2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Machine Learning Guided Exploration for Sampling-based Motion Planning Algorithms

Oktay Arslan Panagiotis Tsiotras

Abstract— **We propose a machine learning (ML)-inspired approach to estimate the relevant region of a motion planning problem during the exploration phase of sampling-based path-planners. The algorithm guides the exploration so that it draws more samples from the relevant region as the number of iterations increases. The approach works in two steps: first, it predicts if a given sample is collision-free (classification phase) without calling the collision-checker, and it then estimates if it is a promising sample, i.e., if it has the potential to improve the current best solution (regression phase), without solving the local steering problem. The proposed exploration strategy is integrated to the $\mathrm{RRT}^{\#}$ algorithm. Numerical simulations demonstrate the efficiency of the proposed approach.**

I. INTRODUCTION

Sampling-based motion planning algorithms need to incorporate efficient exploration strategies in order to gather information about the possibly high-dimensional search space. Most exploration strategies implement a form of rejection sampling in order to collect a large number of collision-free configurations. Rejection sampling is used mainly owing to its implementation simplicity. However, this approach is redundant since not all collision-free configurations have the potential to be part of the optimal solution at a given query. In this paper, we propose to use machine learning ideas in order to estimate the relevant region of the search space, that is, the region where the optimal path is more likely to be found. The result is that future samples are drawn from the relevant region with increased probability, as the number of iterations increases.

The idea of using machine learning fits naturally within the framework of sampling-based algorithms since an incremental dataset describing the topology of the configuration space is readily available from previous samples. Robotic motion planning can be interpreted as a form of learning problem, since the high-dimensional configuration search space is not known explicitly a priori. Therefore, its solution inherently poses a fundamental problem, which is known as the *exploration versus exploitation dilemma*. Specifically, as the motion planner starts gathering more information about the search space, it may favor exploitation of the current knowledge in order to improve the best solution which has been computed so far. This is mainly due to the fact that the space requirements of an exact representation of the configuration space grows very quickly with the problem dimensionality and the number of obstacles, and hence exhaustive search is impractical (for example, it has been proven in [20] that the general motion planning problem is PSPACE-complete). Therefore, a motion planner needs to devote some time for exploitation, i.e., to produce a "good enough" solution based on the available information, as exploration progresses. However, concentrating only on improving the current best solution may cause the planner to get stuck in a local minimum since potentially better solutions may have not been explored yet. Therefore, a motion planner should perform the exploration and exploitation tasks in harmony, striking a balance between the two.

Despite the recent advances in the development of motion planners for high-dimensional spaces using sampling-based methods [15], [16], [14], [1], [18], [13], efficient exploration still remains a challenging issue for sampling-based planners [17]. However, for many problems an admissible heuristic and an approximation of the optimal cost-to-come (or cost-to-go) function is available during the search. In such cases, one can characterize the relevant region of the given task, i.e., the subset of the search space which contains the optimal solution. In this context, exploration can be viewed as a problem of learning the relevant region associated with the given task. Interestingly, the authors of this paper have shown that this region can be approximated incrementally as a by-product of the $\mathrm{RRT}^{\#}$ algorithm [1].

In this work, we follow up on this insight and use machine learning ideas in order to achieve better exploration of the search space. This is based on the simple fact that since incremental sampling-based algorithms collect a lot of data about the planning problem as iterations progress, one can utilize this information to provide informative labels of the collected samples (obstacle or free) and to associate approximate cost (utility) values with each sample. These labels, collectively, can be used to guide the selection of future samples. By employing active learning and by inferencing based on the collected data, one can guide future samples toward the favorable region of the search space without invoking the computationally expensive collision checking and local steering procedures [9], thus speeding up the algorithm.

The proposed adaptive sampling strategy is integrated in the $\mathrm{RRT}^{\#}$ algorithm to guide the future exploration of the search space. We give a detailed explanation of the proposed adaptive sampling strategy in the subsequent sections. Our numerical simulations demonstrate that the proposed adaptive sampling strategy can reduce the number of vertices

Oktay Arslan is a PhD candidate with the Institute for Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, GA 30332-0150, USA, Email: oktay@gatech.edu

Panagiotis Tsiotras is with the faculty of D. Guggenheim School of Aerospace Engineering and the Institute for Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, GA 30332-0150, USA, Email: tsiotras@gatech.edu

in the underlying search graph significantly, yet the path-planner is able to produce high quality paths.

II. Related Work

A plethora of approaches have been developed in order to guide the sampling strategy toward a specific part of the configuration space. A comparative study of these approaches can be found in [10]. The majority of these methods try to address the so-called "narrow passage problem," which deals with drawing more samples from difficult and complex parts of the configuration space [12], [11]. More recently, Bialkowski has proposed an approach which guides samples towards the free space [4]. In [6], the authors have proposed an entropy-guided exploration strategy to guide sampling toward the regions that would yield maximal expected information gain. The approach is elegant but owing to the computations involved it appears to be more appropriate for an off-line construction of the graph, which is the case for multi-query algorithms, such as PRM. In [8] the authors keep a history of extension attempts for each state and the results (success or fail), which are used to compute the utility of each state in an information-theoretic sense. This algorithm guides exploration in a way so as to increase the overall utility.

Most closely related to our work is the work in [7] and [19]. In [7] the authors propose an approximate approach based on machine learning ideas for multi-query algorithms that attempts to reduce collision-checking time. However, the current work differs from this approach in several ways. First, the authors of [7] only focus on adapting the sampling strategy to collect more collision-free samples, not for learning the relevant region. Their exploration strategy is tailored to reduce the variance of an approximated model of the configuration space. Although *active sampling* generates useful samples which yield an accurate model, these configurations are not necessarily related to the solution of the task of interest. This is mainly due to the fact that the approach in [7], similarly to [6], is designed for multi-query algorithms, i.e., the task is not known a priori. Second, our approach is built on fundamentally different assumptions even when only the "learning the configuration space problem" (classification) is considered. In [7] it is assumed that the feature vector and its label (x, y) are jointly Gaussian, and the prediction algorithm is developed by utilizing the nice properties of Gaussian distributions (i.e., conditional distribution of multivariate Gaussian distribution is also a Gaussian). This assumption seems to be reasonable for typical classification problems when there is no information about the underlying structure. However, in the context of sampling-based motion planning, one has some extra information. For example, when rejection sampling is used, we know that the underlying class conditional distribution is a uniform pdf which is defined over a bounded domain and has support which is unknown but can be explored. The use of Gaussian distributions, which do not have compact support, conflicts with the underlying structure of the problem. In our approach, we leverage this key observation and estimate the class conditional distributions directly by estimators on a bounded domain, without resorting to the relaxation of y being continuous.

Finally, in [19], the authors proposed an approach which uses instance-based learning techniques. Their approach stores previous local planning queries and their outcomes (in-collision or collision-free) in a table. Then, a k-NN density estimator computes the probability of a given query point to be in-collision without calling the actual collision checker. The exact collision checking processes is postponed, and is performed only for those points that have a small probability of being in-collision. This idea is also used in our work and can be considered as an alternative way of estimating the posterior distribution. However, it is known that k-NN density estimators have several drawbacks, e.g., they are prone to local noise, yield an estimate distribution with heavy tails, etc; also, the resulting density estimate is not a true pdf since its integral over the whole configuration space diverges [5]. Our approach, on the other hand, does not have any such problems thanks to the nice properties of kernel density estimators.

III. Machine Learning Guided Exploration

A. Problem Formulation

Let $\mathcal{X}$ denote the configuration space, which is assumed to be an open subset of $\mathbb{R}^d$, where $d \in \mathbb{N}$ with $d \geq 2$. Let the *obstacle region* and the *goal region* be denoted by $\mathcal{X}_{\text{obs}}$ and $\mathcal{X}_{\text{goal}}$, respectively. The obstacle-free space is defined by $\mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$. Let the *initial configuration* be denoted by $x_{\text{init}} \in \mathcal{X}_{\text{free}}$. Let $\mathcal{G} = (V, E)$ denote a graph, where V and $E \subseteq V \times V$ are finite sets of vertices and edges, respectively. We will use graphs to represent the connections between a (finite) set of configuration points selected randomly from $\mathcal{X}_{\text{free}}$. Given a vertex $v \in V$, the function $\mathtt{g} : v \mapsto c$ returns a non-negative real number c, which is the cost of the path to v from a given initial state $x_{\text{init}} \in \mathcal{X}_{\text{free}}$. We will use $\mathtt{g}^*(v)$ to denote the optimal cost-to-come value of the vertex v which can be achieved in $\mathcal{X}_{\text{free}}$. Given a vertex $v \in V$, and a goal region $\mathcal{X}_{\text{goal}}$, the heuristic function $\mathtt{h} : (v, \mathcal{X}_{\text{goal}}) \mapsto c$ returns an estimate c of the optimal cost from v to $\mathcal{X}_{\text{goal}}$; we set $\mathtt{h}(v) = 0$ if $v \in \mathcal{X}_{\text{goal}}$. The function $\mathtt{h}$ is an admissible heuristic if it never overestimates the actual cost of reaching $\mathcal{X}_{\text{goal}}$. In this paper, we always assume that $\mathtt{h}$ is an admissible heuristic. We wish to solve the following problem:

Optimal motion planning problem: Given a bounded and connected open set $\mathcal{X} \subset \mathbb{R}^d$, the sets $\mathcal{X}_{\text{free}}$ and $\mathcal{X}_{\text{obs}} = \mathcal{X} \setminus \mathcal{X}_{\text{free}}$, an initial point $x_{\text{init}} \in \mathcal{X}_{\text{free}}$ and a goal region $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$, find the minimum-cost path connecting x_{init} to $\mathcal{X}_{\text{goal}}$.

In sampling-based algorithms, the planning algorithm avoids exhaustive discretization of the search space by randomly drawing configurations which are incorporated into a tree or a graph. The method of random generation of these configurations is called a *sampling strategy*. A good sampling strategy should adapt to the topology of the search space and provide information that can improve the computed solution.

Learning problem: Let $x^*_{\text{goal}} \in \mathcal{X}_{\text{goal}}$ be the point in the goal region that has the lowest optimal cost-to-come value

in $\mathcal{X}_{\text{goal}}$, i.e., let $x^*_{\text{goal}} = \text{argmin}_{x \in \mathcal{X}_{\text{goal}}} \mathtt{g}^*(x)$. The *relevant region* of $\mathcal{X}_{\text{free}}$ is the set of points x for which the optimal cost-to-come value of x, plus the estimate of the optimal cost moving from x to $\mathcal{X}_{\text{goal}}$ is less than the optimal cost-to-come value of x^*_{goal}, that is,

$$\mathcal{X}_{\text{rel}} = \{x \in \mathcal{X}_{\text{free}} : \mathtt{g}^*(x) + \mathtt{h}(x) < \mathtt{g}^*(x^*_{\text{goal}})\}. \quad (1)$$

Points that lie in $\mathcal{X}_{\text{rel}}$ have the potential to be part of the optimal path starting at x_{init} and terminating in $\mathcal{X}_{\text{goal}}$. Our goal is to learn $\mathcal{X}_{\text{rel}}$ and develop a sampling strategy that draws samples only from $\mathcal{X}_{\text{rel}}$. This problem can be formulated as a combination of a classification and a regression problem. First, we need to predict the label of a given arbitrary point $x \in \mathcal{X}$, and then its cost-to-come value needs to be computed approximately, using some regression technique to check if inequality (1) is satisfied.

B. Approach

Before explaining our approach, some terminology needs to be introduced. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the space of input and output values, respectively. Let $x^{(i)} \in \mathcal{X}$ be the *feature vector* of the *i*th example, also called "input" variables, and let $y^{(i)} \in \mathcal{Y}$ be its label, also called the "output" or *target* variable. A pair $\left(x^{(i)}, y^{(i)}\right)$ is called a *training example*. The *training set* is a list of m training examples of the form $\mathcal{D} = \left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\right\}$. In a supervised learning framework, given the training set $\mathcal{D}$, a learning algorithm seeks a function $\hat{y}_\ell : \mathcal{X} \mapsto \mathcal{Y}$ so that $\hat{y}_\ell(x)$ is a "good" predictor for the corresponding value of y. The function $\hat{y}_\ell$ is called a *hypothesis*, for historical reasons. The target variable that needs to be predicted can be continuous or it may take a finite number of discrete values. The learning problem is a *regression* problem when $\mathcal{Y}$ is continuous, and is a *classification* problem if $\mathcal{Y}$ is a discrete set.

Two problems arise in the context of sampling-based algorithms: a classification problem, i.e., the prediction of the label of an unobserved sample x, and a regression problem, i.e., the prediction of the optimal cost-to-come value of the sample x.

C. Learning the Configuration Space

Given the training set $\mathcal{D} = \left\{(x^{(i)}, y^{(i)}) : i = 1, \ldots, m\right\}$ where the pair $\left(x^{(i)}, y^{(i)}\right)$ denotes a randomly drawn point and its label computed by the collision-checker at the *i*th iteration, we wish to find a function $\hat{y}_{cs} : \mathcal{X} \mapsto \{-1, 1\}$ that gives a good prediction for determining if a given point x is in the obstacle space or the free space.

This problem can be solved efficiently via a Bayesian classifier, which makes decisions based on class conditional distributions and priors [5]. The approach computes two approximate probability density functions (pdf) in order to determine where the obstacle-free and obstacle spaces lie in the search space, based on the available data at any given iteration. Then, the classifier uses the Bayesian rule to predict if a given point x is in $\mathcal{X}_{\text{free}}$ or $\mathcal{X}_{\text{obs}}$. A real collision checking is performed *only* for points which are classified as collision-free. All of the samples, regardless if they are in collision or not, are stored in a list which forms the training set $\mathcal{D}$. A kernel density estimator is used to learn the associated class conditional distributions. The kernel density estimator $\hat{f}_X(x)$ for the estimation of the density value $f_X(x)$ at point x is defined as

$$\hat{f}_X(x) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{K}_{\mathbf{H}} \left(x - x^{(i)}\right), \quad (2)$$

where $\mathbf{H}$ is the bandwidth matrix (nonsingular) and $\mathcal{K}_{\mathbf{H}} : \mathbb{R}^d \mapsto \mathbb{R}$ denotes the multivariate kernel function which is defined as follows:

$$\mathcal{K}_{\mathbf{H}}(x) = \frac{1}{\det(\mathbf{H})} \mathcal{K}(\mathbf{H}^{-1} x). \quad (3)$$

We use a diagonal bandwidth matrix for the sake of simplicity, i.e., $\mathbf{H} = \mathtt{diag}(h, \ldots, h)$ where the kernel function $\mathcal{K}$ satisfies the following properties:

i) $\mathcal{K}$ is a density function, that is, $\int_{\mathbb{R}^d} \mathcal{K}(x)\, \mathrm{d}x = 1$ and $\mathcal{K}(x) \geq 0$.
ii) $\mathcal{K}$ is symmetric, that is, $\int_{\mathbb{R}^d} x\mathcal{K}(x)\, \mathrm{d}x = 0$.

Typical kernels involve the Uniform, Gaussian and Epanechnikov kernel functions [22]. In this work, we use the Epanechnikov kernel function

$$\mathcal{K}(x) = \frac{d+2}{2\zeta_d}(1 - x^\intercal x)\mathrm{I}(x^\intercal x \leq 1), \quad (4)$$

where $\mathrm{I}(\cdot)$ is the indicator function and ζ_d is the measure (volume) of the unit sphere in $\mathbb{R}^d$.

D. Proposed Adaptive Sampling Strategy

The proposed adaptive sampling strategy is given in Algorithm 1. First, the algorithm initializes the lists used to store the collected samples with the empty set and initializes the sampling pdf $\hat{f}_X$ with a pdf which is uniform over $\mathcal{X}$. Then, the algorithm incrementally samples from $\hat{f}_X$ in Line 5. In the subsequent step, a collision-checking operation is performed for the randomly generated sample x_{rand}. The sample x_{rand} is stored in either X_{free} or X_{obs} based on the result of the collision-checking.

In Lines 8 and 11, the `DensityEstimator` procedure implements a nonparametric density estimation method. In this work, we have implemented a kernel density estimator that uses the multivariate Epanechnikov kernel with variable, but uniform in all dimensions, bandwidth. In our implementation, the bandwidth h is updated as a function of the size of the training set $\mathcal{D}$ as follows

$$h \propto (\log(|\mathcal{D}|)/|\mathcal{D}|)^{1/d}. \quad (5)$$

The Epanechnikov kernel in (4) has been used instead of, say, a Gaussian kernel because of its finite support. This property makes querying the density value of a given point tractable. For any kernel of finite support, the summation in equation (2) needs to be performed for only the local neighbors of the query point. This neighbor set can be computed efficiently using specific spatial data structures, such as, kd-trees [21]. Simply, the density value of point x is computed using equation (2) and it predicts how likely is for the point x to be in the obstacle-free or obstacle spaces. In

Algorithm 1: Adaptive Sampling Algorithm

1 **AdaptiveSampling**($\mathcal{X}$)
2 $X_{\text{obs}} \leftarrow \emptyset$; $X_{\text{free}} \leftarrow \emptyset$;
3 $\hat{f}_X(\cdot) \leftarrow p_{\text{uniform}}(\cdot|\mathcal{X})$;
4 **for** $i = 1$ *to* N **do**
5 $x_{\text{rand}} \leftarrow \texttt{SampleDensity}(\hat{f}_X)$;
6 **if** $\texttt{OnObstacle}(x_{\text{rand}})$ **then**
7 $X_{\text{obs}} \leftarrow X_{\text{obs}} \cup \{x_{\text{rand}}\}$;
8 $b_{\text{obs}} \leftarrow \texttt{DensityEstimator}(X_{\text{obs}})$;
9 **else**
10 $X_{\text{free}} \leftarrow X_{\text{free}} \cup \{x_{\text{rand}}\}$;
11 $b_{\text{free}} \leftarrow \texttt{DensityEstimator}(X_{\text{free}})$;
12 $\hat{f}_X \leftarrow \texttt{Classifier}(X, b_{\text{obs}}, b_{\text{free}})$;
13 $X \leftarrow (X_{\text{obs}}, X_{\text{free}})$;
14 **return** X;

Line 12, the `Classifier` procedure implements a Bayesian classifier and the label of a given point x is determined by the following Bayesian decision rule:

$$\hat{y}_{cs}(x) = \begin{cases} 1 & \text{if } q_{\text{free}}(x) \geq q_{\text{obs}}(x), \\ -1 & \text{otherwise,} \end{cases} \tag{6}$$

where $q_{\text{obs}}(x) := \eta P(x|y=-1)P(y=-1)$ and $q_{\text{free}}(x) := \eta P(x|y=1)P(y=1)$, where $\eta = 1/P(x)$ is a normalizing coefficient. This classifier separates the configuration space $\mathcal{X}$ into two approximate sets of obstacle $\widehat{\mathcal{X}}_{\text{obs}}$ and obstacle-free $\widehat{\mathcal{X}}_{\text{free}}$ regions, i.e., $\widehat{\mathcal{X}}_{\text{obs}} = \{x \in \mathcal{X} : \hat{y}_{cs}(x) = -1\}$ and $\widehat{\mathcal{X}}_{\text{free}} = \{x \in \mathcal{X} : \hat{y}_{cs}(x) = 1\}$. At each iteration, the class density functions are approximated by the kernel density estimator based on the available data, as follows

$$b_{\text{obs}} = P(x|y=-1) = \frac{1}{|X_{\text{obs}}|} \sum_{x' \in X_{\text{obs}}} \mathcal{K}_{\mathbf{H}_{\text{o}}}(x - x')$$

$$b_{\text{free}} = P(x|y=1) = \frac{1}{|X_{\text{free}}|} \sum_{x' \in X_{\text{free}}} \mathcal{K}_{\mathbf{H}_{\text{f}}}(x - x')$$

where $\mathbf{H}_{\text{o}} = \texttt{diag}(h_{\text{o}}, \ldots, h_{\text{o}})$ and $\mathbf{H}_{\text{f}} = \texttt{diag}(h_{\text{f}}, \ldots, h_{\text{f}})$ are computed according to equation (5) using the sizes of X_{obs} and X_{free}, respectively.

The class priors are computed as the ratio of the samples in each class according to the expression $P(y=-1) = |X_{\text{obs}}|/|X|$ and $P(y=1) = |X_{\text{free}}|/|X|$. Finally, having the $\widehat{\mathcal{X}}_{\text{obs}}$ and $\widehat{\mathcal{X}}_{\text{free}}$ sets, the `Classifier` procedure returns the following pdf which is uniform over $\widehat{\mathcal{X}}_{\text{free}}$:

$$\hat{f}_X(x) = \begin{cases} 1/\mu(\widehat{\mathcal{X}}_{\text{free}}) & \text{if } x \in \widehat{\mathcal{X}}_{\text{free}}, \\ 0 & \text{if } x \in \widehat{\mathcal{X}}_{\text{obs}}. \end{cases} \tag{7}$$

E. Learning the Cost-to-come (or cost-to-go) Value

Given the set of training data $\mathcal{D} = \left\{(x^{(i)}, y^{(i)}) : i = 1, \ldots, m\right\}$ where the pair $\left(x^{(i)}, y^{(i)}\right)$ denotes a randomly drawn point along with its lmc-value (see Ref. [1]) computed by the replanning procedure at the ith iteration, we wish to find a function $\hat{y}_{cv} : \mathcal{X} \mapsto \mathbb{R}$ that gives a good estimate of the cost-to-come value of a given point x.

Due to the incremental setting of the sampling-based algorithms, we consider *locally weighted learning* based methods [3], which are a form of lazy learning, to solve the aforementioned regression problem owing to their easy training. In this method, the training data is stored in memory and only a small subset is retrieved to answer a specific query. Relevance of the data is measured by using a distance function (e.g., nearby points look alike or have similar features). For regression problems, the method fits a surface to nearby points using a distance weighted regression as follows:

$$\hat{y}_{cv}(x) = \frac{\sum_i y^{(i)} w(x, x^{(i)})}{\sum_i w(x, x^{(i)})}. \tag{8}$$

The weighting function $w(x, x')$ measures the relevance of two points and can be defined by using a kernel function, for example, $w(x, x') = \mathcal{K}_{\mathbf{H}_{\text{v}}}(x - x')$ where $\mathbf{H}_{\text{v}} = \texttt{diag}(h_{\text{v}}, \ldots, h_{\text{v}})$ is computed from equation (5) according to size of vertex set V. In the proposed approach, whenever a new sample is examined for inclusion in the graph, first its cost-to-come value is estimated using the lmc-values of the neighbor vertices according to equation (8). Then, the new sample is included in the graph if its approximate cost-to-come value satisfies the following inequality, which is a relaxed version of (1)

$$\widehat{\mathcal{X}}_{\text{rel}} = \{x \in \widehat{\mathcal{X}}_{\text{free}} : \hat{\mathtt{g}}(x) + \mathtt{h}(x) < \texttt{lmc}(x^*_{\text{goal}})\}. \tag{9}$$

F. Integration to the RRT$^{\#}$ *Algorithm*

The proposed approach can be seamlessly integrated to the RRT$^{\#}$ algorithm [1]. In fact, the proposed approach can be used with any single-query sampling-based motion planning algorithm, as long as it provides information of the cost-to-come (or cost-to-go) values for all the vertices. However, it is essential for the planning algorithm to provide accurate estimates of these cost values to achieve a good performance. In this paper we have chosen the RRT$^{\#}$ algorithm to leverage its fast convergence properties, which is the result of using a relaxation step for the local rewiring of the graph. Details of the RRT$^{\#}$ algorithm and its variants can be found in [1] and [2]. Instead of implementing a uniform sampling strategy, the `Sample` procedure of the RRT$^{\#}$ algorithm is replaced with the proposed adaptive sampling strategy. The details of the `Sample` procedure is given in Algorithm 2.

The `Extend` procedure of the RRT$^{\#}$ algorithm is also modified, and relevancy of a new sample is checked by the proposed approach in Algorithm 3 before invoking any collision checker or solving the local steering problem. If the new sample is predicted to be part of the relevant region, then the typical operations of the RRT$^{\#}$ algorithm are performed for the inclusion of the new sample in the current graph. Lines 2-11 implement the Bayesian classifier and compute the posterior distribution of the new sample. If the new sample is predicted to be in the obstacle-free space, then the locally weighted regression is applied in Lines 13-20 to determine if the new information has the potential to improve the current best solution.

Algorithm 2: `Sample` Procedure

Sample($\hat{f}_X$)
- $(X, b_{\text{obs}}, b_{\text{free}}) \leftarrow \hat{f}_X$; $(X_{\text{obs}}, X_{\text{free}}) \leftarrow X$;
- $x \leftarrow \texttt{SampleDensity}(\hat{f}_X)$;
- **while** $\texttt{OnObstacle}(x)$ **do**
  - $X_{\text{obs}} \leftarrow X_{\text{obs}} \cup \{x\}$; $X \leftarrow (X_{\text{obs}}, X_{\text{free}})$;
  - $b_{\text{obs}} \leftarrow \texttt{DensityEstimator}(X_{\text{obs}})$;
  - $\hat{f}_X \leftarrow \texttt{Classifier}(X, b_{\text{obs}}, b_{\text{free}})$;
  - $x \leftarrow \texttt{SampleDensity}(\hat{f}_X)$;
- $X_{\text{free}} \leftarrow X_{\text{free}} \cup \{x\}$; $X \leftarrow (X_{\text{obs}}, X_{\text{free}})$;
- $b_{\text{free}} \leftarrow \texttt{DensityEstimator}(X_{\text{free}})$;
- $\hat{f}_X \leftarrow \texttt{Classifier}(X, b_{\text{obs}}, b_{\text{free}})$;
- **return** $(\hat{f}_X, x)$;

SampleDensity($\hat{f}_X$)
- $(X, b_{\text{obs}}, b_{\text{free}}) \leftarrow \hat{f}_X$; $(X_{\text{obs}}, X_{\text{free}}) \leftarrow X$;
- $P_{\text{p,obs}} = |X_{\text{obs}}|/|X|$; $P_{\text{p,free}} = |X_{\text{free}}|/|X|$;
- **do**
  - $x \leftarrow \texttt{Sample}(\mathcal{X})$;
  - $P_{\text{c,obs}} = b_{\text{obs}}(x)$; $P_{\text{c,free}} = b_{\text{free}}(x)$;
  - $q_{\text{obs}}(x) = P_{\text{c,obs}} \cdot P_{\text{p,obs}}$;
  - $q_{\text{free}}(x) = P_{\text{c,free}} \cdot P_{\text{p,free}}$;
- **while** $q_{\text{free}}(x) < q_{\text{obs}}(x)$
- **return** x;

Algorithm 3: Relevancy Check Procedure

IsRelevant($\mathcal{G}$, X, x)
- $(V, E) \leftarrow \mathcal{G}$; $(X_{\text{obs}}, X_{\text{free}}) \leftarrow X$;
- $P_{\text{p,obs}} = |X_{\text{obs}}|/|X|$; $P_{\text{p,free}} = |X_{\text{free}}|/|X|$;
- $\mathcal{S}_{\text{obs}} \leftarrow \texttt{Near}(X_{\text{obs}}, x, |X_{\text{obs}}|)$; $P_{\text{c,obs}} = 0$;
- **foreach** $x' \in \mathcal{S}_{\text{obs}}$ **do**
  - $P_{\text{c,obs}} = P_{\text{c,obs}} + \mathcal{K}_{\mathbf{H}_o}(x - x')$;
- $\mathcal{S}_{\text{free}} \leftarrow \texttt{Near}(X_{\text{free}}, x, |X_{\text{free}}|)$; $P_{\text{c,free}} = 0$;
- **foreach** $x' \in \mathcal{S}_{\text{free}}$ **do**
  - $P_{\text{c,free}} = P_{\text{c,free}} + \mathcal{K}_{\mathbf{H}_f}(x - x')$;
- $P_{\text{c,obs}} = P_{\text{c,obs}}/|\mathcal{S}_{\text{obs}}|$; $P_{\text{c,free}} = P_{\text{c,free}}/|\mathcal{S}_{\text{free}}|$;
- $q_{\text{obs}}(x) = P_{\text{c,obs}} \cdot P_{\text{p,obs}}$; $q_{\text{free}}(x) = P_{\text{c,free}} \cdot P_{\text{p,free}}$;
- **if** $q_{\text{free}}(x) \geq q_{\text{obs}}(x)$ **then**
  - $\hat{g}(x) = 0$; $w_{\text{total}} = 0$;
  - $\mathcal{X}_{\text{near}} \leftarrow \texttt{Near}(\mathcal{G}, x, |V|)$;
  - **foreach** $x' \in \mathcal{X}_{\text{near}}$ **do**
    - $w(x, x') = \mathcal{K}_{\mathbf{H}_v}(x - x')$;
    - $\hat{g}(x) = \hat{g}(x) + \texttt{lmc}(x')w(x, x')$;
    - $w_{\text{total}} = w_{\text{total}} + w(x, x')$;
  - $\hat{g}(x) = \hat{g}(x)/w_{\text{total}}$;
  - $\texttt{Key}(x) = (\hat{g}(x) + \texttt{h}(x), \hat{g}(x))$;
  - **return** $\texttt{Key}(x) \prec \texttt{Key}(v^*_{\text{goal}})$;
- **return** `False`;

IV. Simulation Results

We have performed several simulations in order to confirm the efficiency of the proposed approach. Here we present the results for a two-link robot moving in the plane to demonstrate that the proposed adaptive sampling strategy is capable of generating a high-number of collision-free samples. The workspace and configuration space of the two-link robot are shown in Figure 1. Objects are intentionally placed in the workspace to form narrow passages in the configuration space. The sampling strategy has also been integrated to the RRT$^{\#}$ algorithm to solve a path planning problem in 2D environment in order to visualize the growth of the search tree.

A. 2D-link Robot

We first tested if the proposed adaptive sampling strategy generates a large number of collision-free configurations. Both uniform and adaptive sampling strategies were used to generate 100,000 samples. In order to demonstrate that the proposed approach eventually draws samples from difficult parts of the configuration space, e.g., narrow passages, all points on the boundary of obstacles were sampled offline and used to initialize the obstacle list X_{obs} of the classifier. By doing so, all narrow passages are blocked at the start of the algorithm.

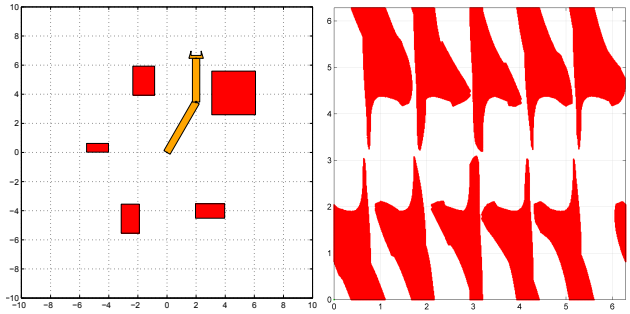


Fig. 1: Workspace and configuration space of a 2-link robot.

Figure 2 compares the ratio of the collision-free samples over the total number of samples ($r = |X_{\text{free}}|/|X|$) in a trial for uniform and adaptive sampling strategies, respectively. The ratio plots for adaptive and uniform sampling strategies are shown in green and blue colors, respectively. It is seen that this ratio converges to one for the proposed approach, whereas it lingers around $r = \mu(\mathcal{X}_{\text{free}})/\mu(X)$ for a uniform sampling strategy, shown in red color.

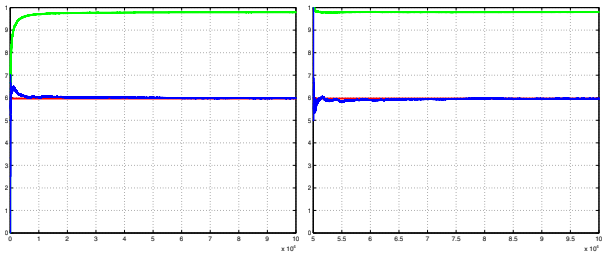
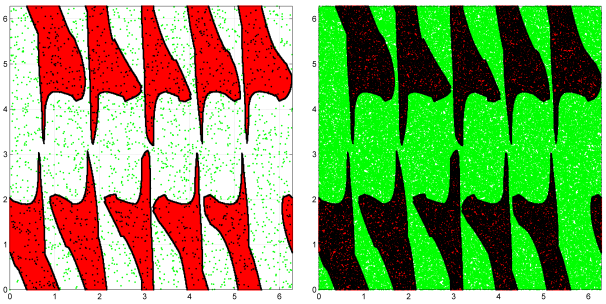
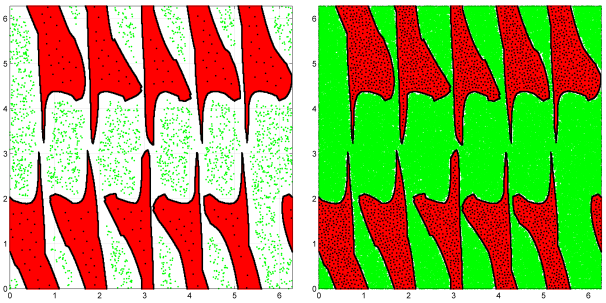


Fig. 2: Ratio of the number of collision-free samples over the total number of samples starting from intermediate iterations: left is with $i = 1$, and right is with $i = 50,001$.

The distribution of samples drawn by the uniform and sampling strategies is shown Figure 3. The free space and configuration space obstacles are shown in white and red, respectively. The samples that are free of and on collisions are shown in green and black, respectively. The proposed adaptive sampling strategy significantly reduces the number of points drawn from the obstacle space and generates samples inside the narrow passages shown, whereas the uniform sampling strategy results in a large number of points on the obstacle space, depending the measure of $\mu(\mathcal{X}_{\text{obs}})$.

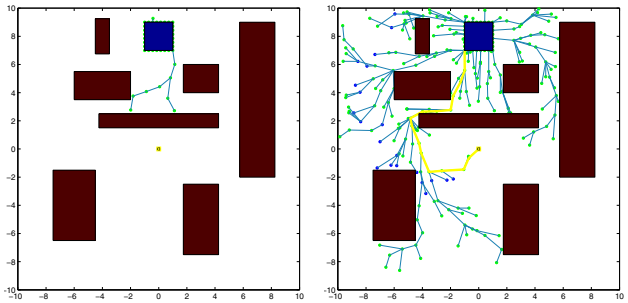


(a) 2,500 samples (b) 100,000 samples

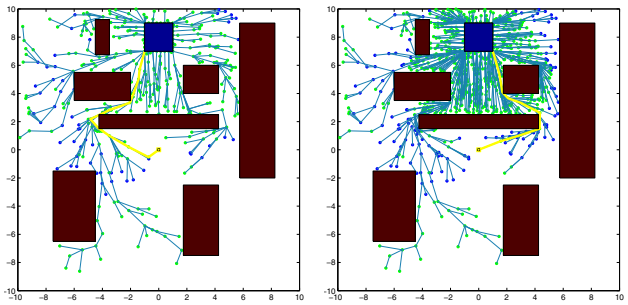


(c) 2,500 samples (d) 100,000 samples

Fig. 3: The distribution of samples randomly drawn by uniform and adaptive sampling strategies is shown in (a)-(b) and (c)-(d), respectively.

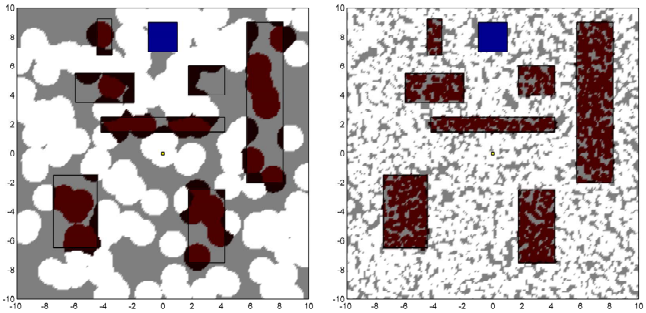


(a) 100 iterations (b) 2,500 iterations



(c) 5,000 iterations (d) 20,000 iterations

Fig. 4: Evolution of the tree shown.



(a) 100 iterations (b) 20,000 iterations

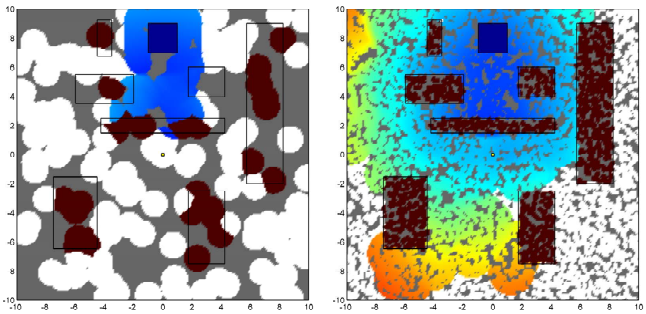
Fig. 5: Learned configuration space.

B. Path Planning in 2D Environment

In this problem, our aim is to find an optimal path that connects a given initial point to the goal region, while minimizing the Euclidean path length in a square environment. The Euclidean distance from a given state to the goal set was used as an admissible heuristic for that state. The growth of the tree at different stages is shown in Figure 4. The initial state is plotted as a yellow square,the goal region is shown in dark blue with (upper middle), and the obstacles are shown in dark red. The minimal-length path is shown in yellow. As shown in Figure 4, the lowest-cost path computed by the algorithm converges to the optimal solution. Note that in these simulations we have used a slightly different implementation of the algorithms, namely, the tree is rooted to the goal set instead of the initial state and the growth of the tree is reversed.

It is seen that once an initial solution is computed, exploration is prevented from going toward the unfavorable regions of the configuration space. This helps to greatly reduce the number of vertices kept in the graph, yet it computes high quality solutions.

The learned configuration space at different stages of the process are shown in Figure 5. The correctly predicted free space is shown in white color, the correctly predicted obstacle space is shown in dark red, the incorrectly predicted obstacle and free spaces are shown in black color, and the unexplored regions (no prediction is available) are shown in gray color. These plots show how the configuration space looks like from the classifier's perspective. The configuration space is densely gridded and all points are queried to the Bayesian classifier. The results are plotted based on the predicted labels. As seen in Figure 5-(b), the classifier builds an almost exact model of the configuration space, at least in the neighborhood of the relevant region.



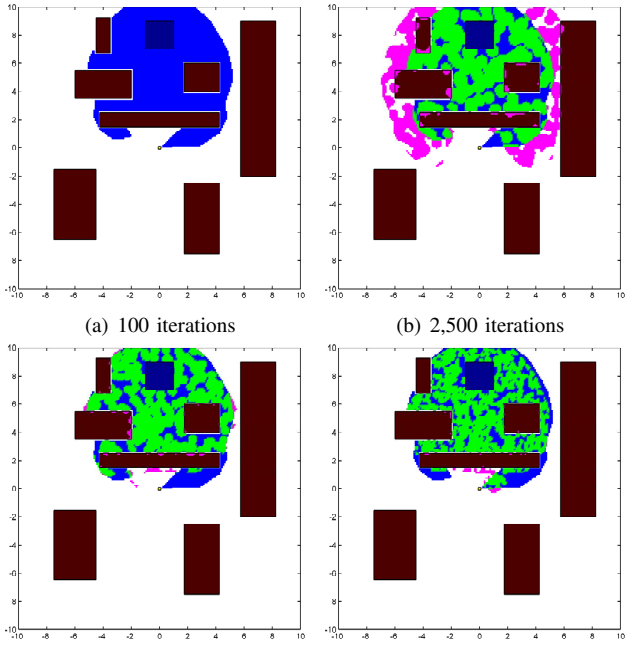
(a) 100 iterations (b) 20,000 iterations

Fig. 6: Approximate f-value function (cost-to-go plus heuristic value).

The approximate f-value function (heuristic value plus cost-to-go) at different stages is shown in Figure 6. Low and

high values of the f-value function are shown in blue and red colors, respectively, and the intermediate values are shown using gradient colors. This function is used to determine if a candidate sample is promising or not.

The approximate relevant region at different stages is shown in Figure 7. The true relevant and approximate regions are shown in blue and purple colors, respectively, and their intersection is shown in green color. In Figure 7-(a), the exact relevant region is plotted based on the true cost-to-go values, which are computed off-line. Since a solution has not been found yet, the algorithm considers initially the whole configuration space as the relevant region. As seen in the next figures, however, the algorithm progressively computes an approximation of the actual relevant region. In these plots the purple and green regions denote the incorrect and correct predictions, respectively.



(a) 100 iterations (b) 2,500 iterations

(c) 5,000 iterations (d) 20,000 iterations

Fig. 7: Approximate relevant region.

V. Conclusion

We have proposed a novel adaptive sampling strategy and have integrated it to the RRT$^{\#}$ algorithm, an asymptotically optimal sampling-based path-planning algorithm. The proposed adaptive sampling strategy utilizes the history of computed information, specifically, the label of the samples and their cost-to-come (or cost-to-go) values, to guide the exploration towards the region of the search space where samples having a great potential to improve the existing solution are more likely to be found. The approach utilizes ideas from machine learning to make predictions about how likely is for a new sample to be part of the free space and improve the current solution, without calling the computationally expensive collision checking and local steering procedures. Simulations demonstrate the effectiveness of the proposed approach, both in terms of reducing the number of samples lying in the obstacle space, and exploring the relevant region efficiently.

References

[1] O. Arslan and P. Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *IEEE International Conference on Robotics and Automation*, pages 2413–2420, Karlsrühe, Germany, May 6–10 2013.

[2] O. Arslan and P. Tsiotras. Dynamic programming guided exploration for sampling-based motion planning algorithms. In *IEEE International Conference on Robotics and Automation*, pages 4819–4826, Seattle, WA, May 26–30 2015.

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

[4] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli. Efficient collision checking in sampling-based motion planning. In *Algorithmic Foundations of Robotics X*, pages 365–380. Springer, 2013.

[5] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[6] B. Burns and O. Brock. Information theoretic construction of probabilistic roadmaps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 650–655, Las Vegas, Nevada, USA, October 27-31 2003.

[7] B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *IEEE International Conference on Robotics and Automation*, pages 3120–3125, Barcelona, Spain, April 18–22 2005.

[8] Brendan Burns and Oliver Brock. Single-query motion planning with utility-guided random trees. In *IEEE International Conference on Robotics and Automation*, pages 3307–3312, Rome, Italy, April 10-14 2007.

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

[10] R. Geraerts and M. H. Overmars. A comparative study of probabilistic roadmap planners. In *Algorithmic Foundations of Robotics V*, pages 43–57. Springer, 2004.

[11] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation*, pages 4420–4426, Taipei, Taiwan, September 14–19 2003.

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

[13] L. Janson, E. Schmerling, A. Clark, and M. Pavone. Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, 4:883–921, 2015.

[14] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[15] 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.

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

[17] S. M. Lavalle and J. J. Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2001.

[18] M. Otte and E. Frazzoli. RRT-X: Real-time motion planning/replanning for environments with unpredictable obstacles. In *The International Workshop on Algorithmic Foundations of Robotics*, Istanbul, Turkey, August 3–5 2014.

[19] J. Pan, S. Chitta, and D. Manocha. Faster sample-based motion planning using instance-based learning. In *Algorithmic Foundations of Robotics X*, pages 381–396. Springer, 2013.

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

[21] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, Amsterdam Boston, 2006.

[22] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London New York, 1986.