# A Machine Learning Approach for Feature-Sensitive Motion Planning

Marco Morales, Lydia Tapia, Roger Pearce, Samuel Rodriguez, and Nancy M. Amato

Parasol Laboratory, Dept. of Computer Science, Texas A&M University
Email: {marcom,ltapia,rap2317,sor8786,amato}@cs.tamu.edu

**Abstract.** Although there are many motion planning techniques, there is no method that outperforms all others for all problem instances. Rather, each technique has different strengths and weaknesses which makes it best-suited for certain types of problems. Moreover, since an environment can contain vastly different regions, there may not be a single planner that will perform well in all its regions. Ideally, one would use a suite of planners in concert and would solve the problem by applying the best-suited planner in each region.

In this paper, we propose an automated framework for feature-sensitive motion planning. We use a machine learning approach to characterize and partition C-space into regions that are well suited to one of the methods in our library of roadmap-based motion planners. After the best-suited method is applied in each region, the resulting region roadmaps are combined to form a roadmap of the entire planning space. Over a range of problems, we demonstrate that our simple prototype system reliably outperforms any of the planners on their own.

## 1 Introduction

The *motion planning* (MP) problem is to find a sequence of valid states to move a movable object (often called a robot) from an initial configuration to a goal configuration. In robotics path planning, valid states are typically collision-free robot configurations. Besides robotics, instances of the MP problem can be found in a range of applications from robotics and CAD to computational biology.

The MP problem is thought to be intractable except for robots with few *degrees of freedom* (DOF) as there is strong evidence that any complete planner will require exponential time in the number of DOFs of the robot [25,26,9]. The complexity of the MP problem has led researchers to investigate heuristic planners, initially based on cell decomposition [7] and potential fields [15]. After the introduction of the Randomized Path Planner (RPP) [2], research has focused on randomized methods. Notable examples include the roadmap-based probabilistic roadmap methods (PRMs) [14] and several tree-based methods that explore the planning space starting from one or two points [3,17,12]. These methods have achieved many remarkable results, including solving several previously unsolved MP problems.

Although many good randomized techniques have been developed, no single method performs optimally for every MP problem. Rather, the performance of each technique depends on the problem instance. For example, some motion planning techniques are better suited for mostly free environments while others show their real strength in cluttered regions. Moreover, since an environment can contain vastly different regions, in many cases there may not even exist a single planner that will perform well in all its regions.

An ideal motion planner would be a meta-planner using a suite of more specialized planners to cooperatively solve an MP problem. It would automatically apply the best-suited planner for each distinct region in the planning space and would produce regional solutions that could be composed to solve the overall MP instance.

We propose an automated framework for feature-sensitive motion planning which uses a machine learning approach for *characterization and partitioning* of C-space. Partitioning is accomplished by recursively subdividing C-space until obtaining regions classified as homogeneous according to a set of features measured for each region. Next, each homogeneous region is matched with the best-suited planner in a library of roadmap-based planners and a regional roadmap is produced. Then, the regional roadmaps are combined to produce a roadmap of the entire planning space. We have implemented a simple prototype system that uses only basic techniques for partitioning C-space and combining regional roadmaps, and which contains only a few motion planning methods. Over a range of problems, we show that our simple prototype system reliably outperforms any of the planners on their own.

## 2 Preliminaries

A robot is a movable object that can be controlled through $n$ parameters or *degrees of freedom* (DOF), 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 $i$th DOF). This space, consisting of all robot configurations (feasible or not) is called *configuration space* (C-space) [18]. 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** Randomized motion planners randomly explore C-space and produce a data structure containing feasible configurations and some information about the connectivity of C-free.

Roadmap-based PRMs [14,22,23] build a graph called a *roadmap* in two steps: *node generation* and *node connection*. In node generation, feasible configurations are generated. In node connection, simple *local planners* identify which pairs of selected nodes are connectable and should be noted at roadmap edges. Many PRM-variants exist. Basic PRM [14] uniformly samples nodes. Other sampling methods increase the proportion of feasible configurations in some regions [21,4,27,17]. For example, OBPRM [1] generates configurations on or near C-obstacle surfaces and Gaussian PRM [5] only retains free samples if they are close to C-obstacles. The performance of these methods varies in different kinds of environments. For example, Basic PRM performs well in open areas while OBPRM performs better in cluttered regions.

Roadmap components can be connected using (combinations of) roadmap-based and tree-based planners [20]. Tree-based planners grow a tree starting with a known feasible configuration and add new nodes to the tree according to some strategy that either makes the tree approach the goal or uniformly cover the space reachable from the nodes already in the tree. Among these techniques we note RPP [2], Ariadne's Clew Algorithm [3], RRTs [17], and a planner for expansive spaces [12].

**C-space Subdivision** Some of the early heuristic motion planners [7] performed uniform C-space subdivision by dividing each of the $n$ DOFs into $m$ equally-sized pieces. The result is a grid of $m^n$ cells which is only feasible to compute for problems with few DOFs.

Deterministic sampling over C-space grids has been studied in [16]. A different approach using randomized methods, non-uniform C-space subdivision, has been explored in two stage planners. Decomposition-based Motion Planning [6] decomposes a high-dimensional problem into subproblems with reduced complexity for real-time planning. It searches for homotopic paths in the workspace, effectively subdividing the first three parameters of C-space while ignoring higher DOFs. Once a workspace path is found, it determines the motions of the joints (the higher DOFs).

We believe that C-space subdivision can be used to identify regions suitable for a particular motion planner. To be general and scalable for high DOF problems, the subdivision cannot be exhaustive or uniform. Instead, it should be feature-sensitive so that subdivision is performed only until the resulting regions have "uniform" features.

**C-space Characterization** Some basic studies of the performance of planners in different types of environments have been performed. A study of several PRM planners [11] in five distinct environments offered criteria for deciding which PRM variant to use. An iterative metaplanner based on region characterization is proposed in [10]. In a first iteration, four Basic PRM roadmap features help in determining whether the space is free or cluttered or if a surface is rough. Then, in a second iteration, more powerful methods could be applied. Finally, filters would remove redundant nodes to keep the remaining as seeds for tree-based planners. Environment subdivision is

suggested, but no mechanism for performing it is proposed. As evidence that automatic environment classification is feasible, several features are measured and their values are shown for several different types of environments (e.g., free vs. cluttered, and rough vs. smooth).

Characterizing C-space regions requires constructing a representation for complex and high dimensional data. Machine learning methods have gained attention over the past few years because of their ability to interpret patterns directly from such data. For example, determining the three-dimensional structure of a protein given only knowledge of the sequence of a protein is a difficult problem addressed by many bioinformatics groups. Some progress has been made in this area using knowledge obtained by a learning method about the sequence to structure relationship [8].

Learning methods are also attractive when the scope of data analysis is too complex for a human (as in C-space characterization). For example, the FIRST dataset is a collection of images of radio sources collected by the Very Large Array. In six years this dataset grew to consist of 32,000 images of two million pixels each. For this difficult problem, the decision tree algorithm C4.5 [24] was trained to identify galaxies with the bent-double morphology. The decision tree was able to classify bent-doubles and find galaxies that had been overlooked during manual searches [13].

## 3 Feature-Sensitive MP Framework

We propose a framework for mapping the connectivity of the C-space for motion planning problems through the coordinated operation of a library of randomized planning methods. Easy regions are recognized and handled quickly while difficult regions are treated with more powerful planners. This framework, sketched in Fig. 1, subdivides the C-space of a problem into a set of overlapping homogeneous regions. It then produces a roadmap in each region using the best-suited planner as determined by the regional features. Finally, the regional roadmaps are integrated to form a roadmap for the general problem.

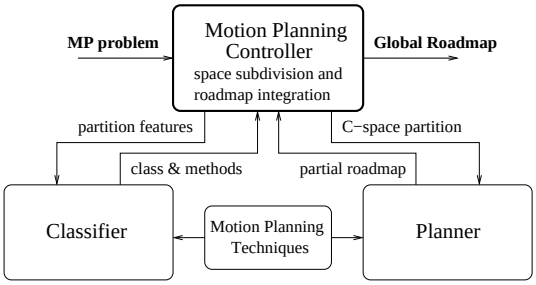



**Fig. 1.** Block diagram of the framework

At the core of this framework is a *motion planning controller* that recursively subdivides the environment and creates a tree representing the subdivision. Tree nodes represent C-space regions and the children of a node correspond to the regions that result from the subdivision of their parent node. Each node stores features extracted from the C-space of its associated region. These features are fed to the *classifier* that matches them to a class and a set of *motion planning techniques*. Based on this output, the size of the subdivision, and the granularity of the subdivisions already performed, the motion planning controller decides whether to further subdivide the current C-space region or mark it as a leaf node. Leaf node regions are sent to the *Planner* which produces a regional roadmap with the assigned motion planner. Then, regional roadmaps corresponding to the children of intermediate nodes are merged into a roadmap representing the parent node's region, so that larger and larger regions are integrated until the global roadmap is obtained in the root node.

Features should help to describe the complexity of the C-space region to be used in characterization and subdivision. They should not depend on the DOFs of the problem nor on the volume of the C-space of the partition. Also, feature extraction should use only a fraction of the resources needed to generate the roadmap.

Subdivisions are made to ease classification and to improve the chances of using the proper method efficiently, and they should be stopped once a good classification is made. Our approach differs from cell decomposition in that we do not perform uniform subdivision, instead we control the subdivision of C-space by the needs of the classification. Also, regions are allowed to overlap; indeed, some overlap may assist in "stitching" together regional roadmaps.

C-space subdivision can be obtained with different methods. Ideally, region boundaries should be decided so that resulting subregions are more homogeneous than the parent region. Information theory, such as that used by the machine learning algorithm C4.5 [24] for creating trees, gives interesting avenues for future exploration. Also, a component based partitioning method might obtain more homogeneous subdivisions.

We can characterize C-space to match it to some environment type as in [10], or to match it to the best-suited planner directly. The former requires studying the features of representative instances of the MP problem that match recognizable classes. The latter is the ideal case, it requires studying the performance of available planners on representative instances of the MP problem. Our prototype system presented here studies the former case, but we are also investigating the latter case.

Roadmap integration should use nodes, if available, in the overlapping portion of the respective regions as starting points for connecting graph components of different subregions with tree-based and roadmap-based planners as in [20].

## 4 Feature-Sensitive MP Framework Modules

In this section we describe the current implementation of the modules of our prototype system. Parts of the framework, such as the machine learning algorithm or the subdivision strategy, can be implemented in different ways. Our implementation, as described here, covers the basic components of the framework. We also discuss how we intend to further enhance the system.

### 4.1 Motion Planning Controller (Subdivision and Integration)

As described in Sect. 3, the motion planning controller creates a tree of C-space subdivisions of the MP instance, builds regional roadmaps in each subdivision, and then integrates them into a global map. First, Algorithm 1 gathers features from the C-space of the entire MP instance and the classifier uses these features to find a matching class (Line 1). Then, the environment and the matching class are sent to the recursive Algorithm 2 which creates a tree of classified C-space subdivisions (Line 2). It uses the selected planning method in each homogeneous region (as defined in Sect. 4.2) and it integrates regional roadmaps into a general roadmap covering the entire C-space. Finally, if Algorithm 2 fails to produce a roadmap, the planner makes one with a default roadmap-based method (Lines 3-4).

**Algorithm 1** Motion Planning controller

**Require:** Environment $E : (robot, obstacles)$, subdivision limit L
**Ensure:** $Tree.roadmap$: a C-space Roadmap for $E$
1: $E.class$ = Classifier.match(getFeatures($E$))
2: $Tree$ = Branch($E$,$class$,L,0)
3: **if** $Tree.roadmap$ == "noroadmap" **then**
4: $Tree.roadmap$ = Planner.makeRoadmap($E$)
5: **end if**

Algorithm 2, Branch, has three main stages: C-space subdivision (Line 1), child generation (Lines 3-11), and regional roadmap generation and integration (Lines 12-21). This algorithm returns a regional roadmap for the region of C-space covered by the node; a roadmap will be generated in every subdivision found.

**Subdivision of C-space Regions** In the first stage, the C-space of the current node is subdivided into two or more overlapping *subdivisions* (Line 1). If the node is too small or the tree has reached its maximum height it is not subdivided and a leaf with no roadmap is returned (Lines 1-2 and 22-23).

In this implementation we have an elementary partitioning strategy that randomly chooses one of the parameters of the robot to partition. A random

**Algorithm 2** Branch

**Require:** C-space region D, class C, subdivision limit L, node height H
**Ensure:** *node* = [region, class, roadmap]
1: (*subdivisions*, *error*) = subdivide(D,L)
2: **if** no error **then**
3: **for all** $region_i$ in *subdivisions* **do**
4: $class_i$ = Classifier.match(getFeatures($region_i$))
5: **if** MeetLeafCriteria($region_i$,$class_i$,C,L,H) **then**
6: $child_i$ = [$region_i$, C, "noroadmap"]
7: **else**
8: $child_i$ = Branch($region_i$,$class_i$, L, H+1)
9: **end if**
10: add $child_i$ to *children*
11: **end for**
12: **if** MeetClassMergingCriteria(*subdivisions*,C) **then**
13: *node* = [D, MergeClasses(*subdivisions*,C), "noroadmap"]
14: **else**
15: **for all** $child_i$ in *children* **do**
16: **if** $child_i$.roadmap == "noroadmap" **then**
17: $child_i$.roadmap = Planner.makeRoadmap($child_i$)
18: **end if**
19: **end for**
20: *node* = [D, MergeClasses(*children*), Planner.mergeRoadmaps(*children*)]
21: **end if**
22: **else**
23: *node* = [D, class, "noroadmap"]
24: **end if**
25: return *node*

point within the range of the chosen dimension is selected, a small $\varepsilon$ (around 10% of the range) is used to divide it into two overlapping ranges. Region overlap is important for roadmap integration as discussed in Sect. 4.3. In our prototype system, we only subdivided on the first positional parameters.

**Children Generation** During the second stage, in each C-space region, features are extracted and fed to the classifier that matches them to a type of environment (Line 4). The region is checked to determine whether it should represent a leaf node (Lines 5-6) or a branch (Lines 7-8). In any case, the result is a child for each subdivision of the current node (Line 10).

Features should capture characteristics of C-space that will give insight into the best planning method for the corresponding region. We create a uniform and quick Basic PRM roadmap from which we extract several features. In addition to giving some insight into C-space structure, features extracted this way will also depend on resolution and sampling strategy. These factors should be taken into account during classification. Our prototype classifier currently recognizes C-space regions as free, cluttered, narrow passage, and

non-homogenous. This along with the features used for classification will be discussed in Sect. 4.2.

A region *meets leaf criteria* (Line 5) when it has reached its limit in size so that there is no room to vary any robot parameter, or when its class is the same as that of the parent whose classification is other than non-homogeneous. If these criteria are not met, the region is branched with a recursive call to Algorithm 2.

**Regional Roadmap Generation and Integration** In the last stage, subdivisions are evaluated to determine whether they should be merged back into their parent without making a roadmap (Lines 12-13) or roadmaps should be produced for leaf subregions through the planner (Lines 15-19) to integrate regional roadmaps into the parent node (Line 20).

Subdivisions *meet class merging criteria* (Line 12) when all of them are leaf nodes belonging to the same class and their parent is non-homogeneous. This means that the parent could not be classified succesfully, but the children were. Since all of them belong to the same class, their regions are merged back into their parent and assigned to its children's class. This prevents having to spend extra effort in merging regional roadmaps. However, in some types of C-space or for certain planners, it may be better to not merge those subregions; this is a topic for future study.

When the subdivisions do not meet class merging criteria the planner is called to make a roadmap for each leaf child (which does not have one yet) and to merge it's children's regional roadmaps into a single roadmap for the current node. Hence, the final roadmap obtained for the root of the tree is a global roadmap. The planner is described in Sect. 4.3.

### 4.2 Classifier

The classifier is given a set of features of a C-space partition, and it outputs a matching class and a confidence level. It is hard to derive analytical expressions defining classes of C-space or rankings for choosing good planners for several reasons: many features that only represent a fraction of C-space can be extracted; there are many MP strategies to map C-space; and the range of MP problems is too diverse in obstacle placement and robot dimensionality. Machine learning methods can be used to perform the classification. For our particular needs of understandable decisions, discrete classifications, and ease of use, decision tree learning methods are appropriate.

Several measurements are made in a small roadmap to describe the complexity of the partition. In our prototype system we used Basic PRM to create the feature roadmap; our future work will investigate other methods as well.

**Roadmap Features** Our objective is a general method that can classify a wide range of environment types and complexities. In our prototype system,

**Table 1.** Features for roadmap $G : (V, E)$, vertices $v \in V$, edges $e(v_\mathrm{j}, v_\mathrm{k}) \in E$, $CC \equiv \{components \in G\}$, $C_\mathrm{M}(cc_\mathrm{k}) = \sum v_\mathrm{i} / \|cc_\mathrm{k}\| \forall v_\mathrm{i} \in cc_\mathrm{k}$, $E_\mathrm{l} = \sum \|e_\mathrm{j}\| \forall e_\mathrm{j} \in E$

| *General Features* | |
|---|---|
| Free node ratio: $V_\mathrm{r} = \lvert V \rvert / V_\mathrm{att}$ | $V_\mathrm{att} \equiv$ number of attempted nodes (1) |
| Components: $C_\mathrm{n} = \lvert CC \rvert$ | (2) |
| Edge ratio: $E_\mathrm{r} = \lvert E \rvert / E_\mathrm{att}$ | $E_\mathrm{att} \equiv$ number of attempted edges (3) |

| *Component-Based Features* for connected component $cc_\mathrm{k}$, 4 features per definition[a] | |
|---|---|
| Node distance: $d(v_\mathrm{i}, v_\mathrm{j}) = \|v_\mathrm{i} - v_\mathrm{j}\|$ | $v_\mathrm{i}, v_\mathrm{j} \in cc_\mathrm{k}, \quad i \neq j$ (4) |
| Edge length: $E_\mathrm{l} = d(v_\mathrm{i}, v_\mathrm{j})$ | $(v_\mathrm{i}, v_\mathrm{j}) \in E, \quad v_\mathrm{i}, v_\mathrm{j} \in cc_\mathrm{k}$ (5) |
| Distance to Center: $d_\mathrm{c} = d(v_\mathrm{i}, C_\mathrm{M}(cc_\mathrm{k}))$ | $v_\mathrm{i} \in E$ (6) |
| Component distance: $D(cc_\mathrm{i}, cc_\mathrm{j}) = d(C_\mathrm{M}(cc_\mathrm{i}), C_\mathrm{M}(cc_\mathrm{j}))$ | (7) |
| Component length: $C_\mathrm{l} = \sum \|e_\mathrm{i}\| / E_\mathrm{l}$ | $\forall e_\mathrm{i} \in cc_\mathrm{k}$ (8) |

[a] Average of the maximum, minimum, mean, and std. deviation over all $cc_\mathrm{k} \in CC$
Notation: $|\cdot| \equiv$ cardinality of $\cdot$, $\|\cdot\| \equiv$ length of $\cdot$.

we extract a set of 23 features; ideally we want to capture the characteristics of each type of space while not being impacted by size and complexity of C-space (Table 1). Specifically, these features should be independent of the dimensionality and volume of the C-space region analyzed.

Three *General features* capture an overall picture of roadmap generation. A *Free node ratio* gives insight into the "freeness" of C-space. In partitions with no obstacles, this ratio will be one whereas in regions filled with obstacles it will be close to zero. *Components* gives an idea of the connectivity of C-space. A count of one would indicate easy connectability. In highly cluttered areas this count will approach the number of nodes sampled. An *Edge ratio* will be low when nodes are difficult to connect such as in cluttered regions, close to the surface of complex C-obstacles, and in narrow passages.

Twenty *Component-based features* are related to the complexity, extension, and coverage of components that map C-free, and the space between them. Each description corresponds to four features: its maximum, minimum, mean, and standard deviation in each component, averaged over the number of components.

Component-based features describe the connectivity of C-free and point to interesting areas of the problem. For example, *Node distance* gives an idea of the distance between nodes in a component. Because packed components will have smaller values than components covering elongated areas, node distance might be important in distinguishing between a cluttered space and a narrow passageway. While the features of *Edge length* and *Distance to center* give

an idea of the span and coverage within a single component, *Component distance* contributes a measure of the distance between components that is related to the amount of infeasible space. The last feature, *Component length*, gives an idea of the volume covered by components. This particular feature is normalized by dividing it by the length of the entire roadmap.

Features are normalized to make them independent of C-space dimensionality and size. With the exception of *Component length*, that is already normalized, each feature is divided by a constant

$$k_{\mathrm{n}} = \max |dimension_i| * |dof|, \quad 1 \leq i \leq |dof| \tag{9}$$

where $|dimension_i|$ is the length of dimension $i$, so to relate $k_{\mathrm{n}}$ to the volume of C-space covered in the sampling. It may be needed to relate the number of sampled nodes for feature collection to $k_{\mathrm{n}}$ to keep it small, but large enough to capture features that properly classify the corresponding C-space region.

**C-space Classes** While MP environments are often non-homogeneous, if partitioned properly, they can be viewed as composed of homogeneous subcomponents. We currently use four classes of C-space that we plan to extend in the future:

- *Free:* very few obstacles in the region.
- *Cluttered:* obstacles clutter the region.
- *Narrow passage:* regions with passages between other types of region.
- *Non-homogeneous:* regions that do not fit in any other class.

**Machine Learning Techniques** There are many learning algorithms that work well for discrete-valued classification. From our learning method we need an algorithm that provides us with understandable decisions for verification, relatively quick learning time, and ease of use. For these needs, we have selected the decision tree method, C4.5 [24].

C4.5 creates a series of decisions based on feature values that leads to a classification of an example. The decision nodes or branches in the trees are selected by choosing the feature that "best" splits the space. Every path along the tree leads to a leaf node or classification based on the set of decisions that lead to that leaf. After a tree is constructed, C4.5 removes paths that do not apply to a general portion of the data through a process called *pruning*.

**Classifier training.** The classifier is trained with examples of free, cluttered, narrow passage, and non-homogeneous environments. Features are extracted, a decision tree is trained, ordered rules are composed from this tree, and the performance evaluated. Because of our small training set, evaluation is performed $k$ times, each time using a different subset of the data for training and evaluation. This process of $k$-fold cross-validation [19], gives a better idea of how the method is working on average. Although the current set of classification rules do not take into account all of the features we examined, these features might be utilized for other environment types such as isolated or blocked spaces.

### 4.3 Planner and Pool of Motion Planning Methods

The planner works as a regular motion planner with a collection of roadmap-based and tree-based planners. It can be called to either make a roadmap for a C-space region or to merge a set of regional roadmaps.

When making a roadmap for a region, it receives the environment, region boundaries, region type as defined by the classifier, and the roadmap used for extracting features in that region. It starts with the feature roadmap to which it applies planners well-suited for that roadmap class. The decision of which planners to use in each class calls for a deeper study as in [11]. In this work, we assign planners to C-space type based on recommendations in [10] and on the strengths of some methods as described by their authors. In future work we will add more planners to our library and will investigate automated methods to match planners to regions. In free regions we do not do additional planning. In narrow passages we apply OBPRM (normals) with about ten times as many nodes as in the feature roadmap. In clutter regions we apply OBPRM (triangles) with about five times as many feature roadmap nodes. In non-homogeneous regions we apply Basic PRM followed of OBPRM (triangles), each method generating as many nodes as in the feature roadmap. In all regions we use a straight-line local planner and a $k$-closest connecting strategy with $k = 5$.

When merging regional roadmaps into their parent region the planner "stitches" them by attempting to connect subregion components focusing on their overlapping sections. This can be done with roadmap-based and tree-based methods as in [20] starting from the nodes in the overlap. But for simplicity, we applied a component connection and a $k$-closest connection. By construction, Algorithm 2 merges only neighboring regions. Our subdivision ensures that regions overlap by small $\varepsilon$. Again, we add the resulting roadmap to the feature roadmap of the parent region.

More investigation is needed to make the classifier learn also about the motion planning methods so that both methods and parameters can be automatically selected and assigned to each partition.

## 5 Experiments

We tested our implementation 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 used our group's C++ motion planning library compiled with gcc 3.2.2 and Perl to control the partitioning. MySQL was used for storing features for training.

**Training the Classifier** We designed a set of 200 environments: 50 free, 50 cluttered, 50 narrow passages, and 50 non-homogeneous. Features were extracted from each environment and saved in a database from which we obtained data to train decision trees. Ten different sets of trees were created using 180 examples for training and a distinct set of 20 examples for testing.

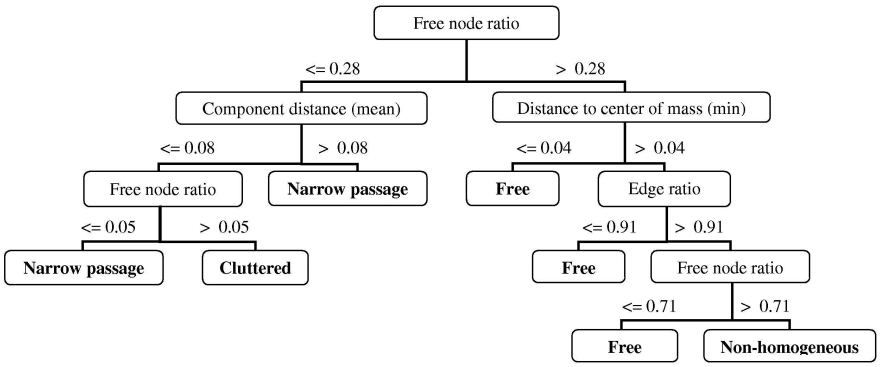


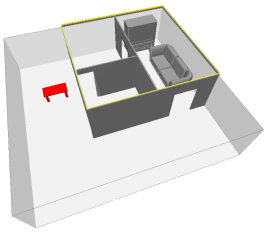

**Fig. 2.** Decision tree selected for classification

Accuracy rates produced by these cross-validation tests ranged from 100% to 85% with an average testing accuracy of 95%. We selected the tree that was able to classify 98.9% of our training and 100% of our testing examples to use for our classification (Fig. 2).

**Test Environments** For testing, we chose four environments with different kinds of obstacles and robots. First, a house environment (Fig. 3) that has three rooms with some furniture. The robot is a table to be moved from outside the house to the smallest room. It has different kinds of regions: free outside the house, cluttered close to the furniture, and a narrow passage must be traversed to reach the last room. Second, the walls-rigid environment (Fig. 4) has 5 chambers connected by varying-size openings. Three of the chambers are filled with small obstacles and the other two are empty. The robot is a stick that must traverse all the chambers. Third, the walls-serial environment (Fig. 5) is similar to the walls-rigid, but all the chambers are free of obstacles. The robot, however, is articulated with two long links, making it more difficult to maneuver. Last, the narrow-clutter environment (Fig. 6) is an open area with a big obstacle that can be traversed through a narrow passage followed by some open space and a clutter of small obstacles. The robot is articulated with 4 small links. It must traverse the entire environment including a narrow passage and a highly cluttered area.

**Experimental Design** We applied Basic PRM, OBPRM, and our framework (C&P, for classify and partition) to the environments to find the minimum number of nodes needed to solve the problem. The minimum number of nodes was found by a binary search, and it was repeated 4 times for each environment and method pair.

Basic PRM employs uniform sampling and $k$-closest connection. OBPRM generates nodes with normal alignment and $k$-unconnected-closest connection (a variant of the standard $k$-closest which only attempts connections to



| Method | Nodes$_{act/att}$ | #CDs | #CCs | solved? |
|---|---|---|---|---|
| Basic PRM | 5798/5798 | 1,385,004 | 9 | yes |
| OBPRM | 489/491 | 3,562,679 | 20 | yes |
| C&P:total | 226/375 | 724,861 | 3 | yes |
| partitioning | 102/225 | 90,917 | *29 | n/a |
| roadmap gen. | 124/150 | 326,076 | 30 | n/a |
| roadmap int. | 0/0 | 307868 | 3 | yes |

* 1 narrow, 4 free, many non-homogeneous

**Fig. 3.** House environment (6 DOF) and performance metrics

| Method | Nodes$_{act/att}$ | #CDs | #CCs | solved? |
|---|---|---|---|---|
| Basic PRM | 3000/3000 | 2,422,749 | 11 | yes |
| OBPRM | 1200/1179 | 11,107,232 | 48 | yes |
| C&P:total | 257/420 | 7,635,344 | 10 | yes |
| partitioning | 143/285 | 164,302 | *35 | n/a |
| roadmap gen. | 114/135 | 72,979 | 30 | n/a |
| roadmap int. | 0/0 | 7,398,063 | 10 | yes |

* 1 narrow, 3 free, many non-homogeneous

**Fig. 4.** Rigid Body Walled environment (6 DOF) and performance metrics

different components). C&P's planner is used as described in Sect. 4.3. All experiments use the straight-line and rotate-at-s 0.5 local planners.

Figures 3, 4, 5, and 6 present results of representative runs. We show the number of nodes generated (act), attempted (att), the number of collision detection calls (#CDs), the number of connected components (#CCs), and we indicate whether the query was solved.

We can see the benefit of our framework in Fig. 3. The number of collision detection calls to solve the house environment using C&P is far smaller than when using Basic PRM and OBPRM. Also, fewer nodes are needed to solve the problem with the C&P framework than with individual methods.

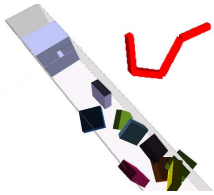
The house environment is classified as a whole as a non-homogeneous environment. When partitioned and classified, useful partitions are found. When the narrow partition was identified, our feature-sensitive framework was able to focus many more OBPRM nodes in that difficult area. Similarly, when the free partitions were found, simple planners were used.

In the Walls Rigid environment (Fig. 4) OBPRM solves the query with far fewer nodes than Basic PRM, but requiring many more collision detection calls. C&P also needs many more collision detection calls than Basic PRM, but it solves the query with only 257 nodes, considerably less than the nodes needed by OBPRM and Basic PRM. During characterization, cluttered and narrow areas of the environment are also discovered as well as free areas that can be considered as already well-mapped.

| Method | Nodes act/att | #CDs | #CCs | solved? |
|---|---|---|---|---|
| Basic PRM | 17801/17801 | 3,183,940 | 20 | no |
| OBPRM | 816/821 | 2,830,528 | 10 | yes |
| C&P:total | 590/795 | 1,460,845 | 2 | yes |
| partitioning | 168/345 | 67,695 | *35 | n/a |
| roadmap gen. | 422/450 | 81,263 | 145 | n/a |
| roadmap int. | 0/0 | 1,311,887 | 2 | yes |

* 2 narrow, 5 free, many non-homogeneous

**Fig. 5.** Serial Walls (2-link) environment (7 DOF) and performance metrics



| Method | Nodes act/att | #CDs | #CCs | solved? |
|---|---|---|---|---|
| Basic PRM | 19461/19461 | 40,056,265 | 110 | no |
| OBPRM | 350/350 | 23,142,065 | 49 | yes |
| C&P:total | 271/435 | 5,966,310 | 60 | yes |
| partitioning | 68/225 | 1,017,267 | *29 | n/a |
| roadmap gen. | 203/210 | 2,129,146 | 143 | n/a |
| roadmap int. | 0/0 | 2,819,897 | 60 | yes |

* 2 narrow, 1 cluttered, many non-homogeneous

**Fig. 6.** Narrow-Clutter environment (9 DOF) and performance metrics

As can be seen in Fig. 5, the Serial Walls environment is quickly solved by C&P. Significant narrow regions are identified and C&P thus focus on those difficult areas. It is important to note that the robot in this environment was articulated and narrow passages are still identified.

C&P shows its strength in the Narrow-Clutter environment (Fig. 6). After the entire environment is classified as non-homogeneous, the planner focuses on the second half of the environment around the narrow passage. The concentrated effort on the narrow passage allows C&P to use considerably fewer collision detection calls than OBPRM to map the narrow region. Basic PRM was never able to find a solution for the narrow region in the alloted number of nodes (20,000).

As mentioned before, C&P joins neighboring subdivisions that belong to the same class. This allows it to focus on entire regions of the same type and prevents oversampling. In general, C&P is able to solve the environments with fewer collision detection calls and nodes than Basic PRM and OBPRM. This is due in large part to its ability to identify difficult regions and focus on these areas. Also, free areas can be discovered and mapped without over-sampling.

## 6 Conclusions

We have proposed an automated framework for feature-sensitive motion planning. A prototype implementation of our machine learning classification and

partitioning approach was shown to achieve results superior to those obtainable by any of the individual planners on their own.

Throughout this paper, we have discussed possible improvements and future directions for our framework. For example, we plan to extend the current framework to recognize and handle more C-space classes such as blocked and isolated regions. Another area of focus is the exploration of partitioning techniques. We suggested two ideas based on information theory and component based partitioning. Throughout the improvement process, we plan on extending our library of motion planning methods and continuing to develop methods of selecting an appropriate planner and its parameters based on previous experience.

**Acknowledgement**

This research supported in part by NSF Grants ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACR-0113971, CCR-0113974, EIA-9810937, EIA-0079874, 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.

## 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. V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 1018–1023, 1999.
6. O. Brock and L.E. Kavraki. Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration places. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1469–1475, 2001.
7. 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.
8. C. Bystroff and D. Baker. Prediction of local structure in proteins using a library of sequence-structure motifs. In *J. Mol. Biol.*, volume 281, pages 565–577, 1998.

9. J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.
10. L. K. Dale and N. M. Amato. Probabilistic roadmaps – putting it all together. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1940–1947, 2001.
11. R. Geraerts and M. H. Overmars. A comparative study of probabilistic roadmap planners. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, December 2002.
12. 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.
13. C. Kamath, E. Cantu-Paz, I. Fodor, and N. Tang. Classifying bent-double galaxies. In *IEEE Comput. in Sci. & Eng.*, volume 4, pages 52–60, 2002.
14. 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.
15. O. Khatib. Real–time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.*, 5(1):90–98, 1986.
16. S. M. LaValle and M. S. Branicky. On the relationship between classical grid search and probabilistic roadmaps. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, December 2002.
17. S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.
18. 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.
19. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
20. M. Morales, S. Rodriguez, and N. M. Amato. Improving the connectivity of PRM roadmaps. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 3, pages 4427–4432, September 2003.
21. 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.
22. M. Overmars. A random approach to path planning. Technical Report RUU-CS-92-32, Computer Science, Utrecht University, The Netherlands, 1992.
23. 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.
24. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
25. 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.
26. 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.
27. 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.