Path Planning for Industrial Robot arms - A Parallel Randomized Approach*

Caigong Qin

Computing Laboratory
University of Oxford
Oxford OX1 3QD, Britain
qin@comlab.ox.ac.uk

Dominik Henrich

IPR, Dept. of Computer Science
University of Karlsruhe
D-76128 Karlsruhe, Germany
dhenrich@ira.uka.de

Abstract. The paper presents a novel approach to parallel motion planning for robot manipulators in 3D workspaces. The approach is based on a randomized parallel search algorithm and focuses on solving the path planning problem for industrial robot arms working in a reasonably cluttered workspace. The path planning system works in the discretized configuration space which needs not to be represented explicitly. The parallel search is conducted by a number of rule-based sequential search processes, which work to find a path connecting the initial configuration to the goal via a number of randomly generated subgoal configurations. Since the planning performs only on-line collision tests with proper proximity information without using pre-computed information, the approach is suitable for planning problems with multirobot or dynamic environments.
The implementation has been carried out on the parallel virtual machine (PVM) of a cluster of SUN4 workstations and SGI machines. The experimental results have shown that the approach works well for a 6-dof robot arm in a reasonably cluttered environment, and that parallel computation increases the efficiency of motion planning significantly.

1 Introduction

The issue of robot motion planning has been studied for more than a few decades and many important contributions to the problem have been made ([10]). One of the most important results is the application of the concept of configuration space. However, it has been shown that the complexity of the generalized movers problem is exponential with respect to the configuration space dimension ([17]) and is PSPACE hard ([16]). Although the configuration space (C-space) approach provides a good framework for theoretical research, motion planning purely based on the approach normally results in a non-practical planner for real-life situations, due to high computation complexity in constructing C-space. In order to avoid the complexity of explicit computation of configuration space (*i.e.* C-free-space and C-obstacles) or its approximation as was done in [13], our method works implicitly in the discretized configuration space with a number of explicit and implicit constraints. The explicit constraints result from the mechanical consideration of the robot, such as, the limitations of joint motions. The implicit ones are derived from collision avoidance between the robot and obstacles. In this way, whether a configuration of the robot is in C-free or C-obstacle space is determined in workspace through collision detections between the robot and the obstacles. In this paper, we will restrict ourselves to considering incomplete (but useful) algorithms working in the discretized configuration space.

To decrease the computation time, some researchers have worked on parallel computations of motion planning ([12]). With parallel processing, not only can some existing sequential algorithms be parallelized but some new parallel algorithms can be designed based on the characteristics of parallelism. We propose a parallel algorithm that would not make much sense for a sequential machine but is fairly effective with parallel processing. The algorithm is implemented using a parallel virtual machine which is a software package that allows a heterogeneous network of parallel and serial computers to work as a single concurrent computational resource. Along with the package, a number of routines are provided in support of user interface. The main advantages of PVM are that it provides a set of user interface primitives that may be incorporated into existing procedural languages and that it is available on most of the network and/or parallel architectures. For further details about PVM, refer to [4].

* The work was under the support of the contract ERBCHRXCT930086 of a Human Capital and Mobility Project of the European Union, and carried out in IPR, Department of Computer Science University of Karlsruhe, Germany.

2 Related Work

Towards autonomous robot systems, motion planning is an important aspect in robotics. It has been attracting a great deal of interest over last 20 years. In the following, we will discuss some relevant work of motion planning.

Glavina [6] proposed an algorithm to solve the "findpath" problem by combining a goal-directed straight-and-slide search and a randomized generation of subgoals. The idea is to conduct the search by following the straight line till the searching point reaches a C-obstacle in the discretized configuration space. Then, the searching configuration point slides along the obstacle boundary only if it reduces the configuration distance which is a function of suitably weighted combination of the configuration variables. The sliding process continues until the point gets stuck at a local minimum with respect to the configuration distance function. Then, a new subgoal is generated randomly. The reachability of the subgoal is tested by the same straight-and-slide searching method from all introduced points (start and goal, and previous subgoals). Eventually, a site graph can be constructed as an abstract representation of the C-free-space. During the process, step-by-step collision tests are carried out in order to detect whether the point is running into a C-obstacle. The algorithm was implemented using a moving polygonal object and the environmental polygonal obstacles in the 2D case.

Our work differs from the previous research in various ways. Rather than working on the moving polygonal object in a 2D case, we consider motion planning for robot manipulators in 3D workspaces. We employ a complete domain-dependent rule base to guide path searching. The number and depth of local minima are reduced through a number of subgoals randomly or purposedly generated in parallel processing. In addition, we utilize some heuristics to reduce the number of collision detections instead of conducting step-by-step collision tests during planning.

Qin [15] presents a solid modelling scheme which is useful for efficient 3D path planners. The scheme makes use of the enhanced version of Gilbert, Johnson and Keerthi's minimum distance algorithm ([5]). This provides an efficient prototype to be tailored to collision detection and to distance computation respectively. We have adopted this scheme in this work. In addition, other heuristics for speeding up collision detection are also investigated by Henrich and Cheng [8], who introduced hierarchical representations of both obstacles and the robot working favourably for complex environments.

Kavraki and Latombe [9] proposed an approach concerning the randomized preprocessing of the configuration space to build up a global network of connected configurations. The idea is to use the *generate-and-test* method to construct a network of randomly but well selected collision free configurations. However, the algorithm requires that each generated configuration be checked to see if it is in C-*free* space, which could be computationally very expensive.

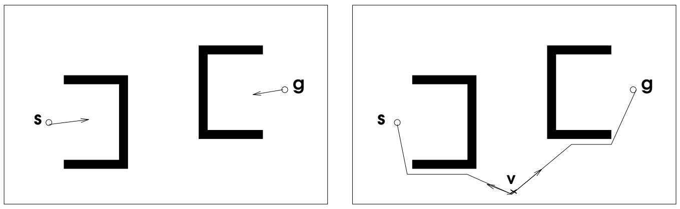
Challou *et al.* [3] presented a parallel motion planner using the parallel formulation of a randomized heuristic search. The algorithm is based on the parallelization of the randomized robot planning method proposed by [1].

Henrich [7] presented an extensive overview of the parallel approaches to robot motion planning. The parallel approaches can be divided into four classes: grid-based, graph-based, potential field, and mathematics programming. The method presented in this paper can be generally classified as grid-based.

3 Outline of the Approach

Our work focuses on developing an effective approach to solving the path planning problem for industrial robot arms operating in a reasonably cluttered environment. This means that the workspace of the robot arm is not maze-like. So it is basically assumed that the path planning problem has a number of acceptable solutions, though the assumption is not compulsory.

The approach is applied in the discretized C-space of the arm. The C-space needs not be represented explicitly with the support of a fast collision test and geometric reasoning. This avoids problems of computational complexity and memory requirement for the transformation between the robot worldspace and the C-space. Given the start and goal configurations, a conventional search (such as depth-first search, best-first search, *etc.*) within the free C-space will readily involve a great deal of backtracking and will therefore require a large amount of computation. The main reason is that the search space is normally very large and has some local minima with respect to some heuristic function of configuration parameters. One of the human-inspired pathfinding strategies is to divide the whole complicated search task into several simpler sub-tasks by setting up proper subgoals between the start and the goal configurations. However, the problem of how to set up subgoals properly is usually not trivial, and it may help by using the generalized Voronoi diagrams (GVD) of the robot's free workspace or other global information.



(1) (2)

Fig. 1.: An illustrative diagram for path search. (1) Searching for a path is costly no matter whether from the start to the goal or reversely. (2) With a via point $\mathbf{v}$, the combination of searching from $\mathbf{v}$ to the start and to the goal respectively is much easier.

Figure 1 shows an example of a path search with dead end C-obstacles. It illustrates that while the direct search from the start to the goal or reversely is not cheap, an indirect search with some via points may be very helpful. This phenomenon, along with the availability of parallel processing, encourages us to come up with a straight *randomized parallel search* algorithm. The general idea of this algorithm is to randomly generate a number of subgoals in the discretized free C-space. Then, parallel searching with each subgoal attempts to find a path connecting the initial configuration with the goal via the subgoal configuration. The purpose of the approach is for the motion planning system to cope with some deep local minima (see Figure 1). In this sense, it is a two-phase search which tries to find a subpath connecting the initial with the randomly generated subgoal and a subpath connecting the subgoal with the final goal. The reasons for not using a three- or multi- phase search are (1) that a path from the initial to the goal via more than two randomly generated subgoals will usually be longer in length and (2) the planner may correspondingly take a longer time than that in a two phase search. But we claim that three or more phase search will be very effective to help robot avoid some very deep local minima if all subgoals are generated under the guidance of some global information.

3.1 Sequential search

For each path search task from one configuration to the other, we have employed an expert system method to perform path searching with the guidance of a set of rules. We use CLIPS [2] as the expert system shell, which is embeded as a module into C++ programs. For generality, the following discussion is for a general n-dof robot arm.

Let $\mathcal{C}^n$ be the n-dimension configuration space. In order to reason and search, we discretize the configuration space $\mathcal{C}^n$ into a rectangloid grid $\mathcal{GC}^n$ (with appropriate modular arithmetic for the angular joint parameters). For each configuration $q = (q_1, q_2, ..., q_n)$ (in $\mathcal{C}^n$) at which the robot is within its workspace, we have

$$q_i^{min} \leq q_i \leq q_i^{max} \ , \qquad i = 1, 2, ..., n. \tag{1}$$

which specifies the limitations of joint motions.

For convenience of reference, we define $\mathcal{S} = \{q : \mathcal{GC}^n | q = (q_1, q_2, ..., q_n) \text{ satisfies } (1)\}$ as the set of configurations at which the robot is within its workspace.

In $\mathcal{GC}^n$, each grid node can be indexed by a point in $\mathcal{Z}^n$, where $\mathcal{Z}$ is the set of integers used in mapping:

$$\mathcal{F} : \mathcal{GC}^n \to \mathcal{Z}^n \ , \ \ i.e. \ \mathcal{F}(q) = (i_1, i_2, ...i_n) \ . \tag{2}$$

where $q \in \mathcal{S}$ and $i_\ell \in \mathcal{Z}$ $(1 \leq \ell \leq n)$. In this sense, $\mathcal{Z}^n$ is regarded as a symbolic abstraction of $\mathcal{GC}^n$. The reasoning system runs directly on $\mathcal{Z}^n$ with other symbolic facts. At the lower level, reasoning is conducted directly on geometric data. The two levels are connected through a number of geometric primitives.

[2] C Language Integrated Production System, developed in NASA.

In our reasoning system, path planning considers only the 1-neighbors of the current configuration as candidates to move to in each step. Then a number of rules are designed to select an optimal candidate according to the following *cost function* $\mathcal{P}$:

$$\mathcal{P}(q) = \begin{cases} +\infty \text{ if } q \text{ does not satisfy constraints (1)} \\ +\infty \text{ if } q \text{ is not in } C\text{-}free \text{ space} \\ f(q) \text{ otherwise} \end{cases}$$

with

$$f(q) = \sum_{i=1}^{n} \omega_i((\mathcal{F}(q_{goal}))_i - (\mathcal{F}(q))_i)^2 \quad (3)$$

where $\omega = (\omega_1, \omega_2, ...\omega_n)$ is a weight vector with $\omega_i > 0$, which is determined heuristically using the model of the robot manipulators.

With the support of some procedural geometrical primitives, a number of rules working in $\mathcal{Z}^n$ have been created to guide the search for a path in $\mathcal{GC}^n$. In addition to rules for initialization, detection of goal reaching and generation of candidates for next movement, there are several other rules devoted to decision-making for the next step. At each step, all the candidates which are previously visited will be removed first; then the candidate with the lowest cost with respect to $\mathcal{P}(q)$ is checked to see if it is out of the robot workspace or in C-obstacles. If it is in C-free space and within the robot workspace, the candidate will be accepted and all others removed; otherwise, it will be removed and the next best candidate will be checked. If the selection procedure results in an empty set of candidates, a rule will force the search to backtrack. For more details on the rule-based search, refer to [14].

3.2 Parallel Search

Given the start and goal configurations, the sequential search algorithm alone may take too much computation time to find a path. To improve the efficiency of the path search, we take advantage of parallel processing to conduct the path search using the randomized parallel search algorithm. In the example shown in Figure 2, a number of processes concurrently conduct a search for a path connecting the start and goal configurations. Each process generates one subgoal randomly in the C-free space and then starts searching for a subpath from the subgoal to the start configuration and for another subpath from the subgoal to the goal configuration. The final path is the proper concatenation of the two subpaths. Whenever a process returns such a final path, all search processes will be terminated. The termination criterion used here is very simple and easy to implement under the PVM.

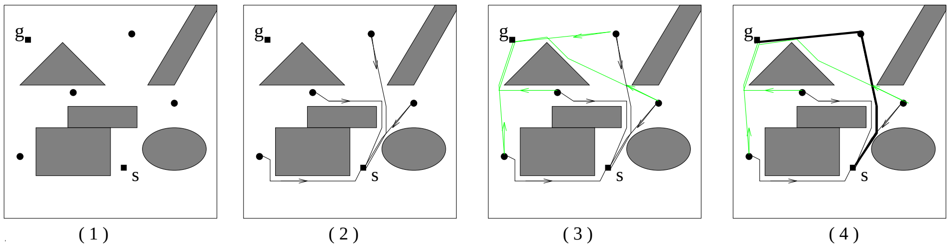


Fig. 2.: An illustrative diagram: (1) Randomly generated subgoals in C-free space; (2) Each process conducts search for a subpath from the subgoal to the start configuration; (3) After step 2, each process starts searching for another subpath from the subgoal to the goal configuration; (4) The first path a process returns will be accepted as the final path.

For each process, search always starts from the subgoal to the start and to the goal configurations respectively, rather than from the start configuration to the subgoal and then from the subgoal to the goal configuration or reversely. It is out of concern that if the start or the goal configuration is in a deep local minimum it can be hard to jump out of such a minimum by starting search from the start or the goal configuration even with help of subgoals. In this case, it may be made easier by searching from a subgoal to the start and the goal configurations respectively. This effect is especially exploited by the parallel algorithm where multiple subgoals are used.

As we know, the final path obtained in this method tends not to be optimal even if each subpath is optimal. But we claim that it is probabilistically optimal provided that the number of random subgoals tends to be infinite.

In some situation without or with sparsely-cluttered obstacles, search for a path from the start to the goal configuration or reversely may be more efficient. To make the method more robust, two extra processes are designated to conduct search directly from the start to the goal configuration and from the goal to the start configuration respectively, while other processes conduct search through subgoals. Therefore, it holds that the time for planning with one process is not less than that with multi-processes.

In all, our approach to parallel processing has some advantages. It needs no heavy load of communications and no load balancing necessary. It is independent of parallel architecture and of low memory space requirements.

4 Heuristics

4.1 Discretization resolution

In a discretised configuration space of the robot arm, the resolution settlement of discretization is also an important issue. There is a trade-off in the granularity of discretisation or resolution: too fine will increase the search space exponentially and too coarse may result in failing to find a path even if there exists one.

We have adopted a heuristic to help set up the discretization resolution of the C-space. Instead of having uniform resolution along each configuration coordinate, we set up the resolution along each coordinate differently estimating the maximum movements of the robot's endeffect at each step the robot moves along the coordinate. The resolution should be so fine that the maximum movement of the robot endeffect is not more than a pre-set distance at each step the robot moves along the coordinate. In this way, generally, the nearer a joint is to the base, the finer the discretization resolution is for the corresponding joint angle.

Formally, for the i-th coordinate q_i of the C-space, let N_i be the number of intervals along q_i. Then, $N_i = \lceil \frac{q_i^{max} - q_i^{min}}{\Delta\theta_i} \rceil$ and $\Delta\theta_i = 2\arcsin(\frac{MaxMove}{2l_i})$, where q_i^{max} and q_i^{min} are the limits of joint motions (see Formula (1)), l_i is the length between the center of the joint to the farthest point the endeffect can reach, and $MaxMove$ is a pre-set distance the robot moves along the coordinate at one step.

4.2 Prediction of Maximum Movement

Geometric reasoning plays an important role in motion planning. [11] used it to approximate the configuration space by calculating the maximum movement of links. Similarly, [2] used the idea to provide analytical formulae for a specific robot model to reason about the occupancy of C-obstacles.

This heuristic is used to improve the rule-based sequential search algorithm by minimizing the number of collision detections. As introduced in §3.1, the rule-based search performs a collision test at least once at each movement step. So, even in the extreme case that there is no obstacle around, the search still must conduct unnecessary collision tests at each step along the line connecting the start and the goal configurations. To cope with the problem, we make use of a heuristic called *prediction of maximum movement*.

The idea of the heuristic is to utilize proximity information in path searching rather than simply conducting collision tests at each step. As we know, the movement of the robot endeffect is not more than the pre-set $MaxMove$ at each step that the robot moves along any one of the coordinate axis directions of the C-space. This also implies that the movement of any other points on the robot arm is not more than $MaxMove$ at each step. Let d be the minimum distance between the robot and obstacles around. Let $FreeSteps$ be $FreeSteps = \lfloor \frac{d}{MaxMove} \rfloor$. Then, we can conclude that the robot arm is ensured to be collision-free for $FreeSteps$ consecutive steps. This helps release the searching from collision tests for $FreeSteps$ steps. Figure 3 shows an example with $FreeSteps = 3$.

5 Experimental Results

The motion planning system is implemented in C++ with the embedded rule-based expert system. Experiments have been carried out on the PVM of a cluster of SGIs and SUN4 workstations. The number of

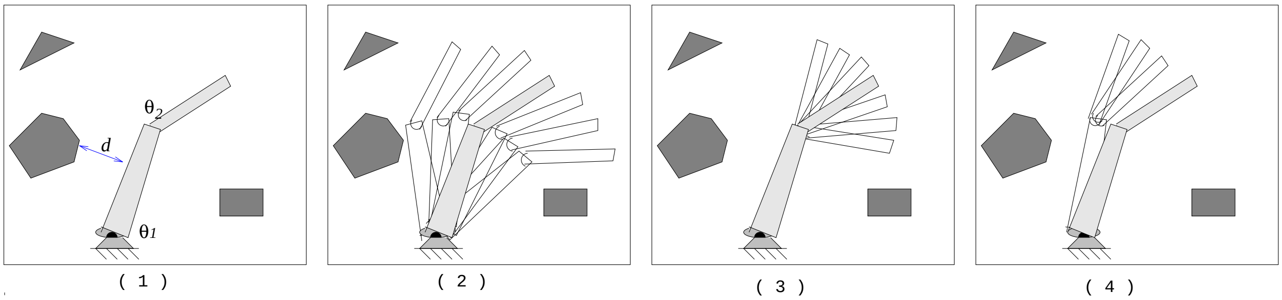


Fig. 3.: An illustration of using the heuristic of prediction of maximum movement with $FreeSteps = 3$. (1) The arm's current configuration. (2) The arm is ensured to be collision-free while moving along θ_1 for 3 consecutive steps in either the positive or negative direction (with θ_2 fixed). (3) Similar as the above with θ_1 and θ_2 exchanged. (4) The arm is collision-free while moving along the coordinate axis (either θ_1 or θ_2) directions of the C-space for 3 consecutive steps.

machines or processors available is limited (up to 45 workstations). In addition, the heterogeneneous machines range from SUN Sparc classic to Sparc20 and from SGI IRIX5.3 R3000 to IRIX5.3 R4000. In this section, we will present some experimental results and show the performance of the planning system with some examples.

Due to the randomness of our parallel algorithm, the time taken to solve one problem may vary more or less from one run to another. All the data presented in this section are calculated by taking the mean value of the corresponding experimental results.

We used a 6-*dof* robot manipulator of PUMA 200 type as the robot model. In our experiments on a SGI machine of IRIX5.3 R4000, it takes 3.835 ms to modify the robot from one configuration to another and to conduct a single collision detection for the case shown in Figure 4. The discretization resolution has been set through the heuristics (see Section 4), so that the joint i of the robot moves $\Delta\theta_i$ at each movement step, where $(\Delta\theta_0, \Delta\theta_1, \Delta\theta_2, \Delta\theta_3, \Delta\theta_4, \Delta\theta_5) = (2.23°, 2.23°, 3.47°, 6.88°, 6.88°, 9.56°)$. The weight vector ω for the cost function $f(q)$ in Formula (3) is set to gives higher priority to the first few joints.

Experiments have also shown that employing the heuristic of prediction of maximum movement improves the performance of the planning system. In the example shown in Figure 4, averagely, the planning takes about 22.69 seconds and conducts 622 collision tests without using the heuristic. Nevertheless, it only takes about 9.64 seconds and conducts 371 collision tests by use of the heuristic on the PVM of a cluster of 10 SUN4 and 4 SGIs.

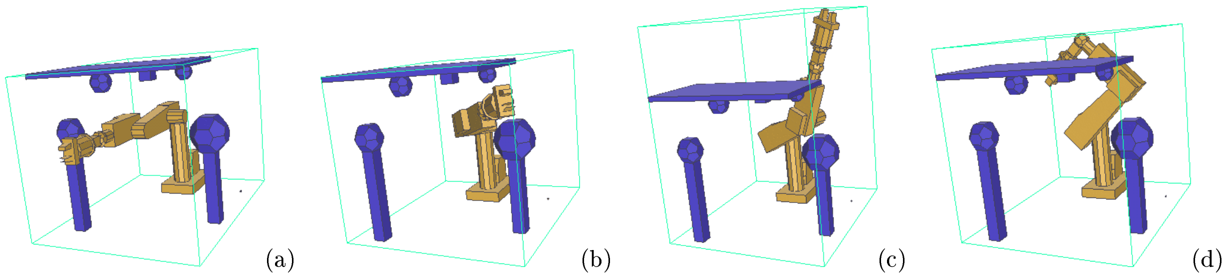


Fig. 4.: Snapshots: (a) The robot is in the initial configuration; (b) and (c) the robot is searching its way towards the goal; (d) the arm reaches the goal.

Another example shown in Figure 5 is to find a path for the robot arm in the workspace occupied with 15 randomly sized and located rectangular obstacles. It takes 32.96 seconds on the PVM of a cluster of available 41 SUN4 and 4 SGI machines. Further experiments show that the efficiency increases with increasing number of processors (see Figure 6 (a)). Figure 6 (b) shows that the more subgoals employed, the better a path length found in the example which is shown in Figure 5.

As we know, *speedup* is defined as the ratio of the time taken to solve a problem on a single processor with the best sequential algorithm to the time needed to solve the same problem on a parallel machine with identical processors. The performance of our algorithm on the example shown in Figure 5 implies the

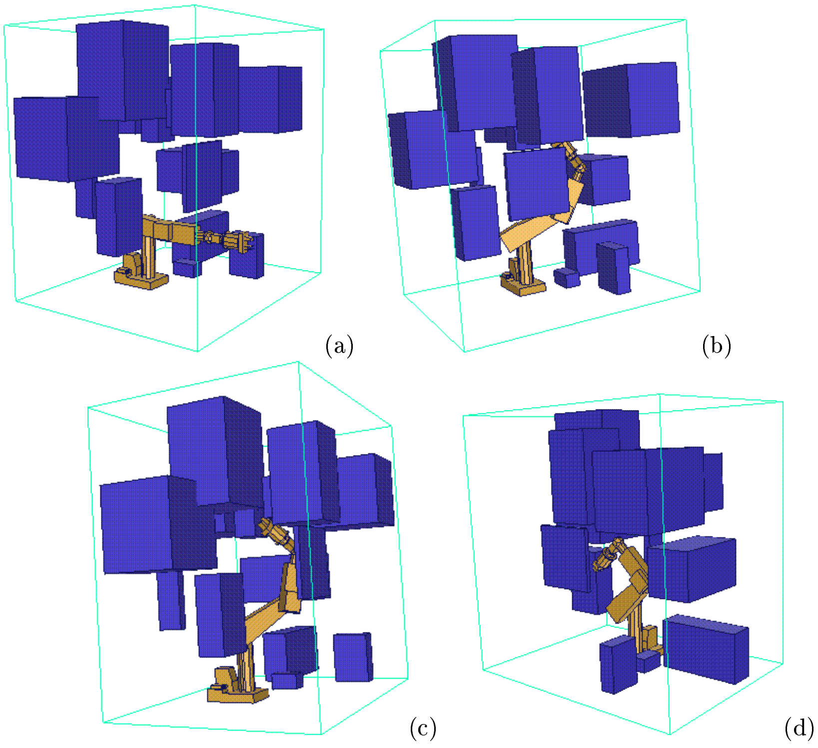


Fig. 5.: The robot arm is in motion. The planning takes 32.96 seconds averagely on PVM of a cluster of 41 SUN4 and 4 SGI machines.

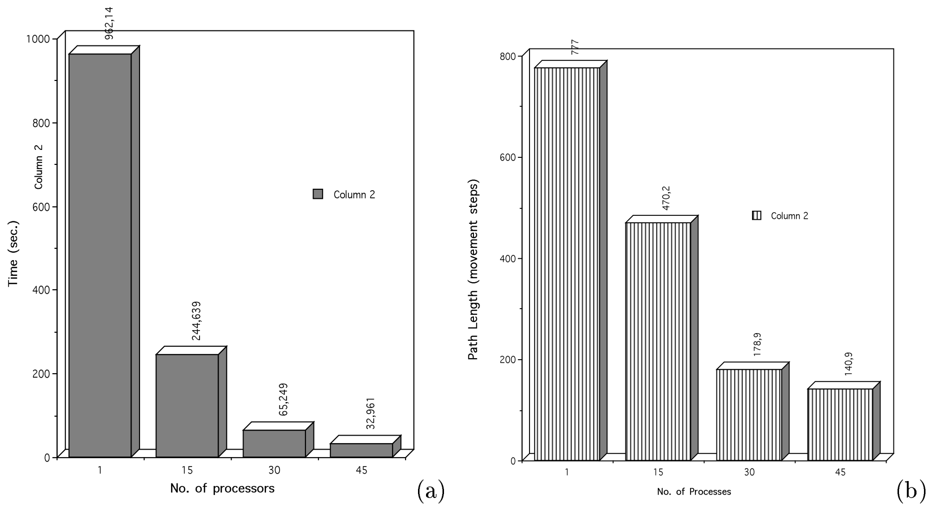


Fig. 6.: (a) The illustrative graph of the performance of the randomized parallel planning algorithm. (b) The more subgoals employed, the better a path length can be found (each process deals with one subgoal).

speedup of 14.75 with 15 processors, though the experiments are not conducted on identical processors and the time taken to find a path on a single processor may not be optimal.

In a further example which is to find a path for the robot arm in the workspace occupied with 20 randomly sized and located rectangular obstacles, planning takes 34.81 seconds on the PVM of a cluster of available 41 SUN4 and 4 SGI machines.

6 Conclusions

This paper presents a novel approach to parallel motion planning for robot manipulators in 3D workspaces. The approach is based on the *randomized parallel search* algorithm and focuses upon solving the path planning problem for industrial robot arms working in a reasonably cluttered workspace. The implementation is carried out on a cluster of SUN4 workstations and SGI machines under the PVM. The experimental results have shown that the approach works well for a 6-dof robot arm in a reasonably cluttered environment, and that computation with multi-processors increases the efficiency of motion planning significantly. However, the method is not recommendable for general redundant robot manipulators, as too large search space may result in high computation time. In addition, the method of the current implementation makes use of randomly generated subgoals for a two-phase parallel search. This may not work well in a very complex workspace, such as a maze-like one. The major problem lies on that no global information is used for subgoal selection. An extension of the work can be to investigate the way of the subgoal selection under the support of global topological information of the environment, for example, using the generalized Voronoi diagrams or information derived from artificial potential fields in the workspace, to improve robustness and efficiency of the method.

Since the planning performs only on-line collision tests with proper proximity information without using pre-computed information, the approach is suitable for planning problems with multirobot or dynamic environments.

References

1. J. Barraquand and J.C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robotics Res.*, 10(6):628–649, 1991.
2. E. Beer. Jam - just another algorithm to solve a motion planning problem: Ein geometrisches bahnplanungsverfahren fuer zweiarmroboter. Dissertation, University of Karlsruhe, 1992.
3. D.J. Challou, M. Gini, V. Kumar, and C. Olson. Very fast motion planning for dexterous robots. In *IEEE Int. Conf. Robotics & Automation*, pages 201–206, May 1995. Nagoya.
4. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V Sunderam. Pvm 3 user's guide and reference manual, May 1993. Electronic publish, Math. Sci. Section, Oak Ridge National Laboratory.
5. E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE J. Robotics & Automation*, 4(2):193–203, April 1988.
6. B. Glavina. Solving findpath by combination of goal-directed and randomized search. In *IEEE Int. Conf. Robotics & Automation*, pages 1718–1723, Cincinnati, Ohio, May 1990.
7. D. Henrich. Parallel processing approaches to motion planning – an overview. In *IEEE Int. Conf. Robotics & Automation*, pages 3289–3294, Minnesota, April 1996.
8. D. Henrich and X. Cheng. Fast distance computation for on-line collision detection with multi-arm robots. In *IEEE Int. Conf. Robotics & Automation*, pages 2514–2519, Nizza, France, May 1992.
9. Lydia Kavraki and Jean-Claude Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE Int. Conf. Robotics & Automation*, pages 2138–45, San Diego, May 1994.
10. Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.
11. T. Lozano-Pérez. A simple motion-planning algorithm for general robot manipulators. *IEEE J. of Robotics and Automation*, RA-3(3), June 1987.
12. T. Lozano-Pérez. Parallel robot motion planning. In *IEEE Int. Conf. Robotics & Automation*, pages 1000–1007, Sacramento, Califonia, April 1991.
13. T. Lozano-Pérez, J. Jones, E. Mazer, and P. O'Donnell. *HANDEY: A Robot Task Planner.* The MIT Press, Cambridge, Massachusetts, 1992.
14. C. Qin and S. Cameron. Motion planning for manipulators using an expert system. To appear, the 12th Int. Conf. on CAD/CAM, Robotics and Factories of the Future, London, August 1996.
15. C. Qin, S. Cameron, and A. McLean. Towards efficient motion planning for manipulators with complex geometry. In *Proc. IEEE International Symposium on Assembly an Task Planning*, pages 207–212, Pittsburgh, USA, August 1995.
16. J.H. Reif. Complexity of the mover's problem and generalizations. In *20th IEEE Symposium on Foundations of Computer Science*, pages 421–427, 1979.
17. J.T. Schwartz and M. Sharir. On the 'piano movers' problem: Ii. general techniques for computing topological properties of real algebraic manifolds. In *Advances in Applied Mathematics*, pages 298–351, Academic Press, 1983.

This article was processed using the TEX macro package with SIRS96 style