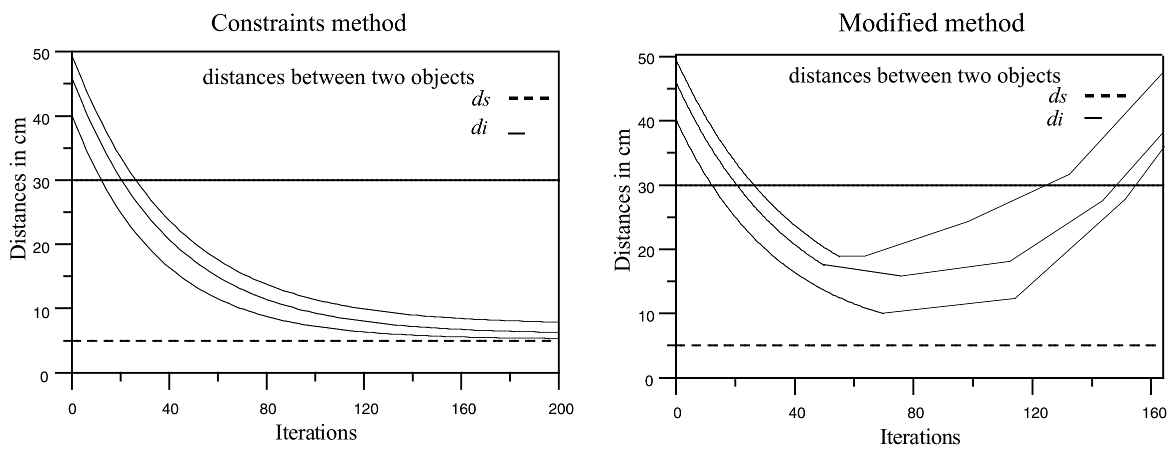

Fig. 2. Distances evolution.

O_1 and O_2 is obtained at points x_1 and x_2 of planes F_{11} and F_{12}. In this situation the constraint remains active on planes F_{11} and F_{12}. This means that the minimal distance which makes the constraint active is always obtained from two points belonging to the facets F_{11} and F_{12}. It is this situation which leads to the blocked condition mentioned above. In order to avoid this problem, a displacement that leads to the modification of a set of constraints must be defined. Consequently, to find the displacement according to those points which give the minimal distance, they must be placed on faces other than F_{11} and F_{12}. This is achieved by calculating distances between x_2 and the different adjacent facets F_{22}, F_{32}, F_{42}, F_{52}, on the obstacle. For each different facet, there is a direction of displacement that is defined by $\vec{V}_1, \vec{V}_2, \vec{V}_3, \vec{V}_4$ which allows the robot to change the constraints on the obstacle. To switch the constraints on the object O_1 the movement must be along the direction $\vec{V}_i$. This direction is obtained by projecting $-\vec{V}_i$ on the facet F_{11}. There are m displacement possibilities in order to avoid the obstacle. Between these possibilities a movement is chosen, and is the one that permits switching rapidly from one set of constraints to another: this displacement is established along the following direction: $min\, i = 1, \dots, m \|\vec{V}_i\| + \|\vec{\vec{V}}_i$. In the example shown in the Figure 1, the method allows to obtain a variation of the configuration vector such that the displacements along $\vec{V}_3$ lead to the possibility of avoiding the obstacle from the top. Thus, the obtained displacement dq (eq. (6)) gives the direction that the robot must follow in order to avoid the dead-lock situation. Finally, to avoid the collisions, this displacement is used as a subgoal in the optimization method under anti-collision constraints.

In the figure 2 we traced the evolution of the distances of three elements of the manipulator showed in figure 1. We can see that in the curves of the original constraints method the trajectory goes forward until the security distance. On the other hand in the modified method, the algorithm anticipates the obstacle, aided by the second displacement, avoiding the convergence towards the security distance.

2.3. *The zig-zaging phenomenon*

This improvement to the formulation proposed by Faverjon solve many of the problems found when a manipulator acts in heavy cluttered environments. Nevertheless, it doesn't completely eliminate the local minima. In fact, the manipulator can still converge towards a local minimum. In order to escape from this minimum the process finds a path that can lead towards another minimum, and when is avoiding this second minimum, it can reconverge to the first one, thus the manipulator moves locally from one minimum to another alternatively (see figure 3), leading to a dead-lock situation. This is called a zig-zaging phenomenon.

This situation can be avoided by searching a solution in a graph. This graph is constructed when the zig-zaging phenomenon is detected and it is obtained from the analysis of the local geometry of the environment.

3. SOLVING BLOCKAGES USING A LOCAL GRAPH

The unblocking procedure consists in building, on line, a graph describing the local geometry of the environment, then an A* search is carried out in the graph, in order to find a sub-goal to use in the optimization process.

The unblocking algorithm, activated when a dead-lock appears, can be stated as follows:

i. Build a graph by sweeping the local environment of the end-effector in the cartesian space.
ii. Test for collisions an evaluate a cost function for each cell.
iii. Search for the best cell with an A* algorithm.
iv. Take this position as a sub-goal in the minimization process.
v. When the sub-goal is reached, retake the path-planning process to the original goal.

3.1. *Blockages detection*

In order to start the procedure to solve the zig-zaging phenomenon, such a situation must be detected. As mentioned above, a dead-lock situation generally appears when the active geometrical constraints remain the same between two iterations of the path-planning process. The variations of the configuration vector are so small, that it can be deduce that the robot does not move. This problem was practically solved with the modifications to

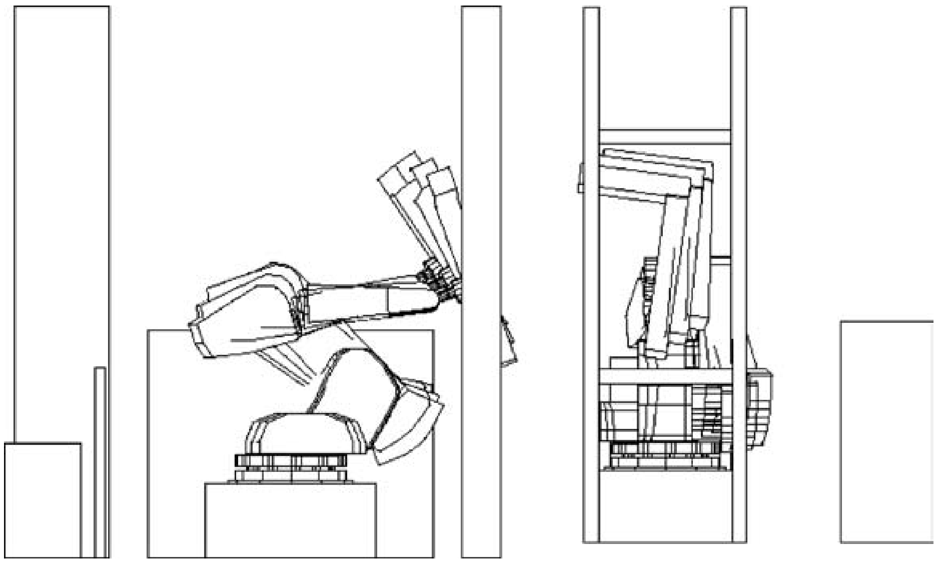

Fig. 3. The zig-zaging phenomenon.

the constraints method mentioned above. Nevertheless, zig-zaging phenomena could appear. This phenomenon can be detected if the variation of the position on several iterations is memorized. Normally, in this kind of blockage the variations in the configuration vector between two iterations are practically the same that those measured at the next iteration. This behavior results from the fact that the manipulator moves from one position to another, to come back to the first one. Then, if this variation is appreciably the same, the unblocking procedure can be activated.

3.2. Graph construction

The use of graphs for path planning in robotics is very extensive (cf. section I). These graphs can be very simple or very complex, the complexity is function of the amount of space explored and the level of discretization. Our graph is limited to the local environment of the end-effector, in such a way that the real-time spirit of our path-planner is respected.

In our case, the graph is a representation of the local environment in the Cartesian space. The node, from which the graph is constructed, represents the position in which the manipulator is in dead-lock situation. From this position the local Cartesian space is swept, and only a few new positions (cells) are chosen. For each position, the euclidean distance to the goal (in the C-space) is measured and the number of active constraints is computed. These values are kept in order to evaluate a cost function for each cell. The cost function will be described latter in this paper.

The searching procedure is stopped when a collision-free cell is detected, such its value of the cost function is lower than the one of the blockage cell. Whereas none cell is detected, a new graph, with a higher discretization of space, is constructed and a new search is performed.

The searching graph represents just some positions of the end-effector in the Cartesian space, in order to reduce the distance calculation and collision tests. The graph showed in the figure 4 corresponds to a representation of the end-effector's local environment in the Cartesian space. From the central cell of the graph (cube), which represents the dead-lock position, the graph is constructed forming a cube containing 26 cells to explore, the 27th is the central one (cell number 14).

3.3. The cost function

The cost function have to reproduce as well as possible the characteristics of each cell (or position) evaluated, but at the same time such a function have to be simple, in order to keep the performance of the local planner. Generally, in path planning and specially in the constraints method, we need to know the final position to reach and the set of active constraints at each iteration, so the cost value for a cell can be obtained from these two variables, using the following equation:

$$F = g + h \tag{9}$$

where g is obtained from the number of active constraints and h is function of the goal distance in the C-space. The position (cell) with the minimum cost is kept as a subgoal for the path-planning process. If two nodes have the same function evaluation, the node with the minimum h is used, since it is the nearest position to the goal.

As mentioned above, the variable g is function of the number of active constraints, obtained by the following expression:

$$g = \exp(-\zeta/k) \tag{10}$$

where k is the number of active constraints at the considered position and ζ is a coefficient which can be changed in function to the distance to the goal. The figure 5 shows the behavior of the estimation g, which moves between $[0, 1]$, for different values of ζ. For the most of the cases, $\zeta = 1$ give good results; nevertheless, for some particular cases (in special when the robot is close to the goal), it could be necessary to increase ζ. The value of g increases very quickly for the first constraints. This exponential behavior is very useful, since by experience we know that a blockage

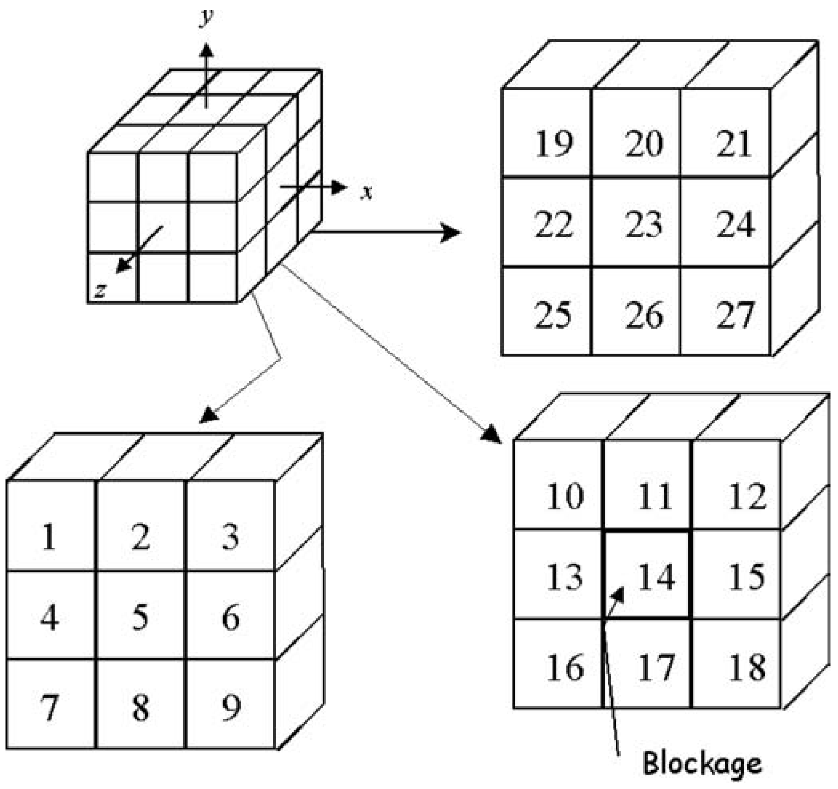

Fig. 4. The local graph.

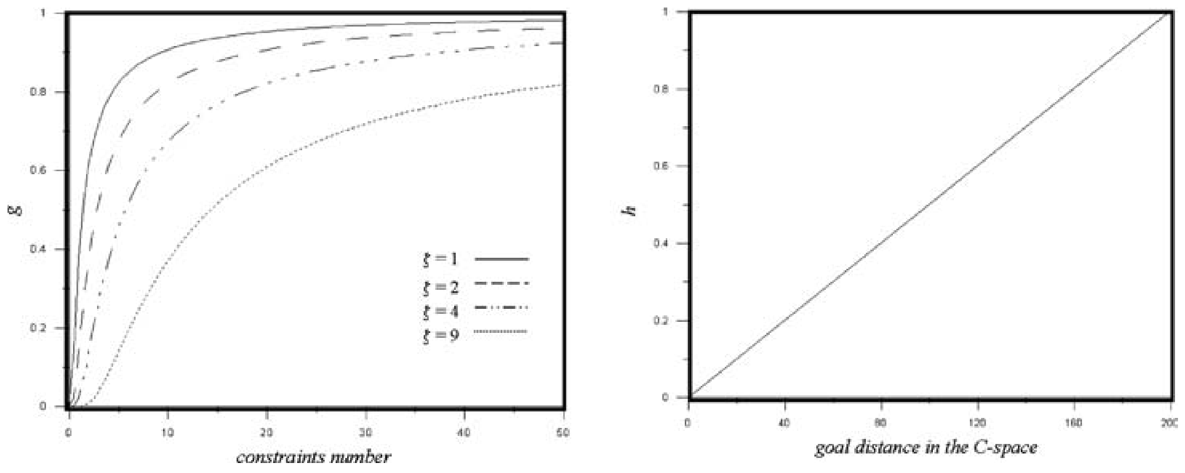

Fig. 5. The two variables behavior.

may appears when only a few constraints are active, so we have to penalize those cells.

The variable h is function of the goal distance in the configuration space and is computed by:

$$h = \frac{\|q_f - q\|}{\|q_f - q_i\|} \tag{11}$$

where q represents the node configuration vector and q_i and q_f the initial and final configuration vectors respectively. In the figure 5 it can be seen the h estimation behavior. This behavior is linear between $[0, 1]$, if the distance to the goal decreases, the cost of h also decreases. If $h > 1$ we take $h = 1$, in order to kept a balance in the cost function between g and h. Taking h as a function of the distance in the C-space tends to give preference to paths with short execution time, which in a real-time planner, is the most interesting response.

3.4. Searching for a path in the local graph

A very effective free-space searching algorithm, in terms of path length optimization, is the A* technique.[25] The basic A* strategy is to compute a set of neighbors, select the most promising one and iterate with it. The process terminates either when the goal is reached or when no further movement is possible. While A* methods can have unbounded running times, path-planning is often halted after a given number of iterations. Due to the amount of space that is considered, the A* technique has a tendency to be a slow planner. If the speed

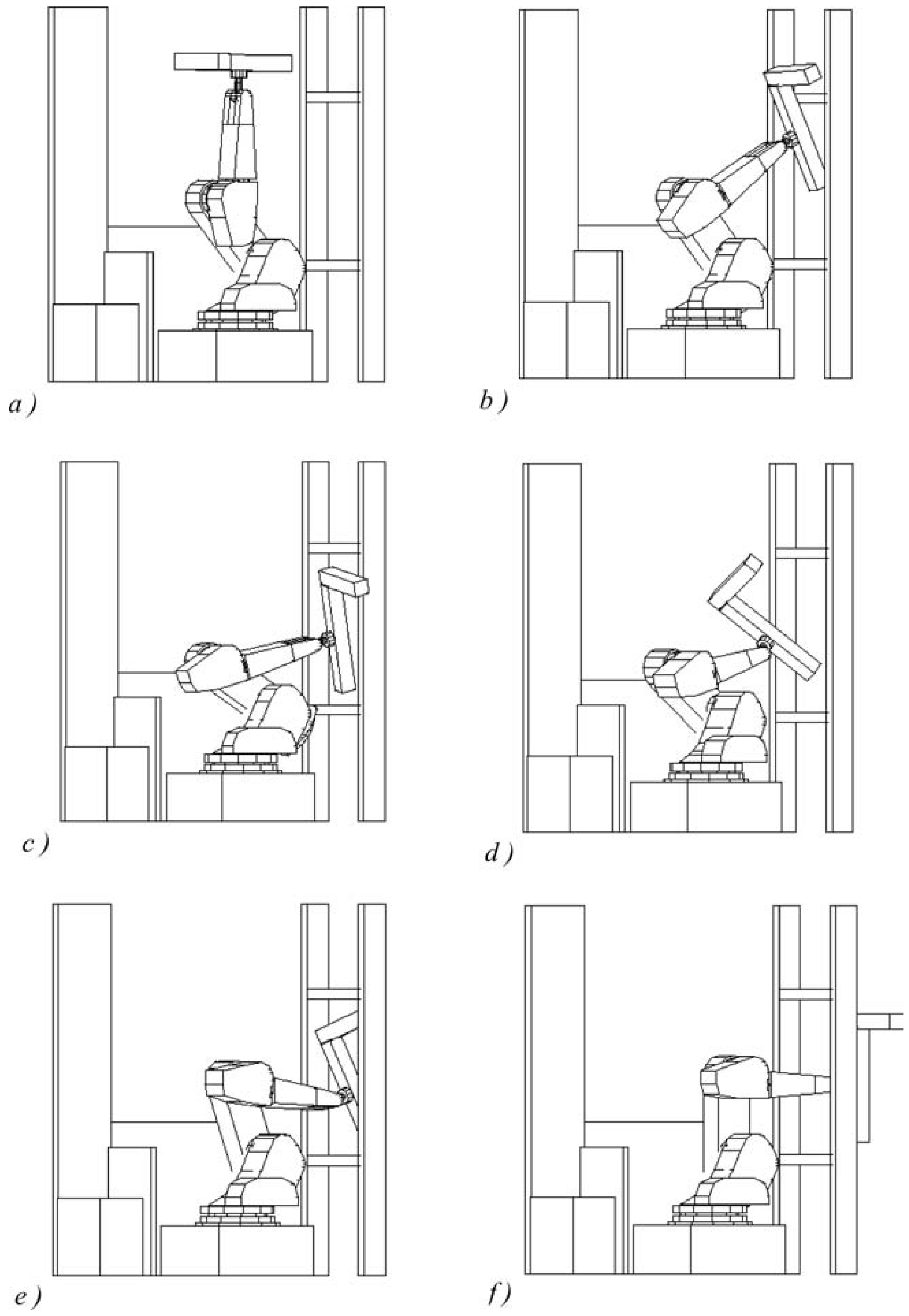

Fig. 6. Example No. 1.

of the algorithm is increased, it could be ideal for real-time path planning. In this approach the amount of space searched (the graph) is reduced to the local environment in order to obtain real-time solutions.

When the function F is computed for all the cells in the graph, the algorithm choose the one with the minimum cost. This new blockage-free position is taken as a subgoal in the optimization process. When the subgoal is reached, the planner switch back to the original goal and the unblocking procedure is done. If a blockage reoccurs a new local graph is constructed in that point and a new search is performed.

If in the first development there are no solutions, a bigger graph is developed and a depper search is performed. However, when a given amount of time has passed out, the search is stopped and failure is reported. The amount of

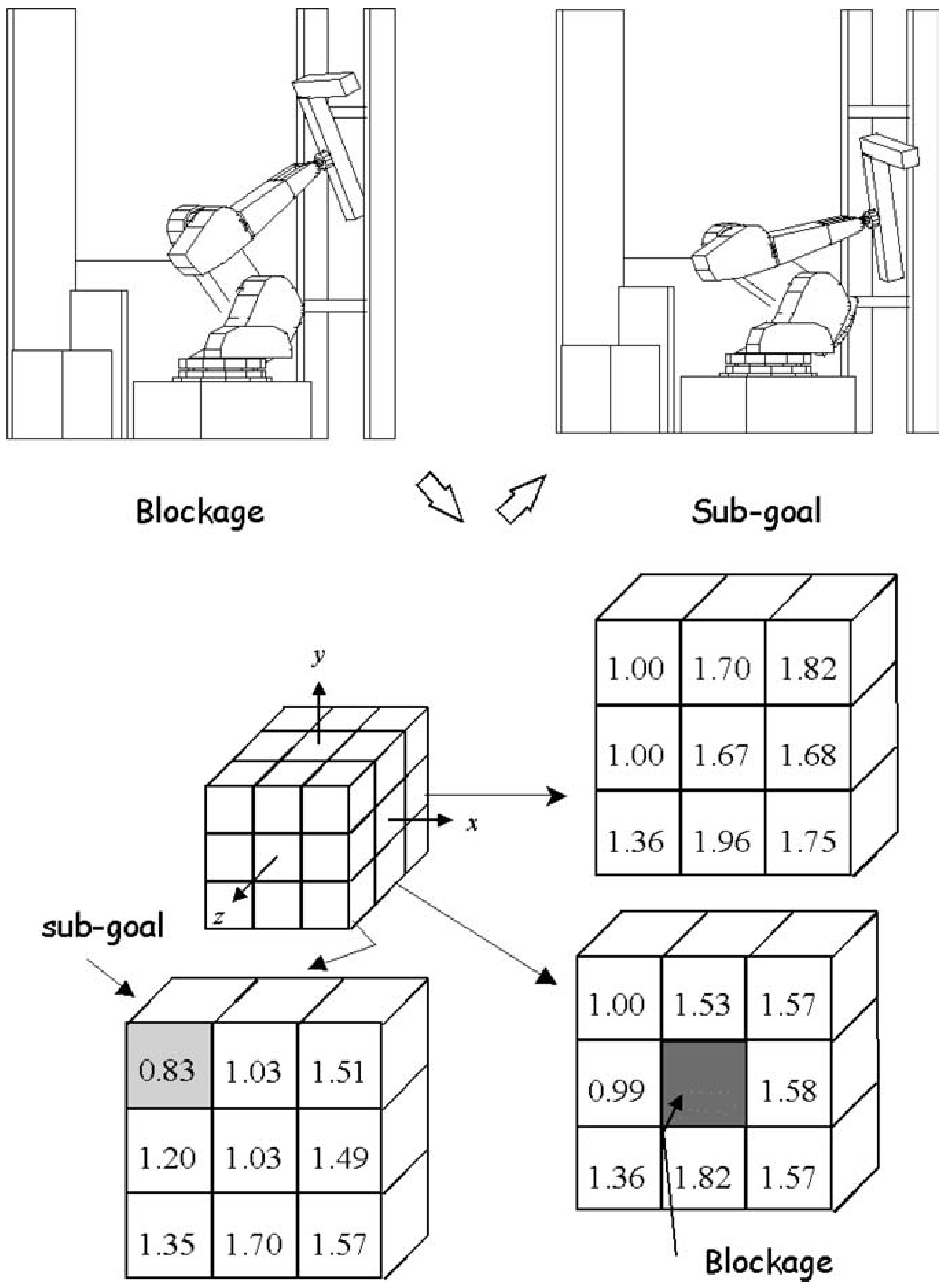

Fig. 7. Local graph for example No. 1.

time used in the search is constrained in order to respect the real-time spirit of the approach. As it can be seen, if no restrictions are indicated, the unblocking algorithm works as a classical A* search, nevertheless the time needed to find the goal position will be increased dramatically.

4. RESULTS

The path planning algorithm, as well as the unblocking algorithm, is installed within SMAR,[23] our CAD-Robotics system, developed in our laboratory using Xlib and Silicon Graphics Inventor libraries. In this example (figure 6) the manipulator must make an insertion of an object of large dimension in a cluttered environment. The dimensions of the object manipulated are larger than the width of the aperture; this fact does not allow an insertion purely horizontal. The planner determine the trajectory from the initial configuration q_i towards the final configuration q_f with an influence distance of 30 cm and a security distance of 5 cm in 138 iterations. The execution time necessary to

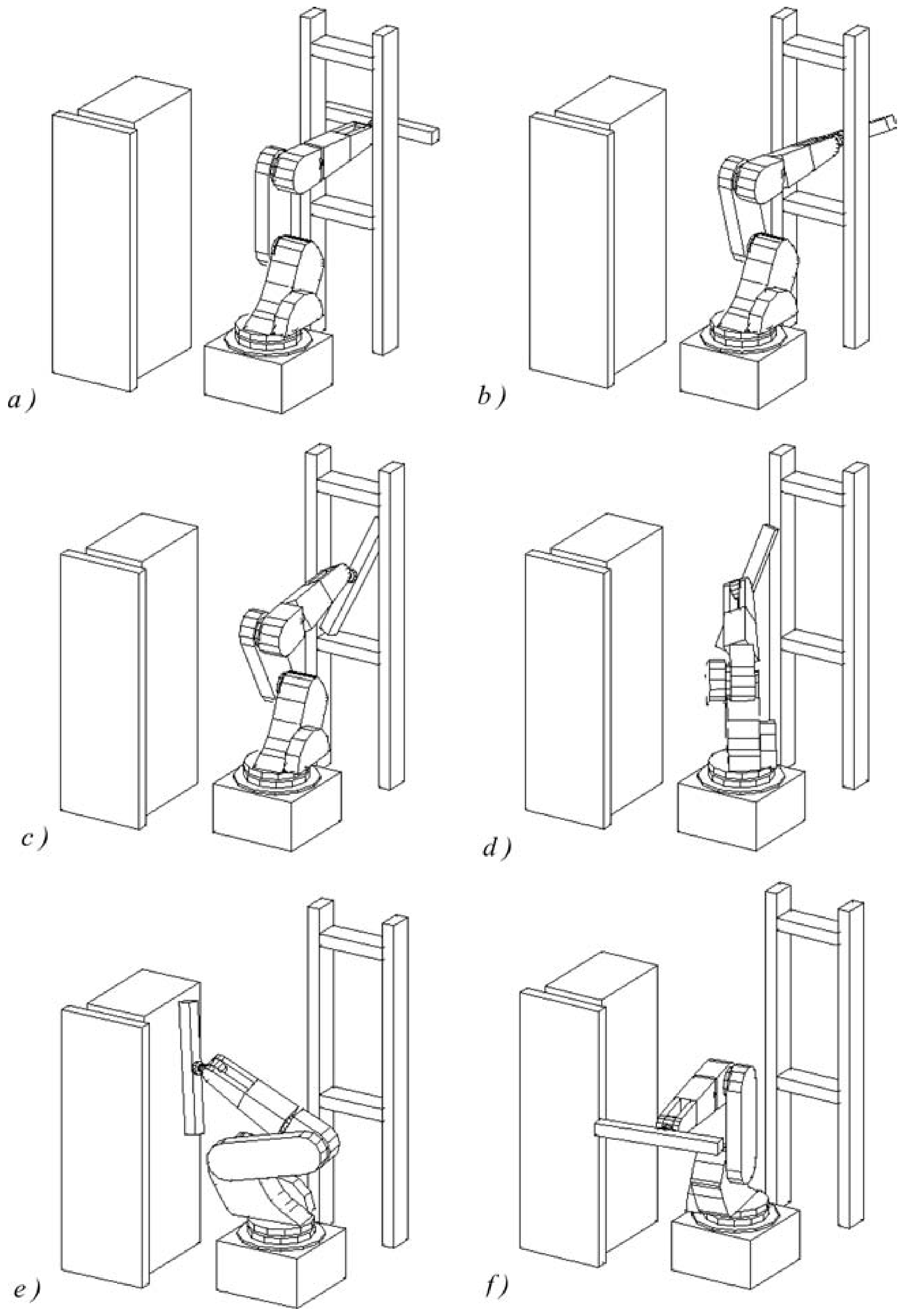

Fig. 8. Example No. 2.

determine the solution for this example, using a Silicon Graphics workstation with a R10000 processor, is 8 seconds. The figure 7 shows how a blockage situation is solved using the local graph. The figure 6-b represents a dead-lock situation, so since this position a local graph is constructed and a sub-goal is found (figure 6-c). Thus, this sub-goal is used in the optimization process, and when this position is reach the original goal is considered again. If another blockage situation appears, a new graph is constructed and a new sub-goal is created. In the case of this example, (figure 6) the unblocking procedure was activated three more times.

Figure 8 shows another example that classical local methods are unable to solve. In this case the dimensions of the bar are larger than the width of the aperture, an also the bar can at any moment get in collision with the manipulator. This situation restricts, even more, the movement of the robot.

For solving this problem, 167 iterations are required, using the same security and influence distances that the previous example, which corresponds to 15 seconds of computation time.

5. CONCLUSION

We have presented a local-based approach for collision-free path-planning for robotic manipulators. In the first part of this paper, a solution for the local minima problem, associated to local planners, has been proposed. Indeed, the new task definition allows the manipulator to avoid these problems even when a little free space is available. However, a zig-zaging phenomenon appears when the manipulator moves in some heavy cluttered environments. In order to solve this problem, the construction of a graph based in the local geometry has been proposed. Then, a free-dead-lock position is founded in the graph with an A* search. The planner takes this new position as a sub-goal to escape from the dead-lock phenomenon.

In this paper we showed cases where the classical local methods are unable to provide a solution. The proposed method has been applied and tested for many robotics applications where the manipulator is placed in some heavy cluttered environments; in such situations the planner finds a collision-free trajectory in almost all cases. These results shown that an A* search, based on a local graph, is a good alternative for the resolution of the dead-lock problems that appear in the classical local methods.

References

1. J. C. Latombe, *Robot Motion Planning* (Kluwer Academic Publishers, 1991).
2. T. Lozano-Pérez, Spatial planning: A configuration space approach, *IEEE Trans. Comput.* **C-32**, 108–120 (Feb. 1983).
3. T. Lozano-Pérez, “A Simple Motion-Planning Algorithm for General Robot Manipulators,” *IEEE Journal of Robotics and Automation* **3**, No. 3, 224–238 (1987).
4. B. Faverjon, “Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator,” *IEEE: Procs. of the Int. Conf. on Robotics and Automation*, Atlanta, Ga (1984), pp. 504–512.
5. D. Jung and K. Gupta, “Octree-Based Hierarchical Distance Maps for Collision Detection,” *IEEE: Proc. of Int. Conf. on Robotics and Automation* (1996) pp. 454–459.
6. R. A. Brooks, “Solving the find-path problem by good representation of the free space,” *IEEE Transactions on Systems, Man and Cybernetics* **SMC-13**(2), 190–197 (March/April, 1983).
7. C. O’Dunlaing and C. K. Yap, “A Retraction Method for Planning the Motion of a Disc,” *Journal of Algorithms* **6**, 104–111 (1985).
8. R. Mahkovic and T. Slivnik, “Generalized Local Voronoi Diagram of Visible Region,” *Proc. Of the IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium (May 1998) pp. 349–355.
9. S. K. Ghosh and D. M. Mount, “An Ouput Sensitive Algorithm for Computing Visibility Graphs,” *IEEE 28th Annual Symposium on Fondations of Computer Science*, Los Angeles, Ca. (1987), pp. 11–19.
10. O. Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” *Int. J. Robotics Research* **5**, No. 1, pp. 90–98 (1986).
11. R. Volpe and P. Khosla, “Artificial potentials with elliptical contours for obstacle avoidance,” *IEEE Proc. of the 26th Conference on Decision and Control*, Los Angeles, Ca. (1987) pp. 180–185.
12. J. H. Chuang, “Potential-Based Modeling of Three-Dimentional Workspace for Obstacle Avoidance,” *IEEE Transactions on Robotics and Automation* **14**, No. 5, 778–785 (Oct. 1998).
13. Y. Wang and G. S. Chirikjian, “A New Potential Field Method for Robot Path Planning,” *IEEE Proc. of Int. Conf. on Robotics and Automation*, San Francisco Ca. (2000), pp. 521–528.
14. B. Faverjon and P. Tournassoud, “A Local Based Approach for Path Planning of Manipulators With High Number of Degrees of Freedom,” *IEEE Proc. of Int. Conf. on Robotics and Automation* (1987) pp. 1152–1159.
15. B. Faverjon and P. Tournassoud, “The Mixed Approach for Motion Planning Learning Global Strategies form a Local Planner,” *Proc. of the 10th Int. Joint Conf. on Artificial Intelligence* Milan (1987) pp. 1131–1137.
16. B. Faverjon and P. Tournassoud, “A Practical Approach to Motion Planning for Manipulators with Many Degrees of Freedom,” *Prep. of the 5th Int. Symposium of Robotics Research* (1989) pp. 65–73.
17. C. W. Warren, J. C. Danos and B. W. Mooring, “An Approach to Manipulator Path Planning,” *Int. J. Robotics Research* **8**(5), 87–95 (1989).
18. S. Caselli and M. Reggiani, “ERPP: An Experience-based Randomized Path Planner,” *IEEE Proc. of Int. Conf. on Robotics and Automation*, San Francisco Ca. (2000) pp. 1002–1008.
19. J. Barraquand and J. C. Latombe, “Robot Motion Planning: A Distributed Representation Approach,” *Int. J. Robotics Research* **10**, 6, 628–649 (1991).
20. J. Barraquand, L. Kavraki, J. C. Latombe, R. Motwani, T. Y. Li and P. A. Raghavan, “Random Sampling Scheme for Path Planning,” *Int. J. Robotics Research* **16**, No. 6, 759–774 (1997).
21. L. Kavraki and J. C. Latombe, “Randomized Preprocessing of Configuration Space for Fast Path Planning,” *IEEE Proc. of Int. Conf. on Robotics and Automation*, San Diego Ca. (1994), pp. 2138–2145.
22. R. Bohlin and L. Kavraki, “Path Planning Using Lazy PRM,” *IEEE Proc. of Int. Conf. on Robotics and Automation*, San Francisco Ca. (2000) pp. 521–528.
23. C. Helguera and S. Zeghloul, “A Local-based Method for Manipulators Path Planning in Heavy Cluttered Environments,” *Procs. Of the IEEE Int. Congress on Robotics and Automation*, San Francisco, Ca. (April 2000) pp. 3467–3472.
24. S. Zeghloul, B. Blanchard and M. Ayrault, “A Robot Modeling and Simulation System,” *Robotica* **15**, Part 1, 63–73 (1997).
25. J. R. Slagle, *Artificial Inteligence:* The heuristic programming approach (McGraw Hill, 1971).