Proceedings of 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation July 29 - August 1, 2001, Banff, Alberta, Canada

Heuristic Methods for Randomized Path Planning in Potential Fields

Stefano Caselli, Monica Reggiani, Roberto Rocchi *
Dipartimento di Ingegneria dell'Informazione
Università di Parma - 43100 Parma - Italy
e-mail: {caselli,reggiani,rocchi}@ce.unipr.it

Abstract

Randomized path planning driven by a potential field is a well established technique for solving complex, many degrees of freedom motion planning problems [5]. In this technique a suitable potential field shapes the search of the path toward the goal. However, randomized path planning can become relatively inefficient when deep local minima are present in the potential field. Indeed, the algorithm usually spends most its running time trying to escape from local minima by means of uninformed random motions. In this paper we present simple yet effective heuristics for escaping local minima, with the goal of improving overall planning performance. We integrate these heuristics into a path planner without sacrificing the overall probabilistic completeness of the algorithm. Experimental results on several test cases show a remarkable performance improvement, up to a factor of 4 for complex problem instances.

1 Introduction

Probabilistic motion planning techniques are among the mainstream methods for solving complex planning problems, possibly involving many degrees of freedom (d.o.f.'s) [5, 13, 19, 1, 16]. These methods have been shown to be probabilistically complete, i.e., they find a solution in bounded time with high probability if such a solution exists. Moreover, probabilistic motion planning algorithms have successfully dealt with large problem instances in such diverse domains as robot path planning in industrial workcells, design for maintainability of complex mechanical systems, and digital actors [15, 9, 16, 18].

There are two major approaches toward probabilistic path planning [2]: *potential field* methods, pioneered by the *Random Path Planner (RPP)* algorithm [4, 5], and *roadmap* methods, initiated by the *Probabilistic RoadMap (PRM)* algorithm [19, 13, 14]. *Potential field* methods drive the search for a path in configuration space (C-space) with the help of a suitable heuristic function; they combine gradient-based, goal-oriented motions with random walks to escape local minima [5, 8]. *Roadmap* methods randomly sample collision-free configurations and connect them into a graph during a preprocessing phase; this graph is exploited as a roadmap to efficiently answer subsequent path queries [13, 19, 1].

Currently, probabilistic roadmap methods are very popular. However, they suffer a major difficulty in dealing with narrow passages offering minimal visibility between connected components of C-free [11]. Furthermore, the expensive preprocessing phase is only amortized if multiple planning queries are made in the same workspace.

On the other hand, probabilistic potential field methods such as RPP are often quite inefficient in dealing with local minima which unavoidably arise in the potential field. This inefficiency is mainly due to the random walk performed to escape these local minima, which in turn is essential to guarantee probabilistic completeness [5]. An additional problem of potential field methods is their difficulty in dealing with traps, i.e., very deep local minima with narrow exit.

On the positive side, potential field methods are well amenable to parallelization and require minimal tuning of parameters. In earlier research we have investigated parallel path planning in a potential field using a random competion approach [8, 6] and exploiting learning techniques [7] across multiple queries. Both techniques help in reducing the impact of traps in the potential field thereby significantly improving average planning performance.

In this paper we aim to increase the efficiency of potential field-based probabilistic planning by replacing or supplementing the random walk with a more informed search, while preserving the probabilistic completeness property of the algorithm. This effort can be seen as dual to recent improvements to the basic PRM approach [1, 17, 10], where random sampling of configurations is biased to make it more effective. Alternative search strategies for escaping local minima are pursued as enhancements to be integrated in a new path planning tool under development at the University of Parma. This tool will also include exploitation of past experience and support for concurrent computation [7]. In this paper, however, heuristic search techniques are investigated and evaluated in terms of improve-

*This work was partially supported by CNR, Italy (CT07), and by ASI, Italy (Project No. 148/3, Dextrous Manipulation for Space Robot Tasks).

ment over a sequential path planner. The combined effect of concurrent computation, learning, and improved heuristic search should make potential field planning a very competitive technique for complex, many d.o.f. problems.

The paper is organized as follows. Section 2 reviews in some detail RPP and its technique for escaping local minima. It also discusses the main sources of inefficiency and outlines directions for improvement. Section 3 describes alternative techniques for dealing with local minima which may be effective in certain cases. Section 4 reports experimental results from several problems assessing the relative effectiveness of the various techniques, both individually and in combination. A final section summarizes the contributions of the paper.

2 Exiting local minima in Random Path Planner

The RPP algorithm [5] constructs a graph whose nodes are the local minima of a potential function U, defined in the discretized configuration space of the robot. The potential function U is obtained from a potential attracting the robot toward the goal (placed in the global minimum of U) and a repulsive potential around obstacles.

Starting from the initial configuration $\mathbf{q}_{init}$ of the robot, the algorithm executes a *gradient motion,* following the negated gradient of the potential U until it finds a local minimum $\mathbf{q}_{loc}$. If $\mathbf{q}_{loc}$ is the global minimum $\mathbf{q}_{goal}$ (i.e., $U = 0$), the algorithm terminates with success since a path has been found; otherwise the algorithm tries to escape from $\mathbf{q}_{loc}$ executing a fairly complex *random motion* phase, to be detailed hereafter. Once the region of attraction of the local minimum has been successfully escaped, the random motion is followed by a gradient motion that will find a (possibly new) local minimum. The algorithm therefore alternates gradient motions with random motion phases until the goal configuration is reached. When the planner finds a path between two local minima, it connects them with an edge in the graph. Hence, an incremental graph construction occurs until the goal configuration is found. Every edge p in the graph connecting two local minima is composed of an "up-hill" path p_u, resulting from the random motion phase, and a "down-hill" path p_d, resulting from the gradient motion phase. Profiling data from RPP execution on a set of problems described in section 4 show that the time spent in the aggregated random motion phases is always more than three times the time required by the gradient motion phases. Moreover, the average time spent executing random motion with respect to the total computation time increases from 43% for 7 d.o.f. problems to 75% for 11 d.o.f. problems.

The random motion phase is carefully implemented in RPP in order to ensure the probabilistic completeness. The algorithm executes up to a number of brownian motions from the local minimum until the domain of attraction of $\mathbf{q}_{loc}$ has been escaped, i.e., $U(\mathbf{q}) < U(\mathbf{q}_{loc})$ for the current configuration, or the maximum number of steps $MAX\ STEPS$ has been reached. The choice of $MAX\ STEPS$ is critical for planning performance. Reaching a new local minimum can be hampered by a too low value of $MAX\ STEPS$, whereas a too high value could yield long and expensive exploration of far C-space areas [5]. When a maximum number of random motions has been reached, the algorithm performs a random backtrack to a previous configuration of the path built so far.

The random motion guarantees the probabilistic completeness of the planning algorithm at the price of an expensive exploration of the configuration space, due to the chaotic brownian search which often moves back to previously explored areas. As shown in [3], the number of steps required to escape from the attraction domain of a local minimum by a random motion, $t(\mathbf{q}_l)$, has a quadratic dependence on the average radius of the attraction basin $R(\mathbf{q}_l)$, i.e., $t(\mathbf{q}_l) = E\left[\left(\frac{R_i(\mathbf{q}_l)}{\Delta_i}\right)\right]^2$, where Δ_i is the step size along the i-th axis.

When the local minimum has a large attraction basin, i.e. high $E\left[\left(\frac{R_i(\mathbf{q}_l)}{\Delta_i}\right)\right]$, and at the same time a large passage to exit, this motion becomes inefficient. The random motion, indeed, must always execute a number of steps at least equal to the square of the attraction radius in order to depart enough from the minimum and thus have sufficient chances to escape the local minimum. Hence, improvements in the random search phase can significantly improve overall planning efficiency, especially in complex or many d.o.f. problems.

3 Heuristic techniques

Consider being blind and trapped inside an unknown labyrinth. In this situation, a way to search for an exit consists in executing random motions hoping to find the exit. However, if the environment is easier, such as if an exit from an unknown but standard room must be found, a motion that follows a straight line in a randomly chosen direction is likely to find an exit faster than a random motion.

This is the rationale at the base of the approach presented in the following. Instead of executing complete but expensive random motions, incomplete but fast motions are exploited in the "up-hill" movement to escape from easy local minima.

Algorithm *StraightLine (SL)* simply selects a random direction from $\mathbf{q}_{loc}$ in the n-dimensional C-space, and follows it until it reaches an obstacle or a configuration such that $U(\mathbf{q}) < U(\mathbf{q}_{loc})$. In case a joint limit is reached, a new admissible random direction is chosen leading to a zig-zag path. With either terminating conditions, *StraightLine* is followed by a *down motion* which eventually leads to a local minimum. If the original local minimum or a minimum with higher potential is reached, the move

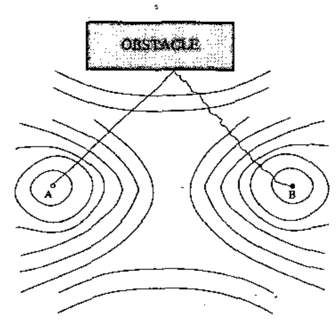

Figure 1: Escaping a local minimum with a straight line move.

is discarded and *StraightLine* tries a new random direction. Up to a maximum number of random directions from $\mathbf{q}_{loc}$ are searched until the local minimum is successfully escaped or the algorithm fails. In practice, a direct exit from the basin of attraction of the local minimum seldom occurs; rather, owing to the intricacies in C-space, the vast majority of successful *StraightLine* moves occur via "collision" with an obstacle. Such a situation is exemplified in Figure 1.

The cost for an "up-hill" motion using the SL algorithm highly depends on p, probability to escape from a local minimum with a single straight line motion. Assuming that X_{SL} is the random variable for the number of straight line motion attempts required to escape from a local minimum, then the probability to escape after k straight line motion is $P\{X_{SL} = k\} = p(1-p)^{k-1}$. The average cost of a single straight line motion is then $R(\mathbf{q}_l) * C_{cc} + E[C_{p_d}]$, where C_{cc} is the cost of a collision detection operation and $E[C_{p_d}]$ is the average cost of a down motion. The average cost to escape from the local minimum using SL is then:

$$E[C(p_{u_{SL}})] = E\{X_{SL}\}(R(\mathbf{q}_l) * C_{cc} + E[C_{p_d}]).$$

The *StraightLine* heuristic turns out to be effective in many "easy" local minima, with high value of p and low $E\{X_{SL}\}$, that are dealt with quite inefficiently by brownian search. These "easy" local minima are likely to occur rather frequently in many planning problems. For difficult local minima, however, SL is likely to be more expensive than a random motion. When the probability p decreases, possibly reaching zero, the value of $E[C(p_{u_{SL}})]$ can indeed be higher than $E[C(p_{u_B})]$.

For high-dimensional C-space, it would be impossible to make a "a priori" classification of minima in "easy" and "difficult" ones, because the computation of U must be limited to reached configurations. A precomputation of U would be intractable so the precomputation is limited to the workspace potentials of control points. Nevertheless, the planner can learn the difficulty of the local minima through the exploration of the attraction basin.

Statistical data collected on five problems discussed in Section 4 illustrate the behavior of SL in

prob.	1-10	11-50	51-300	301-2k	>2k
1	95.9%	4.1%	0%	0%	0%
2	71.4%	12.8%	8.3%	6.3%	1.2%
3	68.8%	17.8%	7.8%	2.8%	2.8%
4	73.9%	10.4%	1.2%	2.4%	12.1%
5	57.7%	21.9%	8.0%	3.4%	9.0%

Table 1: Number of random directions required to escape from the local minima using SL in five different problems. (Each problem solved 100 times).

exiting from local minima. As shown by table 1, the SL method is often effective in its initial attempts, involving a limited number of random directions. When many directions are unsuccessfully explored, the minimum is likely to be very difficult or even impossible to escape using SL.

Hence, *StraightLine* has been modified in a new algorithm, *SL'*, which limits the number of "up-hill" motions using StraightLine to c_{SL}, empirically set as 200 based on values from Table 1. The planner, once reached the limit c_{SL}, autonomously classifies the problem as "difficult" for the StraightLine heuristic. Hence, it reverts to random motion, thereby guaranteeing also the probabilistic completeness of the planning algorithm.

StraightLineSelect (SLS) is also a modified uphill motion heuristic. SLS tries to further optimize the basic idea by early pruning non-promising candidate directions. During each *StraightLineSelect* move the potential $U(\mathbf{q})$ along the straight line or the zig-zag path is tracked by a simple state machine. If along the chosen direction the potential increases monotonically, with high chance the ensuing *down motion* would bring to the same local minimum. These directions are thus discarded without actually performing the down motion. More promising directions are those where $U(\mathbf{q})$ exhibits a non-monotone behavior – a clue for a valley in the potential driving to a new local minimum (Figure 2). Only these promising random directions are thus actually followed by the down motion search in *StraightLineSelect.*

As already done with SL, we empirically define a value, c_{SLS}, based on the data in Table 2 to make a distinction between "easy" and "difficult" problems. Tables 1 and 2 show that more directions are generally required by SLS to escape the local minima. The higher number of SLS attempts, however, usually does not imply a longer execution time. The average cost to escape from a local minimum with the SLS method is:

$$E[C(p_{u_{SLS}})] = E\{X_{SLS}\}(R(\mathbf{q}_l) * C_{cc}) + E\{X_{SLS}\}E[C_{p_d}]q,$$

where q is the probability that the SLS motion crosses a potential valley. When q is small the cost of n SLS escape motions is much lower than the cost of n SL attempts, since down motions requiring costly collision detection operations are avoided. The variation in the SLS heuristics labeled *SLS'* re-

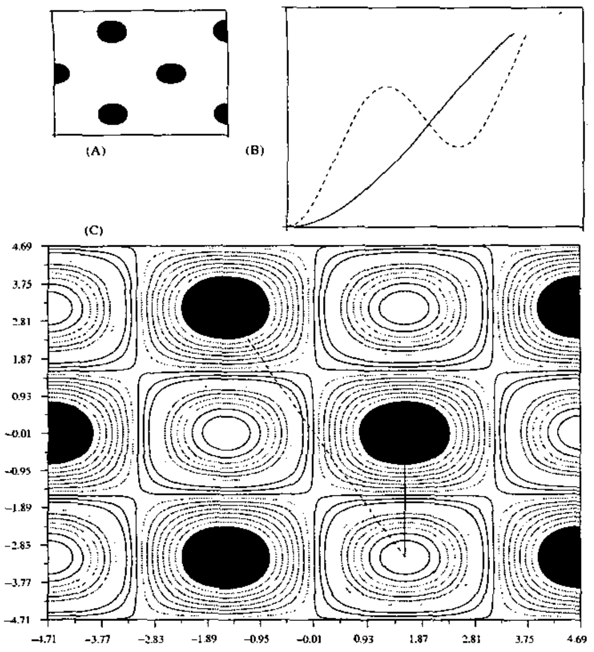

Figure 2: Potential value along a random straight line in the *SLS* heuristic: (a) 2D workspace; (b) potential function; (c) monotone and non-monotone potential distributions along two directions.

prob.	**1-10**	**11-50**	**51-300**	**301-2k**	**>2k**
1	68.8%	27.4%	3.8%	0%	0%
2	34.7%	27.5%	17.8%	13.3%	6.7%
3	14.6%	23.0%	28.3%	14.0%	20.1%
4	27.7%	21.3%	22.1%	12.6%	16.3%
5	9.1%	17.9%	29.5%	26.5%	17.0%

Table 2: Number of random directions required to escape from the local minima arising in the five problems using SLS. (Each problem solved 100 times).

verts to brownian motion after 500 random directions attempted.

A further variation of *StraightLineSelect*, labeled *SLS"* has also been developed. SLS and SLS' behave rather irregularly across the problems examined: for some problems, largely increasing the number of explored directions improves planning performance, whereas for other problems the increase is detrimental. In order to let the planner adapt to the difficulty of the local minimum, the SLS" algorithm reverts to brownian motion when either it has already evaluated $c_{SLS} = 500$ random directions or it has fully explored 10 promising directions with the down motion as well.

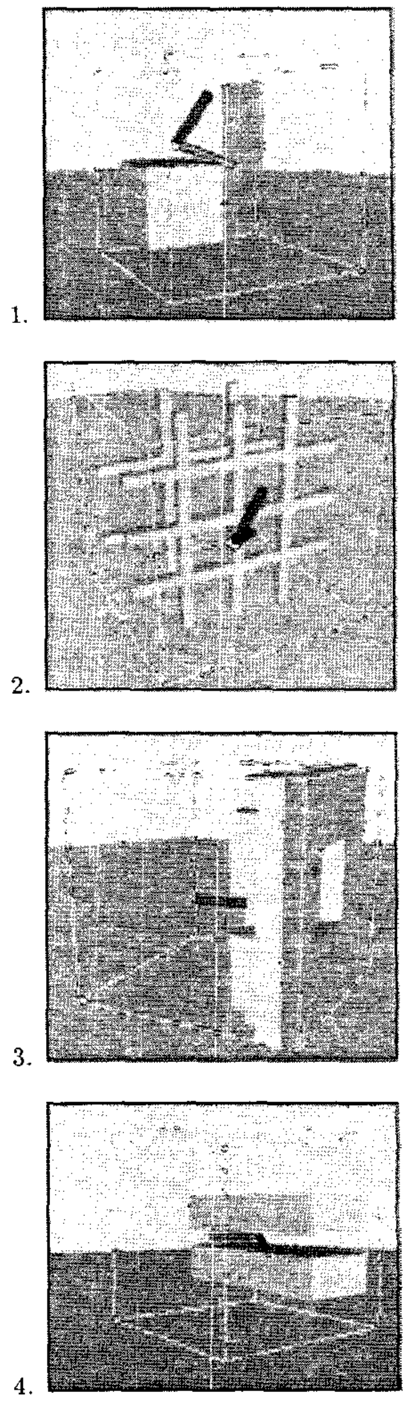

Figure 3: Set of four path planning problems.

problem		BR	SL'	SLS'	SLS"
1	avg	6.64	2.22	8.17	2.69
	std	5.00	0.31	7.17	1.66
2	avg	4.31	6.11	3.95	3.90
	std	2.68	3.62	1.83	1.54
3	avg	6.88	4.07	9.70	6.41
	std	4.40	3.13	7.58	5.72
4	avg	28.47	9.74	28.24	4.85
	std	24.23	12.40	26.19	4.96

Table 3: Planning results for 7 d.o.f. problems in Figure 3. (Times in seconds; 100 independent solutions per problem).

4 Experimental results

We have evaluated the *StraightLine* and *StraightLineSelect*, along with the brownian motion in RPP (labeled as BR in the following tables), on the four problems shown in Figure 3, all referring to a 2 link, 7 d.o.f. robot. The choice of these planning problems as testbed for our algorithms has been motivated by the classification of workspaces proposed by Hwang and Ahuja in [12]. The first workspace belongs to the set of easy problems, first class in Hwang and Ahuja's classification. The space among obstacles is large compared to robot dimensions and no narrow passage is present. The second and third problems are in the second class, i.e. medium level difficulty problems. The planner must be able to find a path for the robot when its movements are restricted in a narrow space among close obstacles. The fourth problem belongs to the last and most difficult class in the classification. The robot is required to enter a narrow slot, where the goal is located. If the entering orientation is incorrect, the robot might be forced to exit the slot in order to reorient itself, a situation likely involving a very deep local minimum in U. Additional problems investigated involve a 9 d.o.f. and an 11 d.o.f. robots in an environment similar to problem 2. In these problems the difficulty stems from the increased number of d.o.f.'s of the robot. All performance results reported in the following have been obtained with a 450 MHz Pentium-II PC with 256 MB main memory, under the Linux operating system.

Table 3 reports average and standard deviation of the planning times achieved by the modified heuristics for the four problems in Figure 3 (all involving a 7 d.o.f. robot). Percentages of solved problems are not reported, since all modified heuristics are (of course) always successful. Table 4 reports summary data obtained in the solution of ten 9 d.o.f. problems (randomly chosen goals for a 3 link, 9 d.o.f. robot in the workspace of Problem 2) and ten 11 d.o.f. problems (randomly chosen goals for a 3 link, 11 d.o.f. robot in the workspace of Problem 2). Table 4 reports average time and standard deviation obtained in solving the 9 d.o.f. and the 11 d.o.f. makespans with the various heuristics

makespan		BR	SL'	SLS'	SLS"
9 d.o.f.	avg	286.15	102.89	105.37	65.97
	std	91.56	24.09	36.56	13.79
11 d.o.f.	avg	389.84	99.71	180.66	105.16
	std	146.79	31.06	71.22	38.24

Table 4: Planning results for makespans of 9 d.o.f. and 11 d.o.f. problems. (Times in seconds; 20 independent solutions per makespan).

(each problem is solved 20 times). All problems are reliably solved by the heuristics in 100% of the cases. Figures 4 and 5 detail the average planning time achieved by the heuristics on the individual 9 d.o.f. and 11 d.o.f. problems. Tables 3 and 4 show remarkable improvements attained by the modified *SL'* and *SLS"* heuristics over the standard RPP (up to a factor 4 for the makespans of 9 d.o.f. and 11 d.o.f. problems). The improvement is larger for the more difficult problems, where RPP random search indeed requires an increased fraction of time.

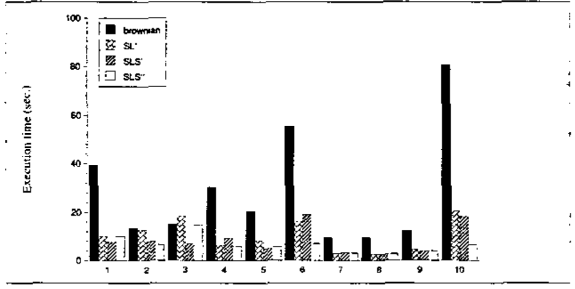

Figure 4: Average planning time for the set of 9 d.o.f. problems. (Times in seconds; 20 independent solutions per problem).

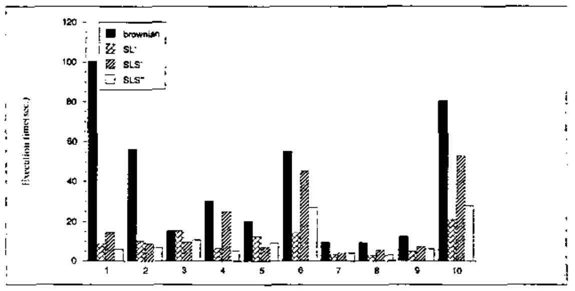

Figure 5: Average planning time for the set of 11 d.o.f. problems. (Times in seconds; 20 independent solutions per problem).

As far as the relative merit of the two *SL'* and *SLS"* heuristics is concerned, no decisive advantage can be seen for either method. Their relative effectiveness seems to depend on the problem, thus implying that the specific U profile tracked by *SLS* is not always the only key to escape the local minimum.

5 Conclusions

We have described simple yet effective heuristics for improving potential field-based path planning performance. Two heuristics, labeled *SL'* and *SLS"*, have been shown to yield remarkable performance benefits for complex problems.

In future work we plan to integrate these heuristics in a parallel path planner under development at the University of Parma and let them operate concurrently.

References

[1] N. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *IEEE International Conference on Robotics and Automation*, pages 113–120, Minneapolis, MN, 1996.

[2] J. Barraquand, L. E. Kavraki, J.-C. Latombe, R. Motwani, T.-Y. Li, and P. Raghavan. A random sampling scheme for path planning. *International Journal of Robotics Research*, 16(6):759–774, 1997.

[3] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241, 1992.

[4] J. Barraquand and J.-C. Latombe. A monte-carlo algorithm for path planning with many degrees of freedom. In *IEEE International Conference on Robotics and Automation*, pages 1712–1717, Cincinnati, OH, 1990.

[5] J. Barraquand and J.-C. Latombe. Robot motion planning: a distributed representation approach. *Internation Journal of Robotics Research*, 10(6):628–649, 1991.

[6] S. Caselli and M. Reggiani. Randomized motion planning on parallel and distributed architectures. In *Euromicro Workshop on Parallel and Distributed Processing*, pages 297–304, Funchal, Portugal, 1999.

[7] S. Caselli and M. Reggiani. ERPP: An experience-based randomized path planner. In *IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2000.

[8] D. J. Challou, M. Gini, and V. Kumar. Parallel search algorithms for robot motion planning. In *IEEE International Conference on Robotics and Automation*, pages 46–51, Atlanta, GA, 1993.

[9] H. Chang and T.-J. Li. Assembly maintainability study with motion planning. In *IEEE International Conference on Robotics and Automation*, pages 1012–1019, Nagoya, Japan, 1995.

[10] D. Hsu. *Randomized Single-Query Motion Planning in Expansive Spaces.* PhD thesis, Dept. of Computer Science, Stanford University, Stanford, CA, 2000.

[11] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Workshop on the Algorithmic Foundations of Robotics*, Houston, TX, 1998.

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

[13] L. E. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE International Conference on Robotics and Automation*, pages 2138–2145, San Diego, CA, 1994.

[14] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transaction on Robotics and Automation*, 12(4):566–580, 1996.

[15] Y. Koga, K. Kondo, J. J. Kuffner, and J.-C. Latombe. Planning motions with intentions. In *SIGGRAPH'94*, pages 395–408, Orlando, FL, 1994.

[16] J.-C. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *Internation Journal of Robotics Research*, 18(11):1119–1128, 1999.

[17] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Fourth Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, 2000.

[18] S.M. LaValle, P.W. Finn, L.E. Kavraki, and J.-C. Latombe. A randomized kinematics-based approach to pharmacophore-constrained conformational search and database screening. *Journal of Computational Chemistry*, 21(9):731–747, 2000.

[19] M. H. Overmars and P. Švestka. A probabilistic learning approach to motion planning. Technical Report UU-CS-1994-03, Department of Computer Science, University of Utrecht, 1994.