

# An Adaptive Framework for ‘Single Shot’ Motion Planning: A Self-Tuning System for Rigid and Articulated Robots\*

Daniel Vallejo  
Department of Computer Systems Engineering  
Universidad de las Américas-Puebla  
Cholula, Puebla 72820 Mexico  
dvallejo@mail.udlap.mx

Ian Remmler Nancy M. Amato  
Department of Computer Science  
Texas A&M University  
College Station, TX 77843  
{irr5509, amato}@cs.tamu.edu

## Abstract

*This paper describes an enhanced version of our previously proposed adaptive framework for single shot motion planning. This framework is versatile, and particularly suitable for crowded environments. Our iterative strategy analyzes the characteristics of the query and adaptively selects planners whose strengths match the current situation.*

*Contributions in this paper include an automatic method for setting and adaptively tuning planner characterizations, and reducing the reliance on programmer expertise present in the original framework. The adaptive refinement enables the system to evolve parameters specifically suited for particular classes of applications. The system now supports articulated robots, which were not supported previously. Our experimental results in complex 3D CAD environments show that our strategy solves queries that none of the planners could solve on their own.*

## 1 Introduction

Automatic motion planning has application in many areas such as robotics, virtual reality systems, and computer-aided design. When many motion planning queries will be performed in an environment, it may be useful to preprocess the environment (as *roadmap* motion planning methods [18, 15] do, for example) in order to reduce the difficulty of subsequent queries.

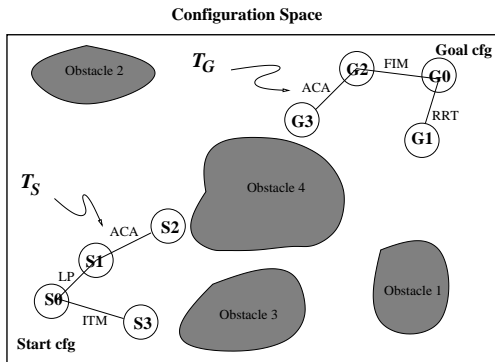
However, if the **start** and **goal** configurations are known *a priori*, and only a few queries will be performed, it is generally not worthwhile to perform an expensive preprocessing stage. In this case, a more directed search of the free configuration space could lead to faster solution times. Motion planning methods that operate in this fashion are often called *single*

*shot* methods. Single shot methods are also useful in dynamic environments in which other techniques, such as roadmap methods, are not suitable.

One of the first randomized planning methods is the Randomized Path Planner (RPP) of Barraquand and Latombe [5], which is a single shot planner. This potential field method uses random walks to attempt to escape local minima, which works well when C-space is relatively free, but is not as effective in cluttered scenarios [15]. Some success has been achieved in adapting the general probabilistic roadmap (PRM) strategy to solve single queries by restricting attention to ‘useful’ regions of C-space [14, 20]. A related idea is to use a sample of free points to specify promising subgoals [7, 11, 12]. Other approaches that have been used are to limit the part of the C-space that is explored. The planner in [16] finds paths for six-dof manipulators using heuristic search techniques. A skeleton of the C-space is built incrementally using a local opportunistic strategy in [8]. The Ariadne’s clew algorithm [7] uses genetic algorithms to help search for a path in high-dimensional C-spaces. The Rapidly-exploring Random Tree (RRT) [17] method grows a tree from a specified start configuration. In [9] a hybrid approach is considered which utilizes a potential function (similar to RPP) on queries, but also saves information from past attempts in a graph to aid future queries in the same environment.

Although much progress has been made, good single shot solutions are needed for problems in cluttered environments. In [22], we present an iterative strategy for single shot motion planning. At any given time, we have a set of sub-queries, any one of which would solve the original motion planning problem, and a set of planning algorithms which can be applied to the sub-queries. We select a sub-query and a planner to be used on it, based on an evaluation process meant to select the best suited local planning method and the most promising sub-query. Anywhere from one to all of the local planners in the bank may be executed when solving a particular query – the goal is to use

<sup>1</sup>This research supported in part by NSF CAREER Award CCR-9624315 (with REU Supplement), NSF Grants IIS-9619850 (with REU Supplement), ACI-9872126, EIA-9975018, EIA-9805823, and EIA-9810937, and by the Texas Higher Education Coordinating Board under grant ARP-036327-017. This work was performed while Vallejo was studying at Texas A&M where he was supported in part by a Fulbright-CONACYT scholarship.



```

Repeat
  <qp,alg> := ExtractBest(PQ);
  PartialPath := Alg(qp);
  if ( qp.start in T_S )
    T_S := T_S + PartialPath; T := T_S;
  else T_G := T_G + PartialPath; T := T_G;
  if (T_S != T_G)
    last := LastCfg(PartialPath);
    for each (cfg in T)
      qp1 := <last,cfg>;
      qp2 := <cfg,last>;
      for each (alg in AlgoBank)
        insert( <qp1,alg>, score(qp1,alg), PQ);
        insert( <qp2,alg>, score(qp2,alg), PQ);
  until ( T_S == T_G )

```

(a) (b)

Figure 1: (a) A graph generated during Single Shot planning. Edges are labeled with the planning algorithm that made the connection, and (b) pseudo-code description of single shot loop.

them cooperatively.

Contributions in this paper include an automatic methodology for adaptively tuning the planner characterizations, thus reducing the reliance on programmer expertise present in the original framework. The adaptive refinement enables the system to evolve parameters specifically suited for particular classes of applications. In addition, the current system has been extended to articulated robots, which were not supported in the previous system. Our experimental results in complex 3D CAD environments show that our strategy solves queries that none of the planners could on their own. For completeness, the system in its entirety is described here, with more emphasis placed on the new additions.

## 2 Single Shot Framework

The framework is outlined in Figure 1. To find a path from the **Start** to the **Goal** configuration, we grow a tree  $T_S$  from the **Start** configuration and another tree  $T_G$  from the **Goal** configuration. In each iteration, we attempt to generate a path that connects  $T_S$  and  $T_G$ , and therefore also the **Start** and **Goal** configurations. In each iteration, we consider all potential query pairs with one configuration in  $T_S$  and one in  $T_G$ , and all the algorithms in the bank, and select the query pair and algorithm combination that is most likely to make a connection. Then, the selected algorithm is executed on the query pair and we add the (partial) path returned to  $T_S$  or  $T_G$ . If the sub-query fails to solve the problem, then we generate query pairs of the form  $(last, cfg)$  and  $(cfg, last)$ , where  $last$  is the last configuration of the partial path returned by the sub-query, and  $cfg$  is a configuration in a different connected component from  $last$ . Each such pair is evaluated, and promising  $\langle QueryPair, Algorithm \rangle$

tuples are added to the priority queue. The process continues in this way, growing  $T_S$  and  $T_G$ , until they can be connected or until some maximum number of iterations is reached.

### 2.1 Evaluation Criteria

The most important operation in each iteration is selecting the “best”  $\langle QueryPair, Algorithm \rangle$  tuple. First, we calculate characteristics of the potential query pairs. Second, based on the query pair and algorithm characteristics, a score is assigned to each  $\langle QueryPair, Algorithm \rangle$  tuple. These scored tuples are stored in a priority queue so that the “best”  $\langle QueryPair, Algorithm \rangle$  tuple can be selected in each iteration.

#### 2.1.1 Query properties and characteristic values

For a given query pair, evaluation criteria are used to characterize the **local properties** of the space near the start and goal configurations and the **global properties** of the space between them (details can be found in [3]).

The evaluation begins by determining general characteristics of individual configurations. This is done by sampling additional configurations near the configuration of interest. For each characteristic property, we compute a numerical value (normalized to  $[0, 1]$ ). Currently, for each configuration  $c$ , we consider the following **local properties**:

- $L_1$ : Clearance (distance to nearest obstacle),
- $L_2$ : Translation (free translational space near  $c$ ),
- $L_3$ : Rotation (free rotational space near  $c$ ),
- $L_4$ : Free (free space near  $c$ ).

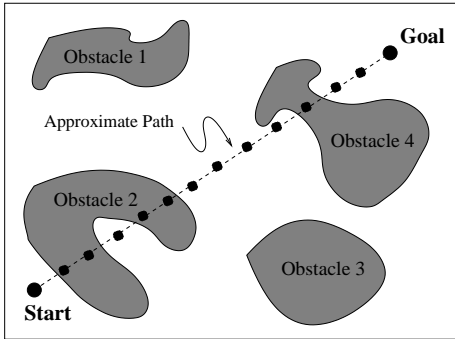


Figure 2: The approximate path  $P$ .

Next, the space between the configurations  $c_1$  and  $c_2$  is evaluated to analyze the type of planning required for the query. This is done in part by evaluating configurations sampled at a coarse resolution on a straight-line connecting  $c_1$  and  $c_2$  (see Figure 2). Currently, the **global properties** (normalized to  $[0, 1]$ ) considered are:

- $G_1$ : Distance between  $c_1$  and  $c_2$ .
- $G_2$ : Proportion of free configurations in  $P$ ,
- $G_3$ : Number of obstacles in collision with  $P$ .

### 2.1.2 Planner characteristics

We assign each algorithm values corresponding to the characteristics which describe the ideal situation in which the planner should be used. Each planning algorithm is assigned values  $s_{L_i}$  and  $g_{L_i}$ ,  $i = 1..4$ , representing local characteristics of ideal start and goal configurations, respectively, and values  $q_{G_i}$ ,  $i = 1..3$ , representing the global characteristics of a query on which that algorithm should be used.

In the original system described in [22], these values were pre-set using programmer expertise. As described in Section 4.3, our current system provides an automated mechanism for deriving, and automatically updating, the values for each planner. We consider two general scenarios, environments that are relatively “OPEN” and cluttered environments that are more “CONFINED,” and derive default characteristic values for each. We have made this distinction since we believe that the selection of the planners for a particular sub-query depends mainly on the global characteristics of the environment considered. Thus, we ask the user to identify the queries as either “OPEN” or “CONFINED” when initiating the single shot query.

We also provide a mechanism for updating the characteristic values assigned to a planner dynamically. If a planner makes “good” progress when trying to solve

a sub-query, the characteristic values of that query are averaged with the current values assigned to the planner. If the planner reduces the original distance between the start and goal configurations by at least half, then the characteristic values of the sub-query are averaged with those of the planner. In this way, the program can “learn” from good experiences.

### 2.1.3 Scoring

The score for a  $\langle \text{QueryPair}, \text{Algorithm} \rangle$  tuple is a weighted average in which we give decreasing importance to the local properties of the start configuration ( $S$ ), the global properties of the query ( $Q$ ), and the local properties of the goal configuration ( $G$ ).

First, for each component ( $S, Q, G$ ), we sum the differences between the computed `QueryPair` value and the algorithm’s assigned value. The score for the tuple  $\langle (s, g), A \rangle$  is:

$$\text{score}((s, g), A) = S(s, A) + \frac{1}{2}Q(s, g, A) + \frac{1}{4}G(g, A)$$

with the best possible score being zero, representing a perfect match between query pair  $(s, g)$  and planner  $A$ .

## 3 Single Shot Algorithm Bank

The algorithm bank should include a diverse set of algorithms so that at least one of them will perform well in each possible situation. See [21] for details.

Our algorithm bank includes several *local planning* methods that are commonly used in PRMs. They are relatively simple, deterministic methods that are fast but not very powerful. The methods currently in our bank are: straight-line in C-space, rotate-at-s (for rigid bodies), and simple  $A^*$ -like planners. All methods are described in detail in [1].

The *directed expansion methods* attempt to grow a connected component of configurations from the **start** to the **goal**. The iterative spread (ISM), translational (ITM) and rotational (IRM) methods try to ‘spread’ away from the **start**, and towards the **goal** when possible. ACA is a variant of the Ariadne’s Clew algorithm [7] which uses random walks instead of the originally proposed Manhattan paths as its local planning method.

The *random expansion methods*, which grow a connected component of configurations from the **start** configuration, are useful in crowded situations. The *random walk* (RWM) method performs a random walk of a given number of steps from a **start** configuration. It is useful for escaping from “local minima.” Our bank also includes a simple version RRT of the

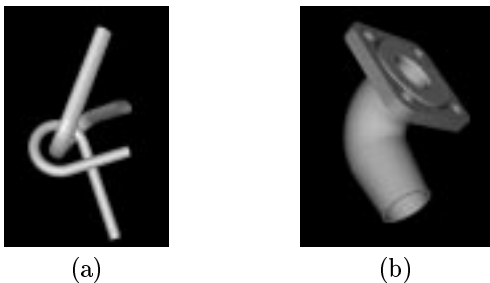


Figure 4: CAD models (a) alpha, (b) flange.

*Rapidly-exploring Random Tree* algorithm proposed in [17].

The *subgoal generation methods* generate subgoals for navigating around obstacles. They are suitable candidates when the `start` or `goal` is in open space and there are obstacles between them. The *first intersection method* (FIM) tries to go directly from the `start` to the `goal` using the straightline planner. If an obstacle interferes, it samples configurations around it in a manner similar to OBPRM [2]. The *recursive midpoint method* (RMM) splits the query at its midpoint into two sub-queries. This continues until each sub-query results in a collision-free connection.

## 4 Experimental Setup

Here we describe the implementation details of our system, the environments that we consider, and the experimental methodology used to obtain the characteristic values for the planners.

### 4.1 Implementation details

Our system consists of approximately 6,000 lines of C++ code, and is built on top of the OBPRM library ([1, 2, 4]), which consists of about 25,000 lines of C++ code. For collision detection, our system supports RAPID [13], V-Clip [19], and CSTK [23, 24]. The experiments were conducted on an HP V2200 system.

### 4.2 Environments studied

For our tests we used several artificial environments with rigid and articulated robots, and complex CAD-type environments with only rigid robots. The artificial environments (Figure 3) were designed so that it would be advantageous to use multiple planners. We tried to include representative environments, some relatively open, and some relatively confined. The CAD models (Figure 4) represent more realistic motion planning problems.

### 4.3 Determining planner characteristic values

The simulation studies described here were used to set the default planner characteristic values for the OPEN and CONFINED cases. A digest of the results is presented here, but complete results can be found in [21].

Of the artificial environments, we considered the Sandwich-Wall and the Stairs environments to be CONFINED, and the Walls and House to be OPEN. For each environment, we used a visualization tool to select about 20 representative configurations in open space, in narrow corridors, etc. We then considered all possible pairs of these configurations as start and goal. For each query pair, we recorded the characteristic values, whether each planner was successful, and the running time of each query. To find our default characteristic values, we averaged the characteristic values of the successful queries for each planner. The standard deviations were all relatively small, indicating that these averages are fairly good choices.

The values for the various planners determined by our experiments for the CONFINED case are shown in Table 1. We chose to distinguish between OPEN and CONFINED environments because, although similar trends were displayed in the two types of environments, some differences in the magnitudes of the characteristic values were noted. Moreover, programmer expertise can usually be relied upon to classify a query and environment as being OPEN or CONFINED. Thus, when initiating the single shot program, the programmer must specify whether the query involves an OPEN or CONFINED environment.

Our experiments also enable us to study the benefits of the characteristics and the individual planners. To see if any of the characteristics were redundant we calculated the correlations between all the planners. Only the translation proportion and free proportion showed a strong correlation. This indicates that it may be sufficient to consider only one of those characteristics. We consider both characteristics anyway because we believe there could be some environments in which they might differ.

We also calculated the correlation between the query successes for the planners. Little correlation was observed, indicating that none of the planners is completely redundant. In some environments, the RRT and the RWM planners seem to be highly correlated, and in these cases one should probably employ only RWM since it is faster than RRT. Something similar occurs with ITM, RRT and RWM. In this case we should perhaps employ the cheaper ones (ITM and RWM), and only the more expensive (RRT) if it is needed. Representative correlation results can be seen in Table 2.

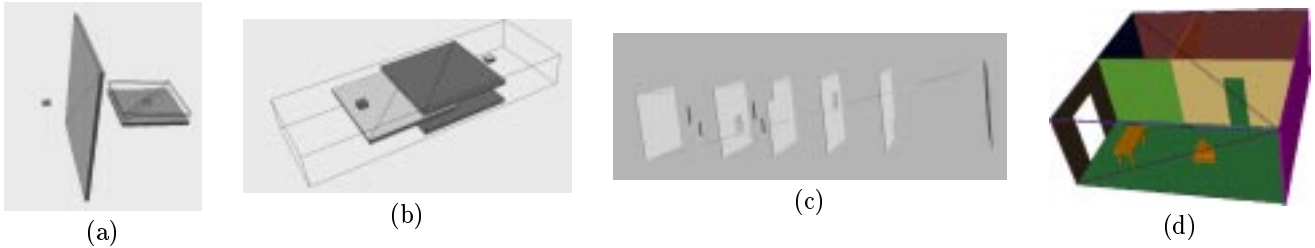


Figure 3: Artificial environments: (a) sandwich-wall, (b) stairs, (c) walls, (d) house-piano.

Alg	Characteristic Values											
	Start				Query			Goal				
	L <sub>1</sub>	L <sub>2</sub>	L <sub>3</sub>	L <sub>4</sub>	G <sub>1</sub>	G <sub>2</sub>	G <sub>3</sub>	L <sub>1</sub>	L <sub>2</sub>	L <sub>3</sub>	L <sub>4</sub>	
LP	.06	.93	.60	.61	.17	.93	.08	.11	.93	.64	.64	
ITM	.06	.94	.58	.58	.17	.82	.17	.10	.94	.65	.64	
IRM	.06	.93	.58	.58	.18	.77	.19	.10	.94	.62	.62	
ACA	.06	.93	.57	.57	.17	.79	.18	.10	.94	.64	.64	
RRT	.06	.93	.57	.57	.17	.83	.16	.09	.94	.65	.65	
RWM	.07	.93	.58	.58	.17	.81	.17	.10	.94	.64	.64	
FIM	.06	.93	.60	.60	.17	.77	.19	.09	.93	.59	.59	
RMM	.06	.94	.56	.55	.30	.39	.30	.08	.94	.56	.55	

Table 1: Planner Characteristic Values (Confined)

	Alg							
	LP	ITM	IRM	ACA	RRT	RWM	FIM	RMM
LP	1	.68	.67	.70	.70	.68	.65	.07
ITM	.68	1	.83	.87	.94	.93	.80	.06
IRM	.67	.83	1	.86	.85	.85	.77	.06
ACA	.70	.87	.86	1	.89	.90	.78	.08
RRT	.70	.94	.85	.89	1	.96	.79	.06
RWM	.68	.93	.85	.90	.96	1	.80	.07
FIM	.65	.80	.77	.78	.79	.80	1	.01
RMM	.07	.06	.06	.08	.06	.07	.01	1

Table 2: Planner Correlation Matrix (Confined)

## 5 Experimental Results

In this section we examine the performance of our prototype implementation of our single shot planner. We consider queries in both artificial environments and in complex mechanical CAD models. We compare the single shot planner with an obstacle-based probabilistic roadmap method (OBPRM), and with the versions of the rapidly-exploring random tree method (RRT) and the Ariadne’s Clew Algorithm (ACA) that we implemented as subroutines in our algorithm bank (see Section 3).

Artificial Environments (Articulated)			
Env.	Method	Time (s)	Solved
Sandwich Wall	OBPRM	549	NO
	RRT	352	NO
	ACA	196	yes
	SS:(ACA)	196	yes
Stairs	OBPRM	1,472	NO
	RRT	636	yes
	ACA	285	yes
	SS:(IRM,IRM,ITM,RWM,ACA)	302	yes
Walls	OBPRM	1,903	NO
	RRT	1,236	yes
	ACA	468	yes
	SS:(LP,ACA)	517	yes

Table 4: Artificial Environment Results (Articulated)

Artificial Environments (Rigid Bodies)			
Env/Robot	Method	Time (s)	Solved
Sandwich Wall/block	OBPRM	10	yes
	RRT	864	NO
	ACA	946	NO
	SS:(ACA,RRT,RMM)	136	yes
Stairs/block	OBPRM	3,865	yes
	RRT	7,893	NO
	ACA	19,419	NO
	SS:(FIM,ACA,IRM,ACA,IRM,ITM,FIM,FIM)	3,061	yes
Walls/Stick	OBPRM	446	yes
	RRT	421	yes
	ACA	8	yes
	SS:(LP,RRT,ACA)	78	yes
House/Piano	OBPRM	70	yes
	RRT	2,014	NO
	ACA	16	yes
	SS:(IRM,RMM)	46	yes

Table 3: Artificial Environment Results

From Table 3, Table 4, and Table 5 we can see that the single shot system works well for both rigid and

articulated robots in the artificial environments, and rigid robots in the CAD environments. The framework generally selected algorithms appropriate for the query, and the various planners do seem to work together to construct solutions that are competitive with the best of the individual methods. While there are cases when other methods outperform the single shot system, our performance is generally good. This indicates that the single shot system tends to avoid the pathological behavior of any of the individual methods.

## 6 Conclusion and Future Work

We have described a framework which enables multiple algorithms to work together to solve a query, and

CAD Environments			
Env.	Method	Time (s)	Solved
Alpha1.5	OBPRM	3,495	yes
	RRT	5,555	NO
	ACA	1,699	yes
	SS:(ACA)	1,699	yes
Alpha1.2	OBPRM	94,150	NO
	RRT	49,455	yes
	ACA	83,254	NO
	SS:(ACA,ACA,ITM,IRM)	68,934	yes
Flange.85	RRT	17,275	NO
	ACA	233	yes
	SS:(ACA)	233	yes

Table 5: CAD Environment Results

have shown that this yields better results on average than any algorithm alone. In this paper, we have extended our previous work [22] by showing that an automatic characterization of the planning algorithms can be used to effectively match algorithms with particular queries. We have also proposed a methodology for adaptively tuning these values so that the system can evolve over time for particular classes of applications. In addition, we have shown the generality of our approach by applying it to various types of environments and robots. There are still many interesting issues to be studied. For example, determining the best planning algorithms for the algorithm bank – perhaps some current algorithms should be removed, and others might be added. Similar questions exist for the characteristic values used for the queries and algorithms.

## Acknowledgement

The alpha puzzle was designed by Boris Yamrom of the Computer Graphics & Systems Group at GE’s Corporate Research & Development Center. GE also provided us with the Flange environment.

## References

- [1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 630–637, 1998.
- [2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 155–168, 1998.
- [3] N. M. Amato, C. V. Jones, and D. Vallejo. An adaptive framework for ‘single shot’ motion planning. Technical Report 98-025, Dept. of Computer Science, Texas A&M University, Nov 1998.
- [4] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 113–120, 1996.
- [5] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, 1991.
- [6] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 529–536, 2000.
- [7] P. Bessiere, J. M. Ahuactzin, E.-G. Talbi, and E. Mazer. The ariadne’s clew algorithm: Global planning with local methods. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, volume 2, pages 1373–1380, 1993.
- [8] J. Canny and M. C. Lin. An opportunistic global path planner. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 39–48, 1990.
- [9] S. Caselli and M. Reggiani. ERPP: An Experience-based Randomized Path Planner. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1002–1008, 2000.
- [10] H. Chang and T. Y. Li. Assembly maintainability study with motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1012–1019, 1995.
- [11] P. C. Chen and Y. K. Hwang. SANDROS: A dynamic graph search algorithm for motion planning. *IEEE Trans. Robot. Automat.*, 14(3):390–403, 1998.
- [12] B. Glavina. Solving findpath by combination of directed and randomized search. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1718–1723, 1990.
- [13] S. Gottschalk, M.C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. Technical Report TR96-013, University of N. Carolina, Chapel Hill, CA, 1996.
- [14] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expensive configuration spaces. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2719–2726, 1997.
- [15] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.
- [16] K. Kondo. Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free space enumeration. *IEEE Trans. Robot. Automat.*, 7(3):267–277, 1991.
- [17] J. J. Kuffner and S. M. LaValle. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 995–1001, 2000.
- [18] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.
- [19] B. Mirtich. V-clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electric Research Lab, Cambridge, MA, 1997.
- [20] M. Overmars and P. Svestka. A probabilistic learning approach to motion planning. In *Proc. Workshop on Algorithmic Foundations of Robotics*, pages 19–37, 1994.
- [21] D. Vallejo. *An Adaptive Framework for ‘Single Shot’ Motion Planning*. PhD thesis, Dept. of Computer Science, Texas A&M University, December 2000.
- [22] D. Vallejo, C. V. Jones, and N. M. Amato. An adaptive framework for ‘single shot’ motion planning. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2000. To appear.
- [23] P. Xavier. Fast swept-volume distance for robust collision detection. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1162–1169, 1997.
- [24] P. G. Xavier and R. A. LaFarge. A configuration space toolkit for automated spatial reasoning: Technical results and ldrd project final report. Technical Report SAND97-0366, Sandia National Laboratories, 1997.