Accepted at: The Third ASCE Specialty Conference on Robotics for Challenging Environments, Robotics´98, Albuquerque, New Mexico, April 26-30, 1998.

Parallel on-line motion planning for industrial robots

C. Wurll, D. Henrich and H. Wörn[1]

This paper presents a new approach to parallel motion planning for industrial robot arms with six degrees of freedom in an on-line given 3D environment. The method is based on the A-search algorithm and needs no essential off-line computations. The algorithm works in an implicitly descrete configuration space. Collisions are detected in the cartesian workspace by hierarchical distance computation based on the given CAD model. By decomposing the 6D configuration space into hypercubes and cyclically mapping them onto multiple processing units, a good load distribution can be achieved. We have implemented the parallel motion planner on a workstation cluster with 9 PCs and tested the planner for several benchmark environments. With optimal discretisation, the new approach usually shows linear, and sometimes even superlinear speedups. In on-line provided environments with static obstacles, the parallel planning times are only a few seconds.*

Keywords: motion planning, parallel processing, search alogorithms

1. Introduction

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

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

For dynamic environments, three different cases can be distinguished. In the first case, the environment contains dynamic obstacles (i.e. objects on a conveyor belt, or additional robots) with known or partially known movements. In the second case, the robot grips different objects. There, the kinematic chain of the robot, including the gripped object, will change. The third case occurs in the area of virtual engineering.

---

1 Institute for Process Control and Robotics, Prof. Dr.-Ing. H. Wörn, University of Karlsruhe, Faculty for Informatics, P.O. Box 69 80, D-76128 Karlsruhe, Germany, E-Mail: [wurll, dHenrich, woern@ira.uka.de]

After every assembly operation, the product, or the environment will change its geometry. All three cases of dynamic environments implicate a modification of the configuration space (C-space), which has to be considered during planning motions.

An extensive introduction to this problem is presented by Fujimura. In several examples he explains the different approaches used for the motion planning problem of autonomous, mobile robots [Fujimura91]. Fiorini discusses a motion planner for industrial robots based on velocity adaption, but he plans only with a 2 DOF workspace with two robots and known movements, but without any other obstacles [Fiorini96]. Ralli proposes a potential-field approach based on the explicit calculation of the workspace and C-space. If a new object appears, the new path is searched in a few seconds, but the planner works for 5 DOF in a very small search space[2], which is unfavourable for industrial robots [Ralli96].

Generally speaking, speeding up the computation will enable the motion planner to cope better with dynamic environments in practice. One approach is based on the introduction of parallel processing. Mazer reports good planning times based on parallel genetic algorithms for 6 DOF robots in simple problems, but the planner will fail in industrial environments [Mazer93]. Challou presents a parallel formulation of the informed randomized search and achieves good results. But the neccessary pre-computed heuristics are unfavourable for dynamic environments [Challou95]. An extensive overview and a classification of the different parallel methods can be found in [Henrich97a]. The facit of the overview is that parallel processing is an efficient method for speeding up motion planning.

Summarizing, up to now, no planners for 6 DOF robots exist, which have the ability of dealing with dynamic obstacles and having low on-line computation times. Our motivation has been the development of a planner for industrial robots satisfying these requirements. We focus on industrial robots, which constitutes a considerable part of robots being used.

The remainder of the paper is organized as follows: In Section 2 the basic approach of our motion planner is introduced. Section 3 describes the necessary enhancements for parallelizing the sequential approach. Section 4 shows the experimental results, and the paper ends with the conclusion and an outlook of the future investigation in Section 5.

2. Sequential Approach

Most of the off-line motion 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 workspace.[3] This strategy enables the planner to cope with on-line provided environments and moving obstacles.

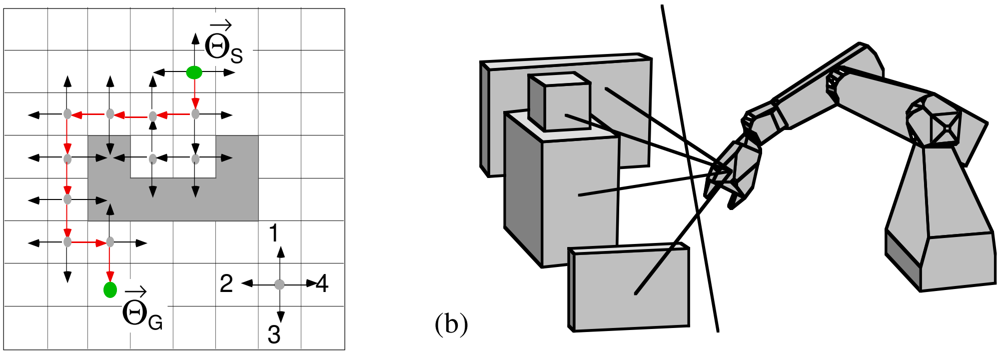
For searching in the implicit C-space, we apply the well known A*-search algorithm [Hart86]. The main task of the A*-algorithm consists of the expansion and the processing of configurations, which are saved in the priority list OPEN. In every iteration, the best configuration of OPEN is expanded.

According to a heuristic evaluation function, these successors will be considerd in the following iterations. After the expansion, the parent configuration is saved in the hashing table CLOSED. The search continues until the goal is found, or the OPEN list is empty. In the latter case the algorithm stops with no solution.

---

2 Taking the reported space of $1.252*10^6$ states, the workspace for a Puma260 with 6 DOF would have a discretisation of about 10 *cm*.

3 Verwer is one of the first, who has used this strategy [Verwer90]. The reported planning times are acceptable for 2 DOF examples but not for 5 DOF in order to cope with dynamic environments.



(a) (b)

Figure 1: (a) A-search in the implicit C-space from the start configuration $\vec{\Theta}_S$ to the goal configuration $\vec{\Theta}_G$, (b) collision detection by distance computation in the workspace*

In Figure 1a, an example for a 2D search is given. The dots indicate investigated configurations and the arrows give reference to the corresponding successors.

Collisions are detected by a fast, hierarchical distance computation in the 3D workspace, based on the given CAD model of the environment and the robot[4] [Henrich92, Henrich97b] (see Figure 1b). With the help of the "maxmove-tables", introduced in [Katz96], the cartesian distances are then transformed into joint intervals in order to define the state ("free" or "prohibited") of the regarded configuration. For obtaining similar joint intervals, thus implicating an efficient distance exploitation, the optimal joint discretisation is automatically computed based on the method of [Qin96].

3. Parallel approach

For parallelizing the A*-algorithm, the configurations in OPEN and CLOSED must be accessible to all processors, in order to distribute the whole work. These lists can either be managed by one dedicated processor or each processor can have its own local lists. In a message passing system, each access to a global list would lead to an enormous communication effort, thus, the local method was preferred.

3.1 Mapping

The work distribution is the key aspect of parallelization. Therefore, the C-space is decomposed into d-dimensional hypercubes of size b in each dimension. For parallel processing, the hypercubes are cylcically mapped on the p available processors by the following function[5]:

$$f(\Theta) := 1 + \left(\sum_{i=1}^{d} \left\lfloor \frac{\theta_i}{\Delta\theta_i * b} \right\rfloor \right) \bmod p$$

According to the automatically computed discretisation $\Delta\Theta_i$, every configuration $\Theta = [\theta_1, \ldots, \theta_d]$ is mapped uniquely to one hypercube or to one processor. Thus, the OPEN list of each processor contains configurations of the multiple mapped hypercubes.

---

4 At present we are evaluating our approach in a traditional manufactoring environment. Additionally our system is also able to plan motions in environments unkown to the user (e.g. space, construction, underwater work, etc.), as the required environment models can either be established by CAD-systems or by sensor systems.

5 The operator $\lfloor \bullet \rfloor$ denotes the next lower integer number.

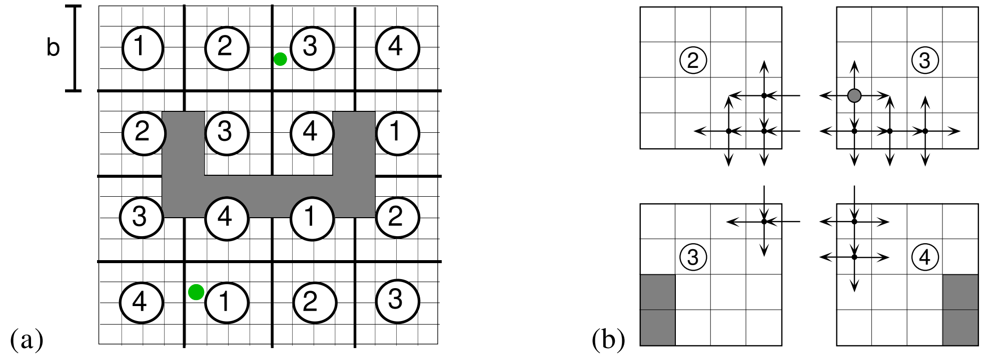


Figure 2: C-space decomposition with the cube size b (a) and the enlarged clipping of four neighboured hypercubes (b)

Figure 2a demonstrates the decomposition of a 2D C-space for $p = 4$ processors and the cube size $b = 4$. The 2D cubes are mapped cyclically onto the four processors according to $f(\Theta)$. Figure 2b shows an enlarged clipping of four neighbouring hypercubes. After the search has reached the hypercube boundaries, the expanded successors are sent to their corresponding processors. The received configurations will then be inserted in a local OPEN list. For the parallel version, in each iteration, every processor expands the best configuration of OPEN until the list is empty or the goal configuration is found. If the OPEN lists of all processors are empty, the algorithm stops with no solution.

3.2. Load Balancing

The performance of the parallel algorithm essentially depends on the load balancing mechanism. In our approach, we have implemented a static distribution mechanism, which can be influenced by modifying the cube size b. Considering the C-space decomposition, small sizes lead to more cubes (in different areas of the C-space) being mapped on to a single processor. Thus, implicating a good load distribution. In contrast, larger sizes will worsen the load balancing. This is mainly due to the coarse decomposition, thus, the processors are longer idle, until they receive work. Additionally, larger cube sizes lead to less cubes beeing mapped on to one processor.

For validating the performance of this load balancing mechanism, we have solved a benchmark problem[6] with different cube sizes b. On each of the $p = 8$ processors, we measured the number of collision detections, which is the most expensive function. Figure 3 shows the experimental results. For larger values, the coarse C-space decomposition leads to an irregular load distribution. In some cases, only a few cubes are covering the complete solution space, thus, some processors becomes idle. Additionaly, due to the sparse mapping, the searching processors expand unneccessary configurations, because they receive no better ones. For smaller cube sizes, the load is nearly equally distributed.

3.3. Message Combining

Lower cube sizes result in a good load distribution, but increase the number of messages. Too many messages, however, usually leads to a bottleneck in the communication network and slows down the calculation times. Combining several messages to form one message seems to be a way out of this problem.

---

6 We used the benchmark problem STAR from Section 4.

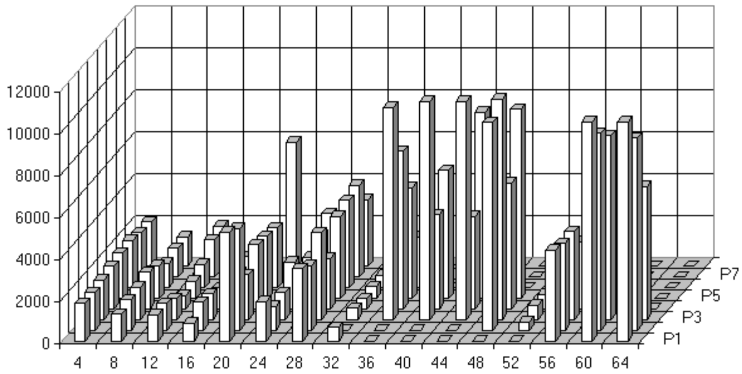


Figure 3: Number of distance computations as a measure for the load distribution depending on the cube size b for the processors P_1 - P_8

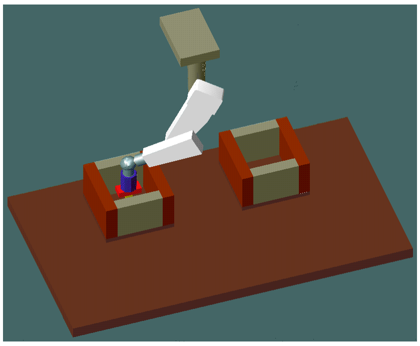
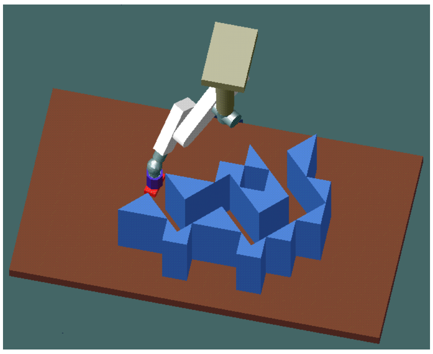
(a)  (b) 

Figure 4: The benchmark problems STAR (a) and DETOUR (b) for the 6 DOF robot Puma260

Evaluation runs for two benchmark problems[7] with a cube size $b = 2$ and $p = 8$ processors have shown that the combining of configurations to one message over several iterations leads to longer planning times. For example, without combining, the benchmark problem STAR was solved in 8 *sec*, but with the combining of messages over 20 iterations the planner has needed 20 *sec*. [Sandmann97].

This is mainly due to the fact, that the remote configurations are sent too late. Thus, every processor does not receive better configurations and expands the worse ones. At this stage, no message combining was included in the further runtime tests.

4. Experimental Results

We have implemented the parallel motion planner on a workstation cluster. The cluster consists of 9 PC's, each with 133 Mhz Intel Pentium processors and 64 Mbyte memory. The parallel communication is established by an Ethernet based bus network. For more details see [Wurll97a].

For testing the motion planner, we have developed five benchmark problems for a 6 DOF robot, a Puma260. As an example, the benchmark problems STAR and DETOUR are shown in Figure 4, developed in [Katz96][8].

---

7 We used the benchmark problem STAR and DETOUR from Section 4.

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

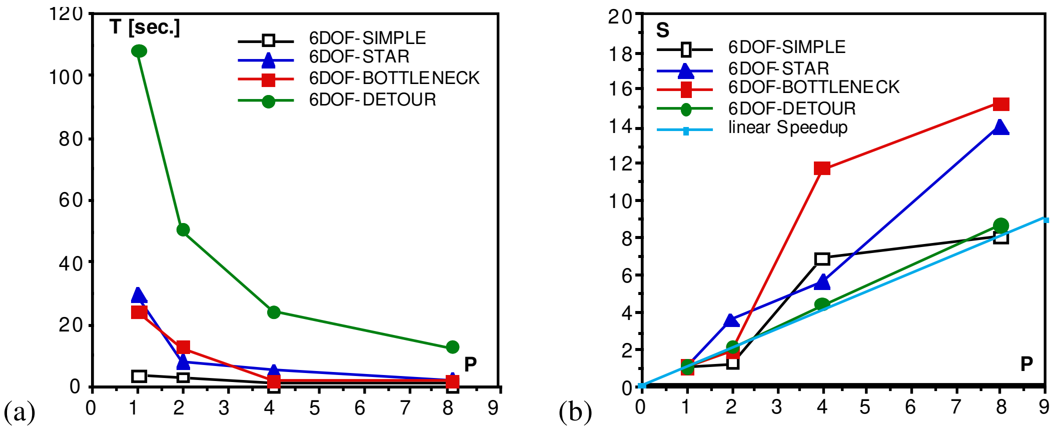


Figure 5: Planning time T (a) and the Speedup S (b) for P processors for the different benchmark problems

We ran every benchmark problem 12 times, deleted the lowest and highest planning times and computed the average of the remaining 10 values. The cartesian resolution was chosen to be 20 *mm*, which leads (for a Puma260) to the discretisation $\Delta\Theta = [1.91°;1.96°;2.79°;5.66°;5.66°;20.66°]$.

According to the upper and lower joint limits of the Puma260, the C-space consists of $2.99*10^{11}$ states, which is at least 3 magnitudes greater, than in the examples of the references.

To calculate the speedups, we have run 8 parallel processes of the parallel algorithm on 1, 2, 4 and 8 processors, thus, guaranteeing the same C-space decomposition. This method of measuring was necessary in order to obtain a fair comparison, because the search performance essentially depends on the C-space decomposition (see Section 3.2).

Figure 5 presents the parallel planning times and the achieved speedups for the benchmark problems. It can be seen that the parallelizing results in a reduction in planning times, and that the speedups are linear, and sometimes even superlinear. Three of four planning times range below 5 seconds. Only the benchmark problem DETOUR needs about 20 seconds planning time [Wurll97b].

5. Conclusion and Future Work

In this paper, we have introduced a new approach to parallel motion planning for industrial robot arms with 6 DOF. The algorithm works in an implicit and discretisized 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 motion planner to work reasonably fast in dynamic environments.

The parallelization with static load balancing results in an equal load distribution and shows linear and sometimes even superlinear speedups. Further acceleration of the motion planner is possible by distributing communication, which can be done using a mesh-based communication network [Wurll97a].

Based on these results, we now focus on developing a motion planner which is able to cope with moving obstacles, such as other robots. With some modification, our approach is also suitable for tasks in the area of virtual engineering. Instead of planning the path for robots, we are able to search a trajectory for the subcomponents, which have to be mapped onto another object .

To further increase the speed of the algorithm, we are currently working on a hierarchical on-line discretisation of the C-space, thus reducing the enormous size of the search space [Wurll97b]. The planning strategy will also be enhanced by a multidirectional search [Beeh97]. Additionally, for running the computed trajectories on a real robot, we are working on a path smoothing algorithm [Bordon97].

Acknowledgement

The project "Scalable algorithms for parallel motion planning in dynamic environments" is funded by the German Basic Research Framework (DFG-Schwerpunktpro–gramm) "Efficient algorithms for discrete problems and their applications". Further information about the project can be found on the Web page of the PaRo group (Parallel robotics) of the IPR at http://wwwipr.ira.uka.de/~paro/.

References

[Beeh97] Beeh F.: „Erweiterung des parallelen Bewegungsplaners“, Diplomarbeit in Bearbeitung, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Bordon97] Bordon U.: „Parallele Glättung von Robotertrajektorien“, Studienarbeit in Bearbeitung, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Challou95] Challou D.J.; Boley D.; Gini M.; Kumar V. : „A parallel formulation of informed randomized search for robot motion planning problems“, IEEE Int. Conf. on Robotics and Automation, Nagoya, Japan, May 21-27, 1995.

[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.

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

[Henrich97a] Henrich D., „Fast motion planning by parallel processing - A review“, Accepted for: Journal of Intelligent and Robotic Systems, 1997.

[Henrich97b] Henrich D., Gontermann St., Wörn H.: „Schnelle Kollisionserkennung durch parallele Abstandsberechnung“ In: 13. Fachgespräch Autonome mobile Systeme (AMS´97), Stuttgart, 6. + 7. Oktober, Springer-Verlag, Reihe „Informatik Aktuell“, 1997.

[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.: „Konzeption einer Entwicklungsumgebung unter ROBCAD für die parallele Bewegungsplanung“, Diplomarbeit, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Mazer93] Mazer E., Ahuactzin J.M., Bessiere P., Chatroux T.: „Parallel motion planning with the Ariadne´s clew algorithm“, ISER-93, 1993.

[Qin96] Qin C., Henrich D.: „Path planning for industrial robot arms - A parallel randomized approach“, In Proc. of the International Symposium 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“, Diplomarbeit, Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe, 1997.

[Verwer90] Verwer B.J.H.: „A muliresolution work space, multiresolution configuration space approach to solve the path planning problem“, IEEE Conf. on Robotics and Automation, 1990.

[Wurll97a] Wurll C., Henrich D.: „Ein Workstation-Cluster für paralleles Rechnen in Robotik-Anwendungen“, In: APS´97, 4. ITG / GI - Fachtagung Arbeitsplatz-Rechensysteme, Koblenz-Landau, 1997.

[Wurll97b] Wurll C., Henrich D., Wörn H.: „Parallele Bewegungsplanung in dynamischer Umgebung“, Interner Bericht der Fakultät für Informatik, Nr. 20/97, Universität Karlsruhe, 1997.