In: The 24th Annual Conference of the IEEE Industrial Electronics Society (IECON'98), Aachen, Germany, August 31 - September 4, 1998.

On-line path planning by heuristic hierarchical search

Dominik HENRICH, Christian WURLL and Heinz WÖRN

Institute for Process Control and Robotics (IPR)
Computer Science Department, University of Karlsruhe
Kaiserstrasse 12, D-76128 Karlsruhe, Germany
e-mail: [dHenrich, Wurll, Woern] @ira.uka.de
http: //wwwipr.ira.uka.de/~paro/

***Abstract* – In this paper, the problem of path planning for robot manipulators with six degrees of freedom in an on-line provided three-dimensional environment is investigated. As a basic approach, the best-first algorithm is used to search in the implicit descrete configuration space. Collisions are detected in the Cartesian workspace by hierarchical distance computation based on the given CAD model. The basic approach is extended by three simple mechanisms and results in a heuristic hierarchical search. This is done by adjusting the stepsize of the search to the distance between the robot and the obstacles. As a first step, we show encouraging experimental results with two degrees of freedom for five typical benchmark problems.**

I. INTRODUCTION

The issue of robot path planning has been studied for a couple of decades and many important contributions to the problem have been made [Hwang92]. Path planning algorithms are of great theoretical interest, but are rarely used in practice because of their computational complexity [Kamal96]. Here, we make a step in the direction of practical path planning.

The problem of path planning we focus on is the following: Input is the geometry of an industrial robot manipulator with (rotational) $d \leq 6$ degrees of freedom in an environment with static obstacles given by their current location. Optionally, there may be a couple of dynamic obstacles. All geometric objects (obstacles, robot) are represented by a number of 3-dimensional convex[1] polyhedrons. Additionally, the start and goal configuration of the robot are given. The output of the problem is a sequence of pairwise neighboring and collision-free robot configurations connecting the start configuration with the goal configuration.

Many of the future robotic tasks (e.g. recycling, robot guidance, tele-operation, assembly and disassembly, medical surgery) can often only be completed in dynamic environments. Therefore, powerful on-line path planners for industrial robots with six degrees of freedom (DOF) are needed. The *on-line* capability[2] means that the planner does not require any time-consuming off-line computations in order to directly react to dynamic changes in the environment.

For dynamic environments, several different cases can be distinguished. In the first case, the environment contains dynamic obstacles (e.g. objects on a conveyor belt, or additional robots) with known or partially known movements. In another case, the robot grips different objects or changes the tool. There, the kinematic chain of the robot, including the gripped object, will change. The next case occurs in the area of virtual engineering. After every assembly operation, the product or the environment will change its geometry. All these cases of dynamic environments implicate a modification of the configuration space (C-space), which has to be considered during planning paths.

An introduction to motion planning in dynamic environments is given in [Fujimura91]. In several examples, different approaches, especially for mobile robots, are presented. In [Fiorini96], a path planner for industrial robots based on velocity adaptation is discussed. It plans only for a 2 DOF workspace for two robots and their off-line known movements. In [Ralli96], a potential-field approach based on the explicit calculation of the workspace and the C-space is proposed. When a new object is detected, the new path is sought within a few seconds, but the planner works only with 5 DOF in a very small search space, which is unfavorable for industrial applications.

In summary, to date, no planners for 6 DOF robots exist which can deal with dynamic environments and have low on-line computation times. Our aim is to develop a path planner satisfying these requirements for robots with up to 6 DOF [Wörn98]. We focus on industrial robots, which

[1] This is no severe restriction, because every non-convex polyhedron can be modelled by (multiple) convex polyhedrons.

[2] Here, "on-line" does not include to meet given time constrains as required for "real-time".

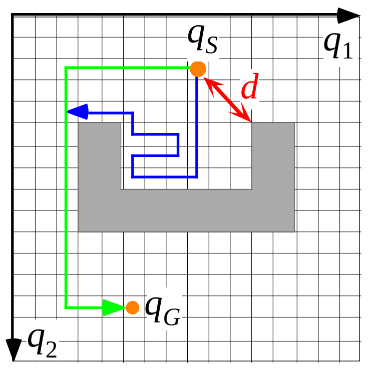
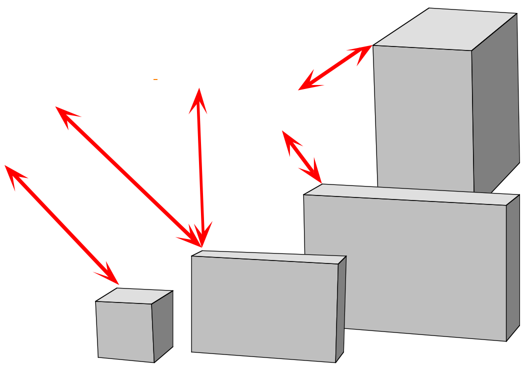


Figure 1: A 2D illustration of the path search in the implicit C-space from the start q_S to the goal q_G using the Cartesian obstacle distance d for collision detection



constitute a considerable fraction of all robots used currently and in future.

The paper is organized as follows: In Section II, we shortly review our basic approach to practical path planning for six DOF. Then, we present a new approach to automatically adapt the stepsize by hierarchical search in Section III. The experimental results for two DOF using a set of benchmark problems are given in Section IV. The paper closes with conclusions and future work in Section V.

II. BASIC APPROACH

Most of the off-line path planners are based on an explicit representation of the *free C-space*. The free C-space computation consists of the obstacle transformation into the C-space and the construction of a free-space representation. Both tasks are very time and memory consuming, and their calculation effort increases with the robot's DOF. In order to avoid these time consuming obstacle transformations, one can search in an implicitly represented C-space and detect collisions in the Cartesian workspace. See Figure 1 and Figure 2. This strategy enables the planner to cope with on-line provided environments and moving obstacles [Henrich98a].

In the basic approach, the implicit and d-dimensional C-space is subdivided in dimension i into subsections of distance Δq_i. Thus, the C-space is covered by hypercubes of edge lengths Δq_i in dimension i. Each hypercube is represented by a *node* in a d-dimensional grid. As an abstract distance measure between two nodes, we use the number of hops between these nodes. A node is called collision-free if all configurations covered by the corresponding hypercube are collision-free.

For searching in this discrete and implicit representation of the C-space, any best-first search mechanism can be applied. We choose a variation of the well known A*-search algorithm [Hart68, Korf92]. Robot configurations (nodes) still to be processed are stored in OPEN, while already processed nodes are stored in CLOSED. Contrasting to the original A*, here, no reopening of nodes in CLOSED is performed.

The evaluation function $f(n) = (1 - w)\, g(n) + w\, h(n)$ is used, where $g(n)$ is the number of nodes of the path from the start node to node n, and $h(n)$ is the "Airplane" or the "Manhattan" distance in C-space between node n to the goal node. Increasing the weight $w \in [0, 1]$ beyond 0.5 generally decreases the number of investigated nodes while increasing the length of the solution path being generated. To improve the on-line capabilities of the path planner, our search is strongly directed to the goal by setting $w = 0.99$ [Sandmann97].

Collisions are detected by a fast, hierarchical distance computation in the 3D workspace, based on the polyhedral model of the environment and the robot provided by common CAD systems [Henrich92, Henrich97e]. With the help of the "MaxMove Tables", introduced in [Katz96], the Cartesian distances are then transformed into joint angles in order to determine whether the current node collides or not. For obtaining similar joint intervals, thus implicating an efficient distance exploitation, a heuristically selected joint discretisation of $\Delta q = (2^\circ, 2^\circ, 4^\circ, 4^\circ, 6^\circ, 6^\circ)$ is used.

The basic approach has been extended in several ways. By using an optimal discretization method based on a predefined Cartesian value, smaller search spaces with an unchanged solution accuracy can be received [Henrich98e]. By multi-directional search with goal switching, the planning system is enabled to select one of the preferable goal configuration by itself [Henrich98c]. By running a parallel version of the path planner on a workstation cluster with 9 PCs, the planning times can be reduced to only a few seconds [Henrich98a]. Besides validating this approach by simulations, we already have connected the planner to a real robot for executing the calculated paths [Wurll98c]. Finally, a hierarchical search can be applied, which we are

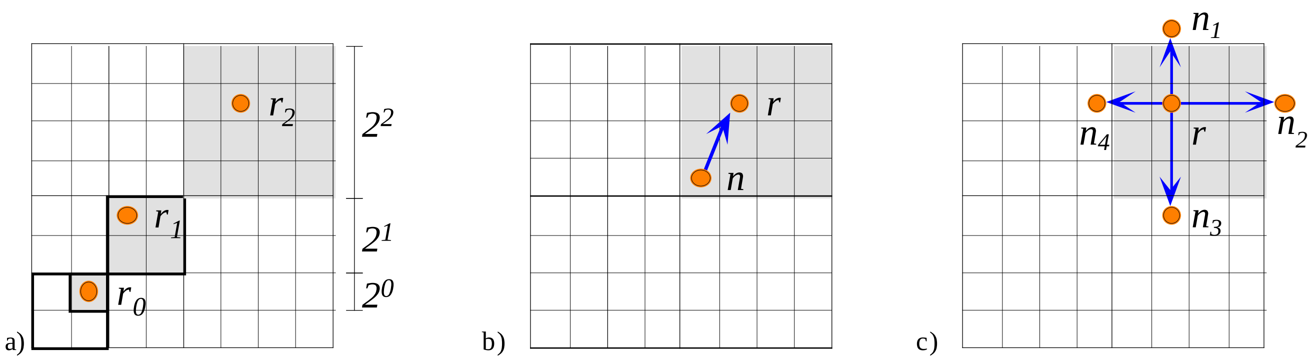


Figure 3: Three additional mechanisms for hierarchical search: a) Introducing representatives r_i for cubes of size 2^s on hierarchy level s, b) Mapping some node n onto its representative r, c) Generating successor nodes n_i of a representative r

investigating here.

III. HIERARCHICAL SEARCH

This basic approach has several advantages. First of all, it is a systematic search and, therefore, complete, i.e., a solution is found if one exists. Additionally, if the reopening of nodes in CLOSED is allowed and if $w = 0.5$, the search is optimal concerning the path length. Finally, no obstacle transformations are necessary. This implicates that this path planning approach can cope with on-line environments or dynamic obstacles.

On the other hand, this approach uses a constant discretisation of C-space. Thus, the stepsize b of the search is fixed and it cannot adapt to wider or narrower environments. This results in huge search spaces and memory overflow for more difficult problems.

Of course, one could increase the stepsize of the search by using a coarser discretisation of the C-space in order to reduce the search space. But this has the drawback that no solution may be found for bottlenecks in C-space or cluttered environments because more configurations are colliding. Therefore, we need a variable stepsize of the search: large steps in free areas of C-space and small steps in the vicinity of obstacles.

A. *Extensions of basic search mechanism*

To introduce a variable stepsize of the search, we use the basic approach of Section II. Due to the distance calculation as collision check, we are able to determine the free space around each processed node. Depending on the free space, multiple neighboring basic hypercubes are merged to bigger hypercubes resulting in larger stepsizes, or the original hypercubes are used resulting in small stepsizes. To realize this idea, we only have to extend the best-first search algorithm by three mechanisms (for details, see [Sandmann97, Wurll97b]):

The first mechanism is a successive (implicit) subdivision of large hypercubes of the C-space by always halving the edge length b in each dimension. Starting with the largest hypercube of edge length $b_{\max}$, smaller hypercubes of edge lengths $b_{\max}/2$, $b_{\max}/4$, $b_{\max}/8$, …, $b = 1$ are successively generated. The subdivision and the location of each hypercube are fixed during the search process. All hypercubes with edge length b are said to be on *hierarchy level* s with $s = \log_2(b)$. Additionally, each hypercube has a representative in its interior, which is a single node with edge length $b = 1$ (see Figure 3a).

The second mechanism maps all nodes of one hypercube onto its representative node. This mechanism depends on the location of the representative within the hypercubes (see Figure 3b). To calculate the representative r_s on hierarchy level s of the configuration (basic node) $q = (q_1, …, q_d)$ in grid coordinates, we use the following function:

$$r_s(q_i) = q_i - q_i \bmod 2^s + (2^s - 1) \operatorname{div} 2,$$

where div and mod stand for integer division and modulo operation, respectively. (In Figure 3, the origins are in the upper left hypercube and the y-axis points downwards.)

The third mechanism is a new node expansion operation, where the successor nodes of the current node lay just outside of the corresponding cube. The mapping (second mechanism) then generates the appropriate hypercube on the desired hierarchy level (see Figure 3c).

B. *Resulting procedure*

To describe the resulting hierarchical search algorithm, it is sufficient to regard one iteration after the start node has been expanded. For the current node n selected from OPEN, the successor nodes n'_i ($i = 1, …, 2d–1$) are generated according to the mechanism in Figure 3c. (Due to the predecessor node of n needs not to be generated, $2d–1$ successor nodes are sufficient). For these nodes, the representative a_i with the maximum hierarchy level is generated according to the mechanism in Figure 3b. If the corresponding cube is collision-free and not yet element of OPEN or CLOSED, it becomes the final successor node for n'_i. Otherwise, the representative b_i of the next smaller hierarchy level is generated and checked. This continues until a collision-free

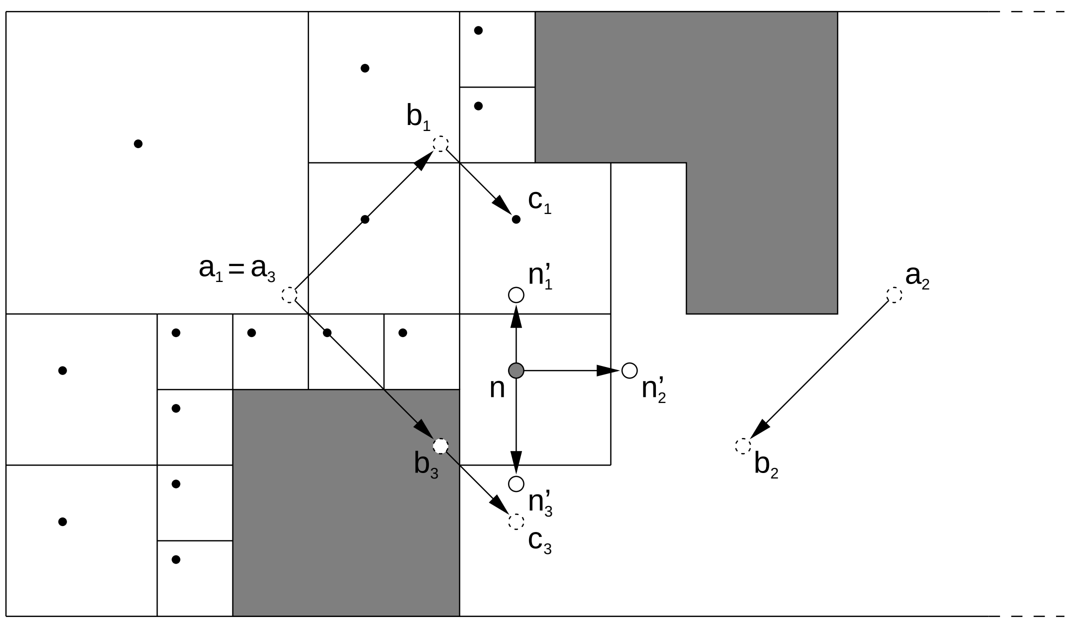


Figure 4: Example for the hierarchical search process. The current node n generates the successor nodes n'_1, n'_2, and n'_3. The successor nodes n'_i, for $i = 1, \ldots, 3$, are mapped successively on representatives of different hierarchy levels a_i, b_i, and c_i.

successor is found or the lowest hierarchy level has been reached.

An example iteration for the 2D case is illustrated in Figure 4. For the current node n, we assume that its predecessor is on the left, thus, only three preliminary successors n'_1, n'_2, and n'_3 have to be generated. For n'_1, subsequently the representatives a_1, b_1, and c_1 are generated. The corresponding hypercubes of a_1 and b_1 are not collision-free. Additionally, c_1 may already be element of OPEN or CLOSED. Thus, the preliminary successor n'_1 has to be disregarded. For the preliminary successors n'_2, and n'_3, the representatives b_2 and c_3 on hierarchy levels 2 and 1 are generated, respectively. These successors are collision-free and not in OPEN or CLOSED and, thus, become the final successors to be inserted in OPEN. The preliminary successors n'_2, and n'_3 are deleted.

To speed up the selection of successor nodes, we first scan upwards and then downwards, always starting with the hierarchy level of the current node n.

C. Modified evaluation function

To take the most advantage of the hierarchical search, we may give preference to nodes in OPEN on higher hierarchy levels. This can be realized by modifying the evaluation function $f(n)$ according to

$$f(n) = [(1 - w)\ g(n) + w\ h(n)] \ / \ (s + 1)$$

where s indicates the hierarchy level of node n. This function assigns less cost to nodes representing larger cubes, resulting in larger stepsizes of the search and, therefore, in smaller number of expanded nodes. Additionally, it keeps the solution path away from obstacles, thus enabling a fast path execution by a real robot.

IV. EXPERIMENTAL RESULTS

For testing the path planner, we have developed five benchmark problems in the 2-dimensional C-space and for the 6 DOF robot Puma260 [Katz96] [3]. The 2 DOF version of the benchmark TRAP is shown in Figure 5. The other four 2 DOF benchmark problems are given in Figure 6. As a first step, here, we present the results for robots with two DOFs.

For a point robot with two DOFs, the workspace and the C-space are identical. There is no need to transform the Cartesian distances into corresponding joint angles movements. Instead, the calculated minimum distance to obstacles directly expresses the maximum collision-free movement of the robot. There is no loss in distance benefit due to distance transformation as in the six DOF case (see Section II).

In Figure 5, the search results of three different search methods for the two DOF benchmark TRAP are given. As $h(n)$ the Manhattan distance in C-space has been used. On the left, the basic approach of Section II with $w = 0.5$ is shown. It expands a huge number of nodes (1216) to find a solution path which touches the obstacles. In the middle, the hierarchical search using the unchanged evaluation function is applied. Here, many fewer nodes (244) are expanded and most of them are close to the obstacle. The solution path is still in the vicinity of the obstacle. On the right, the hierarchical search with the modified evaluation

[3] The data of these benchmark problems can be downloaded from the Web page at http://wwwipr.ira.uka.de/~paro/skalp/

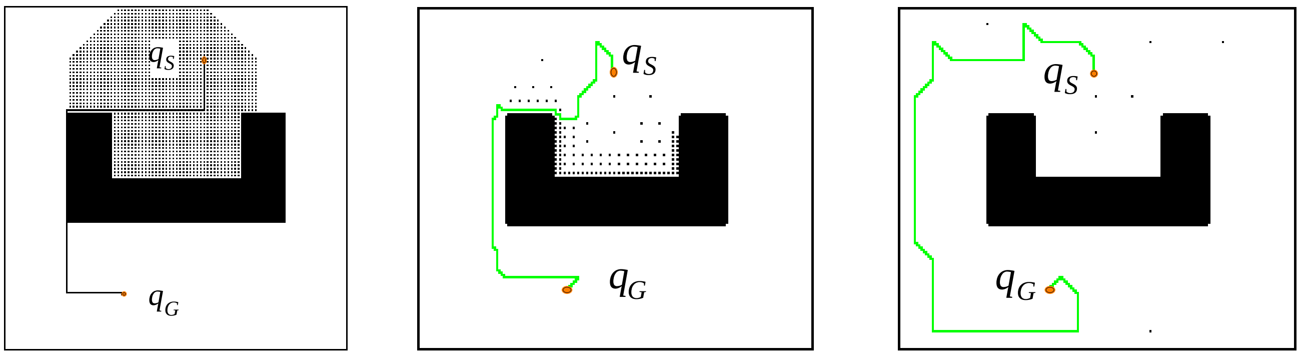


Figure 5: Example for equidistant search (left), hierarchical search (middle), and hierarchical search with weighting of high levels (right). Besides the obstacle (black), the expanded nodes (dots) and the solution path (line) are shown.

function from Section III.C is shown. It results in only very few expanded nodes (52) and none of them nearby the obstacle. Additionally, the path runs almost in the middle of the free space. All three solution paths have smaller or larger jags and need to be smoothed before being executed by a real robot.

The run-time results for the heuristic hierarchical path planner with the modified evaluation function are given in Figure 7. As a run-time measure in the two DOF case, we used the number E of expanded nodes. Additionally, we restricted the maximum hierarchy level used by the search to b_{max}, which indicates the maximum cube size in one dimension. For $b_{max} = 1$, the hierarchical search expands the same nodes as the basic approach of Section II. An increasing b_{max} results in an increasing use of the hierarchy levels for the search. The results show that the number of expanded nodes decreases exponentially with increasing number of used hierarchy levels. Thus, the more hierarchy levels are allowed, the faster the search will be.

V. CONCLUSION AND FUTURE WORK

In this paper, we have extended our approach to path planning for industrial robot arms with six DOF. The basic approach works in an implicit and equidistantly discretisized C-space and collisions are detected in the Cartesian workspace using distance computation. This avoids the time and memory consuming obstacle transformation and C-space calculation. The method is based on the modified A*-search algorithm and needs no essential off-line computation. This approach enables the path planner to work reasonably fast for on-line environments.

The hierarchical search avoids the disadvantages of searching with constant stepsizes by using a stepsize which adapts itself in C-space to the environment. The solution paths are found quickly and stay away from the obstacles. For the two DOF benchmark problems, an increasing number of used hierarchy levels leads to an exponentially decreasing number of expanded nodes. This results in a run-time reduction of two magnitudes.

Currently, we are extending the proposed heuristic hierarchical search in several ways. First, a complete search algorithm based on [Beeh97] is being developed, since the heuristic search may not find all solution paths. Second, the approach is applied to the six DOF case. For this, the collision detection is improved such that more nodes and, therefore, bigger cubes on higher hierarchy levels can be used during the search [Osterroht98]. Finally, the solution path has to be smoothed before it can be executed by a real robot [Bordon98].

ACKNOWLEDGMENT

This work is part of the project "Scalable algorithms for parallel path planning in dynamic environments" funded by the German Basic Research Framework (DFG-Schwerpunktprogramm) "Efficient algorithms for discrete problems and their applications". Thanks to Frank Beeh and Michelle Specht for proofreading.

REFERENCES

[Beeh97] Beeh F.: "Erweiterung des parallelen Bewegungsplaners", Master’s Thesis, Institute for Process Control and Robotics, University of Karlsruhe, 1997.

[Bordon98] Bordon U.: "Parallele Glättung von Robotertrajektorien", Semester Term Thesis, Institute for Process and Robotics, University of Karlsruhe, 1998.

[Fiorini96] Fiorini P., Shiller Z.: "Time optimal trajectory planning in dynamic environments", IEEE Int. Conf. on Robotics and Automation, vol. 2, pp. 1553-1558, 1996.

[Fujimura91] Fujimura K.: "Motion planning in dynamic environments", Berlin, Heidelberg, Springer, 1991.

[Hart68] Hart P. E., Nilsson N. J., Raphael B.: "A formal basis for the heuristic determination of minimum cost paths". In: IEEE Trans. Systems, Science and Cybernetics, Jan. 1968, pp 100-107.

[Henrich92] Henrich D., Cheng X.: "Fast Distance Computa-

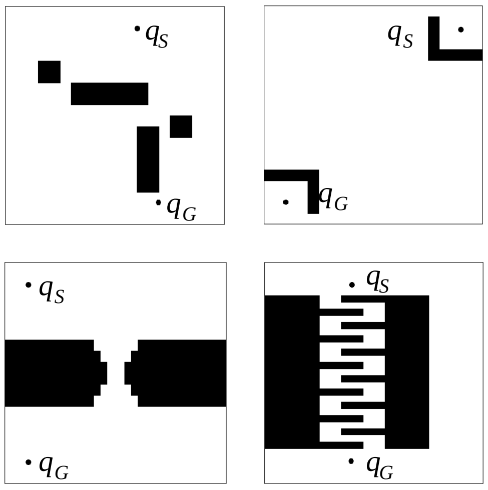


Figure 6: Benchmark problems with start and goal configuration q_S and q_G in a 2 DOF C-space. Top row: SIMPLE, STAR, Bottom row: BOTTLENECK, DOTOUR

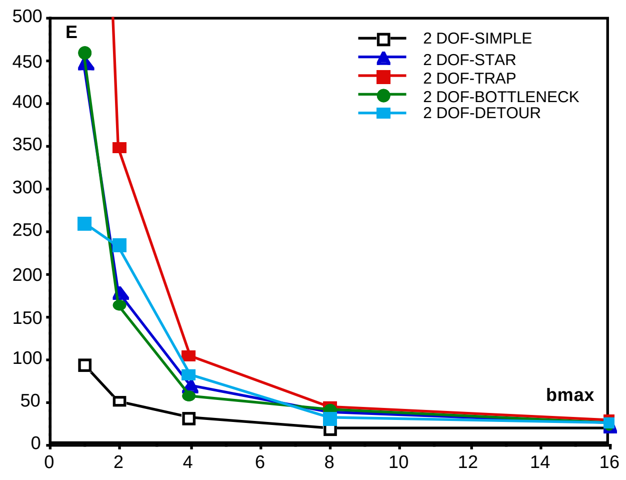


Figure 7: Number E of expanded nodes for hierarchical path search when the maximum cube size is b_{max}

tion for On-line Collision Detection with Multi-Arm Robots", In: Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France, May 10.-15., 1992, pp. 2514-2519.

[Henrich97e] Henrich D., Gontermann S., Wörn H.: "Kollisionserkennung durch parallele Abstandsberechnung". In: 13. Fachgespräch Autonome Mobile Systeme (AMS'97), Stuttgart, 6.-7. Oktober 1997, Springer-Verlag, Reihe "Informatik Aktuell".

[Henrich98a] Henrich D., Wurll Ch., Wörn H.: "6 DOF path planning in dynamic environments – A parallel on-line approach ". In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA'98), Leuven, Belgium, May 16-21, 1998.

[Henrich98c] Henrich D., Wurll Ch., Wörn H.: "Multi-directional search with goal switching for robot path planning ". In: The 11th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems (IEA/AIE'98), Castellan, Spain, July 1-4, 1998.

[Henrich98e] Henrich D., Wurll Ch., Wörn H.: "On-line path planning with optimal C-space discretization". In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98), Victoria, Canada, October 12-16, 1998

[Hwang92] Hwang Y. K., Ahuja N.: "Gross motion planning – A survey", ACM Computing Surveys, vol 24, no 3, Sept. 1992.

[Kamal96] Kamal L., Gupta K., del Pobil A.P.: "Practical motion planning in robotics: Current approaches and future directions", IEEE Robotics & Automation Magazine, Dec. 1996.

[Katz96] Katz G.: "Integration eines parallelen Bewegungsplaners in ROBCAD", Master's Thesis, Institute for Process Control and Robotics, University of Karlsruhe, 1996.

[Korf92] Korf R. E.: "Search". In: Encyclopaedia of Artificial Intelligence", S. C. Shapiro (ed.), vol 2, pp 1460-73, 2nd Edition, John Wiley&Sons, New York, 1992.

[Osterroht98] Osterroht T.: ”Hierarchische parallele Bewegungsplanung für dynamische Umgebungen”, Master's Thesis, Institute for Process Control and Robotics, University of Karlsruhe, University of Karlsruhe, 1998.

[Ralli96] Ralli E., Hirzinger G.: "A global and resolution complete path planner for up to 6 DOF robot manipulators", IEEE Int. Conf. on Robotics and Automation, Minnesota (ICRA'96), 1996.

[Sandmann97] Sandmann S.: "Entwicklung eines parallelen Bewegungsplaner", Master's Thesis, Institute for Process Control and Robotics, University of Karlsruhe, 1997.

[Wörn98] Wörn H., Wurll Ch., Henrich D.: "Automatic off-line programming and motion planning for industrial robots". In: The 29th Int. Symp. on Robotics, (ISR'98), Birmingham, United Kingdom, April 27-30, 1998.

[Wurll97b] Wurll Ch., Henrich D., Wörn H.: "Parallele Bewegungsplanung in dynamischen Umgebungen", Technical Report 20/97, Computer Science Department, University of Karlsruhe, 1997.

[Wurll98c] Wurll Ch., Henrich D., Wörn H., Damm M., Schloen J., Maier W.: "A distributed planning and control system for industrial robots". In: The 5th IEEE International Advanced Motion Control Workshop (AMC'98), Coimbra, Protugal, June 29 – July 1, 1998.