C-space Subdivision and Integration in Feature Sensitive Motion Planning

Marco Morales, Lydia Tapia, Roger Pearce, Samuel Rodriguez, and Nancy M. Amato
{marcom,ltapia,sor8786,rap2317,amato}@cs.tamu.edu

Technical Report TR04-004
Parasol Lab
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

September 20, 2004

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 developed a prototype meta-planner that, through analysis of the problem features, subdivides it into regions and determines which planner to apply in which region of the problem. The results obtained with our prototype system were very promising even though it utilized simplistic strategies for all components. Even so, we did identify that problem subdivision and partial regional solution integration strategies are crucial to 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 integrating partial solutions, we propose a new method that concentrates attention on integrating neighboring areas of the regional solutions. We present results that show the performance gain achieved by utilizing these new strategies.

1 Introduction

The *motion planning* (MP) problem is that of finding a sequence of valid states for a movable object, usually called a robot, to get from an initial to a goal state. A robot's state or *configuration* is a set of parameters that describe the position, orientation, and any linkages associated with the robot. A configuration is said to be valid if it satisfies all constraints given in the MP problem (e.g., in many cases the configuration is valid if it is collision free). Many applications address the problem of path planning from robotics and CAD to computational biology.

There is strong evidence that any complete planner will require exponential time in the number of *degrees of freedom* (DOF) of the robot [20, 21, 6]. Thus, the motion planning problem is thought to be intractable except for robots with few DOFs. Initially, heuristic methods based on cell decomposition [5] and potential fields [9] were explored to address this complexity but after the introduction of the Randomized Path Planner (RPP) [2] randomized methods have been the focus of extensive research. 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, 10, 7] are notable examples. Remarkable results, including solutions for previously unsolved MP problems, have been obtained with randomized methods.

Although there are many randomized planners, their performance varies depending on their individual strengths and weaknesses and on the construction of the problem instance to solve. Some planners are best suited for problems with few obstacles while others show their real strengths in cluttered areas. In addition, many environments have vastly different regions, so there may not be any planner that can deal efficiently with all the pieces of the problem.

In our previous work [14], we proposed a meta-planner for motion planning that would oversee the coordinated application of multiple planners. We used a machine learning approach to characterize and partition C-space into regions that were suited to one of the methods in a library of roadmap-based motion planners. After the best-suited method was applied in each region, the resulting regional roadmaps were combined to form a roadmap of the entire planning space.

Our prototype meta-planner for feature-sensitive motion planning demonstrated the promise of this approach by outperforming any of the individual planners on a variety of problem instances [14]. However, our results also illustrated some performance bottlenecks in the prototype system. For example, our rather naive subdivision strategy was not actually feature sensitive and hence did not always result in the most natural subdivisions. Of greatest impact, however, was our straight-forward brute force strategy for integrating regional roadmaps – in many cases this step accounted for the majority of the computation costs!

In this paper, we propose new methods aimed at these issues. We describe methods for subdividing a region into subproblems so that resulting subregions are more homogeneous than the regions that results from our original strategy. We also propose mechanisms to reduce the cost of integrating regional solutions without affecting the overall quality of the overall solution.

2 Preliminaries

A robot is a movable object that can be controlled through n parameters or *degrees of freedom*, each corresponding to a unique component (e.g., object positions, object orientations, link angles, or link displacements). 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 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-obstacle*). 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. And, although it is intractable to compute C-space, 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.

Randomized Motion Planning Algorithms. Since computing C-space is intractable, randomization has been successfully applied in a number of planners that probabilistically sample and map C-space. The roadmap-based PRMs [8, 16, 17] and several tree-based planners [2, 3, 10, 7] stand out for their ability to solve many previously unsolved problems.

PRMs make a roadmap of the free C-space. First, collision-free configurations are found to make up the vertices of the roadmap. Then, pairs of vertices are selected and simple *local planners* identify which nodes can be connected to form roadmap edges.

There are many PRM variants based on different heuristics. Some methods generate configurations uniformly for a problem instance [8], while other meth-

ods increase the proportion of feasible configurations in some regions [15, 4, 22, 10].

Tree-based planners grow a tree starting with a known feasible configuration and make it grow to either uniformly cover the space reachable from the nodes already in the tree or to get closer to the goal. Roadmap components can be connected using (combinations of) roadmap-based and tree-based planners [13].

Although there are many variants of randomized planners, no single variant 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 generated on or near C-obstacle surfaces, performs better in cluttered regions. On the other hand, Basic PRM [8], a PRM where configurations are generated through uniform random sampling, performs best in free regions.

Feature-Sensitive Motion Planning Framework. In [14], we proposed a machine learning based characterization and partitioning approach to motion planning problems. The C-space of the instance is recursively partitioned, so that in each level a region may be subdivided into different overlapping subregions. Subdivisions stop when a region is classified as homogeneous (based on a set of features measured for each region). Then, the best suited planner available is applied to find a regional solution. When all the subregions have been mapped, their roadmaps are combined to produce a regional roadmap. At the end of this process we have a global roadmap for the environment.

The classification of a region was performed with the use of a trained decision tree. This method requires the collection of descriptive features of each region. We use 21 different features that are easy to collect during and after the creation of a small PRM roadmap. Based on the values of these features, the trained decision tree classifies a region as homogeneous (free, cluttered, or narrow passageway) or non-homogeneous.

The feature-sensitive framework depends on the effectiveness of the partitioning into homogeneous regions. Previously, we applied a technique that selects both the positional dimension to subdivide and the point to split the selected dimension at random. Regions were partitioned so that they overlap in 10% to 15% of their neighboring area. While this method eventually created homogeneous regions, it created numerous regions. Also, while the classification was performed using descriptive features collected for the space, this information, although available, was not considered when creating and selecting the partition.

The most expensive step in our prototype system was the integration 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 without utilizing any proximity information. As a consequence, shown in our previous results, this step took from 50 to 90 percent of the total collision-detection calls. In this paper we propose attempting connections between nodes in or close to the overlapping regions of neighboring partitions during the integration step to reduce the cost of merging regional maps.

3 C-space Subdivision

In the Feature-Sensitive Motion Planning Framework we create a C-space subdivision tree. Each node of this tree represents a region of the problem, and each child node is a subregion. The union of the child nodes covers the parent region.

The main goal of our partitioning is that resulting subdivisions are more homogeneous than the parent region. Thus, we need a proper mechanism to find the best place to separate distinct subregions.

In our previous work [14], we used an elementary subdivision strategy that uses a random splitting point from a randomly selected positional DOF so to define an orthogonal boundary to the selected DOF. The boundaries are extended beyond the splitting point by a small ε to leave an overlapping area covered by the two neighboring partitions.

Random subdivisions limit our chances to find proper subdivisions that are more homogeneous than the parent partition. Also, while characterizing the region we obtain features that can measure homogeneity, but we do not use them when selecting a partition. Moreover, considering only positional DOFs limits our framework, because we will miss any subdivision in higher degrees of freedom. Here we propose strategies that use knowledge available about the region, and make it easier for the characterization to find a homogeneous class so that the right planners can be matched faster. In addition, our strategies consider every DOFs when partitioning a region.

All our partitioning strategies follow the same basic approach. During region characterization we generate a small PRM roadmap to classify the region. Valid and invalid configurations used to create this roadmap are used to determine what is the best DOF to subdivide and the best splitting point within that DOF. Subregions overlap by some value ε that is a small fraction of the region. Since the strategies proposed

here make orthogonal subdivisions, they can be sensitive to alignment. We leave for future studies the exploration of other subdivision strategies that address the alignment problem.

Random subdivision. We extend our random subdivision strategy that only made partitions on the positional DOFs to consider all the DOFs as candidates for subdivision.

Subdivision based on gaps The gap-based subdivision (Algorithm 1) searches for gaps, areas along a DOF where no free nodes could be found during feature gathering. The largest normalized gap is selected to base the partitions 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.

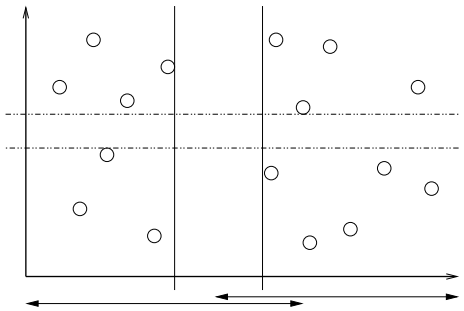

Figure 1: Two DOF example: Largest gap (x axis) is split in two overlapping pieces, a smaller gap (y axis) is not split yet

Normalization is needed to give every DOF equal chances of being considered for partition. We divide each gap by the maximum range of its corresponding DOF.

Subdivision based on information gain

Our subdivision based on information gain is inspired by decision tree based machine learning algorithms, such as ID3 [18] and C4.5 [19], that construct from a dataset of examples the series of decisions that best leads to a certain classification. Branches in the learned tree are attribute-value tests found to best partition the examples. During construction of the tree, a property of the dataset, entropy, is used to compute the information gain of various partitions [12].

Our feature-sensitive meta-planner also builds a tree, but a tree of subregions. Rather than learning a tree for classification, we need to find the best partition within a DOF that creates homogeneous subdivisions. Information gain can be computed for regional prospective partitions to measure which partition best separates the sampled configurations into homogeneous subregions. In this case, homogeneity can be calculated using C-space roadmap features already required for region classification [14]. Because it is very telling of the characteristic of the space, we use valid and invalid configurations to measure the homogeneity of a C-space region.

Algorithm 1 Subdivide Gap

Require: bounding box bb, valid samples s_v
Ensure: overlapping boxes leftBB, rightBB
1: split_DOF = NULL, split=NULL, largest_gap=0
2: **for all** DOF d_i **do**
3: values = sort(project s_v on d_i)
4: (gap,right_val) = findMaxGap(values)
5: **if** gap > largest_gap **then**
6: largest_gap=gap, split_DOF= i, split = right_val
7: **end if**
8: **end for**
9: ε = findEpsilon(bb, split_DOF,start,split,end)
10: leftBB = subregion(bb, split_DOF, start, split-ε)
11: rightBB = subregion(bb, split_DOF, split+ε, end)
12: **Return** leftBB,rightBB

Information gain is defined in terms of *entropy* which measures the diversity of a set of examples. 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

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

where $p_k \log_2 p_k = 0$ if $p_k = 0$.

In our case $c = 2$, and our examples are either valid or invalid configurations, so entropy of a set of samples S from a region r is defined as

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

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 more to be of one value as illustrated in Fig. 2.

We use entropy to determine what is the best parameter to subdivide. This can be done by obtaining the information gain of each DOF D after subdividing D into two pieces. 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. We evaluate the information gain of splitting in D into two

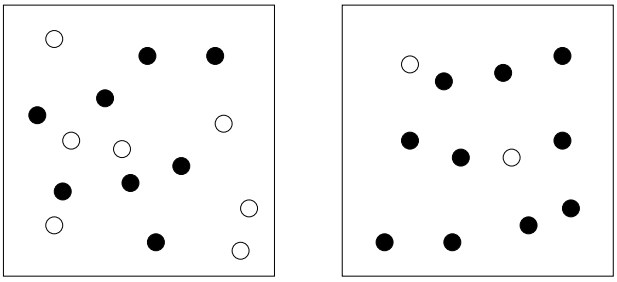

Figure 2: (left) high entropy set: similar number of black and white elements; (right) low entropy set: most elements are of one type

subregions $S_{D,i}$ through the point m with

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

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

The subdivision that gives the largest gain is applied as shown in Algorithm 2.

Algorithm 2 Subdivide Gain

Require: bounding box bb, valid samples s_v, invalid samples s_i
Ensure: overlapping boxes leftBB, rightBB
split_DOF = NULL, split=NULL, best_gain=0
E_s = Ent($|s_v|$,$|s_i|$), $t_s = |s_v| + |s_i|$
for all DOF d_i **do**
  splits = d_i points between valid-invalid samples
  for $j = 0$; j ¡ splits.size(); j++ **do**
    $l_v = |s_v$.subregion(bb,d_i,start,splits[j])|
    $l_i = |s_i$.subregion(bb,d_i,start,splits[j])|
    $r_v = |s_v| - l_v$, $r_i = |s_i| - l_i$
    gain = E_s
    gain = gain - $[(l_v + l_i)/t_s]$*Ent(l_v, l_i)
    gain = gain - $[(r_v + r_i)/t_s]$*Ent(r_v, r_i)
    if gain > best_gain **then**
      split_DOF = i, split = splits[j]
      best_gain=gain
    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

4 Roadmap Integration

Once every child of a node in the C-space subdivision tree has been mapped, partial solutions need to be integrated into a global solution for the parent node. Neighboring child nodes have an overlap region that is used to "stitch" their corresponding roadmaps.

In our previous work [14] we integrated neighboring regions by applying a component connection and a k-closest connection over all nodes. This strategy is not ideal because it will attempt connections in regions that have been mapped, resulting in excessive mapping effort while integrating partial solutions. In fact, the integration of regional solutions was the bottleneck of our original framework that in our experiments took from 50 to 90 percent of the total collision-detection calls. What we need is to focus connection attempts in and near to overlapping sections of the subregions to integrate.

The flexibility of our framework allows for many integration strategies. Here, we propose 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.

Integration of neighboring regions. Neighboring regions can be connected by selecting pairs of nodes from the overlap and its surroundings. When we 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 from leftNodes to rightNodes and vice-versa. We propose two mechanisms to fill out these lists.

The first mechanisms, *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 that is biased to nodes close to the overlap (Algorithm 3).

Integration of feature and regional roadmaps. When we obtain features for a node, we produce a feature roadmap that should be integrated with the node solution. Since the feature roadmap has a small number of nodes, it is connected to the regional roadmap by trying connections from all the nodes in the feature roadmap to the k closest nodes of the regional solution. This way no time is wasted in mapping areas that are already well mapped.

Algorithm 3 Roadmap Integration: Overlap + Random

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

```
1: for all nodes n_i in leftRdmp do
2:   if n_i in overlap(leftBB,rightBB) then
3:     leftNodes.add(n_i)
4:   else if n_i satisfyGaussianDistribution() then
5:     leftNodes.add(n_i)
6:   end if
7: end for
8: for all nodes n_i in rightRdmp do
9:   if n_i in overlap(leftBB,rightBB) then
10:    rightNodes.add(n_i)
11:  else if n_i satisfyGaussianDistribution() then
12:    rightNodes.add(n_i)
13:  end if
14: end for
15: roadmap = mergeMaps(leftRdmp, rightRdmp)
16: roadmap.connectPairs(leftNodes, rightNodes)
17: Return roadmap
```

5 Experiments

Our experiments compare the performance of the different subdivision and integration methods proposed here and the ones previously used. Our meta-planner is coded in C++ and Perl, it uses the Parasol motion planning library, our group's collection of randomized planners. Our 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 OS.

We performed experiments in an environment of barriers, Plates, that form several narrow openings as shown in Fig. 3. The robot is a rigid body with six DOFs. The environment has two sets of plates forming a sandwich and a set of six plates that form narrow passages where the robot must maneuver between. To test roadmaps we set a query where the robot moves from inside one of the sandwiches to the other.

We also tested an environment with a linked robot with 7 DOFs (Fig. 4), Walls. It is formed by 6 chambers

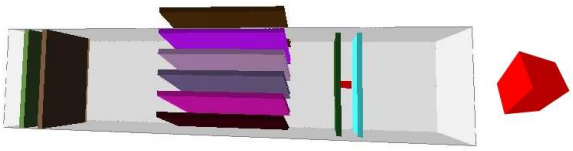

Figure 3: Plates environment, robot enlarged on right

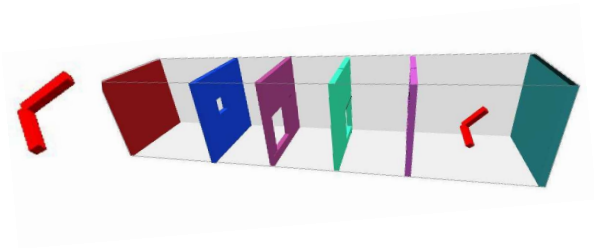

Figure 4: Walls environment, robot enlarged on left

separated by small openings. We 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 while solving the query.

We measure performance and map quality by counting the number of collision detection calls (#CDs), number of nodes attempted (att) and obtained (act), and the number of connected components (#CCs) in each stage of our framework: partitioning, mapping of regions, and integration of solutions.

Subdivision methods. First, we tested our subdivision methods. In order to isolate the analysis of subdivisions we used the basic integration method in these tests. A summary of representative results is shown in Table 1. 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 Basic subdivision.

The subdivisions formed by the Gap method are more expensive to integrate. For example, in the one case, Walls, where the Gap method does not show improvement, there were many subdivisions created, requiring more effort in integration. Also, in both environments, the Gain method produced subdivisions

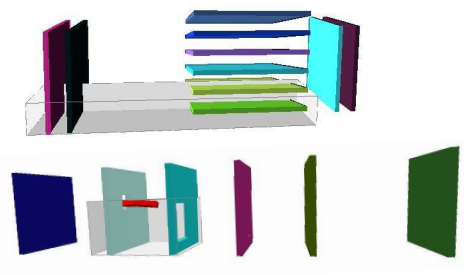

Figure 5: Subdivision examples

that are faster to map, but with an increased cost in partitioning over the Basic method. Moreover, our information Gain technique produces roadmaps with much less nodes than the other two methods. Figure 5 shows examples of subdivisions.

Table 1: Subdivision Results

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

Integration methods. We tested our integration methods with our basic random subdivision to measure the performance of our different integration methods. Table 2 shows representative runs of our meta-planner. Significantly improved performance, based on CD calls, is similar for the *overlap* and *overlap+random* methods. But, roadmaps with less components are produced with *overlap+random*. The basic integration consumes an excessive number of collision detection calls, as we noted in our previous work.

Table 2: Integration Results with a basic subdivision

Environment: Plates Robot: 6 DOF rigid body Method	Nodes act/att	#CDs	#CCs
Basic totals	1415/2250	3451872	11
partitioning	991/1550	850855	72
mapping	424/700	127826	34
integrations	na	2473191	11
Overlap totals	1241/1700	1702085	107
partitioning	508/775	983463	66
mapping	734/925	210365	177
integrations	na	508257	107
Overlap + Random totals	824/1275	2277185	16
partitioning	532/775	1309900	71
mapping	292/500	142819	24
integrations	na	824466	16
Environment: Walls **Robot: 7 DOF serial** **Method**	**Nodes act/att**	**#CDs**	**#CCs**
Basic 3-D totals	612/900	1497292	1
partitioning	211/300	84313	23
mapping	401/600	28996	12
integrations	na	1383983	1
Overlap totals	876/1350	602796	4
partitioning	302/450	182768	36
mapping	574/900	52640	19
integrations	na	367388	4
Overlap + Random totals	620/900	575863	2
partitioning	211/300	84313	23
mapping	409/600	28863	12
integrations	na	462687	2

On the other hand, our two proposed strategies reduce the CD calls in the integration while having similar costs during subdivision and mapping of regions. All our experiments solved the query assigned.

6 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 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 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 our earlier basic random subdivision. Although Gap produced solutions with less 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 less nodes. This shows that using information already available from the characterization to determine the partition location increases the quality of subregions.

The new integration methods we proposed demonstrated significant performance improvements over our previous naive integration. 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, Samuel Rodriguez, and Nancy M. Amato. Improving the connectivity of PRM roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 3, pages 4427–4432, September 2003.

[14] 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.

[15] 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.

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

[17] 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.

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

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

[20] 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.

[21] 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.

[22] 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.