Model-Based Motion Planning

Brendan Burns Oliver Brock
Laboratory for Perceptual Robotics
Department of Computer Science
University of Massachusetts Amherst

Technical Report 04-32

September 2004

Abstract

Robotic motion planning requires configuration space exploration. In high-dimensional configuration spaces, a complete exploration is computationally intractable. To devise practical motion planning algorithms in such high-dimensional spaces, computational resources have to be expended in proportion to the local complexity of a configuration space region. We propose a novel motion planning approach that addresses this problem by building an incremental, approximate model of configuration space. The information contained in this model is used to direct computational resources to difficult regions. This effectively addresses the narrow passage problem by adapting the sampling density to the complexity of that region. Each sample of configuration space is guaranteed to maximally improve the accuracy of the model, given the available information. Experimental results indicate that this approach to motion planning results in a significant decrease in the computational time necessary for successful motion planning.

1 Introduction

Complete motion planning requires understanding a robot's configuration space. The acquisition of such an understanding is computationally challenging. The general motion planning problem has been shown to be PSPACE complete [32]. Sampling-based motion planning methods [21, 25] address this difficulty by constructing a connectivity graph which implicitly approximates the structure of the configuration space, thus minimizing exploration. However, even sampling-based methods can require an exponential number of uniformly placed samples [36] to build a successful representation of configuration space.

Uniform sampling, which forms the basis of almost all multi-query sampling-based motion planning approaches, makes the implicit assumption that configuration space is uniformly complex. This forces a motion planner to expend the amount of computation required by the most complex area of configuration space on all areas of configuration space. This shortcoming of uniform sampling is generally referred to as the "narrow passage problem" [18]. A great deal of research in sampling-based motion planning is concerned with the resolution of this problem [2, 6, 17, 19]. The approaches proposed in the literature so far use heuristics to filter configurations after, computationally expensive, examination in the configuration space but prior to insertion into the roadmap. We propose an alternate adaptive approach which actively selects configurations before examination.

A practical motion planner has to exploit the structure inherent in a particular instance of the motion planning problem to avoid the computational complexity bounds of the general motion planning problem. This can be achieved by expending computation on each region in proportion to the complexity of that region. For sampling-based motion planning, this corresponds to adapting the planner's sampling density to match the complexity of the configuration space. More samples are drawn from complex areas of configuration space, constructing the detailed roadmap required for motion planning in those areas. Simple regions of configuration space receive minimal sampling and consequently, a simple representation. Adapting sampling this way greatly reduces the computation necessary to construct a roadmap for practical motion planning problems. Redundant exploration in simple areas is avoided, saving computation on unnecessary collision and edge checks. The collision and edge checks that do occur are necessary. Computational effort is directed to regions where detailed information is required to solve the problem.

In the following we introduce a novel motion planning technique which directs computational resources in this manner. This new approach incrementally constructs and refines an approximate statistical model of the entire configuration space. The model indicates the areas of configuration space which are complex and the areas which are simple. The model acts as a guide for our sampling strategy. Sampling is performed in proportion to the model's estimate of complexity. As areas of configuration space become understood, i.e., the model represents them accurately, they receive no further sampling.

All sampling-based motion planners construct an approximate model of the configuration space from which paths may be derived. The proposed motion planner also uses its model to guide the sampling of configuration space. To do this, the model used by the planner must provide an approximate picture of the entire configuration space, enabling predictions of unobserved regions of the space. It should also provide a confidence value for those predictions, indicating areas which are poorly understood. The traditional model, a connectivity graph or roadmap, is built from path-segments and configurations. Consequently, it cannot provide these capabilities. A more expressive approximate model is required.

For such a model, we use locally weighted regression [3], a statistical approach from machine learning. This model provides a computationally efficient representation of the configuration space. The model provides efficient predictions about the state of unseen regions of configuration space and its confidence in these predictions. Sampling is directed to regions of uncertainty to maximally improve the accuracy of the model. In addition, the proposed motion planning approach exploits the efficient predictions provided by the model to avoid invocations of a costly collision checker. Using locally weighted regression we have developed a practical implementation of a model-based motion planner. Experimental results indicate that model-based motion planning is capable of significantly reducing the amount of exploration required to build an accurate model of relatively high-dimensional configuration spaces.

2 Related Work

2.1 Motion Planning

All complete motion planners build a model of configuration space. Depending on the motion planning algorithm, different types of models are constructed. Details of a number of approaches, with a particular focus on sampling-based methods that most directly relate to our work, are given in the following.

For cell decomposition planners [24], the model is a collection of labeled cells and a connectivity graph connecting them. Exact cell decomposition methods partition configuration space into non-overlapping regions that either contain obstacle or free space. For approximate cell decomposition [1, 27], the cells are labeled as free, obstructed, or mixed. These methods generally proceed by recursively subdividing mixed

cells until a pre-determined minimal cell size is reached. Whereas cell decomposition methods are complete, approximate cell decomposition methods are only complete to the resolution of the smallest allowed cell. This notion of completeness is referred to as resolution completeness.

Artificial potential field methods [22] model configuration space as a potential function which gives a numeric value to a particular configuration. Descending the potential function via gradient descent gives a path from start to goal configurations. Because they descend a gradient, potential field methods are susceptible to getting stuck in local minima which are not the goal location. Consequently, motion planners based on artificial potential fields are incomplete. Navigation functions [4, 11, 23] address this problem by providing local minima-free artificial potentials. The computation of such navigation functions, however, is difficult in the general case.

The silhouette method [8] builds an exact roadmap of the configuration space by incrementally sweeping a plane along each dimension of the configuration space and extracting the curves which connect locally extremal points. These curves form the basis for the roadmap which is then used for motion planning. Because the method requires sweeping along every dimension in configuration space, the method is exponential in the degrees of freedom of the robot.

Due to the computational complexity of exact modeling of configuration space obstacles, the most efficient techniques for motion planning for robots with many degrees of freedom construct approximate models by sampling configuration space. The probabilistic roadmap (PRM) method [21] uses a connectivity graph as its model of configuration space. This graph, called roadmap, consists of milestones (vertices) and collision-free paths (edges), implicitly captures the free space connectivity of the configuration space. PRM methods derive their computational efficiency from the fact that the roadmap can be constructed through the examination of only a small fraction of the configuration space. This gain in computational efficiency comes at the cost of replacing the notion of completeness with probabilistic completeness. Probabilistic completeness guarantees that any given path in a configuration space will be found with a probability which increases as sampling increases.

The uniform sampling strategy used by the original PRM algorithm makes the implicit assumption that all areas of configuration space are equally complex. For any practical motion planning problem, however, some regions require a high density of samples to explore completely, while others only require a few samples. Uniform sampling is forced to sample everywhere with the high density required by the most complex area of configuration space. This presents a serious computational challenge when the configuration space contains small regions of free space that are surrounded by obstacles. Such a structure is often referred to as a "narrow passage" [18]. Much of the research in sampling-based motion planning over the last decade has focused on the design of sampling strategies to address the narrow passage problem. A survey several of these methods follows.

In the traditional PRM algorithm, samples that are found to be inside of configuration space obstacles are discarded. Some heuristic sampling strategies attempt to use these obstructed samples to find nearby free points, assuming that points near obstacles are more likely to be important than more distant free points. These methods use heuristics based on obstacle surface properties [2] or shrinking and growing obstacles [18] to modify colliding samples into free ones. Although modifying obstructed samples can discover free points near obstacles, it often requires the computationally expensive examination of a large number of additional obstructed configurations before a free configuration is found. Furthermore, not all points near obstacles are critical to the development of a successful roadmap.

Other approaches address the problem of obstructed configurations by minimizing the probability that an obstructed configuration will be examined. The medial-axis strategy [14, 16, 26, 38] selects configurations near to the medial axis of the work space [14, 16, 38] or configuration space [26], since these configura-

tions are less likely to be obstructed. A significant challenge to this approach is the difficulty of finding configurations near the medial axis for articulated robots.

The computational cost of edge validation in large roadmaps has lead to the development of sampling strategies that attempt to minimize unnecessary samples. The Gaussian sampling strategy [6] and the bridge test [17] ensure that most configurations in the roadmap are close to obstacles or lie inside a narrow passages, respectively. These configurations are presumed to be most helpful in capturing the free space connectivity. Visibility-based PRM planners [35] specify nodes in configuration space which act as "guards." These nodes capture a region of configuration space containing every configuration for which a collision-free, straight-line path exists to to the guard (the configuration is "visible" to the guard). Nodes are only inserted into the roadmap if they are not in a guard's captured region, or if they are "connection" nodes which permit indirect paths between two previously disconnected guards. This results in a smaller roadmap without redundant nodes or connections but it requires the expensive computation the guard's visibility, which in effect is determined by performing an edge validation. In a different approach, Fuzzy PRM planners [29] address the costs of edge checking by incrementally validating edges based upon an estimate of their probability of being free. The probability that an edge is collision-free is estimated by a calculation based upon the length of the edge.

The methods discussed so far perform preprocessing by constructing a roadmap for the entire configuration space. Any subsequent motion planning query can then be answered efficiently by performing graph search in the roadmap. Due to this property, they are called multi-query methods. To avoid the exploration of the entire configuration space, so-called single-query methods end exploration upon the solution of a particular motion planning problem. The selection of regions to explore during the planning process is performed using heuristics. Lazy PRM planners [5], for example, biases exploration toward those regions close the the initial and final configuration. Rapidly-exploring random trees (RRTs) [25] are another single-query approach to sampling-based motion planning. They use simulated diffusion to build a connectivity tree in configuration space. RRTs are often used for single query motion planning when the start and end configurations are known. In these situations, trees are rooted at both the start and end configurations and grown until they intersect. The approach of diffusion is also taken by expansive spaces [19], an approach quite similar to RRTs.

Recent work in sampling-based motion planning has examined the use of more expressive approximate models of configuration space. The entropy-guided motion planning approach [7] maintains a bounding box around connected components of the roadmap and uses these boxes to influence the selection of configurations lying between these components. Entropy-guided motion planners demonstrate that exploiting the information represented in more expressive models allows this approach to obtain significant improvements other planners.

Other work [28] uses machine learning to choose which motion planner should be applied to different regions of configuration space. The approach constructs a decision tree [31] from hand selected attributes of prior planning experience. This decision tree is used to select a motion planning approach which is thought to be suited to a particular region of configuration space.

Hsu [20] also proposes an adaptive sampling method. The work assembles a weighted collection of different sampling strategies. Each sampling strategy is used to choose a configuration with a probability proportional to its weight. The weights are adapted in response to the samplers performance. Several strategies for weight updating are proposed. While Hsu's work can adapt to a changing understanding of the configuration space, it does not adapt to different regions of the configuration space. Different sampling strategies may become weighted more heavily, but they are applied uniformly across the configuration space.

2.2 Machine Learning

In this work, we view the construction of an approximate model for configuration space as a classification task that can be solved using machine learning techniques. In general, a classification can be seen as a mapping of some input x to an output label y. For configuration space, this mapping is described in detail in Section 3.1.

Locally weighted regression [3] is an efficient approach to modeling arbitrary functions based on samples. This approach possesses a number of attractive characteristics, making it well-suited to the task of modeling configuration spaces in motion planning problems. Models based on locally weighted regression are locally adaptive to the structure of the underlying function, their training cost is constant, regardless of the number of training examples, and an efficiently computable closed form derivation of an active learning strategy [10] exists.

In contrast to traditional machine learning, active learning considers the task of learning from examples when the learner has control over the selection of examples from which it learns. In particular, active learning focuses on selecting examples which maximize the accuracy of the learner using a minimum of training examples. Approximating configuration space is an active learning situation since we can choose to examine any configuration with the collision checker. We use the derivation of an active learner for locally weighted regression presented by Cohn et al. [10]. Further details concerning locally weighted regression and active learning are in Section 3.1.

Decision tree algorithms, such as C4.5 [30], perform classification by constructing a tree where each branch represents partitioning on a particular input value. Each leaf of the tree corresponds to a classification. Others [28] have chosen to use decision trees to aid motion planning. Decision trees are often prone to overfitting training data and can have difficulty adapting to numerous local features of a space.

Support vector machines [34] map input values into a high-dimensional space. In this high-dimensional space, a number of linear separations is used to model different categories. One of the challenges of support vector machines is the design of a customized kernel function suited to the particular classification task. Designing this kernel function for configuration space classification is still an open problem. Active learning with support vector machines has also been explored [37].

Mixture models [13] approximate a complicated surface through the mixture of a number of simpler distributions. Gaussian distributions are a popular choice for the mixture distributions. These distributions are fit to training data using the expectation maximization (EM) algorithm [12]. Several choices in constructing the model such as the type of distribution and the number of distributions to mix are critical to the success of the model. Additionally, the EM algorithm can be quite computationally expensive. Cohn et al. [10] also provide an active learning formulation for Gaussian mixture models.

Neural networks [33] can be used to approximate a classification function. Neural networks consist a number of layers connected together by weighted links. The input value enters through the input node and is propagated forward through the network resulting in the presentation of values on the output nodes. A number of different structures for the nodes and methods for training the weights of the links exist. Like mixture models, the choice of structure and training for the neural network play a significant role in its success. Training times for the neural networks can be prohibitive.

3 Model-Based Motion Planning

To build a computationally efficient, sampling-based motion planner we propose the use of an approximate model of the entire configuration space. This approximate model is constructed incrementally, as a solution

to the motion planning problem is computed. It provides information about which areas of the configuration space are well understood and which are understood poorly. The sampling strategy associated with such a model uses this information to adapt sampling densities so that areas receive sampling in proportion to their complexity. Regions that remain poorly understood due to their inherent complexity will be sampled more densely. Regions which are accurately represented by the model are not sampled further.

In addition to exploiting the information contained in the model to adapt the sampling density, the model can also generate predictions about unexplored regions of configuration space. If the model is capable of making a prediction with high confidence, a planning method should choose to accept the prediction, thereby reducing the number of required collision checks significantly. Only uncertain paths should be explored. Neither paths which are likely to be obstructed nor paths which are likely to be free are worth the computation required to explore them. If a path is likely to be free, exhaustively checking it represents excess computation for information already obtained from the model. If a path is likely obstructed, verifying that it is obstructed is likewise a waste of computation. Interestingly, by far the greater savings in computation comes from not verifying free paths since the computation required to verify a path is unobstructed is generally greater than the computation required to determine a path is obstructed.

The graph-based models underlying current sampling-based motion planners only contain information about free configurations and free path segments connecting them. Because of this, graph-based models do not allow predictions about unexplored space. To find a suitable model for such predictions, we view the approximation of configuration space as a classification problem and choose from methods developed in the field of machine learning (see Section 2.2). The resulting sampling strategy is fundamentally different from those of previous sampling-based planners. The proposed model-based planner shares two characteristics with current sampling-based techniques: it samples configuration space and the resulting path is based on path segments. As we will see in this section, every other aspect of model-based planning differs significantly from previous sampling-based planning methods.

We now present the algorithmic details of the proposed model-based motion planning approach. First, we present the model used to approximate configuration space. This model consists of an underlying representation based on locally weighted regression [3] and a sampling strategy based on active learning [10]. Subsequently, we present a model-based motion planner that exploits the predictive power of this model to avoid edge validations when finding a collision-free path.

3.1 Underlying Representation

Machine learning traditionally concerns itself with the approximation of a function $f(x) \rightarrow y$ where x is the (possibly multi-dimensional) input, and y is the (possibly multi-dimensional) prediction. For configuration space approximation, the input to the function is a location in configuration space. In practice, we normalize this configuration space input to a unit hyper-cube. The output of the approximating function is a continuous value in the range $[-1, 1]$. Training configurations are labeled with a negative one if the configuration is obstructed and a positive one if the configuration is free. This choice of values is arbitrary. It could be modified to ensure that the mean of the distribution of values was zero, an attractive feature for some machine learning algorithms. When predicting the state of a location, the continuous output of the approximate function is rounded up or down to create the discrete prediction of obstructed or free.

Locally weighted regression (LWR) [3] is a lazy, supervised learner. Supervised learning algorithms build approximate models from labeled examples or training data. In the case of configuration-space modeling, the training data is configurations labeled with a value corresponding to their obstructed or free state. Lazy learners delay the examination of training data until a prediction needs to be made. These characteristics make LWR a computationally efficient choice for our modeling needs since the time to add data to the

model is constant and querying is at most linear in the number of points contained by the model.

Locally weighted regression makes predictions by fitting a local surface to nearby training points. The critical element in this fitting is the distance weighting function, which calculates the influence that a particular observed location will have on the prediction of some unobserved location. We have chosen to use a Gaussian function for distance weighting:

$$w(x, x') = e^{-k(x-x')^2}$$

In this equation, a smoothing parameter k adjusts the spread of the Gaussian. A larger spread incorporates information from more distant configurations. In the following equations, x may be a multi-dimensional vector. In these cases, $(x - x')^2$ is computed as the dot-product of the difference of the two points. To perform the regression we use the LOESS technique proposed by Cleveland and Devlin [9]. Following Cohn et al. [10], we fit a Gaussian distribution to the region surrounding our query point. A Gaussian distribution is a statistical model which provides a distribution of outputs y for some input x. For the purpose of approximating the configuration space, x is a vector whose value is a point in configuration space and y is a value classifying the state (obstructed or free) of the point. The parameters of the Gaussian distribution fit to the local region are derived below, where x is the query point whose classification we are interested in and x_i,y_i are members of the sets of input and output training data X, Y.

The means of the Gaussian are calculated as follows:

$$\mu_x = \frac{\sum_i (w(x, x_i) x_i)}{\sum_i w(x, x_i)}$$

$$\mu_y = \frac{\sum_i (w(x, x_i) y_i)}{\sum_i w(x, x_i)}$$

The variances of the Gaussian are calculated as follows:

$$\sigma_x^2 = \frac{\sum_i (w(x, x_i)(x_i - \mu_x)^2)}{\sum_i w(x, x_i)}$$

$$\sigma_y^2 = \frac{\sum_i (w(x, x_i)(y_i - \mu_y)^2)}{\sum_i w(x, x_i)}$$

The covariance is given by:

$$\sigma_{xy} = \frac{\sum_i (w(x, x_i)(x_i - \mu_x)(y_i - \mu_y))}{\sum_i w(x, x_i)}$$

The conditional variance is given by:

$$\sigma_{y|x}^2 = \sigma_y^2 \frac{\sigma_{xy}^2}{\sigma_x^2}$$

Using this parameterization, we can then calculate the expected value of the output $\hat{y}$. This value is the prediction for our query point x.

$$\hat{y} = \mu_y + \frac{\sigma_{xy}}{\sigma_x^2}(x - \mu_x)$$

The variance of this prediction ($\sigma_{\hat{y}}^2$) is given by:

$$\begin{aligned} \sigma_{\hat{y}}^2 \quad = \quad & \frac{\sigma_{y|x}^2}{(\sum_i w(x, x_i))^2} \left(\sum_i w(x, x_i)^2 + \right. \\ & \left. \frac{(x - \mu_x)^2}{\sigma_x^2} \sum_i w(x, x_i)^2 \frac{(x_i - \mu_x)^2}{\sigma_x^2} \right) \end{aligned}$$

So far we have shown how to use a collection of samples to construct an approximate model of configuration space. Because the model is fit locally to a particular query point, it is only computed to answer a particular query. This means that the addition of data to the model only requires constant computation. We next discuss how this model can be constructed incrementally by placing samples that maximally improve the model.

3.2 Sampling Strategy

The task of modeling configuration spaces differs in one important way from many traditional machine learning tasks. While a learner can not generally select the training data available to it, a sampling-based motion planner has the ability to query any location in configuration space. The information obtained from this query is then added to the model. Because the planner can select configurations to explore and explorations are costly, each exploration should be carefully chosen to maximize the resulting improvement of the model. The subfield of active learning concerns itself with precisely these sorts of situations where a machine learning algorithm is empowered to select for training data for itself. A variety of methods for selecting a next query have been suggested. We use a method which is statistically near-optimal at improving locally weighted regression models [10]. This approach selects configurations in an attempt to minimize the variance of the resulting statistical model.

The variance in a model can be seen as the sum of three terms [15]: the variance of the output given identical inputs (i.e. the "noise" in the data); the bias of the model toward particular outputs; and the variance of the model. In general, we have no control over the noise in the data, and in the particular case of motion planning there is no noise (although one imagines this is a place in which sensor error could be taken into account). Since the model is assumed to be unbiased, error in our prediction is due solely to the variance in the model. A strategy which examines configurations in an effort to reduce this variance is assured that from its current perspective, the configuration it selects is the one that leads to the greatest improvement in model accuracy.

Cohn et al. [10] provide the derivation used to estimate the expected variance of a learner resulting from the selection and examination of a particular configuration $\tilde{x}$.

The new mean of the Gaussian fit to the local region is given by:

$$\tilde{\mu}_x = \frac{(\sum_i w(x, x_i))\mu_x + w(x, \tilde{x})\tilde{x}}{(\sum_i w(x, x_i)) + w(x, \tilde{x})}$$

The new variance and expected variance of the Gaussian fit to the local region are:

$$\tilde{\sigma}_x^2 = \frac{(\sum_i w(x, x_i))\sigma_x^2}{\sum_i w(x, x_i) + w(x, \tilde{x})} + \frac{(\sum_i w(x, x_i))w(x, \tilde{x})(\tilde{x} - \tilde{\mu}_x)^2}{(\sum_i w(x, x_i) + w(x, \tilde{x}))^2}$$

$$\langle \sigma_y^2 \rangle = \frac{(\sum_i w(x, x_i))\sigma_y^2}{\sum_i w(x, x_i) + w(x, \tilde{x})} + \frac{(\sum_i w(x, x_i))w(x, \tilde{x}) \left(\sigma_{y|\tilde{x}}^2 + (\hat{y}(\tilde{x}) - \mu_y)^2 \right)}{(\sum_i w(x, x_i) + w(x, \tilde{x}))^2}$$

The expected covariance and squared covariance of the Gaussian fit to the local region are:

$$\langle \tilde{\sigma}_{xy} \rangle = \frac{(\sum_i w(x, x_i))\sigma_{xy}}{\sum_i w(x, x_i) + w(x, \tilde{x})} + \frac{(\sum_i w(x, x_i))w(x, \tilde{x})(\tilde{x} - \mu_x)\hat{y}(\tilde{x}) - \mu_y)}{(\sum_i w(x, x_i) + w(x, \tilde{x}))^2}$$

$$\langle \tilde{\sigma}_{xy}^2 \rangle = \langle \tilde{\sigma}_{xy} \rangle^2 + \frac{(\sum_i w(x, x_i))^2 w(x, \tilde{x})^2 \sigma_{y|\tilde{x}}^2 (\tilde{x} - \mu_x)^2}{(\sum_i w(x, x_i) + w(x, \tilde{x}))^4}$$

The expected conditional variance is calculated from these terms:

$$\langle \tilde{\sigma}^2_{y|x} \rangle = \langle \tilde{\sigma}^2_y \rangle - \frac{\langle \tilde{\sigma}^2_{xy} \rangle}{\tilde{\sigma}^2_x}$$

As previously, we can put these terms together to calculate the variance, this time expected, of the model's prediction:

$$\begin{aligned} \langle \tilde{\sigma}^2_{\hat{y}} \rangle \quad = \quad & \frac{\langle \tilde{\sigma}^2_{y|x} \rangle}{(\sum_i w(x, x_i) + w(x, \tilde{x}))^2} \times \\ & \left[\sum_i w(x, x_i)^2 + w(x, \tilde{x})^2 + \frac{(x - \tilde{\mu_x})^2}{\tilde{\sigma}^2_x} + \right. \\ & \left. \left(\sum_i w(x, x_i)^2 \frac{(x_i - \tilde{\mu}_x)^2}{\tilde{\sigma}^2_x} + w(x, \tilde{x}) \frac{(\tilde{x} - \tilde{\mu_x})^2}{\tilde{\sigma}^2_x} \right) \right] \end{aligned}$$

This equation gives us the expected variance of the prediction for a particular configuration x. To obtain an estimate of the variance of the model in general, we calculate the variance for a set of reference points distributed throughout the configuration space.

The proposed sampling strategy then consists of selecting a configuration $\tilde{x}$ which maximizes the expected reduction in variance of the model. In practice, the $\tilde{x}$ is selected by evaluating the expected reduction in variance of a number of candidate points selected at random. Cohn et al. [10] note that hill-climbing may also be used to find $\tilde{x}$, but we have not found this to be necessary. The result is a sampling strategy that only queries sample points at which the model has high variance. A large reduction in variance translates into a large improvement of the model as a consequence of examining a sample. Since the variance of the model will be low in regions that are well understood, this sampling strategy naturally directs computational resources to complex regions of configuration space.

The combination of locally weighted regression and its active learning method give a practical manner for the approximate model to guide the sampling of configurations. Whenever a configuration space sample is required by the motion planner, the sampling strategy examines the state the locally weighted regression of configuration space. A configuration which minimizes the expected variance of the model is selected. The configuration is observed to see if it is obstructed or free and this information is added to the approximate model of the configuration space. If the configuration is free it is added to the roadmap.

The proposed approach of representing configuration space is evaluated experimentally in Section 4.1.

3.3 A Model-Based Motion Planner

We can devise a model-based motion planner by replacing the underlying model and sampling strategy in the traditional PRM approach. The resulting motion planner differs as follows. Before roadmap construction begins, an initial approximate model is constructed from a small number of configurations selected uniformly at random. These initial configurations are not added to the roadmap. Since they are drawn uniformly at random, they are less likely to be useful to the roadmap and would result in an increased number of edge validations. Once the initial model has been built, the algorithm proceeds in the traditional manner, but using the model-based sampling strategy, as described in Section 3.2. This strategy selects a configuration with maximum expected improvement to the model. This configuration is checked for collision and, if free, is incorporated into the roadmap. Here, another important difference to traditional PRM planners has to be noted: the configuration is added to the model, irrespective of whether it is free or obstructed. The

MODELBASEDMOTIONPLAN(INIT, ITERATION) : ROADMAP

do *init* times
  Select a random configuration x
  Add x to the initial data set D
Construct a model M from D
do *iteration* times
  Select a configuration x based upon active learning
  Add x to the model M
  if x is free
    Add x to the roadmap R
    foreach x_i in the set of n configurations nearest to x
      Connect x to x_i if the path is possible.

Figure 1: A model-based motion planner

traditional PRM framework discards obstructed samples. The resulting algorithm is given in Figure 1. In Section 4.2 experimental results for this model-based planner are presented.

This version of a model-based planner does not take full advantage of the information available in the model. The sampling strategy exploits the model to guide sampling, but the planner does not use the predictive power of the model. A better model-based motion planning algorithm also exploits the predictive power to reduce the overall computational requirements of solving a particular motion planning problem.

3.4 Predictive Edge Validations

Edge validation is one of the most expensive parts of the construction of a traditional roadmap [17] (cf. Figure 4.2). The approximate model presented in Section 3.1 also provides an opportunity to ease this computational cost. The ability of the model to efficiently predict the state of unexplored space can be used to predict if an edge is collision-free. The outcome of this prediction is used to determine if the computationally expensive examination of the edge using a collision checker is warranted. We show how locally weighted regression presented above can be augmented to allow predictive edge validations. We also develop a model-based motion planner that uses this capability.

We first modify the locally weighted regression algorithm to give predictions for a line rather than a point. When predicting if an edge is free or obstructed, the contribution of a particular sample in the model is determined by the distance to the nearest point on that edge. For some edge e, and a training point x_i the weight is given by:

$$w'(e, x_i) = e^{-k(\text{NearestPoint}(e,x_i)-x_i)^2}$$

This weight function, w', is substituted for the weight function, w, in the derivation of the Gaussian's parameters detailed earlier in Section 3.1. In order to calculate the prediction ($\hat{y}$) for the edge, the distance between the edge and the mean of the regressed Gaussian is needed. We use the point on the edge nearest to the Gaussian's mean (NearestPoint$(e, \mu_x) - \mu_x$)) and calculate the prediction of $\hat{y}$ as follows:

$$\hat{y} = \mu_y + \frac{\sigma_{xy}}{\sigma_x^2}(\text{NearestPoint}(e, \mu_x) - \mu_x).$$

Now that locally weighted regression can give predictions for a line, we might simply use a single prediction for an edge. However, this may result in false predictions for edges that are split between free and obstructed

```
PREDICTIVEEDGECHECK(E, MODEL) : BOOLEAN
  obsProb := obstructedProbability(e,model)
  if (obsProb > minObstructedProbability)
        return false
  else if (1-obsProb > minFreeProbability)
        return true
  else
        e' = firstHalf(e)
        if (PredictiveEdgeCheck(e', model))
              e" = secondHalf(e)
              return PredictiveEdgeCheck(e", model)
        else
              return false
```

Figure 2: The predictive edge checking algorithm

space. Because prediction averages over the entire edge, the resulting prediction is uncertain (as it should be).

To address this, and adapt our line prediction to the task of edge checking, we first calculate a prediction for the entire line. If the probability of the edge being free or obstructed is above a threshold, that prediction is taken for the entire edge. If the prediction is uncertain, the edge is divided in half, and each half is recursively tested. If either half is predicted obstructed, the entire edge is predicted obstructed, otherwise the entire edge is predicted free. This algorithm is given in Figure 2. Section 4.1 provides experimental evidence that this method can achieve satisfactory accuracy while at the same time significantly reducing the computational cost of edge validation.

3.5 A Model-Based Motion Planner with Predictive Edge Validation

As mentioned previously, edge validation is the predominant computational cost in the construction of a traditional probabilistic roadmap. This fact can readily be seen in the experimental results shown in Figure 4.2. For the traditional PRM planner, more than three-quarters of the time is spent validating edges. This computational cost is often unnecessary. Many edges are actually redundant. They could be removed from the final roadmap without affecting the completeness of the roadmap. Computation used to check such redundant edges is wasted since a path along another edge was already possible.

We have shown that the approximate model used to guide sampling can provide predictions about the state of unobserved edges (Section 3.4). This enables an alternative to the expensive construction a roadmap in the traditional sense. We can construct a predictive roadmap in which certain edges have been validated using the model without invoking a collision checker. Like a traditional roadmap, graph search is used to find a path in the predictive roadmap from start to goal location. However, since the path has only been predictively validated, it is necessary to verify the path is collision free. When edges are verified they are marked as such so that they are only checked once. Graph search in the predictive roadmap is is biased toward verified edges. This ensures that redundant edges in the roadmap are never examined.

For most edges in a potential path, the prediction that it is free is correct and the edge is retained as part of the path. Occasionally, edges classified as collision-free by the model are found to be obstructed when validated with a collision checker. On such occasions, the offending edge is removed from the predictive

```
PREDICTIVEMODELBASEDMOTIONPLAN(INIT, ITERATION, START, END) : PATH
    do init times
        Select a random configuration x
        Add x to the initial data set D
    Construct a model M from D
    do iteration times
        Use active sampling to determine configuration x
        Add x to model M
        if x is free according to model M
          Add x to the roadmap R
          foreach x_i in the set of its n neighbors
              if path between x and x_i is free in M
                Connect x to x_i
    return EXTRACTPATH(START, END, R)

EXTRACTPATH(START, END, ROADMAP) : PATH
    do
        p := DykstraPathPlan(start, end, roadmap)
        for each edge e(x_i, x_j) in p
          if e is in collision
              resample between x_i and x_j
                and add configurations to roadmap
          else
              mark e validated
    while p is not a valid path
    return p
```

Figure 3: A predictive model-based motion planner

roadmap and repair to the path is attempted. Repair consists of resampling configurations in the vicinity of the removed edge and attempting to reconnect the graph through the newly sampled configurations. The repair process is similar to those used by others [5, 29, 18]. Infrequently, the repair process fails. In such a case, roadmap construction is resumed until a new candidate path between start and goal is found. This process repeats until a path is found (see Figure 3).

The predictive roadmap algorithm marks a middle ground between multi-query motion planners and single-query motion planners. The predictive roadmap contains general but incomplete information relevant to the construction of any possible path. The verification and repair of edges in the roadmap as a result of a particular planning query focuses computation on those areas most relevant to the solution of that query. As a consequence, the predictive model-based motion planning approach is ideally suited for dynamic environments. In the absence of a particular motion planning query, computational resources are directed toward improvement of the overall configuration space model. A particular query can exploit this information to quickly find a collision free path. Such a planner is expected to outperform single-query methods presented in the literature since they do not have such initial information available to them.

The predictive model-based motion planner presented here shares some characteristics with traditional

sampling-based motion planners, but also distinguishes itself from them in several important aspects. Similarly to traditional approaches, it obtains information about the configuration space by sampling. By doing so, it takes advantage of the computational gains achieved by sampling-based methods over complete motion planners. Like traditional sampling-based approaches, the solution to the motion planning problem is ultimately found by building a graph-based representation, or roadmap. Model-based motion planners, however, build a more expressive underlying model of configuration space information. The information contained in this model is exploited to adapt the local sampling density to the complexity of the configuration space region. This adaptation effectively addresses the narrow passage problem. The model is also used to make predictions about unexplored regions. These differences permit the motion planner to make maximum expected progress toward finding a solution to the motion planning problem, given the information represented in the model. Due to this important conceptual and algorithmic distinction, we consider model-based motion planning to be a novel category of sampling-based motion planners.

4 Experimental Validation

To provide empirical evidence in support of the proposed approach, we perform a number of experiments. We evaluate the accuracy of the proposed model for representing configuration space and for edge prediction (see Section 4.1). The performance of the two model-based motion planners introduced in Sections 3.3 and 3.5 is compared with the traditional PRM approach [21] and a motion planner based on the hybrid bridge test [17] (see Section 4.2).

4.1 Validation of Proposed Model

To assess locally weighted regression's ability to be an accurate approximate representation of the configuration space empirically, we build an approximate model based on a selection of labeled training configurations and then test it on a test set of previously unseen configurations. Training configurations are chosen using active learning (Section 3.2), and test configurations are chosen uniformly at random. The accuracy of the resulting models as a function of the number of configurations used to train them is given in Figure 4. The experimental environment is shown in Figure 8. Environments containing arms with three, four, and six degrees of freedom are used for testing. The results are averaged over ten experimental runs.

We also evaluate the accuracy of the predictive edge checker (Section 3.4). The model is trained using a number of labeled training configurations chosen by active learning. It is then tested on a number of test edges. The test edges are randomly selected, straight-line trajectories between two free configurations whose distance is less than the threshold used for connecting neighbors in the roadmap construction algorithm. This experimental scenario approximates the situation in a real motion planner as closely as possible. All results shown are averages over ten runs of the algorithm.

The results indicate that locally weighted regression can accurately estimate the state of unseen regions of configuration space. Even with only two or three hundred training configurations, the approximate model is capable of predicting the state of both edges and configurations with sufficient accuracy.

We are also interested in the time that the approximate model takes to predict a particular configuration or edge, since one of the uses of the approximate model is speedy estimations of the state of a configuration or edge. These results are shown in Figures 5 and 7. From the results it is clear that the computational cost of predictive configuration checking or predictive edge validation is linear in the number of configurations used to build the model. This is consistent with the computational complexity of locally weighted regression. It is also our judgment from the accuracy versus time trade-off that predictions about individual

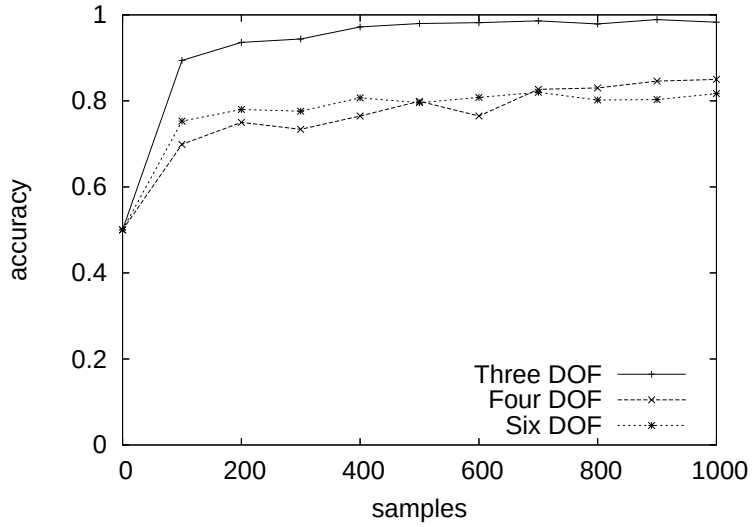
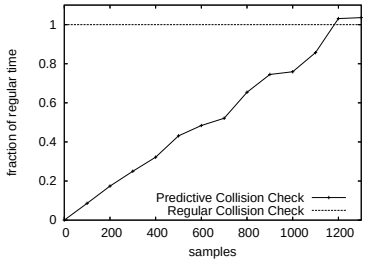
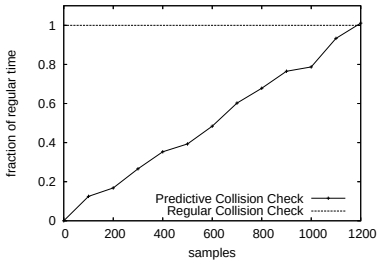


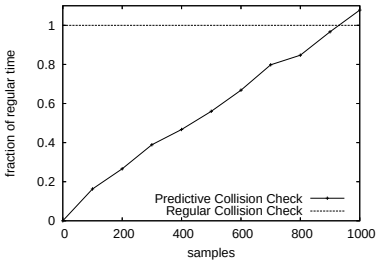
Figure 4: Accuracy of predictive collision checking as a function of the number of training configurations for three, four and six degrees of freedom



(a) 3 DOF



(b) 4 DOF



(c) 6 DOF

Figure 5: Execution time of predictive collision checking as a function of the number of training configurations for three, four and six degrees of freedom

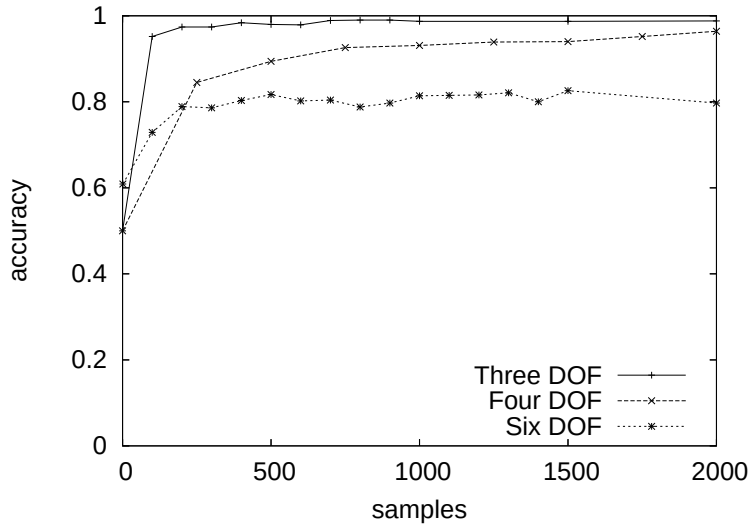
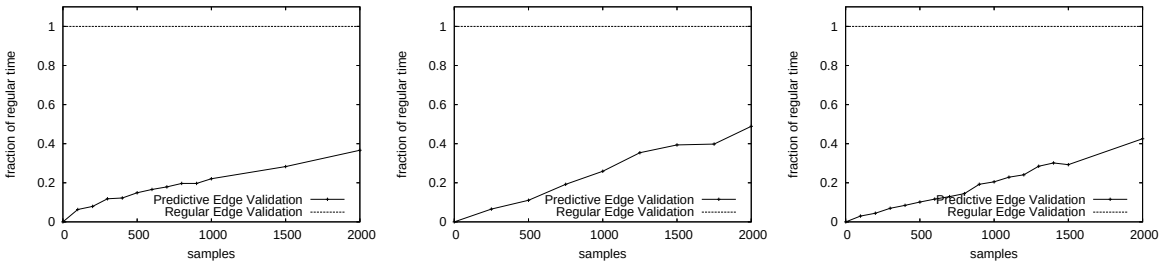


Figure 6: Accuracy of predictive edge validation as a function of the number of training configurations for three, four and six degrees of freedom



(a) 3 DOF (b) 4 DOF (c) 6 DOF

Figure 7: Execution time of predictive edge validation as a function of the number of training configurations for three, four and six degrees of freedom

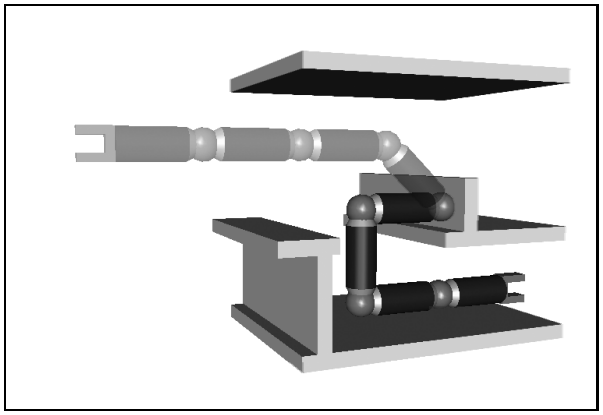


Figure 8: The initial (transparent) and final (solid) configuration of a twelve degree of freedom arm in the experimental environment.

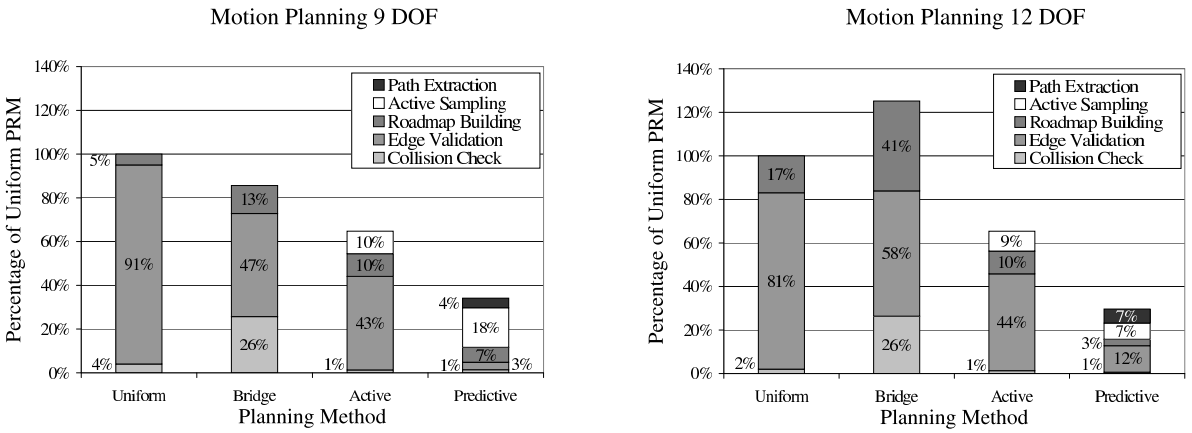
configurations are not worthwhile. Even though a two times speed-up with predictive collision checking is possible, the absolute time gain does not outweigh the reduction in accuracy. In our experience, successful roadmaps generated with the proposed active learning sampling strategy contain relatively few configurations or milestones (fewer than a thousand for all of the motion planning experiments we performed). Therefore, the overall gain in computation time offered by predictive collision checking is less than half a second. However, the detrimental effect of mis-prediction on the construction of a roadmap is large.

Predictive edge validation, on the other hand, can offer a significant reduction of computational cost with a limited detrimental effect. Since an edge prediction can be obtained with good accuracy at a tenth of the computational expense of performing an edge validation with a collision checker, its use in motion planning is more compelling. In addition, the misprediction of an edge is less damaging to roadmap construction than the misprediction of a milestone. This is especially true in the case of edges thought to be free that later prove to be obstructed. Often, they are repaired to produce valid paths.

4.2 Evaluation of Model-Based Motion Planning

To evaluate the proposed model-based planning approach, we perform path planning experiments with two simulated arms with nine and twelve degrees of freedom (DOF). The twelve degree-of-freedom version of the arm with its initial and final configuration are shown in Figure 8. Note that the final configuration of the robot is inside the most confined region of the workspace. We make the assumption that the corresponding configuration space region exhibits very high complexity. Consequently, even the predictive model-based motion planner presented in Section 3.5 will build a relatively detailed model of the entire configuration space before finding a solution. In other words, any planner which can successfully plan for the experimental scenario, is likely to have explored the entire configuration space and will be able to answer subsequent queries in constant time. By choosing the experimental scenario in this manner, the difference between the multi-query approach to motion planning and the hybrid multi-query/single-query nature of predictive model-based motion planning is minimized. The experimental results given here therefore represent a fair comparison.

In our experimental evaluation we compare the performance of a PRM planner with uniform sampling [21], a PRM implementation using hybrid bridge sampling [17], the model-based motion planner described in Section 3.3, and the predictive model-based motion planner described in Section 3.5. Each



(a) Motion planning with nine degrees of freedom

(b) Motion planning with twelve degrees of freedom

Figure 9: Time to find a successful path for a nine and twelve degree of freedom arm as a percentage of the PRM planner with uniform sampling and the percentage of the time taken by each algorithmic component of sampling-based motion planning.

algorithm runs until a successful path between the start and goal positions is found. In the case of the predictive roadmap, the time to verify and repair (if necessary) the candidate path in the predictive roadmap was included in the overall time. Verification and repair time is labeled "Path Extraction" in the bar graphs in Figure 4.2. The times given represent the average performance over ten runs of each algorithm. They are reported as fractions of the time that it took the traditional PRM planner with uniform sampling to solve the problem. We performed experiments for nine and twelve degree of freedom arms. The nine degree of freedom arm consisted of three links connected by spherical joints. The twelve degree of freedom arm consisted of four links also connected spherical joints (see Figure 8).

From the results in Figure 4.2, it can be readily seen that active sampling is an improved sampling technique over hybrid bridge sampling and uniform sampling. Its choice of points results in faster motion planning even though querying the model to determine the next sample imposes an additional computational cost. Further, the amount of improvement (around a 40% decrease in time) appears constant as the degree of freedom increases, suggesting that the performance of the approach may degrade gracefully for motion planning in very high-dimensional configuration spaces.

The use of the predictive roadmap results in even greater increases in performance (nearly a three times speed-up). It is interesting to contrast where computation is spent in the three traditional roadmap methods versus the predictive roadmap. In each of the three traditional roadmap methods at least a third of the time (much more in the case of uniform sampling) is spent checking edges for validity. Since the predictive roadmap does its edge checking in the model, the amount of time it spends checking edges is significantly less (less than ten percent). Of course, the predictive roadmap pays a price for the potential inaccuracy of its roadmap, using a third of its computational time to perform path verification and repair, but this cost is clearly offset by the computational savings from predictive edge checking.

Lastly, it should be noted that the performance of the PRM planner with hybrid bridge sampling degrades with respect to the PRM planner with uniform sampling from the nine degree-of-freedom example to the twelve degree-of-freedom example. This observation illustrates one of the shortcomings of existing sam-

pling strategies: they are dependent upon the environment and parameter settings. In our example, the bridge sampling techniques falsely identifies many configurations as important and enters them into the roadmap. These false classifications introduce a number of unnecessary edges into the roadmap. The time required to validate these unnecessary edges results in an overall decrease in performance. Since model-based motion planning is inherently adaptive to the environment, such problems cannot arise.

5 Conclusion

We have proposed a novel sampling-based motion planning approach. Its main distinguishing feature is the underlying representation of configuration space information used for path planning. We refer to this representation as a model of configuration space. The proposed model differs from the graph-based representations of previous sampling-based approaches in that it generalizes across information obtained from our exploration of the configuration space and allows predictions about unexplored parts of configuration space. The model also provides a measure of confidence for these predictions. A model-based motion planner exploits the information from the model to provide a significant reduction in computational requirements for solving high-dimensional motion planning problems.

Model-based motion planning adapts the sampling density to the local complexity of configuration space regions. As a result, computational resources are expended in proportion to the local difficulty of a configuration space region. This effectively addresses the narrow passage problem of sampling-based motion planning techniques that has been the focus of much research over the past decade. The adaptation of the sampling density is achieved by incrementally sampling those regions of configuration space for which the model is least certain of its predictions. By proceeding in this fashion, every additional sample ensures maximum expected improvement for the model. Adapting sampling in this manner results in a necessarily detailed roadmap in complex regions and a simple roadmap in simple regions.

Model-based motion planning is capable of avoiding much configuration space exploration by exploiting the predictive power of the underlying model. Rather than performing edge validations by repeatedly invoking a computationally costly collision checker, the model can be efficiently queried to predict whether an edge is free or obstructed. This prediction is based on known sample points in proximity to the edge in question. Since in the traditional PRM framework edge validations are the most costly operation, this results in a substantial increase in computational efficiency.

Model-based motion planning provides a novel approach to motion planning, since it combines characteristics of multi-query and single-query approaches. This characteristic makes it ideally suited for motion planning in dynamic environments, where parts of the model are repeatedly invalidated and require updating. This interesting property is also a consequence of the predictive capabilities of the underlying model. The model provides an approximate global understanding of configuration space that allows to answer a specific query efficiently by directing computational resources to those regions most likely to contain the required solution.

We present experiments demonstrating that the underlying model is indeed capable of representing configuration space accurately and that it is capable of providing accurate and efficient predictive edge validations. We also compare the performance of model-based motion planning to other sampling-based motion planning techniques. Our experiments show that the exploitation of information represented in the model, as performed by the proposed method, results in a dramatic reduction of the the computational cost of motion planning in high-dimensional configuration spaces.

References

[1] N. Ahuja, R. T. C. dn R. Yen, and N. Bridwell. Interference detection and collision avoidance among three dimensional objects. In *Proceedings of the First AAAI Conference*, pages 44–48, 1980.

[2] N. Amato, B. Bayazid, L. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*. AK Peters, 1998.

[3] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

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

[5] R. Bohlin and L. E. Kavraki. Path planning using lazy PRM. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 521–528, San Francisco, USA, 2000.

[6] V. Boor, M. Overmars, and F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of the International Conference on Robotics and Automation*, 1999.

[7] B. Burns and O. Brock. Information theoretic construction of probabilistic roadmaps. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 650–655, Las Vegas, 2003.

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

[9] W. Cleveland and S. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83:596–610, 1988.

[10] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical methods. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[11] C. I. Connolly and R. A. Grupen. One the applications of harmonic functions to robotics. *Journal of Robotic Systems*, 10(7):931–946, 1993.

[12] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via theem algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[13] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, 2001.

[14] M. Foskey, M. Garber, M. C. Lin, and D. Manocha. A Voronoi-based hybrid motion planner. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 1, pages 55–60, Maui, USA, 2001.

[15] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[16] C. Holleman and L. E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 1408–1413, San Francisco, USA, 2000.

[17] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proceedings of the International Conference on Robotics and Automation*, 2003.

[18] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, pages 141–154. A K Peters, 1998.

[19] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of the International Conference on Robotics and Automation*, volume 3, pages 2719–2726, 1997.

[20] D. Hsu and Z. Sun. Adaptive hybrid sampling for probabilistic roadmap planning. Technical Report TRA5/04, National University of Singapore, 2004.

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

[22] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[23] D. E. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of the International Conference on Robotics and Automation*, pages 1–6, 1987.

[24] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[25] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Iowa State University, 1998.

[26] J.-M. Lien, S. L. Thomas, and N. M. Amato. A general framework for sampling on the medial axis of the free space. In *Proceedings of the International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[27] T. Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(10):681–698, 1981.

[28] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. Amato. A machine learning approach for feature-sensitive motion planning. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2004.

[29] C. L. Nielsen and L. E. Kavraki. A two level fuzzy PRM for manipulation planning. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1716–1722, Takamatsu, Japan, 2000.

[30] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.

[31] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 802–807, 1993.

[32] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 421–427, 1979.

[33] D. Rumelhart and J. McClelland, editors. *Parallel Distributed Processing*. MIT Press, 1986.

[34] B. Schölkopf and A. J. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, 2002.

[35] T. Siméon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Journal of Advanced Robotics*, 14(6):477–494, 2000.

[36] A. G. Sukharev. Optimal strategies of the search for an extremum. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 11(4):119–137, 1971. Translated from Russian, *Zh. Vychisl. Mt. i Mat. Fiz.*, 11(4):910-924.

[37] S. Tong and D. Koller. Support vector machine active learning with applications to text classifiction. *Journal of Machine Learning Research*, 2:45–66, 2001.

[38] Y. Yang and O. Brock. Adapting the sampling distribution in prm planners based on an approximated medial axis. In *Proceedings of the International Conference on Robotics and Automation*, pages 4405–4410, New Orleans, USA, April 2004.