In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'98), Victoria, Canada, Oct. 12-16, 1998

On-line path planning with optimal C-space discretization

Dominik HENRICH, Christian WURLL, and Heinz WÖRN

Institute for Process Control and Robotics (IPR)
Computer Science Department, University of Karlsruhe
P.O. Box 69 80, D-76128 Karlsruhe, Germany
e-mail: [dHenrich, Wurll, Woern]@ira.uka.de, http: //wwwipr.ira.uka.de/~paro/

Abstract

This paper is based on a path planning approach we reported earlier for industrial robot arms with 6 degrees of freedom in an on-line given 3D environment. It has online capabilities by searching in an implicit and descrete configuration space and detecting collisions in the Cartesian workspace by distance computation based on the given CAD model. Here, we present different methods for specifying the C-space discretization. Besides the usual uniform and heuristic discretization, we investigate two versions of an optimal discretization for an user-predefined Cartesian resolution. The different methods are experimentally evaluated. Additionally, we provide a set of 3-dimensional benchmark problems for a fair comparison of path planner. For each benchmark, the run-times of our planner are between only 3 and 100 seconds on a Pentium PC with 133 MHz.

Keywords: industrial robots, path planning, on-line algorithms, search algorithms, discretization

1 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.

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[1] means that the planner does not require any time-consuming off-line computations in order to directly react to dynamic changes in the environment.

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 motion planner for industrial robots based on velocity adaptation is discussed. It plans only for a 2 DOF workspace for two robots and their offline 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. We focus on industrial robots, which constitute a considerable fraction of all robots used currently and in future.

The remainder of the paper is organized as follows: In Section 2 the basic approach of our on-line path planner is introduced. Section 3 describes and compares different methods for C-space discretization. Section 4 introduces a set of benchmark problems for robots with 6 DOF based on 2 DOF examples. Experimental results are given in Section 5. The paper closes with the conclusion and an outlook to the future research in Section 6.

2 Basic approach

Most of the off-line path planners are based on some explicit representation of the *free C-space*. This representation can either be retrieved by transforming the obstacle into the C-space and approximating the free-space or by randomly sampling the C-space and interconnecting the samples by collision-free links. Both approaches are very time consuming and not suited for on-line calculations, especially, if a full geometric CAD model for the robot and the obstacles is used. In order to avoid these time consuming calculations, one can search in an implicitly represented C-space and detect collisions in the workspace. This strategy enables the planner to cope with on-line provided environments. See Figure 1 and 2.

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

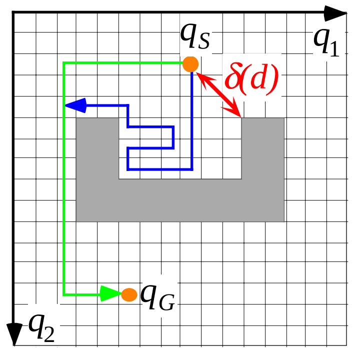

Figure 1: A 2D illustration of the path search in the implicit C-space from the start q_S to the goal q_G using a transformation $\delta(d)$ of the obstacle distance d for collision detection.

For searching in the implicit C-space, any best-first search mechanism can be applied. We choose a variation of the well known A*-search algorithm [Hart68]. 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. As 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 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 cost of the solutions 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 configuration collides or not.

3 C-space discretization

As mentioned previously, the path planning takes place in a discretized configuration space of the robot manipulator. The resolution settlement of discretization is also an important issue. There is a trade-off in the granularity of discretization or resolution: too fine will increase the search space and too coarse may result in failing to find a path even if there exists one.

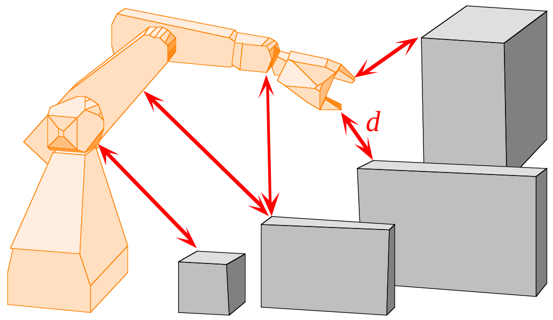

Figure 2: Collision detection in the explicit workspace by computing the minimum distance d between robot and obstacles

Formally, for the i-th coordinate q_i of the C-space, let N_i be the number of intervals along q_i. Then we can determine N_i by[2]

$$N_i = \left\lfloor \frac{q_i^{\max} - q_i^{\min}}{\Delta q_i} \right\rfloor,$$

where q_i^{max} and q_i^{min} are the limits of joint motions and Δq_i is the resolution of joint i. The question is now how to determine the Δq_i's.

3.1 Discretization methods

We now investigat different methods to determine the discretization resolution $\Delta q = (\Delta q_1, \ldots, \Delta q_D)$ of a D-dimensional C-space. In the most simple method, the user specifies a *uniform discretization* for all joints of the robot manipulator, thus, $\Delta q_i = c$ for some constant c. With a reasonable joint resolution of one degree, the uniform discretization result in huge C-spaces. For example, a discretization of the Puma260's joints with $\Delta q = (1°, 1°, 1°, 1°, 1°, 1°)$ results in a C-space with $2.13 {*} 10^{15}$ states.

To avoid the huge search space of uniform discretization, usually a *heuristic discretization* is applied. Here, reasonable Δq_i are estimated by the user to balance the resulting Cartesian movement Δx_i when the different joints i are moved for Δq_i. The underlying problem is illustrated in Figure 3a. For the Puma260, one may choose $\Delta q = (1°, 2°, 3°, 4°, 5°, 6°)$. In this way, generally, the nearer a joint is to the base the finer the discretization resolution is for the corresponding joint angle.

Instead of having a uniform or a heuristic resolution along each configuration coordinate, an *optimal discretization* can be calculated. Therefore, the resolution along each coordinate is set according to the maximum movement of the robot endeffect at each step the robot moves along this coordinate. The result of this discretization is illustrated in Figure 3b. Analytically, this can be achieved by setting

[2] Here, $\lfloor x \rfloor$ denotes the next smaller integer of x.

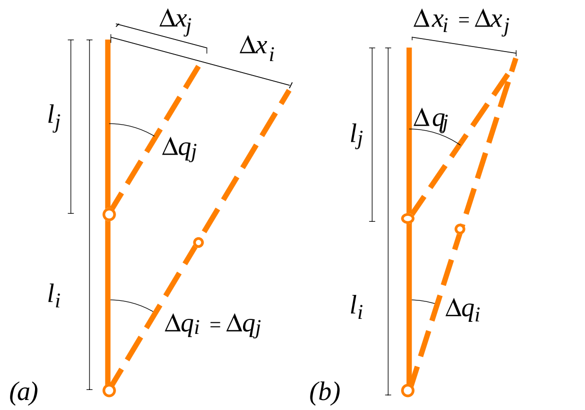

Figure 3: (a) The uniform discretization ($\Delta q_i = \Delta q_j$) results in different Cartesian movements $\Delta x_i \neq \Delta x_j$ when different joints i, j are moved. (b) The optimal discretization results in equal maximum Cartesian movement $\Delta x_i = \Delta x_j$ when different joints i, j with distance l_i, l_j to the endeffect are moved. [Beeh97]

$$\Delta q_i = 2\arcsin\left(\frac{MaxMove}{2l_i}\right),$$

where l_i is the distance between the center of joint i to the farthest point the endeffect can reach, and *MaxMove* is a pre-set distance the robot may move at one step along the coordinate [Qin96b].[3] Altogether, the optimal discretization results in Cartesian movements Δx_i of joint i which meets the condition $\Delta x_{max} \leq MaxMove$, where $\Delta x_{max} = \max\{\Delta x_i, \forall i\}$. For *MaxMove* = 10 mm of a Puma260, the optimal discretization equals to Δq = (0.96°, 0.98°, 1.40°, 2.83°, 2.84°, 10.33°). The size of the corresponding C-space is $1.88*10^{13}$ states. This is two magnitudes less than for the uniform discretization ($\Delta q_i = 0.96^4°$) applying the same maximum movement.

Additionally to the optimal discretization with the strict condition $\Delta x_{max} \leq MaxMove$, this condition can be released to Δx_{mean} (*MeanMove*, where $\Delta x_{mean} = \text{mean}\{\Delta x_i, \forall i\}$.[5] This is interesting in applications where it is not necessary to meet a given upper bound for the Cartesian movement of the robot. Instead, it may be sufficient that the robot's Cartesian movement equals to the pre-defined value *MeanMove* in average.

The *ideal discretization* would result in Δx_i = *MaxMove*, for all i. Unfortunately, the Δx_i depend on the current configuration. Thus, a variable resolution along each coordinate is necessary. This cannot be achieved for an implicit representation of the C-space as needed here for on-line path planning. In the case where an explicit representation of the C-space is applicable, a configuration dependent resolution of C-space can be calculated. For example, a neural network representation of the C-space can be adapted in a training phase to achieve a good C-space discretization [Ralli96].

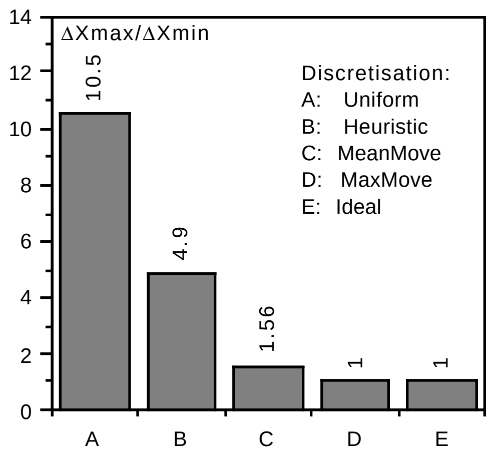

Figure 4: The ratio of maximum to minimum Cartesian movement ($\Delta x_{max}/\Delta x_{min}$) of a robot in the worst case configuration when different joints i are moved for Δq_i. The different discretization methods are indicated by A *through* E.

3.2 Experimental comparisons

To give a better insight to the effects of the different discretization methods, experimental comparisons are presented in this subsection.

The most important property of C-space discretization for path planning is a small number of configuration states while meeting a given (Cartesian) accuracy of robot's motion. This can be achieved by balancing the resolution of the joint discretization such that the resulting Cartesian movements Δx_i are similar for the different joints i. As indicator for this balance, we use the ratio $\Delta x_{max}/\Delta x_{min}$, where $\Delta x_{\min} = \min\{\Delta x_i, \forall i\}$ and $\Delta x_{\max}$ is defined as above. The ratios for the different discretization methods are shown in Figure 4. The ratios differ about a factor of ten between uniform and optimal discretization.

The resulting number of states in the discrete C-space for different resolutions are given in Figure 5. There is only a small difference between the MaxMove and MeanMove discretization. Huge differences in C-space size occure between the optimal and the other discretization methods. One can clearly improve the resolution when changing from uniform to heuristic to optimal discretisation without changing the C-space size.

[3] This kind of geometric reasoning was used by [Lozano87] to build up the approximation of the configuration space by calculating the maximum movement of links. Similarly, [Beer92] used this idea to provide analytical formulae for a specific robot model to reason on the occupancy of C-obstacles.

[4]

[5] Please note, that the Δx_i depent on the current configuration of the robot. We have omitted this in the formalism for a better clarity. Anyhow, in the most formulae, the worst case is assumed, which restricts the robot to some straight configuration.

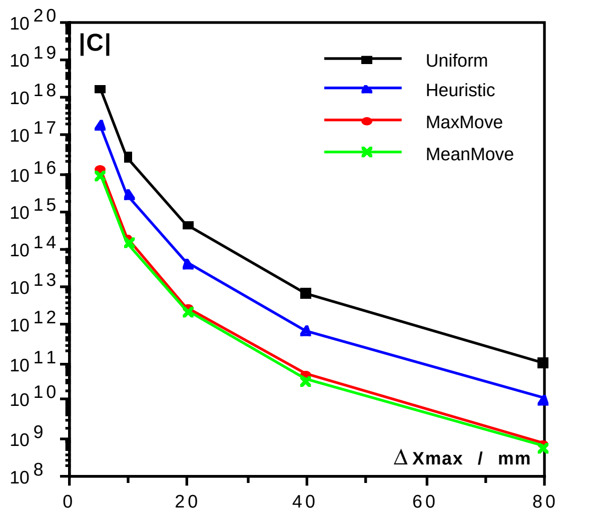

Figure 5: Size |C| of the discretized C-space for different discretization methods and different maximum Cartesian movements Δx_{max} per motion step

The above discussed movement ratios and C-space sizes assume the worst case. For industrial robots, the worst case holds true when the endeffect has the greatest distance to the currently regarded joint, thus, the manipulator is in some straight configuration. For most applications of path planning, the robot is far from being in this worst case configuration all the time. There are many other type of configurations adopted in the average case. Thus, it is interesting how well the discretization methods work in the average case.

For both optimal discretization approaches (*MaxMove* and *MeanMove*), a histogram of the configuration dependent ratios is given in Figure 6 and Figure 7. Therefore, all configurations of a Puma260 which result in different Cartesian movements per joint are regarded. For these configurations, each joint i is moved by Δq_i and the resulting Cartesian movement Δx_i calculated. The Δx_i can be determined by the maximum distance between two corresponding points of the robot in the two configurations.[6] Then the ratio $\Delta x_{max}/\Delta x_{min}$ is clustered into intervals of size 0.1 for rendering.

We can observe that by using the optimal discretization with condition $\Delta x_{mean} \approx$ *MeanMove*, many more ratios are nearby one. Thus more of the robot movement are similar to the pre-defined *MeanMove*. Of cause, there will be movements greater than *MeanMove* and neither a maximum number of such moves nor an upper bound for this moves is guaranteed. The ideal discretization achieves per definition the best ratio of one for all configurations.

[6] Actually, we approximate this calculation by a pre-computed „MaxMove Table" to look up the Δx_i's only for the first three joints [Katz96].

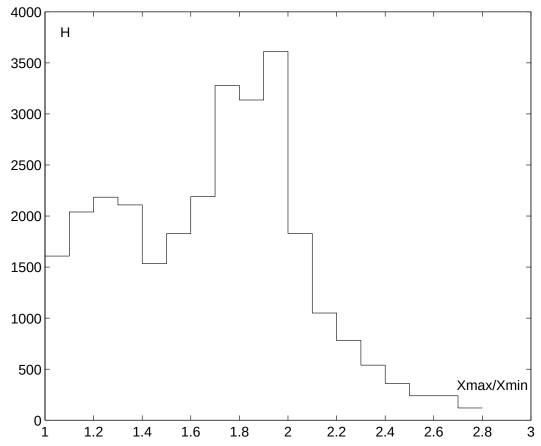

Figure 6: Histogram H of ratios of maximum to minimum Cartesian robot's movement ($\Delta x_{max}/\Delta x_{min}$) per joint step for optimal discretization $\Delta x_{max} \leq$ MaxMove

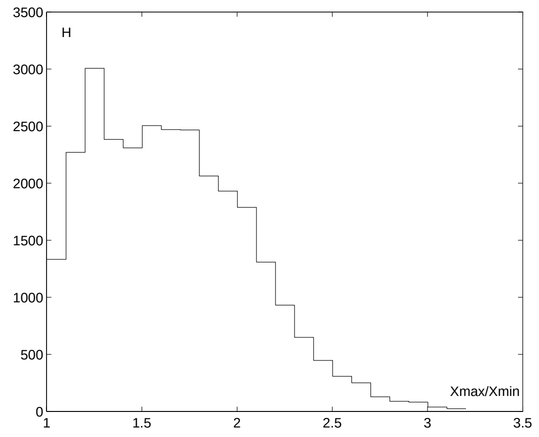

Figure 7: Histogram H of ratios of maximum to minimum Cartesian robot's movement ($\Delta x_{max}/\Delta x_{min}$) per joint step for optimal discretization with $\Delta x_{mean} \approx$ MeanMove

4 Benchmark problems

As a basis for an objective evaluation of the path planner, a set of test environments with corresponding problem specification (*benchmark problem*) is used. Here, the problem emerges that the planner may use different robots. Since the robots differ in their construction (e.g. geometry and kinematic), one cannot compare a problem specification for a robot *A* in a test environment with the same problem specification for a robot *B* in the same test environment. Therefore, the test environments are not specified in the workspace but schematically in a 2D configuration space with increasing level of difficulty.

The levels of difficulty SIMPLE, STAR, and BOTTLENECK were presented in [Hwang96]. A new level of difficulty DETOUR is introduced including a

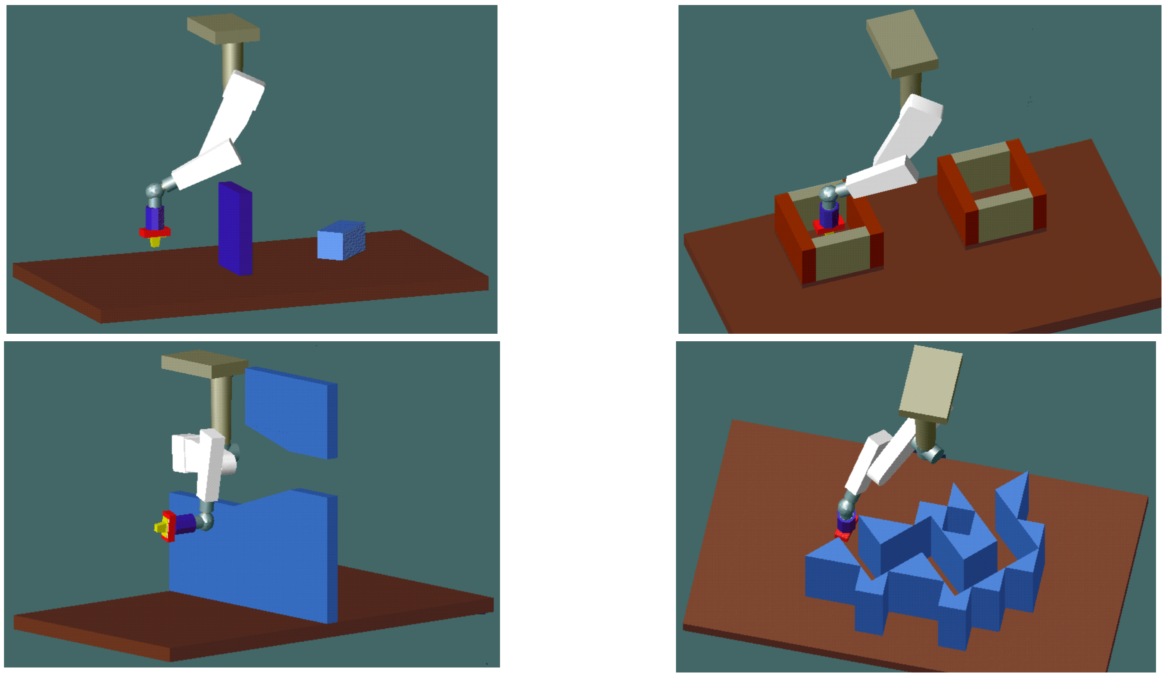
Figure 8: The 3-dimensional test environments for a Puma260 with 6 DOF. Top row: SIMPLE and STAR, bottom row: BOTTLENECK and DETOUR

shorter path nearby obstacles and a longer path staying away from obstacles. This enables us to investigate the path planner's ability to find a reasonable trade-off between fast[7] and short paths.

Based on these schemes, corresponding test environments together with their problem specification have to be prepared for each type of robot. In Figure 8, examples for a Puma260 in the robot simulation tool ROBCAD are shown.[8] More details about the generation of test environments based on the presented levels of difficulty can be found in [Katz96].

5 Run-time results

We have implemented the path planner on a PC workstation with a 133 MHz Intel Pentium processor and 64 Mbytes of main memory. To compare the run-times, we have run every benchmark problem 12 times, deleted the lowest and highest planning times, and computed the average of the remaining 10 values. A optimal discretization with *MaxMove* = 20 mm is used, which leads to the discretization Δq = [1.91°, 1.96°, 2.79°, 5.66°, 5.66°, 20.66°] for a Puma260. According to the upper and lower joint limits of the Puma260, the C-space consists of $2.99*10^{11}$ states.

[7] „Fast“ in the sense of robot execution time

[8] These benchmark problems can be downloaded from the Web page at http://wwwipr.ira.uka.de/~paro/gkatz/benchmark.html

The planning times for the benchmark problems are presented in Figure 9. Three of four planning times were below 30 secs. Only the benchmark problem DETOUR needs little more planning time [Sandmann97].

6 Conclusion and future work

In this paper, we have introduced a new approach to path planning for industrial robot arms with 6 DOF. The algorithm works in an implicit and discretized C-space and the collision detection is done in the Cartesian workspace by distance computation. This avoids the time- and memory consuming obstacle transformation and C-space calculation. The method is based on the A*-search algorithm and needs no essential off-line computation. This approach enables the path planner to work reasonably fast in dynamic environments.

By using the optimal discretization method, the C-space resolution can be determined automatically based on one pre-defined Cartesian value. This leads to smaller search spaces with a unchanged solution accuracy. Additionally, the optimal discretization leads to a better exploitation of the available free-space, when distance computations are used for collision detection.

The method of optimal discretization can be improved by using a more accurate estimation of the Cartesian movement. The linear approximation is sufficient for small movements as they occur in our application. For larger movements per step, an error correction according to [Baginski97] may be appropriate.

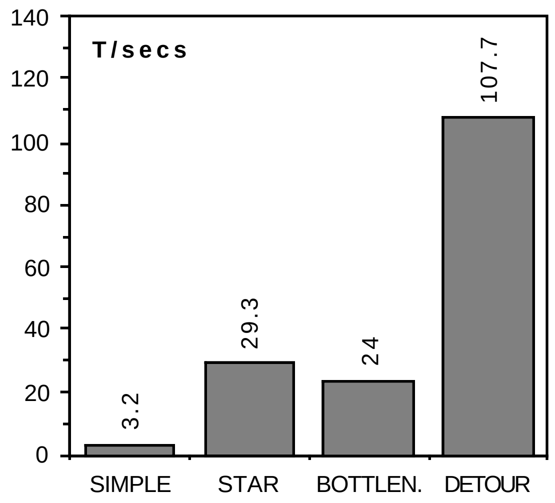

Figure 9: Average run-time T in seconds for different benchmark problems provided on-line to the path planner

Based on these results, we focus next on developing a path planner which is able to cope with moving objects, such as gripped workpieces. To further increase the speed of the algorithm, we are currently working on a hierarchical on-line discretization of the C-space, further reducing the enormous size of the search space. The planning strategy can also be enhanced by a multi-directional or parallel search. Additionally, for executing the computed trajectories with a real robot, we are developing a path smoothing method.

Acknowledgments

This work is part of the project "Scalable algorithms for parallel motion planning in dynamic environments" funded by the German Basic Research Framework (DFG-Schwerpunktprogramm) "Efficient algorithms for discrete problems and their applications".

References

[Baginski97] Baginski B.: "Efficient Dynamic Collision Detection using Expanded Geometry Models". In: Proc. of IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems IROS'97, Grenoble, September 7-11, 1997, pp 1714-1719.

[Beeh97] Beeh F.: "Erweiterung des parallelen Bewegungsplaners", Master's Thesis, Computer Science Department, University of Karlsruhe, 1997.

[Beer92] Beer E.: "JAM - Just another algorithm to solve a motion planning problem: Ein geometrisches Bahnplanungsverfahren fuer Zweiarmroboter", Ph.D. Thesis, University of Karlsruhe, 1992.

[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 minmum cost paths", IEEE Trans. Syst. Sci. Cybern, pp. 100-107, 1968

[Henrich92] Henrich D., Cheng X., "Fast Distance Computation for On-line Collision Detection with Multi-Arm Robots", IEEE Int. Conf. on Robotics and Automation, Nice, France, May 10.-15., pp. 2514-2519, 1992.

[Henrich97e] Henrich D., Gontermann S., Wörn H.: "Schnelle Kollisionserkennung durch parallele Abstandsberechnung". In: 13. Fachgespräch Autonome Mobile Systeme (AMS'97), Stuttgart, October 6-7, 1997, Springer-Verlag, Reihe "Informatik Aktuell", pp 131 - 142.

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

[Hwang96] Hwang K. H..: "Completeness vs. Efficiency in Real Applications of Motion Planning", IEEE Workshop on Practical Motion Planning in Robotics, Minneapolis, MN, April 1996.

[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.: "Konzeption einer Entwicklungsumgebung unter ROBCAD für die parallele Bewegungsplanung", Master's Thesis, Computer Science Department, University of Karlsruhe, 1996.

[Lozano87] Lozano-Perez T.: "A simple motion-planning algorithm for general robot manipulators", IEEE Jour. of Robotics and Automation, vol RA-3, no 3, June 1987.

[Qin96b] Qin C., Henrich D.: "Path planning for industrial robot arms - A parallel randomized approach", In Proc. of the Int. Symp. on Intelligent Robotic Systems (SIRS´96), Lissabon, Portugal, pp. 65-72, July 22-26, 1996.

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

[Sandmann97] Sandmann St.: "Entwicklung eines parallelen Bewegungsplaners", Master's Thesis, Computer Science Department, University of Karlsruhe, 1997.