C-space Subdivision and Integration in Feature-Sensitive Motion Planning*

Marco A. Morales A., Lydia Tapia, Roger Pearce, Samuel Rodriguez, Nancy M. Amato
Parasol Laboratory
Department of Computer Science
Texas A&M University
College Station, Texas, 77843-3112, USA
{*marcom,ltapia,rap2317,sor8786,amato*}*@cs.tamu.edu*

***Abstract*— There are many randomized motion planning techniques, but it is often difficult to determine what planning method to apply to best solve a problem. Planners have their own strengths and weaknesses, and each one is best suited to a specific type of problem. In previous work, we proposed a meta-planner that, through analysis of the problem features, subdivides the instance into regions and determines which planner to apply in each region. The results obtained with our prototype system were very promising even though it utilized simplistic strategies for all components. Even so, we did determine that strategies for problem subdivision and for combination of partial regional solutions have a crucial impact on performance. In this paper, we propose new methods for these steps to improve the performance of the meta-planner. For problem subdivision, we propose two new methods: a method based on 'gaps' and a method based on information theory. For combining partial solutions, we propose two new methods that concentrate on neighboring areas of the regional solutions. We present results that show the performance gain achieved by utilizing these new strategies.**

I. Introduction

The *motion planning* (MP) problem is to find a sequence of valid states taking a movable object, usually called a robot, from an initial state to a goal state. A robot's state or *configuration* is represented by a set of parameters that describe the position and orientation of all robot components; these parameters are often referred to as the robot's *degrees of freedom* (DOF). A configuration is said to be valid if it satisfies every predefined constraint (e.g., in many cases the configuration is valid if it is collision free). Motion planning arises in many applications ranging from robotics and CAD to computational biology.

There is strong evidence that any complete planner will require exponential time in the number of DOF of the robot [6], [19], [20]. Thus, the motion planning problem is considered intractable except for robots with few DOFs. Initially, this complexity was addressed with heuristic methods based on cell decomposition [5] and potential fields [9], but after the introduction of the Randomized Path Planner (RPP) [2], randomized methods have been the focus of extensive research. Notable examples include the roadmap-based probabilistic roadmap methods (PRMs) [8] and several tree-based methods that explore the planning space starting from one or two points. [3], [7], [10]. Randomized methods have yielded remarkable results, including solutions for previously unsolved MP problems.

Although there are many randomized planners, their performance varies depending on their individual strengths and weaknesses in dealing with different types of C-space and on the construction of the problem instance. In addition, many environments have vastly different regions and there may not be any planner that can deal efficiently with all regions.

In our previous work [13], we proposed to follow a machine learning approach in developing a feature-sensitive meta-planner to oversee the coordinated application of multiple planners. It characterized and partitioned C-space into regions that are well-suited for a strategy chosen from a library of roadmap-based motion planners. In each region, the corresponding strategy is applied and the resulting regional roadmaps are combined to form a roadmap of the entire planning space.

In experiments performed, our feature-sensitive meta-planner demonstrated the promise of this approach by outperforming any of the individual planners on a variety of problem instances [13]. However, our results also illustrated some performance bottlenecks. For example, our rather naive subdivision strategy was not truly feature sensitive and therefore did not always result in the most natural subdivisions. Of greatest impact, however, was our straight-forward brute force strategy for combining regional roadmaps into the global roadmap – in many cases this step accounted for the majority of the computation costs!

In this paper, we propose new methods to address these issues. We present methods for subdividing a region into subregions that are more homogeneous than their parent regions. Also, we present mechanisms to reduce the cost of integrating regional solutions without affecting the quality of the overall solution.

* This research supported in part by NSF Grants EIA-0103742, ACR-0081510, ACR-0113971, CCR-0113974, ACI-0326350, by the DOE, and by the Texas Higher Education Coordinating Board grant ATP-000512-0261-2001. Morales supported in part by a Fulbright/Garcia Robles (CONACYT) Fellowship. Rodriguez supported in part by a LSAMP Bridge to Doctorate Fellowship.

II. PRELIMINARIES

A robot is a movable object whose position and orientation can be described by n parameters, or DOFs, each corresponding to a component (e.g., object positions, object orientations, link angles, or link displacements). Hence, a robot's placement, or configuration, can be uniquely described by a point $(x_1, x_2, ..., x_n)$ in an n dimensional space (x_i being the ith DOF). This space, consisting of all possible robot configurations (feasible or not) is called *configuration space* (C-space) [11]. The subset of all feasible configurations is the *free C-space* (C-free), while the union of the unfeasible configurations is the *blocked C-space* (*C-obstacles*). Thus, the MP problem becomes that of finding a continuous trajectory for a point in C-space connecting the start and the goal configurations that completely lies in C-free. In general, it is intractable to compute C-space, but we can often determine whether a configuration is feasible or not quite efficiently, e.g., by performing a collision detection test in the *workspace*, the robot's natural space.

A. Randomized Motion Planning

Since computing C-space is intractable, several planners have successfully exploited randomization to sample and map C-space. Notable examples include Roadmap-based PRMs [8], [15], [16] and tree-based planners [2], [3], [7], [10]. These planners have solved many previously unsolved problems.

PRMs build a roadmap of the free C-space by sampling collision-free configurations. Then, simple *local planners* identify which pairs of selected samples can be connected to form roadmap edges. Many PRM variants have been proposed. Some methods generate configurations uniformly [8], while other methods increase the proportion of feasible configurations in some regions [4], [10], [14], [21]. On the other hand, tree-based planners set the root of a tree at a known feasible configuration and grow from it either uniformly to cover the space reachable from the nodes already in the tree or to get closer to the goal.

None of the planners available performs well for every problem. The characteristics of the problem often dictate the performance of the method applied. For example, OBPRM [1], a PRM where configurations are generated near C-obstacle surfaces, performs better in cluttered regions. On the other hand, the original PRM [8] which uniformly samples configurations from C-space performs best in free regions.

B. Feature-Sensitive Motion Planning Framework

In [13], we proposed a machine learning based characterization and partitioning approach to motion planning. The C-space of the instance is recursively partitioned, so that in each level a region may be subdivided into different overlapping subregions. When a region is classified as homogeneous, based on a set of measured features, a planning strategy that is well suited for the corresponding features is applied to generate a regional roadmap. When all the subregions have been mapped, their roadmaps are combined to produce a roadmap for the entire region. This leaves a global roadmap for the environment at the top of the recursion.

The classification of a region was performed with a trained decision tree [18]. This method requires the collection of descriptive features of each region. During and after the creation of a small PRM roadmap, general features (based on the counts and ratios of nodes, edges, and components) and connected component features (based on the size, span, and distances between components) are measured. The values of these features allow a trained decision tree to classify a region as homogeneous (free, cluttered, or narrow passage) or non-homogeneous.

The performance of this feature-sensitive framework depends on how effectively the problem is partitioned into homogeneous regions. Previously, we applied a technique that selects at random both the positional dimension to subdivide and the point to split the selected dimension. Regions were partitioned leaving an overlap in the neighboring area of 10% to 15%. While this method eventually created homogeneous regions, it fragmented the problem indiscriminately. Also, although descriptive features collected for the space were available, they were not considered when creating the partition.

The most expensive step in our meta-planner was the combination of regional solutions. For this operation, we used a simple brute force approach that performed k-closest connection over all the nodes in the regional roadmaps. As a consequence, in our previous results this step took from 50% to 90% of the collision-detection calls. In this paper, we propose attempting connections between nodes in or close to the overlapping regions of neighboring partitions to reduce the cost of merging regional maps.

III. C-SPACE SUBDIVISION

The Feature-Sensitive Motion Planning Framework creates a C-space subdivision tree. Each node of this tree represents a region of the problem, its child nodes are subregions whose union covers the parent region. The main goal of our partitioning is that the resulting subregions should be more 'homogeneous' than the parent region as measured by their features.

In [13], we applied an elementary subdivision strategy. It split a randomly selected positional DOF at a random point along an orthogonal boundary. The boundaries were extended beyond the splitting point by a small ε to create an overlapping area between the two neighboring subregions. Random subdivisions limit our chances to find proper subregions that are more homogeneous than the parent partition. Also, while characterizing the region we obtained features that can measure homogeneity, but we did not use them when determining a partition. Moreover, considering only positional DOFs prevented the meta-planner from

finding possibly useful subdivisions in higher degrees of freedom. In this work, we propose subdivision strategies that use regional features in defining partitions that increase the homogeneity of the subregions so the recursion can be stopped sooner. In addition, the proposed strategies make partitions considering every DOF.

The partitioning strategies presented utilize features measured from the small PRM roadmap generated to characterize the region. Valid and invalid samples determine the selection of the best DOF and point within the selected DOF to place an orthogonal boundary. Boundaries are extended by some ε that is a small fraction of the region in order to make neighboring regions overlap. Since the resulting subregions are orthogonal, they may be sensitive to alignment. This is an issue we leave for further studies.

A. Subdivision based on gaps

The gap-based subdivision searches for gaps, areas along a DOF where no free nodes could be found. The largest normalized gap is selected as a boundary between subregions as shown in Fig. 1. This largest gap is likely to represent the largest region of homogeneity, an obstacle-prone, hard-to-map area. Two subregions are defined using the gap, one covering the area to the right of the gap and one including the area left of the gap and the gap itself. Normalization is needed to give every DOF equal chances of being considered for partition, so, we divide each gap by the maximum range of its corresponding DOF.

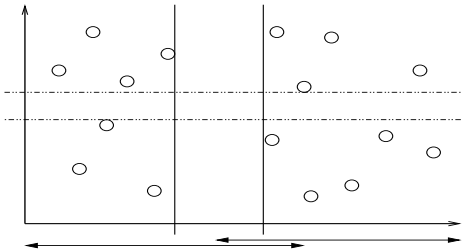


Fig. 1. Two DOF example: Largest gap (x axis) is split in two overlapping pieces, a smaller gap (y axis) is not split yet

B. Subdivision based on information gain

We present a subdivision strategy inspired by decision-tree-based machine learning algorithms, such as ID3 [17] and C4.5 [18], that construct the series of decisions that best leads to a certain classification from a dataset of examples. Branches in the learned tree are attribute-value tests found to best partition the examples. During construction of the tree, entropy (a property of the dataset) is used to compute the information gain of various partitions [12].

Our feature-sensitive meta-planner also builds a tree of subregions. Rather than learning a tree for classification, we are defining partitions that best create homogeneous subdivisions. Information gain can be computed for regional prospective partitions to measure which partition best separates the sampled configurations into homogeneous subregions. Homogeneity can be calculated using C-space roadmap features already required for region classification [13]. We use valid and invalid configurations to measure the homogeneity of a C-space region.

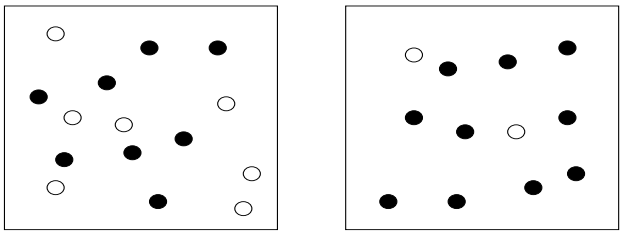
Information gain is defined in terms of *entropy* which measures the diversity of a set. The entropy of a set S of examples relative to a c-wise classification that has a proportion of p_i examples in the class i is defined as

$$Ent(S) = \sum_{k=1}^{c} -p_k \log_2 p_k \quad (1)$$

where $p_k \log_2 p_k = 0$ if $p_k = 0$. In order to classify valid or invalid configurations, $c = 2$ and the entropy of a set of samples S_r from a region r is defined as

$$Ent(S_r) = -p_v \log_2 p_v - p_i \log_2 p_i \quad (2)$$

where p_v is the proportion of valid configurations in S, and p_i is the proportion of invalid configurations in S. Intuitively, a region has higher entropy when the proportion of valid and invalid configurations is similar, it has lower entropy when the configurations tend to be more of one value as illustrated in Fig. 2.



(a) (b)

Fig. 2. (a) high entropy set: similar number of black and white elements; (b) low entropy set: most elements are of one type

We use entropy to compute the information gain obtained when subdividing each DOF D into two pieces, so we can choose the subdivision that yields the highest gain. Algorithm 1 shows our approach. We project the valid and invalid configurations on each DOF and find the middle points between a valid and an invalid configuration. These middle points are candidate splitting points. The information gain of splitting the region S in parameter D through the point m into two subregions $S_{D,i}$ is

$$Gain(S, D, m) = Ent(S) - \sum_{i=1}^{2} \frac{|S_{D,i}|}{|S|} Ent(S_{D,i}) \quad (3)$$

where $|S_{D,1}|$ is the total number of samples to the left of m in dimension D and $\|S_{D,2}\|$ is the total number of samples to the right of m in dimension D.

IV. Combination of Regional Roadmaps

Once every child of a node in the C-space subdivision tree has been mapped, regional roadmaps need to be

Algorithm 1 Information Gain Subdivision

Require: bounding box bb, valid/invalid samples (s_v, s_i)
Ensure: overlapping boxes leftBB, rightBB

split_DOF = NULL, split=NULL
Calculate entropy, E_s, of inital samples
for all DOF d_i **do**
  for all possible split points **do**
    Define the set of valid (l_v) and invalid (l_i) samples left of the split point
    Define the set of valid (r_v) and invalid (r_i) samples right of the split point
    gain = InformationGain(s_v,s_i,E_s,l_v,l_i,r_v,r_i)
    if The gain of this split is greatest **then**
      Save the split_DOF and split point
    end if
  end for
end for
ε = findEpsilon(bb, split_DOF,start,split,end)
leftBB = subregion(bb, split_DOF, start, split-ε)
rightBB = subregion(bb, split_DOF, split+ε, end)
Return leftBB, rightBB

combined into a global roadmap for the parent node. Neighboring child nodes have an overlap that is used to "stitch" their corresponding roadmaps together.

In our previous work [13] we integrated neighboring regions by applying a component connection and a k-closest connection over all nodes. Attempting connections in regions that have already been mapped results in excessive mapping effort while yielding little benefit. In fact, combination of regional roadmaps was the bottleneck of our original framework – it generally took from 50% to 90% of the collision-detection calls. Instead, what we need is to focus connection attempts in and near to overlapping sections of the subregions that we hope to combine.

The flexibility of our framework allows for many combination strategies. Here, we propose both an alternative approach to our initial method that is applied in neighboring regions and a method to combine the feature roadmap of a node with its regional roadmap.

A. Combination of neighboring region roadmaps

Neighboring regions can be combined by connecting selected pairs of nodes from the overlap and its surroundings. To merge two roadmaps, left and right, that overlap in the range [a,b] from a given DOF, we form two lists of nodes: leftNodes and rightNodes. Connection attempts are done with regular local planners between leftNodes and rightNodes. We introduce two mechanisms to fill out these lists.

The first mechanism, *overlap*, consists on filling leftNodes with all nodes from the left roadmap in the range [a,b]. Similarly, rightNodes is filled with all nodes from the right roadmap in the same range.

The second mechanism, *overlap+random*, applies the *overlap* method and then additional nodes from the left (right) roadmap are added to leftNodes (rightNodes) with some probability using a Gaussian distribution biased to nodes close to the overlap (Algorithm 2).

Algorithm 2 Roadmap Combination: Overlap + Random

Require: leftBB, leftRdmp, rightBB, rightRdmp, k
Ensure: leftRdmp and rightRdmp combined in roadmap

for all nodes n_i in leftRdmp **do**
  if n_i is in overlap between leftBB and rightBB **then**
    leftNodes.add(n_i)
  else if accept given GaussianLikelihood(n_i) **then**
    leftNodes.add(n_i)
  end if
end for
for all nodes n_i in rightRdmp **do**
  if n_i is in overlap between leftBB and rightBB **then**
    rightNodes.add(n_i)
  else if accept given GaussianLikelihood(n_i) **then**
    rightNodes.add(n_i)
  end if
end for
Attempt connections between leftNodes & rightNodes
roadmap = leftRdmp + rightRdmp + connections
Return roadmap

B. Integration of feature and regional roadmaps

The feature roadmap produced for characterizing a region can also be used to improve the regional roadmap. Since the feature roadmap has a small number of nodes, in this work we connect it to the regional roadmap by trying connections from all the nodes in the feature roadmap to the k-closest nodes of the regional solution.

V. Experiments

We compared the performance of the different subdivision and combination methods proposed against the ones previously used when mapping two environments: Plates (Fig. 3) and Walls (Fig. 4). The meta-planner was coded in C++ and Perl, using the Parasol motion planning library, our group's collection of randomized planners. Results were obtained on an Intel Pentium 4 2.8 GHz processor with 512 MB of RAM and 512 KB of cache, running Linux 2.4.20-9.

The Plates environment forms several narrow openings with barriers as shown in Fig. 3. It has two sets of parallel vertical plates (sandwiches) and a set of six parallel horizontal plates that form narrow passages. The robot is a rigid body with six DOFs. We set a query where the robot moves from inside one of the sandwiches to the other.

The Walls environment is formed by six chambers separated by small openings as shown in Fig. 4. The seven-DOF robot has two links connected by a single joint. We

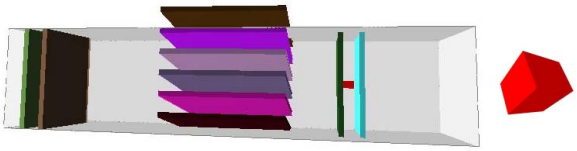


Fig. 3. Plates environment, robot shown enlarged on right

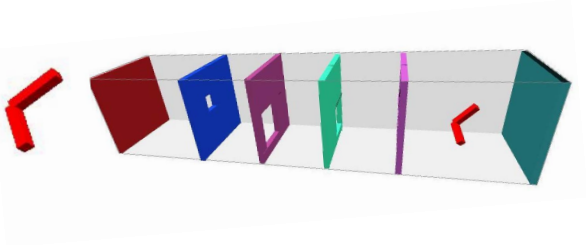


Fig. 4. Walls environment, robot shown enlarged on left

set a query where the robot moves from a chamber on one extreme to the chamber on the other extreme of the environment. The size of the robot forces it to fold and unfold as it goes through the openings.

We measure performance and map quality by counting the number of collision detection calls (#CDs), the number of nodes attempted (att) and obtained (act), and the number of connected components (#CCs) in each stage of the meta-planning: partitioning, mapping of regions, and combination of roadmaps.

A. Subdivision methods

First, we tested the subdivision methods proposed. In order to isolate the analysis of subdivisions, we used a basic combination method in these tests. A summary of representative results is shown in Table I. We present results for our previously introduced basic Random subdivision in three DOFs and for subdivisions based on gaps and on information gain. In all but one case, Gap and Gain offer performance improvements from 41% to 70% in comparison with the Random subdivision.

The subregions formed by the Gap method are more expensive to combine. For example, in the one case, Walls, where the Gap method does not show improvement, it created many subdivisions, requiring more effort to combine them. Also, in both environments, the Gain method produced subregions that are faster to map, but with an increased cost in partitioning over the Random method. Moreover, our information Gain technique produces roadmaps with much fewer nodes than the other two methods. Figure 5 shows examples of subdivisions.

B. Roadmap combination methods

We tested the subregion combination methods applying only the basic Random subdivision. A summary of representative results is shown in Table II. All the runs solved

TABLE I
SUBDIVISION RESULTS

Method		Nodes act/att	#CDs	#CCs
Environment: Plates. Robot: 6 DOF rigid body				
Random 3-D	totals	583/825	2023513	19
	partitioning	301/465	692373	50
	mapping	282/360	551140	55
	combination	na	780000	19
Gap	totals	1175/1500	1205003	10
	partitioning	551/750	463937	52
	mapping	624/750	109261	25
	combination	na	631805	10
Gain	totals	212/340	1343651	2
	partitioning	110/190	678228	42
	mapping	102/150	79218	15
	combination	na	586205	2
Environment: Walls. Robot: 7 DOF serial				
Method		**Nodes act/att**	**#CDs**	**#CCs**
Random 3-D	totals	612/900	1497292	1
	partitioning	211/300	84313	23
	mapping	401/600	28996	12
	combination	na	1383983	1
Gap	totals	945/1950	2567767	1
	partitioning	787/1550	887565	151
	mapping	158/400	31598	23
	combination	na	1648604	1
Gain	totals	1360/2300	456012	2
	partitioning	336/500	239421	56
	mapping	1024/1800	81353	28
	combination	na	135238	2

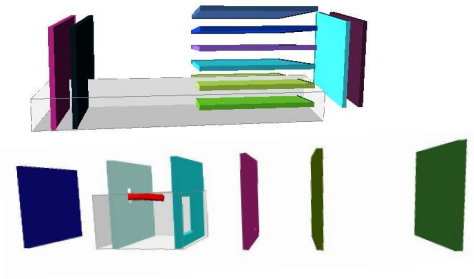


Fig. 5. Subdivision examples

the query assigned. The basic combination consumes an excessive number of collision detection calls, as we noted in our previous work. The *overlap* and *overlap+random* methods yielded similar and significant improvement in performance (based on #CDs), but roadmaps with fewer components are produced with *overlap+random*. In addition, the two proposed strategies reduce the CD calls in region combination while having similar costs during subdivision and mapping of regions.

VI. CONCLUSIONS

We introduced techniques to improve feature-sensitive motion planning. Our early work on this topic produced a framework based on machine learning that subdivides

TABLE II
INTEGRATION RESULTS

Method		Nodes act/att	#CDs	#CCs
Environment: Plates. Robot: 6 DOF rigid body				
Brute Force	totals	1415/2250	3451872	11
	partitioning	991/1550	850855	72
	mapping	424/700	127826	34
	combination	na	2473191	11
Overlap	totals	1241/1700	1702085	107
	partitioning	508/775	983463	66
	mapping	734/925	210365	177
	combination	na	508257	107
Overlap + Random	totals	824/1275	2277185	16
	partitioning	532/775	1309900	71
	mapping	292/500	142819	24
	combination	na	824466	16
Environment: Walls. Robot: 7 DOF serial				
Method		**Nodes act/att**	**#CDs**	**#CCs**
Brute Force	totals	612/900	1497292	1
	partitioning	211/300	84313	23
	mapping	401/600	28996	12
	combination	na	1383983	1
Overlap	totals	876/1350	602796	4
	partitioning	302/450	182768	36
	mapping	574/900	52640	19
	combination	na	367388	4
Overlap + Random	totals	620/900	575863	2
	partitioning	211/300	84313	23
	mapping	409/600	28863	12
	combination	na	462687	2

a problem into regions which are then mapped with a randomized planner determined by the features of the region. Here, we proposed methods for subdivision of the problem that consider some features previously obtained. This information is used to generate subregions that are more homogeneous than the original region and that are easier to classify. We also proposed more efficient integration methods that do not waste effort in mapping areas that specialized planners have already mapped.

Our results show that our subdivisions based on gaps and on information gain produce improvements up to 70% in comparison with our earlier basic random subdivision. Although Gap produced solutions with fewer collision detection calls, it produces subdivisions that are more expensive to map. The cost of partitioning with information gain is slightly higher than Gap, but it produces roadmaps with less effort and with fewer nodes. This demonstrates that using information already available from the characterization to determine the partition location increases the quality of subregions.

The new methods we proposed to combine regional roadmaps demonstrated significant performance improvements over our previous naive combination. Connections that are focused on overlapping sections of neighboring regions save many unnecessary collision detection calls.

REFERENCES

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, pages 155–168, Natick, MA, 1998. A.K. Peters. Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[2] J. Barraquand and J. C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, 1991.

[3] 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.

[4] R. Bohlin and L. E. Kavraki. Path planning using Lazy PRM. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 521–528, 2000.

[5] R. A. Brooks and T. Lozano-Pérez. A subdivision algorithm in configuration space for findpath with rotation. In *Proc. Int. Conf. Artif. Intel.*, pages 799–806, 1983.

[6] J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[7] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages SA1–SA18, 2000.

[8] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[9] O. Khatib. Real–time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.*, 5(1):90–98, 1986.

[10] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[11] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[12] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[13] Marco Morales, Lydia Tapia, Roger Pearce, Sam Rodriguez, and Nancy M. Amato. A machine learning approach for feature-sensitive motion planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, Utrecht/Zeist, The Netherlands, July 2004.

[14] C. L. Nielsen and L. E. Kavraki. A two level fuzzy PRM for manipulation planning. Technical Report TR2000-365, Computer Science, Rice University, Houston, TX, 2000.

[15] M. Overmars. A random approach to path planning. Technical Report RUU-CS-92-32, Computer Science, Utrecht University, The Netherlands, 1992.

[16] M. Overmars and P. Svestka. A probabilistic learning approach to motion planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 19–37, 1994.

[17] J.R. Quinlan. Induction of decision trees. In *Machine Learning*, volume 1, pages 81–106, 1986.

[18] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[19] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.

[20] J. T. Schwartz and M. Sharir. On the "piano movers" problem II: General techniques for computing topological properties of real algebraic manifolds. *Adv. Appl. Math.*, 4:298–351, 1983.

[21] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1024–1031, 1999.