Very Fast Motion Planning for Highly Dexterous Articulated Robots

Daniel J. Challou, Maria Gini, and Vipin Kumar
Department of Computer Science
University of Minnesota, Minneapolis, MN 55455
Fax: (612) 625-0572
challou@cs.umn.edu (612) 626-7503
gini@cs.umn.edu (612) 625-5582
kumar@cs.umn.edu (612) 624-8023

KEY WORDS AND PHRASES

Collision avoidance, parallel algorithms, path planning, robot motion planning.

INTRODUCTION

Due to the inherent danger of space exploration, the need for greater use of teleoperated and autonomous robotic systems in space-based applications has long been apparent. However, the need for such systems has intensified lately because they will be necessary to carry out a variety of important missions. Free-flying robots carrying multiple highly dexterous robot arms have been proposed for aiding in the construction of the space station Freedom, and for assisting in satellite maintenance. Autonomous and semi-autonomous robotic devices have been proposed for carrying out routine functions associated with scientific experiments aboard the shuttle and space station. Finally, research into the use of such devices for planetary exploration continues [4].

To accomplish their assigned tasks, all such autonomous and semi-autonomous devices will require the ability to move themselves through space without hitting themselves or the objects which surround them. In space it is important to execute the necessary motions correctly when they are first attempted because repositioning is expensive in terms of both time and resources (e.g., fuel). Finally, such devices will have to function in a variety of different environments. Given these constraints, a means for fast motion planning to insure the correct movement of robotic devices would be ideal.

Unfortunately, motion planning algorithms are rarely used in practice because of their computational complexity [6]. Fast methods have been developed for detecting imminent collisions [10, 11], but the more general problem of motion planning remains computationally intractable. However, in this paper we show how the use of multicomputers and appropriate parallel algorithms can substantially reduce the time required to synthesize paths for dexterous articulated robots with a large number of joints.

We have developed a parallel formulation of the Randomized Path Planner proposed by Barraquand and Latombe [1]. We have shown that our parallel formulation is capable of formulating plans in a few seconds or less on various parallel architectures including: the nCUBE2 multicomputer with up to 1024 processors (nCUBE2 is a registered trademark of the nCUBE corporation); the CM-5 (CM-5 is a registered trademark of the Thinking Machines Corporation), and a network of workstations [3, 5]. (The results obtained on the CM-5 presented in this paper are based upon a beta version of the software and, consequently, are not necessarily representative of the performance of the full version of the software.)

One might argue that massively parallel machines are not a viable platform for space based applications due to their prohibitive cost. However, due to the continuing progress in VLSI design and economy of scale resulting from their widespread use, the cost of processors that massively parallel machines employ is expected to decrease. When this occurs, it will be feasible to build large scale

parallel computers with substantial raw computing performance at a relatively small cost.

Working projects that utilize embedded parallel processing, such as the autonomous land vehicle Navlab [8], indicate their viability. The fact that embedded parallel systems can also perform other tasks efficiently, such as image processing and image recognition, justifies their use in planning applications as well.

RANDOMIZED PARALLEL MOTION PLANNING

Most motion planning algorithms decompose the search space into discrete components called cells [9]. The motion planning problem then becomes one of computing a decomposition and searching through sequences of contiguous cells to find a path through free space (i.e. a sequence of configurations that involves no collisions with obstacles).

Unfortunately, as more degrees of freedom are added to the robot most methods become computationally impractical [9]. The only existing motion planning methods capable of synthesising plans in reasonable time frames (i.e., times on the order of minutes [6]), for robots with more than three degrees of freedom utilize an approximate decomposition of the configuration space (C-Space). The C-space is the space defined by parameters that uniquely specify the position of the robot. To obtain such performance, most methods precompute a significant portion of the C-space. Total precomputation is impossible because of both the time required to perform the computation and the amount of memory required to store the resulting C-Space. Unfortunately, precomputation relegates such methods to static workspaces, and hence they are not well suited to the space-based applications described earlier.

Our method is a parallel formulation of the Randomized Path Planner proposed by Barraquand and Latombe [1]. Space is represented with bitmap arrays. The configuration space is discretized and searched using best first search with random walks. Artificial potential fields are used as the heuristic to guide the search. The potential fields are precomputed, but their computation requires at most a few tens of seconds (and it is readily parallelizable). Furthermore, the method works with discrete representations of the environment, so it can readily be coupled with fast methods of producing such representations, such as the method proposed by [7].

The path is constructed incrementally as follows. A new configuration is randomly generated from the current configuration at the start of each step. If the heuristic value of the new configuration is smaller than the current value, and the move does not cause a collision, then the new configuration is added to the path and the search process is resumed. Otherwise another neighbor is investigated. When none of the neighbors has a smaller value than the current configuration, a random walk is executed and then the search process resumes. This process is repeated until a solution is found.

We first broadcast a bitmap representation of the workspace and the desired goal location to all processors, and then check for a message indicating that a processor has found a solution. Each processor runs the same basic program. The only interprocessor communication is the initial broadcast and the termination check. The search and random walks are the means by which the search-space is partitioned, as they insure that each processor searches different parts of the C-Space.

Although the method is only probabilistically complete, a large number of experimental results indicate that with a sufficient number of processors a solution is always found in very short time frames [3, 2].

DISCUSSION OF RESULTS

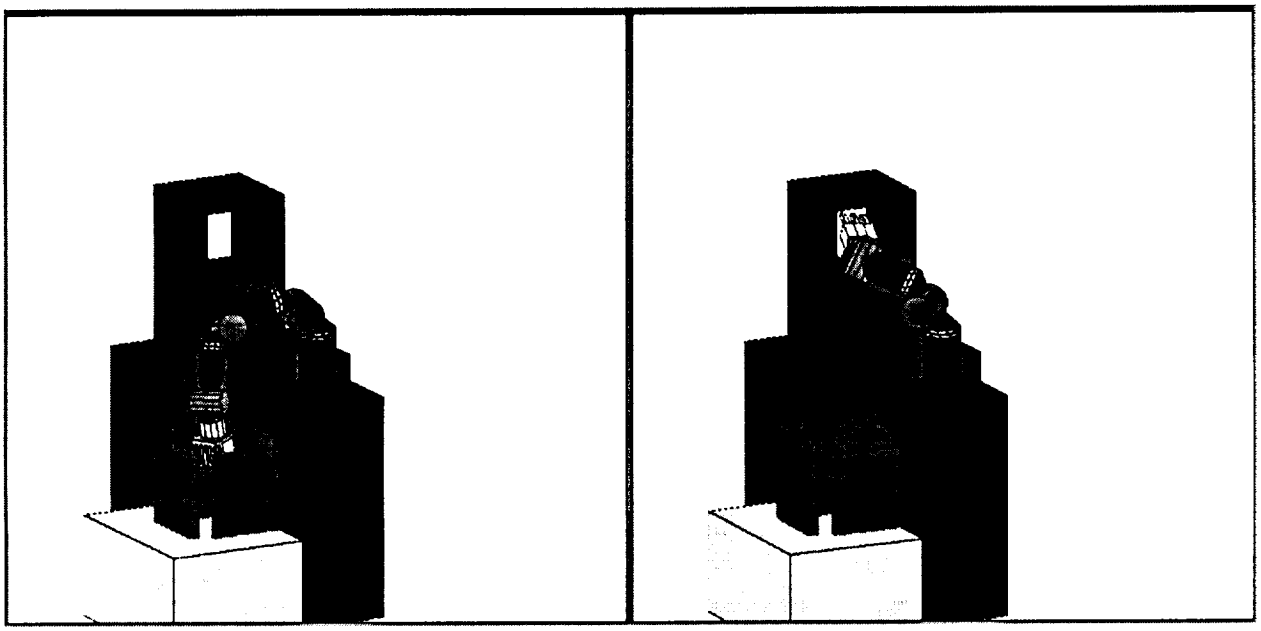
Figure 1 shows the start and goal configurations for one of our test cases for motions of a seven degree of freedom Robotics Research arm operating in a 128^3 cell workspace. Each cell in the robot's workspace represents a volume of 2.1 cubic centimeters. Each joint has up to 128 discrete positions (2.8125 degrees per position). The table shows the results on up to 256 processors on the CM-5 multicomputer. Each processor requires approximately 13.1 megabytes of random access memory.

The table indicates the benefits of parallelizing the planner. For the problem instance shown just

32 processors are required to cut the average solution an order of magnitude to under ten seconds, and 64 processors cut the average solution time to under five seconds.

In addition to delivering paths in shorter time frames, another important property of the parallel formulation is that, when it is executed with a larger number of processors, it tends to produce better solutions. We have observed this behavior in all the experiments we have performed to date. In the example, 32 processors yield a solution path length about one fourth as long as the average solution path length delivered by one processor, and 128 processors reduce the average solution path length by an order of magnitude. The variance in time to solution behaves similarly, that is, it falls off as the number of processors attempting to solve the problem increases.

The performance falls off and the average time taken to solve the problem moves toward a constant value as we increase the number of processors. This is because we hit a point where the number of processors required to insure that one processor will find a solution in the minimum possible time is optimal or near optimal for the problem instance. The probability that the random component of the algorithm will ensure that different processors are exploring different parts of the search space decreases as we add more processors. When we reach that point, then adding more processors will just result in more processors doing redundant work (in the average case).



No Processors	1	32	64	128	256
Avg Search Time	102.34	8.39	5.36	3.37	2.32
Std Dev	108.33	5.24	3.26	2.17	1.26
Avg Path Length	4264	1178	1351	967	531
Std Dev	5196	1550	1942	1277	476
Avg Speedup	1.00	12.02	19.09	30.37	44.11

Figure 1: The figure shows the start and goal configurations for a seven degree of freedom Robotics Research arm. The robot is reaching from the box in front of it, up and into the box on the left. The table shows data for at least 64 runs on a CM-5 multicomputer. All times are in seconds.

We have developed fast performance prediction methods that can be used to determine whether the number of processors available is adequate or excessive [2]. Because of the way the random planner escapes local minima and generates successors, as the minimum solution length and the degrees of freedom of the robot increase the number of different (not necessarily optimal) solution paths increases dramatically. The number of solution paths with similar lengths increases dramatically as well. This increased solution density enables the planner to perform well in instances where deterministic methods would encounter difficulty.

If a priori knowledge about obstacles allows a coarser discretization of C-space, (such as the 64 discrete positions used by [11]), then our experimental results [2] indicate that we can cut the planning time by at least a factor of three. Thus, coarser discretizations coupled with faster processors, such as Digital Equipment's alpha chip, would enable our system to deliver sub-second performance using a reasonable number of processors.

We are currently in the process of parallelizing the computation of the 3D artificial potential field maps. Preliminary results indicate that it is possible to complete the heuristic computation process in real-time. As a result, given a discrete 3D picture of an environment, our planner will be able to formulate motion plans in very fast time frames.

ACKNOWLEDGMENTS

This work was supported in part by Contract Number DAAL03-89-C-0038 between the Army Research Office and the University of Minnesota for the Army High Performance Computing Research Center. Additional support was furnished by NSF/CDA-9022509, DA/DAAH04-93-6-0800, NSG/IRI-9216941, and the Center for Advanced Manufacturing, Design and Control of the University of Minnesota.

We would like to sincerely acknowledge Mike Hennessey and Max Donath for helping us model the Robotics Research arm; Jean Claude Latombe at Stanford University for providing access to implementations of the Random Path Planner; David Strip and Robert Benner at Sandia National Laboratories for providing access to the nCUBE2.

REFERENCES

[1] J. Barraquand and J. C. Latombe. Robot motion planning: A distributed representation approach. *Int'l Journal of Robotics Research*, 10(6):628–649, 1991.

[2] D. Challou. Parallel search algorithms for robot motion planning. Ph.D. dissertation., The University of Minnesota, 1994.

[3] D. Challou, M. Gini, and V. Kumar. Parallel search algorithms for robot motion planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, volume 2, pages 46–51, 1993.

[4] S. Dubowski and E. Papadopoulos. The kinematics, dynamics, and control of free-flying and free floating space robotic systems. *IEEE Trans. Robotics and Automation*, 9(2):541–543, 1994.

[5] M. Gini, D. Challou, V. Kumar, and B. Camel. Parallel algorithms for path planning. In *Proc. Virtual Reality Systems '93, Teleoperation '93, and Beyond Speech Recognition '93.* Sig-Advanced Applications, Inc., New York, 1993.

[6] Y. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

[7] J. P. Jones. Real-time construction of three-dimensional occupancy maps. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1993.

[8] T. Kanade, M. L. Reed, and L. E. Weiss. New technologies and applications in robotics. *Communications of the ACM*, 37(3):58–67, March 1994.

[9] J. C. Latombe. *Robot Motion Planning.* Kluwer Academic Publ., Norwell, MA, 1991.

[10] C. A. Shaffer. A real-time robot arm collision avoidance system. *IEEE Trans. Robotics and Automation*, RA–8(2):149–160, 1992.

[11] T. S. Wikman, M. Branicky, and W. S. Newman. Reflexive collision avoidance: a generalized approach. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, volume 3, pages 31–36, 1993.