

Non-Parametric Informed Exploration for Sampling-Based Motion Planning

Sagar Suhas Joshi¹ and Tsiotras Panagiotis²

Abstract—Efficient exploration of the search space is crucial for faster convergence in sampling-based motion planning. An effective sampling method must first concentrate on quickly finding a good initial solution and then focus the search on regions that can potentially improve the current best solution. In this paper, we propose a non-parametric exploration technique that addresses these challenges. The proposed algorithm prioritizes search by utilizing heuristics. After an initial solution is found, the method generates samples in the “ L_2 -informed set”, while leveraging collision data to reduce the number of samples in the obstacle space. We demonstrate the efficiency of the proposed approach with several benchmarking experiments.

I. INTRODUCTION

Perhaps the most celebrated search algorithm is A* [1]. A* conducts a prioritized exploration of the discretized search space by expanding nodes with the lowest f value (sum of cost-to-come plus the heuristic estimate of cost-to-go). The use of an admissible heuristic leads to a focused search, and A* only expands nodes relevant to the current query of interest. However, such deterministic search algorithms are impractical for problems in high dimensions due to the computational overhead associated with a priori discretizing the search space. Randomized methods avoid this overhead by building a graph online. Incremental sampling based algorithms, in particular, Rapidly-exploring Random Trees (RRT) [2] have become popular recently due to their ability to handle higher dimensions and kino-dynamic constraints. The RRT procedure implicitly computes the Voronoi region of each vertex in the search tree. Randomization ensures that the search is biased towards vertices with the larger Voronoi regions. This leads to a rapid exploration of the search space. The RRT algorithm guarantees probabilistic completeness, but it can only return sub-optimal solutions. This sub-optimality can be attributed to the fact that the RRT algorithm, though rapidly exploring the search space, does not “exploit” the information gained to improve the current solution.

In general, every sampling-based motion planning (SBMP) algorithm needs to perform two fundamental tasks, namely, exploration and exploitation. The task of the exploration module is to acquire information about the search space by generating new samples and incorporating the feasible ones into the connectivity graph. The exploitation module, on the other hand, processes this information to improve the current solution. The RRT* algorithm [3] implements a “local rewiring” procedure for exploitation. This is achieved by rewiring the tree in the neighborhood of the newly generated sample to improve the cost-to-come values of the vertices. The RRT[#] algorithm [4] goes one step further

and performs “global rewiring” by using value iteration or policy iteration [5] to connect each vertex optimally to the current graph. The BIT* algorithm [6] generates batches of samples and prioritizes the search according to the potential solution quality using a heuristic. The FMT* algorithm [7] expands the search outwards in the cost-to-arrive space and uses “lazy” dynamic programming to reduce the number of collision checks. This interplay of exploration and exploitation endows algorithms like RRT* and others with asymptotic optimality, i.e., the probability of finding the optimal solution approaches unity as the number of samples approach infinity.

An efficient sampling strategy is naturally crucial for speedy convergence to the optimal solution. The easiest method to implement, used widely in practice, is uniform rejection sampling. However, for a single-query scenario, after an initial solution is found, the search should, ideally, focus on a subset of the space that can possibly further improve the current solution. Sampling outside this subset is redundant and wastes computational resources. It is thus critical to discover the first solution quickly in order of define this subset. Although uniform rejection sampling leads to rapid exploration of the search space, the use of (A*-like) heuristics can focus the search and help find the initial solution faster and with better quality. Also, rejection sampling does not utilize two sources of information that are available during the planning process: the previous collision checks and the planner’s current graph structure. The collision checks provide data regarding the location of the obstacles, while the graph structure can provide insight about areas that are already well represented by the planner, and areas that need to be explored further. Moreover, the performance of any rejection sampling method degrades as the state dimension increases, since sampling a state that can improve current solution becomes a rare event.

In this paper, we address the above issues by proposing a non-parametric sampling method. The method generates new samples from a set of “kernel vertices” located in the relevant region of the search space. The proposed algorithm uses heuristics and graph state information to focus the search during the initial phase of planning. After a solution is found, new points are sampled only in the vicinity of kernel vertices that can potentially provide better solution. Data from the past collision checks is leveraged to learn the location of the obstacles and to avoid sampling in the obstacle space.

II. RELATED WORK

Intelligent sampling techniques to accelerate convergence in single query cases is a topic that has been investigated

extensively in the literature. Akgun and Stilman [8] introduce local biasing to sample in the vicinity of the path vertices after an initial solution is found. Arslan and Tsiotras extend the basic RRT[#] algorithm to perform a more selective vertex inclusion procedure in [9]. After an initial solution is found, an adaptive rejection sampling method includes future samples only from the relevant region (namely, the region which can potentially have better solutions). Gammell et al [10] use admissible heuristics, L_2 norm in this case, for defining the L_2 -informed set, as the set of all states that could provide a better solution than the current solution. The authors of [10] discuss a direct sampling procedure to generate samples in the informed set: a prolate hyperspheroid with focal points at the start and goal locations and transverse diameter equal to the current best solution cost. This direct informed sampling technique has 100% recall, i.e., it covers all the points that can potentially improve the current solution. This gives a scalable approach to efficiently focus search in higher dimensional cases. The above approaches modify the sampling strategy after an initial solution is found. However, all these methods rely on uniform rejection sampling until an initial solution is discovered. Collision checking and graph structure information is not utilized in either of these methods.

Many other approaches use machine learning to improve the sampling process. Zucker et al [11] use deterministic search to compute features and policy gradient to optimize the sampling distribution. Recent work by Ichter et al [12] uses a conditional variational autoencoder (CVAE) to learn sampling from demonstrations. Kernel density estimation is used in [13] to model the free and obstructed space and obtain class conditional distributions. However, this method randomly selects kernels to generate samples. This can slow down exploration, as the sampling is not biased outwards (towards the leaf vertices). We build on this work, but attach a key value to every kernel to obtain a more efficient and prioritized exploration.

Substantial work has also been done to incorporate heuristics and graph state information to guide exploration in randomized algorithms. Urmson and Simmons in their heuristically-guided RRT (hRRT) [14] calculate a heuristic-based quality measure for every vertex in the tree. Diankov and Kuffner [15] propose a continuous randomized version of A* in their RA* algorithm. Burns and Brock propose an information-theoretic framework in a multi-query [16] and a single-query setting [17] to guide exploration towards regions of maximum information gain. The Exploring/Exploiting tree planner of [18], modulates the variance of Gaussian distributions based on success/failure of new samples. Expansive search trees (EST) [19] and its variants are especially relevant to our work. The EST algorithm picks a vertex with probability inversely proportional to the number of vertices in the neighborhood, for generating the next sample. Guided EST [20] adds an A* cost and an exploration term into the vertex weight to focus the search. Persson and Sharf [21] present a formal analysis for the generalization of the A* cost. Their SBA* algorithm uses graph density and constriction value to estimate the likelihood of sampling a

new and free point in the neighborhood of a vertex. However, the algorithm shows bias towards improving the current optimal paths (exploitation) rather than finding a connection to the goal (exploration). This is addressed by replacing some SBA* iterations with RRT* iterations and balancing this strategy with simulated annealing. The utility-guided random trees algorithm [22] selects the vertex with the least number of expansion attempts. A new direction for expansion is calculated based on the history of previous attempts. However, the algorithm targets pure exploration and does not use heuristics to focus the search. The common theme in EST variants is to include the graph density along with the A* cost to extract a prioritized exploration of the search space. Although we use similar concepts in this paper, a key difference is that the proposed algorithm generates samples only in the informed set after the first connection is made. This saves the computational effort of redundant exploration. Secondly, while the EST algorithm and its variants expand vertices, the proposed algorithm generates samples in the vicinity of the set of well dispersed “kernel vertices” inside the informed set. A new kernel vertex is initialized only if it lies in the informed set and it is at a threshold distance away from all the existing kernel vertices. Thus, although the number of vertices may increase indefinitely as the iterations progress, the number of kernel vertices remain fixed after the entire informed set is explored. This greatly reduces the computational load of otherwise storing information and processing every vertex for expansion.

In the following sections, we first formally define the problem, followed by a detailed description of the proposed Non-Parametric informed (NP-informed) sampling method. Although the sampler can be paired with any exploitation module, we provide benchmarking experiments with RRT* (local rewiring) for exploitation to demonstrate the efficiency of the approach.

III. PROBLEM DEFINITION

A. Motion Planning Problem

Let $\mathcal{X}$ denote the search space, which is assumed to be a subset of $\mathbb{R}^d$, here d is a positive integer and $d \geq 2$. Let $\mathcal{X}_{\text{obs}}$ denote the obstacle space and $\mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$ denote the free space. Let $\mathcal{G} = (V, E)$ represent the graph that signifies the connectivity between the finite set of vertices V with edges $E \subseteq V \times V$. Let $\mathbf{x}_{\text{init}} \in \mathcal{X}_{\text{free}}$ denote the initial state and $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$ represent the goal region. Given a vertex $v \in V$, the function $g : v \rightarrow r$, returns a non-negative real number r which is the cost of the path from v to the given initial state $\mathbf{x}_{\text{init}}$ (cost-to-come). Consider points $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$. The heuristic function h is defined as $h : (\mathbf{x}_1, \mathbf{x}_2) \rightarrow r$, where r is the estimate of the optimal cost between $\mathbf{x}_1$ and $\mathbf{x}_2$. We set $h(\mathbf{x}_1, \mathbf{x}_2) = 0$ if $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}_{\text{goal}}$. In this paper, we only consider admissible heuristics, i.e., the heuristic never overestimates the cost between two points. Thus, given an initial state $\mathbf{x}_{\text{init}} \in \mathcal{X}_{\text{free}}$ and a goal region $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$, the optimal motion planning problem is one of finding the minimum cost path from $\mathbf{x}_{\text{init}}$ to $\mathcal{X}_{\text{goal}}$. Sampling-based algorithms solve the motion planning problem by randomly drawing points from $\mathcal{X}_{\text{free}}$ and incorporating them into the

graph $\mathcal{G}$. This method of generating configurations is called the sampling strategy.

B. Exploration Problem

Consider the estimate of the cost path from $\mathbf{x}_{\text{init}}$ to $\mathbf{x}_{\text{goal}} \in \mathcal{X}_{\text{goal}}$ constrained to pass through $\mathbf{x} \in \mathcal{X}$

$$\hat{f}(\mathbf{x}) = h(\mathbf{x}, \mathbf{x}_{\text{init}}) + h(\mathbf{x}, \mathbf{x}_{\text{goal}}). \quad (1)$$

Let V_{goal} be the set of vertices in $\mathcal{X}_{\text{goal}}$. The vertex in $\mathcal{X}_{\text{goal}}$ with minimum cost-to-come can be designated as $v_{\text{goal}}^* = \arg \min_{v \in V_{\text{goal}}} g(v)$. The current optimal solution cost is then $g(v_{\text{goal}}^*)$. We define a set of points $X_{\hat{f}}$, that can potentially provide a better solution as

$$X_{\hat{f}} = \{\mathbf{x} \in \mathcal{X}_{\text{free}} \mid \hat{f}(\mathbf{x}) < g(v_{\text{goal}}^*)\}. \quad (2)$$

In this paper, we consider the Euclidean distance as the heuristic function. The L_2 -informed set [10] is thus defined as:

$$X_{\hat{f}} = \{\mathbf{x} \in \mathcal{X}_{\text{free}} \mid (\mathbf{x} - \mathbf{x}_{\text{centre}})^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{x}_{\text{centre}}) < 1\}, \quad (3)$$

where, $\mathbf{S} \in \mathbb{R}^{n \times n}$, is a symmetric, positive-definite hyper-ellipsoid matrix and

$$\mathbf{x}_{\text{centre}} = \frac{\mathbf{x}_{\text{init}} + \mathbf{x}_{\text{goal}}}{2}. \quad (4)$$

The procedure for calculating $\mathbf{S}$ and direct-sampling the interior of the hyper-ellipsoid in (3) is given in [10]. Thus, the strategy of the sampling algorithm must be to find an initial solution quickly and then focus on generating samples in $X_{\hat{f}}$, while at the same time use collision data to model the topology of $\mathcal{X}_{\text{free}}$.

IV. NON-PARAMETRIC INFORMED SAMPLING

A. Algorithm Overview

As mentioned in the previous sections, the proposed method generates new samples in the neighborhood of a set of kernel vertices $K = \{k_1, k_2, \dots, k_n\}$, $n \in \mathbb{Z}^+$ (see Fig.1). The set of kernel vertices is a subset of graph vertices, i.e., $K \subset V$. Thus, not all graph vertices are kernel vertices. Let the state (i.e., location vector) of the kernel vertex k be μ_k . Let d_k be the number of edge connections from kernel vertex k . This represents the “graph density” around k . Let p_k be the number of sampling attempts from k and let $\hat{f}_k$ be the estimate of the cost path constrained to pass through μ_k , i.e. $\hat{f}_k = \hat{f}(\mu_k)$. The set of *informed kernel vertices* is defined as

$$K_{\text{inf}} = \{k \in K \mid \hat{f}_k < g(v_{\text{goal}}^*)\}. \quad (5)$$

The outline of the proposed method is given in Algorithm 1. The algorithm is initialized by a kernel vertex at $\mathbf{x}_{\text{init}}$. A new sample $\mathbf{x}_{\text{rand}}$ is generated using conventional direct informed sampling with probability ϵ_k . This adds robustness and aids in exploration. Else, a kernel vertex is chosen to generate a new sample $\mathbf{x}_{\text{rand}} \in X_{\hat{f}}$ in its vicinity (Alg. 1, lines 9-10). The Extend procedure (Alg. 1, line 11) involves conventional rapidly exploring random graphs (RRG) modules: 1) finding the vertex nearest to $\mathbf{x}_{\text{rand}}$ in graph $\mathcal{G}$; 2) local steering from the nearest vertex to initialize a new vertex with state $\mathbf{x}_{\text{new}}$;

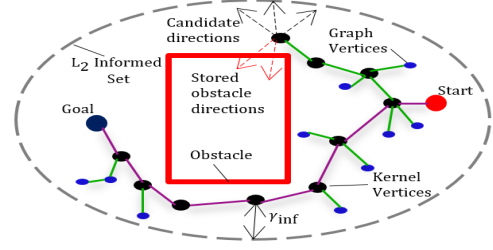


Fig. 1: Schematic of NP-informed sampling, showing kernel vertices(black) generating new samples in the L_2 -informed set after an initial solution is found.

3) making new edge connections by finding vertices in the neighborhood of $\mathbf{x}_{\text{new}}$. Approximate nearest neighbors can be found efficiently using data structures such as KD-trees. We use FLANN [23] for nearest neighbor calculations. The Extend procedure also outputs δ_{min} : the distance of $\mathbf{x}_{\text{new}}$ to the closest kernel vertex. Step 3 can be used to search the neighborhood of $\mathbf{x}_{\text{new}}$ for kernel vertices to compute δ_{min} . If no kernel vertex is found, a separate KD-tree built with kernel vertex states $\mu_k, \forall k \in K$, as input is used for calculating δ_{min} . If this distance is greater than a threshold value η' , then a new kernel vertex is initialized at $\mathbf{x}_{\text{new}}$ (Alg. 1, lines 12-14). Thus, a new kernel vertex is initialized

Algorithm 1: Sampling Algorithm Flow

```

1  $V \leftarrow \{\mathbf{v}_{\text{init}}\}; E \leftarrow \phi; \mathcal{G} \leftarrow (V, E);$ 
2  $\mu_{k_1} \leftarrow \mathbf{x}_{\text{init}}; p_{k_1} \leftarrow 0; K \leftarrow \{k_1\};$ 
3 for  $i = 1 : N$  do
4    $g(v_{\text{goal}}^*) \leftarrow \min_{v \in V_{\text{goal}}} g(v);$ 
5    $u_{\text{rand}} \sim \mathcal{U}[0, 1);$ 
6   if  $u_{\text{rand}} < \epsilon_k$  then
7      $\mathbf{x}_{\text{rand}} \leftarrow \text{InformedSampling}()$ 
8   else
9      $k_i \leftarrow \text{chooseKernel}(K_{\text{inf}}, \mathcal{G});$ 
10     $\mathbf{x}_{\text{rand}} \leftarrow \text{generateSample}(k_i);$ 
11     $\delta_{\text{min}}, \mathbf{x}_{\text{new}} \leftarrow \text{Extend}(\mathbf{x}_{\text{rand}});$ 
12    if  $\delta_{\text{min}} \geq \eta'$  then
13       $\mu_{k_{\text{new}}} \leftarrow \mathbf{x}_{\text{new}}; p_{k_{\text{new}}} \leftarrow 0;$ 
14       $K \leftarrow K \cup \{k_{\text{new}}\};$ 
15     $\text{Exploitation}(\mathcal{G});$ 
16 return  $\mathcal{G}$ 

```

only if it is η' distance away from the existing kernel vertices. This avoids clumping all kernels in the same region and hence aids exploration. Though the graph vertices can be arbitrarily close, the kernel vertex inclusion procedure leads to evenly spaced kernels representing salient regions in $X_{\hat{f}}$. Also, as the set $X_{\hat{f}}$ is bounded, there can only be a finite number of kernel vertices.

B. Kernel Selection

The procedure for kernel selection is described in Algorithm 2. A value q_k is calculated for every $k \in K_{\text{inf}}$ and the kernel with minimum q value is selected to generate the next sample. We consider the position information given by $\hat{f}_k$, the graph density information given by d_k , and the number of previous sampling attempts p_k to calculate the

value q_k . The normalizing terms $(\hat{f}_{\max}, \hat{f}_{\min})$, $(p_{\max}, p_{\min})$ and $(d_{\max}, d_{\min})$, are as defined in Algorithm 2, lines 2-4. The value q_k is a weighted linear combination with weights $\lambda_1, \lambda_2, \lambda_3$, such that $\sum_i \lambda_i = 1$, as defined in (6). Here, the first term containing $\hat{f}_k$ represents the “greedy” term that penalizes selection of kernel vertices away from $\mathbf{x}_{\text{init}}$ and $\mathbf{x}_{\text{goal}}$ and subsequently generating samples in those regions.

$$q_k = \lambda_1 \frac{\hat{f}_k - \hat{f}_{\min}}{\hat{f}_{\max} - \hat{f}_{\min}} + \lambda_2 \frac{p_k - p_{\min}}{p_{\max} - p_{\min}} + \lambda_3 \frac{d_k - d_{\min}}{d_{\max} - d_{\min}}. \quad (6)$$

The graph density (third) term promotes sampling in the regions that are relatively unexplored. The second term adds random exploratory behavior and brings uniformity in the selection of kernels. Different behaviors can be extracted by modulating the values of $\lambda_1, \lambda_2, \lambda_3$. A high value of λ_1 would focus search, minimizing exploratory behavior. Increasing λ_3 biases the search outwards (towards the leaf vertices), in the regions of low graph density and prioritizes exploration. A very high value of λ_2 would lead to random selection of kernel vertices. As described in the previous section, a new kernel is initialized only if it is more than a threshold distance away from the existing kernels. Hence, the new kernel would have low d and p values. This results in bias towards choosing new kernel vertices and exploring new regions.

Algorithm 2: Kernel Selection Algorithm

```

1 chooseKernel ( $K_{\text{inf}}, \mathcal{G}$ ):
2    $\hat{f}_{\max} \leftarrow \max_{k \in K_{\text{inf}}} \hat{f}_k$ ;  $\hat{f}_{\min} \leftarrow \min_{k \in K_{\text{inf}}} \hat{f}_k$ ;
3    $p_{\max} \leftarrow \max_{k \in K_{\text{inf}}} p_k$ ;  $p_{\min} \leftarrow \min_{k \in K_{\text{inf}}} p_k$ ;
4    $d_{\max} \leftarrow \max_{k \in K_{\text{inf}}} d_k$ ;  $d_{\min} \leftarrow \min_{k \in K_{\text{inf}}} d_k$ ;
5    $q^* \leftarrow \infty$ ;
6   foreach  $k \in K_{\text{inf}}$  do
7      $q_k =$ 
8        $\lambda_1 \frac{\hat{f}_k - \hat{f}_{\min}}{\hat{f}_{\max} - \hat{f}_{\min}} + \lambda_2 \frac{p_k - p_{\min}}{p_{\max} - p_{\min}} + \lambda_3 \frac{d_k - d_{\min}}{d_{\max} - d_{\min}}$ ;
9     if  $q_k < q^*$  then
10        $q^* \leftarrow q_k$ ;  $k^* \leftarrow k$ ;
11   return  $k^*$ ;

```

C. Sample Generation

After a kernel k is chosen, a new sample is generated by deciding the direction and magnitude of travel from μ_k . The magnitude of the travel in a given direction needs to be restricted so as to confine the samples to $X_{\hat{f}}$. Consider a point $\mathbf{p} \in X_{\hat{f}}$ and a direction vector $\hat{\mathbf{e}}$. Let the maximum distance that can be travelled from point $\mathbf{p}$ in the direction $\hat{\mathbf{e}}$ until the surface of hyper-ellipsoid (defined in (3)) is reached be γ_{inf} . Thus,

$$(\mathbf{p} + \gamma_{\text{inf}} \hat{\mathbf{e}} - \mathbf{x}_{\text{centre}})^T \mathbf{S}^{-1} (\mathbf{p} + \gamma_{\text{inf}} \hat{\mathbf{e}} - \mathbf{x}_{\text{centre}}) = 1, \quad (7)$$

Let, $\mathbf{p}' = \mathbf{p} - \mathbf{x}_{\text{centre}}$. Then, simplifying the above equation, we get

$$\gamma_{\text{inf}}^2 (\hat{\mathbf{e}}^T \mathbf{S}^{-1} \hat{\mathbf{e}}) + \gamma_{\text{inf}} (2 \hat{\mathbf{e}}^T \mathbf{S}^{-1} \mathbf{p}') + (\mathbf{p}'^T \mathbf{S}^{-1} \mathbf{p}' - 1) = 0. \quad (8)$$

Solving this quadratic equation, we obtain two solutions. Only the positive solution is considered, as we wish to move in the direction given by $\hat{\mathbf{e}}$. Thus,

$$\gamma_{\text{inf}} = \frac{-(\hat{\mathbf{e}}^T \mathbf{S}^{-1} \mathbf{p}') + \sqrt{(\hat{\mathbf{e}}^T \mathbf{S}^{-1} \mathbf{p}')^2 + (1 - \mathbf{p}'^T \mathbf{S}^{-1} \mathbf{p}') (\hat{\mathbf{e}}^T \mathbf{S}^{-1} \hat{\mathbf{e}})}}{(\hat{\mathbf{e}}^T \mathbf{S}^{-1} \hat{\mathbf{e}})}. \quad (9)$$

In the proposed algorithm, we consider the point $\mathbf{p}$ to be the kernel vertex state μ_k for $k \in K_{\text{inf}}$. Hence, from equations (3),(5) we have $\mathbf{p}'^T \mathbf{S}^{-1} \mathbf{p}' < 1$ in (9). Thus, γ_{inf} will always have a real, positive value. The sample generation

Algorithm 3: Step Limit for informed Sampling

```

1 InformedStepLimit ( $\mathbf{p}, \hat{\mathbf{e}}, \mathbf{S}$ ):
2    $\mathbf{p}' \leftarrow \mathbf{p} - \mathbf{x}_{\text{centre}}$ ;
3    $\gamma_{\text{inf}} \leftarrow$ 
4      $\frac{-(\hat{\mathbf{e}}^T \mathbf{S}^{-1} \mathbf{p}') + \sqrt{(\hat{\mathbf{e}}^T \mathbf{S}^{-1} \mathbf{p}')^2 + (1 - \mathbf{p}'^T \mathbf{S}^{-1} \mathbf{p}') (\hat{\mathbf{e}}^T \mathbf{S}^{-1} \hat{\mathbf{e}})}}{(\hat{\mathbf{e}}^T \mathbf{S}^{-1} \hat{\mathbf{e}})}$ ;
5   return  $\gamma_{\text{inf}}$ ;

```

Algorithm 4: Sample Generation Algorithm

```

1 generateSample ( $k$ ):
2   while true do
3      $u_{\text{rand}} \sim \mathcal{U}[0, 1]$ ;
4     if  $u_{\text{rand}} < \epsilon_s$  then
5        $\mathbf{e}^* \leftarrow \text{generateDirection}()$ 
6     else
7       for  $i = 1 : T$  do
8          $\mathbf{e}_i \leftarrow \text{generateDirection}()$ ;
9         foreach  $\mathbf{e}_{\text{obs}} \in O_k$  do
10          if  $(\mathbf{e}_i^T \mathbf{e}_{\text{obs}}) > u_{\text{obs}}$  then
11             $u_{\text{obs}} \leftarrow (\mathbf{e}_i^T \mathbf{e}_{\text{obs}})$ ;
12          if  $u_{\text{obs}} < u^*$  then
13             $u^* \leftarrow u_{\text{obs}}$ ;  $\mathbf{e}^* \leftarrow \mathbf{e}_i$ ;
14        $\gamma_{\text{inf}} \leftarrow \text{InformedStepLimit}(\mu_k, \mathbf{e}^*, \mathbf{S})$ ;
15        $u_{\text{rand}} \sim \mathcal{U}[0, 1]$ ;
16        $\gamma \leftarrow (u_{\text{rand}})^{\frac{1}{d}} \min(\gamma_k, \gamma_{\text{inf}})$ ;
17        $\mathbf{x}_{\text{rand}} \leftarrow \mu_k + \gamma \mathbf{e}^*$ ;
18       if IsFree( $\mathbf{x}_{\text{rand}}$ ) then
19         return  $\mathbf{x}_{\text{rand}}$ ;
20       else
21          $O_k \leftarrow O_k \cup \{\mathbf{e}^*\}$ ;
22          $k \leftarrow \text{random}(K_{\text{inf}})$ ;  $p_k \leftarrow p_k + 1$ ;

```

procedure is outlined in Algorithm 4. A random exploration direction is generated with probability ϵ_s . Otherwise, the set of past directions O_k that resulted in collision for kernel k are considered. Before an initial solution is found, the non-greedy direction $(\mu_k - \mathbf{x}_{\text{goal}}) / \|\mu_k - \mathbf{x}_{\text{goal}}\|_2$ for kernel k , is also considered as an obstacle direction in order to bias the exploration efforts to reach $\mathcal{X}_{\text{goal}}$. A number of candidate directions T are produced and the dot product is calculated with each direction vector in O_k . The cost associated with a candidate direction $\mathbf{e}_i$ is the maximum of these dot product values (Alg. 4, lines 9-11). The candidate direction with the

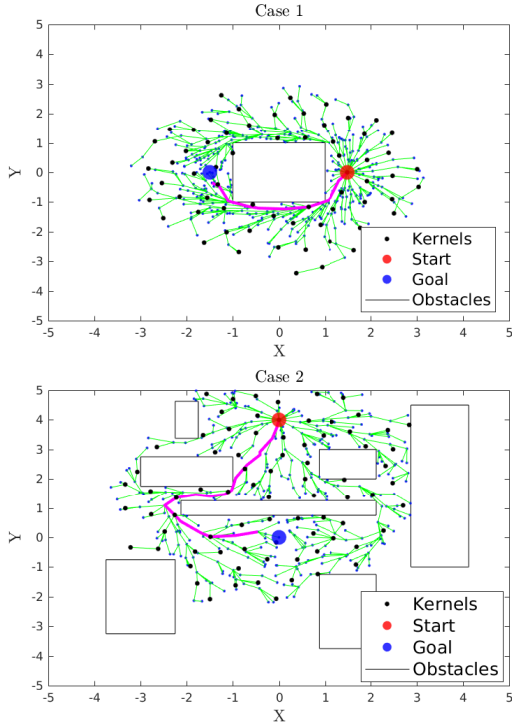


Fig. 2: (Top) Single obstacle environment; (Bottom) Multiple obstacle environment. The figure shows 500 iterations of NP-informed sampling in both cases.

least cost is then chosen for exploration (Alg. 4, lines 12-13). This direction is used to calculate γ_{inf} , the maximum step-size, that will ensure that samples remain in $X_{\hat{f}}$. The minimum of γ_{inf} and γ_k is considered to calculate the travel magnitude (Alg. 4, line 16). Here, γ_k represents the radius of the ball around μ_k in which new samples would be generated if $\gamma_k < \gamma_{\text{inf}}$. The sample generation algorithm terminates if the new sample $\mathbf{x}_{\text{rand}} \in \mathcal{X}_{\text{free}}$, else this direction of travel is stored in O_k and a new kernel vertex in K_{inf} is chosen randomly to generate a new sample (Alg. 4, line 22).

V. NUMERICAL EXPERIMENTS

The performance of the NP-Informed sampling was benchmarked against conventional Informed Sampling and Uniform rejection sampling. The above exploration strategies were paired with local rewiring for exploitation. A C++ implementation of the above samplers was used (Informed sampling implementation based on the open-source OMPL version [24]). Please see: <https://github.gatech.edu/DCSL/NP-Informed-Sampling>. All experiments were run on a 64-bit PC with 64 GB RAM and Intel Xeon(R) Processor. The operating system used was Ubuntu 16.04. Data was recorded over 100 trials for all the cases. The algorithms were tested in a single obstacle and a multiple obstacle setting in $\mathbb{R}^2, \mathbb{R}^3, \mathbb{R}^4$ and $\mathbb{R}^6$ (Fig. 2). A (hyper)cube problem environment $\mathcal{X}$ of width 10 units was considered in both cases. In the first case, a single (hyper)cube obstacle with width of 2 units was placed with its center at the origin. The obstacle lies between the start and the goal point, $\mathbf{x}_{\text{init}} = [1.5, 0, \dots, 0]^T$, $\mathbf{x}_{\text{goal}} = [-1.5, 0, \dots, 0]^T$. The second case consists of multiple obstacles with $\mathbf{x}_{\text{init}} = [0, 4, \dots, 0]^T$,

TABLE I: Case 1: Average time/length of first solution

Dimension		2D	3D	4D	6D
Step Size		0.3	0.6	1.0	2.0
Time (ms)	Inf	7.7(4.8)	10.4(6.1)	11.5(17.5)	50.1(94.2)
	NP-Inf	4.3(1.7)	3.2(1.6)	2.6(1.5)	37.9(43.9)
Length	Inf	6.5(1.6)	8.0(1.6)	9.3(2.8)	10.4(3.4)
	NP-Inf	5.8(0.8)	7.0(1.0)	7.7(1.3)	8.5(1.6)

TABLE II: Case 2: Average time/length of first solution

Dimension		2D	3D	4D	6D
Step Size		0.3	0.6	1.0	2.0
Time (ms)	Inf	17.2(6.6)	13.6(10.8)	9.2(11.2)	51.4(142.6)
	NP-Inf	16.7(6.4)	4.9(2.8)	3.3(2.5)	30.2(33.3)
Length	Inf	10.3(1.5)	8.6(2.1)	8.8(2.5)	9.6(2.9)
	NP-Inf	9.0(0.8)	7.6(1.2)	8.0(1.3)	8.2(1.6)

$\mathbf{x}_{\text{goal}} = [0, 0, \dots, 0]^T$. The 2D environment in case 2 was extended to higher dimensions by imparting a length of 2 units (symmetrically) to each obstacle in dimensions $d \geq 3$. A goal bias of 10% was used in Informed and Uniform rejection sampling. Experiments were performed with following the step-sizes (maximum edge length) η : 0.3, 0.6, 1., 2. for $\mathbb{R}^2, \mathbb{R}^3, \mathbb{R}^4, \mathbb{R}^6$ respectively. Simulations were run for 2, 3, 4, 8 seconds in $\mathbb{R}^2, \mathbb{R}^3, \mathbb{R}^4, \mathbb{R}^6$ respectively. The parameters of the proposed algorithm were set as follows. All simulations were performed with $\epsilon_k = 0.5$, i.e., 50 % split between conventional Informed sampling and NP-informed sampling. Weighting constants $(\lambda_1, \lambda_2, \lambda_3)$ in (6) were set to $(3/9, 2/9, 4/9)$ before an initial solution is found and then changed to $(0, 1/2, 1/2)$. This causes the sampler to first focus on finding a good initial solution and then prioritize exploring the informed set. The probability of generating a random direction ϵ_s (Alg. 4, line 4) was initialized to 0 and changed to 0.5 after an initial solution was found. The maximum magnitude of travel for generating a random sample γ_k was initialized to 1.5η for every kernel and then decreased by a small quantity ($\Delta\gamma = 0.01$) every time a sampling attempt was made from that kernel. The threshold distance for initializing a new kernel vertex: (η') was set to η . The number of candidate directions generated T , were 5 in $\mathbb{R}^2, \mathbb{R}^3$ and 10 in $\mathbb{R}^4, \mathbb{R}^6$.

The simulation results are summarized in Tables I, II and Fig. 3. Tables I and II document the average time (in milliseconds) until the first solution was found and its corresponding length, while the numbers in the brackets signify the standard deviation. It can be seen that the NP-Informed sampling consistently finds a first solution of better quality (lesser length) and with smaller standard deviation than either Informed sampling or Uniform rejection sampling. This can be attributed to the use of heuristics during the kernel vertex selection, which focuses and prioritizes the search.

The convergence behavior of all three algorithms is illustrated in Fig. 3. The plots show that NP-informed sampling gets a “head start” over the other two samplers, as it finds first solution of better quality. The figure also shows the plot of collision fraction versus time, where

$$\text{Collision Fraction} = \frac{\# \text{Positive collisions}}{\# \text{Calls to collision checker}}. \quad (10)$$

Thus, the collision fraction represents the fraction of samples generated in $\mathcal{X}_{\text{obs}}$ out of all the samples generated. Theoret-

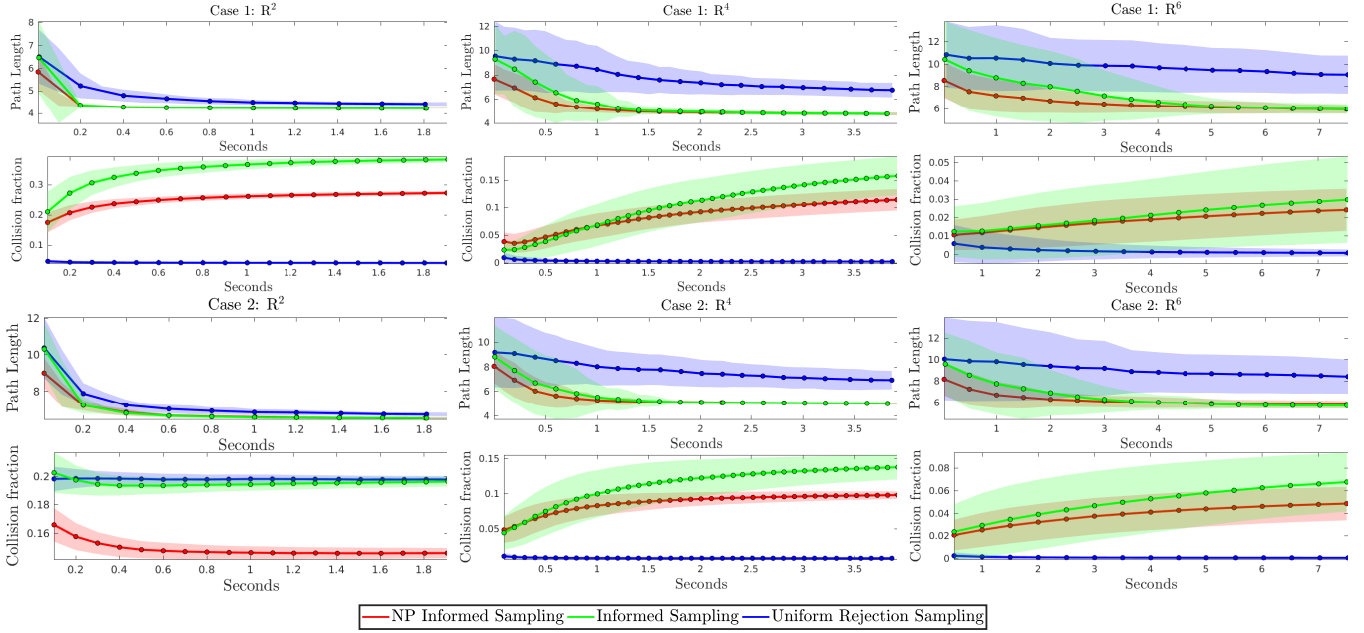


Fig. 3: Convergence behavior and fraction of samples generated in obstructed space for case 1 and 2 in $\mathbb{R}^2$, $\mathbb{R}^4$ and $\mathbb{R}^6$. Results are averaged over 100 trials and the standard deviation is shaded

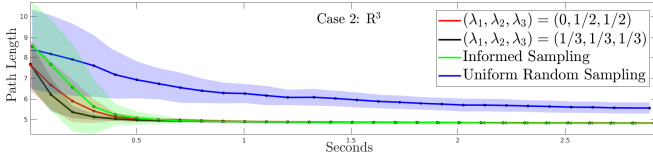


Fig. 4: Convergence for NP-informed sampling with different weights in (6) set after an initial solution is found

ically, the probability of generating a sample in $\mathcal{X}_{\text{obs}}$ would be $|\mathcal{X}_{\text{obs}}|/|\mathcal{X}|$ for uniform rejection sampling and $|\mathcal{X}_{\text{obs}} \cap \mathcal{X}_{\text{ellipse}}|/|\mathcal{X}_{\text{ellipse}}|$ for informed sampling. Here, $\mathcal{X}_{\text{ellipse}}$ represents the set of points contained inside the hyper-ellipsoid defined in (3) and $|\cdot|$ represents volume of the set. However, the proposed method generates new samples in the vicinity of kernel vertices that are initialized in $\mathcal{X}_{\text{free}}$. Moreover, past collision data is used to avoid sampling in the obstacle space. This results in the reduction of the collision fraction for NP-informed sampling. After the NP-informed sampling and the informed sampling converge to similar path lengths (which means that $|\mathcal{X}_{\text{ellipse}}|$ is comparable for both), the NP-informed sampling shows lesser collision fraction than informed sampling. This can be seen in Fig. 3, especially Case2: $\mathbb{R}^2$, where $|\mathcal{X}_{\text{obs}}|/|\mathcal{X}|$ is highest. Rejection sampling shows lowest collision fraction in higher dimensions as the ratio $|\mathcal{X}_{\text{obs}}|/|\mathcal{X}|$ diminishes rapidly.

VI. CONCLUSION

Informed sampling provides a scalable approach to focus search by direct sampling the interior of a prolate hyper-spheroid (the L_2 informed set). However, the algorithm has to rely on uniform rejection sampling until an initial solution is found. Informed sampling can generate a large fraction of samples in the obstacle space if the ratio $|\mathcal{X}_{\text{obs}} \cap$

$\mathcal{X}_{\text{ellipse}}|/|\mathcal{X}_{\text{ellipse}}|$ is high. These issues are addressed in this paper by proposing a non-parametric informed sampling technique. One of the drawbacks of the NP-informed sampling is the computational complexity associated with choosing and generating samples from a kernel vertex. This results in a lesser number of vertices produced compared to informed sampling. However, NP-informed sampling can compensate by finding first a solution of better quality to attain a better convergence profile.

The proposed approach presents many promising directions for future research. An efficient balance between exploration using Informed sampling and exploitation using NP-informed sampling can be attained. An example of this is illustrated in Fig. 4, which shows NP-informed sampling with two different sets of weights, signifying different behavior after an initial solution is found. The black plot with bias for exploitation shows better convergence than the pure exploratory NP-informed sampling (red plot). This can be extended to have a reward system for kernel vertices that produce “good” samples. The proposed approach can also be used to efficiently generate samples in the “relevant region set” defined in [9]. The relevant region set considers the estimate of g value (sum of edge-costs from start state) for cost-to-come instead of the heuristic approximation used in Informed set (2). Relevant region does not have a 100% recall like the Informed set. However, it can possibly have a higher precision (probability of sampling a point in the set that improves the current solution) as the iterations progress. It also reduces the dependence on heuristics, which can be misleading in many cases. A comparative study along the above ideas is the focus of ongoing research.

Acknowledgment: This work has been supported by NSF award IIS-1617630.

REFERENCES

- [1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 7 1968.
- [2] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
- [3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.
- [4] O. Arslan and P. Tsotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6–10 2013, pp. 2421–2428.
- [5] —, "Incremental sampling-based motion planners using policy iteration methods," in *IEEE 55th Conference on Decision and Control*, Las Vegas, NV, Dec. 12–15 2016, pp. 5004–5009.
- [6] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE International Conference on Robotics and Automation*, Seattle, WA, May, 25–30 2015, pp. 3067–3074.
- [7] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, May 2015.
- [8] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, Sept. 25–30 2011, pp. 2640–2645.
- [9] O. Arslan and P. Tsotras, "Dynamic programming guided exploration for sampling-based motion planning algorithms," in *IEEE International Conference on Robotics and Automation*, Seattle, WA, May 26–30 2015, pp. 4819–4826.
- [10] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, Aug. 2018.
- [11] M. Zucker, J. Kuffner, and A. J. Bagnell, "Adaptive workspace biasing for sampling-based planners," in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 19–23 2008, pp. 3757–3762.
- [12] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," *arXiv preprint arXiv:1709.05448*, 2017.
- [13] O. Arslan and P. Tsotras, "Machine learning guided exploration for sampling-based motion planning algorithms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, Sept. 28–Oct. 2, 2015, pp. 2646–2652.
- [14] C. Urmson and R. Simmons, "Approaches for heuristically biasing rrt growth," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, Las Vegas, NV, Oct. 27–31 2003, pp. 1178–1183.
- [15] R. Diankov and J. Kuffner, "Randomized statistical path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, Oct. 29–31 2007, pp. 1–6.
- [16] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Robotics: Science and Systems*, Cambridge, MA, June 8–11 2005, pp. 105–112.
- [17] —, "Single-query entropy-guided path planning," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 4 2005, pp. 2124–2129.
- [18] M. Rickert, O. Brock, and A. Knoll, "Balancing exploration and exploitation in motion planning," in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 19–23 2008, pp. 2812–2817.
- [19] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *IEEE International Conference on Robotics and Automation*, vol. 3, Albuquerque, NM, April 25–29 1997, pp. 2719–2726.
- [20] J. M. Phillips, N. Bedrossian, and L. E. Kavraki, "Guided expansive spaces trees: a search strategy for motion-and cost-constrained state spaces," in *IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 26–30 2004, pp. 3968–3973.
- [21] S. M. Persson and I. Sharf, "Sampling-based a* algorithm for robot path-planning," *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 10 2014.
- [22] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *IEEE International Conference on Robotics and Automation*, Rome, Italy, April 10–14 2007, pp. 3307–3312.
- [23] M. Muja and D. G. Lowe, "FLANN, fast library for approximate nearest neighbors," in *International Conference on Computer Vision Theory and Applications*, vol. 3, Lisbon, Portugal, Feb. 5–8 2009.
- [24] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, Dec. 2012.