2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

A Fast Marching Gradient Sampling Strategy for Motion Planning using an Informed Certificate Set

Shenglei Shi, Jiankui Chen and Youlun Xiong

Abstract—**We present a novel fast marching gradient sampling strategy to accelerate the convergence speed of sampling-based motion planning algorithms. This strategy is based on an informed certificate set which consists of the robot states with exact collision status as well as the minimum distance and the gradient to the nearest obstacle. The informed certificate set covers almost the whole planning space such that it contains rich information for the planner. The best quality point in this set is selected as the marching seed to guide the search graph move steadily to the goal set. The distance and gradient information of the marching seed helps to generate a new sample with almost sure collision status. When a feasible solution has been found, this set can construct the restricted subset that can improve current path quality. This marching gradient sampling strategy is applied to the RRT and RRT* algorithms. Simulation experiments demonstrate that the convergence speed to a feasible solution or to the optimal solution is almost twice faster than that of the safety certificate algorithms.**

I. Introduction

Probabilistic sampling-based algorithms have been shown as a particularly successful approach to high-dimensional robot motion planning problems. The most popular algorithms are PRM (Probabilistic roadmap) [1], [2] and RRT (Rapidly exploring random tree) [3]. However, both algorithms are only probabilistic complete that the probability of the planner fails to find a solution, if one exists, decays to zero as the number of samples approaches infinity. Recently, Karaman and Frazzoli [4] showed how asymptotic optimality can also be achieved with these methods and proposed the PRM* and RRT* algorithms.

These algorithms randomly sample robot configurations in the collision-free space, then connect them with local planning method to obtain a graph data structure. The collision-free samples generation and local planning are enabled by a “Boolean BlackBox” collision checking module. Hence, the basic components in sampling-based motion planning algorithms include: (1) a sampling strategy to generate a sequence of points in the free configuration space; (2) a local planning method to return a path between two given configurations; (3) a collision-checking module to determine the collision status of a sample point or a local path.

Collision checking is widely considered to be the most expensive computation bottleneck in sampling-based motion planning algorithms. Accordingly, the design of faster collision checking module to realize rapid convergence algorithms has emerged as a central topic in robotic motion planning. Janson et al. [5] applied a lazy collision checking module in the local planning process. That is, the algorithm ignores the presence of obstacles to virtually connect a new sample to its neighbor nodes, and then truly connect the optimal collision-free local path. The similar lazy collision checking idea is also adopted in a candidate graph that allows the existence of in-collision edges [6]. For problems to minimize the path length in Euclidean space, Gammell et al. [7] show that the solution quality can only be improved by directly sampling a prolate hyper-ellipsoid subset. Pan and Manocha [8] utilized the k-nearest neighbors of a new sample to predict its collision probability. Bialkowski et al. [9] stored additional information, a lower bound on a sample’s minimum distance to the obstacles, to the data structure. This information is called safety certificate such that when a new sample is located within a safety certificate region of an old point, it’s collision-free. Finally, rapid convergence property can also be achieved through other techniques. Otte and Correll [10] adopted a parallelization framework for sampling-based motion planning algorithms. Message passing, best solution sharing, and admissible subset restricting are particular techniques to increase the convergence speed and the solution quality. For in-time online problems, Karaman et al. [11] proposed an anytime motion planning framework to improve the practical performance.

In this paper, we place a stronger requirement on the collision-checking procedure and assume that it returns the safety certificate as well as the gradient to leave away from the nearest obstacle. We fully exploited the capabilities of this informed certificate set in addition to its fast collision checking characteristic. We find this set can guide the search graph grow in a marching way to move steadily and quickly to the goal set. In most sampling-based motion planning algorithms, the in-collision samples are always discarded. If given the gradient information, the in-collision samples can be flicked to valid collision-free samples.

II. Fundamentals and Motivations

The sampling-based motion planning problem definition and the safety certificate method are introduced first. Then we will briefly explain why this certificate set is very informed to guide the search graph grow in a marching way, and how the gradient information assists fast motion planning.

Let $\boldsymbol{X}=[0,1]^d$ denote the configuration space, where the dimension, d, is an integer larger than or equal to two. Each

S. Shi, J. Chen and Y. Xiong are with the State Key Laboratory of Digital Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan, 430074, P.R. China (e-mail: shshlei@hust.edu.cn; Chenjk@hust.edu.cn; famt@hust.edu.cn).

point in this space represents a configuration $\boldsymbol{x}$, and this space can be partitioned into two regions: collision-free region $\boldsymbol{X}_{\text{free}}$ and in-collision region $\boldsymbol{X}_{\text{obs}}$. The motion planning problem is denoted by a triplet $(\boldsymbol{X}_{\text{free}}, \boldsymbol{x}_s, \boldsymbol{X}_{\text{goal}})$, where $\boldsymbol{x}_s$ is the starting collision-free configuration, and $\boldsymbol{X}_{\text{goal}}$ is an open subset of $\boldsymbol{X}_{\text{free}}$. The planner seeks a collision-free feasible path $\boldsymbol{\sigma}:[0,1]\rightarrow\Re^d$, that $\boldsymbol{\sigma}(0)=\boldsymbol{x}_s$, $\boldsymbol{\sigma}(1)=\text{cl}(\boldsymbol{X}_{\text{goal}})$ and $\boldsymbol{\sigma}(\tau)\in\boldsymbol{X}_{\text{free}}$ for all $\tau\in(0,1)$ where $\text{cl}(\cdot)$ is the closure of a set. The sampling-based motion planners usually use a random sampling sequence to return a graph in which the path can be searched. Let $G=(V,E)$ denote the output of the algorithm, where G is the graph defined by a set of nodes $V\subset\boldsymbol{X}_{\text{free}}$ and the set of edges between them $E\subset V\times V$.

Bialkowski, et al. [9] defined a five-tuple augmented graph $AG=(V,E,\boldsymbol{S}_{\text{free}},\boldsymbol{S}_{\text{obs}},\text{Dist})$ to store additional data to realize the safety certificate method. $\boldsymbol{S}_{\text{free}}\subset\boldsymbol{X}_{\text{free}},\boldsymbol{S}_{\text{obs}}\subset\boldsymbol{X}_{\text{obs}}$ denote the sets of points with exact collision status checked by normal method. The map $\text{Dist}:\boldsymbol{S}_{\text{free}}\cup\boldsymbol{S}_{\text{obs}}\rightarrow\Re_{\geq 0}$ stores the distance to the obstacles for points in $\boldsymbol{S}_{\text{free}}$ or to the free space for points in $\boldsymbol{S}_{\text{obs}}$. The dataset $(\boldsymbol{S}_{\text{free}},\boldsymbol{S}_{\text{obs}},\text{Dist})$ constructs a certificate region, when new samples are generated in this region, its collision status is well-known. Hence, this method reduces the normal expensive collision checking numbers.

The certificate method builds up a knowledgeable data set $(\boldsymbol{S}_{\text{free}},\boldsymbol{S}_{\text{obs}},\text{Dist})$ in addition to the search graph (V,E). Here, we present a sampling strategy that guide the search graph grow in a marching way to move steadily to the goal set $\boldsymbol{X}_{\text{goal}}$. In the early stage of the motion planning algorithms a random sample usually fails to be added to V but will be added to $\boldsymbol{S}_{\text{free}}$ or $\boldsymbol{S}_{\text{obs}}$ because of the local planning characteristic. If a random sample is far away from the graph (V,E) it will be pulled close to (V,E). If the local path is in collision the new pulled point is not added to V either. Hence, the free set $\boldsymbol{S}_{\text{free}}$ or in-collision set $\boldsymbol{S}_{\text{obs}}$ contains exploratory information for V, where the exploration performance is the core of sampling-based motion planning algorithms. We define the point $\boldsymbol{x}$ in $\boldsymbol{S}_{\text{free}}$ or $\boldsymbol{S}_{\text{obs}}$ that is closest to $\boldsymbol{X}_{\text{goal}}$ as the marching seed. Then sampling around the marching seed will guide the graph (V,E) move steadily to $\boldsymbol{X}_{\text{goal}}$ as shown in Fig. 1. The conventional random sampling strategy is still adopted to increase the exploration ability of the overall algorithm.

The gradient information is motivated by the optimization algorithms for motion planning in [12], [13], as it has the characteristic to "flick" the in-collision samples quickly. Given a point $\boldsymbol{x}\in\boldsymbol{X}$ (even if it's in $\boldsymbol{X}_{\text{obs}}$), the corresponding gradient $\boldsymbol{g}$ to leave away from the nearest obstacle, and a new sample $\boldsymbol{x}'$ near $\boldsymbol{x}$ and along the direction $\boldsymbol{g}$. Then it's reasonably certain that the new sample $\boldsymbol{x}'$ has better collision properties than $\boldsymbol{x}$. Comparing the sampling-based motion planning algorithms with the optimization-based motion planning algorithms, we can find that: (1) the sampling-based motion planning algorithms are "try-check" framework, i.e., sample a configuration randomly and check its collision status to determine reserve or discard; (2) the optimization-based motion planning algorithms are "try-correct" framework, i.e., given a configuration of the robot then the algorithm uses the interactive information (e.g., distance and gradient) with the environment to improve current performance. In practical applications, the correct procedure is more efficient to let the algorithm converge to a feasible solution or to the optimal solution with respect to a given cost functional.

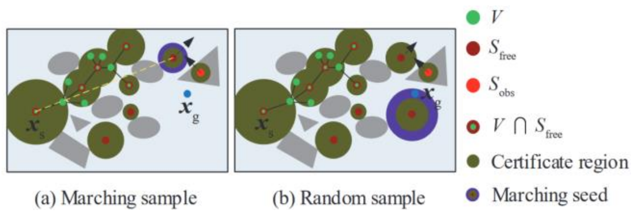


Fig. 1: In the early stage of the motion planning algorithms the set $\boldsymbol{S}_{\text{free}}$ always contains much more exploration information than V. Sampling around the marching seed (the best quality point in $\boldsymbol{S}_{\text{free}}$) helps the exploring tree grow quickly to the goal set $\boldsymbol{X}_{\text{goal}}$, while the traditional random sampling explores the unreached space and enriches $\boldsymbol{S}_{\text{free}}$.

III. Marching Gradient Sampling Strategy

In this section, we present the basic version of the marching gradient sampling strategy: that in which the gradient information is obtained in the configuration space.

A. Notations and Symbols

The symbols $\wedge,\vee$ and $\neg$ denote the logic operations AND, OR, and INVERT. The function $\text{Cost}(\boldsymbol{x}',\boldsymbol{x})$ returns the cost of the optimal path in (V,E) that starts from the node $\boldsymbol{x}'$ and reaches $\boldsymbol{x}$, while if either $\boldsymbol{x}'$ or $\boldsymbol{x}$ is not in V then the cost is infinity. The function $\text{MinCost}(\boldsymbol{x}',\boldsymbol{S})$ returns two tuples $(c,\boldsymbol{x})$ of the minimum cost from $\boldsymbol{x}'$ to the point set $\boldsymbol{S}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n\}$ and the corresponding optimal point $\boldsymbol{x}\in\boldsymbol{S}$. The functions $\text{LineCost}(\boldsymbol{x}',\boldsymbol{x})$ and $\text{MinLineCost}(\boldsymbol{x}',\boldsymbol{S})$ have similar meanings, except that the cost is calculated without considering the obstacles and the restriction in (V,E).

B. Marching Gradient Feasible Sampling

The primitive subroutines, including the *Sampling process*, *Nearest neighbor*, *k-Nearest neighbors*, *Near neighbors*, and *Steering*, are all the same as in [4]. Here, we just present the different additional data structure and primitive subroutines.

1) *Data Structure*: Let $AG=(V,E,\boldsymbol{S}_{\text{free}},\boldsymbol{S}_{\text{obs}},\text{Dist},\text{Grad})$ represent the 6-tuple augmented graph, $V,E,\boldsymbol{S}_{\text{free}},\boldsymbol{S}_{\text{obs}}$ and Dist are the same as in [9]. The map $\text{Grad}:\boldsymbol{S}_{\text{free}}\cup\boldsymbol{S}_{\text{obs}}\rightarrow\Re^d$ stores the gradient information to leave away from the nearest obstacle. V and E are initialized according to a particular algorithm (e.g., RRT), $\boldsymbol{S}_{\text{free}}$ and $\boldsymbol{S}_{\text{obs}}$ are initialized as empty sets. And for convenience, we assume the augmented data structure AG can be accessed directly and do not list it as an input for the algorithms presented in this paper.

2) *Set Source Indicator*: The subroutine $I(i) \in \{0,1,2\}$ returns an indicator random variable for the event that how to generate the ith sample. Specifically, (a) if $I(i)=0$ the ith sample is generated by the traditional sampling strategy used by a particular motion planning algorithm; (b) if $I(i)=1$ the ith sample is generated by using the information in $\boldsymbol{S}_{\text{free}}$; (c) if $I(i)=2$, the process is same as in (b) except that $\boldsymbol{S}_{\text{free}}$ is replaced by $\boldsymbol{S}_{\text{obs}}$. A practical method is:

$$I(i) = \text{MaxIndex}\left(\text{rand}(),\ i_f * \text{rand}(),\ i_o * \text{rand}()\right) \qquad (1)$$

where $\text{rand}()$ generates a uniformly distributed random variable in $[0,1]$, i_f, i_o are weighting factors that determine how often the sets $\boldsymbol{S}_{\text{free}}$ and $\boldsymbol{S}_{\text{obs}}$ should be exploited, and MaxIndex returns the index of the maximum value.

3)*Marching seed*: If the indicator variable $I(i)$ is one or two, then the ith sample is generated according to the point $\boldsymbol{x}_{\text{best}}$ in $\boldsymbol{S}_{\text{free}}$ or $\boldsymbol{S}_{\text{obs}}$ that has the best quality with respect to $\boldsymbol{X}_{\text{goal}}$. And a particular quality metric is the distance:

$$\boldsymbol{x}_{\text{best}} = \text{Nearest}(\boldsymbol{S}_i, \boldsymbol{X}_{\text{goal}}) \qquad (2)$$

where $\boldsymbol{S}_i$ is $\boldsymbol{S}_{\text{free}}$ or $\boldsymbol{S}_{\text{obs}}$. Intuitively, sampling around $\boldsymbol{x}_{\text{best}}$ (Algorithm lines 1.4, 1.9 and Fig. 1(a)) will guide the search graph move steadily to the goal set $\boldsymbol{X}_{\text{goal}}$.

4) *Random direction generator*: Given a unit vector $\boldsymbol{g} \in \Re^d$, the function $\text{SampleDirection}: \boldsymbol{g} \rightarrow \boldsymbol{g}' \in \Re^d$ returns a random unit vector $\boldsymbol{g}'$ that has the property $\boldsymbol{g}^{\text{T}}\boldsymbol{g}' \geq 0$.

5) *Gradient guided sample generator*: Given a point $\boldsymbol{x}$ in $\boldsymbol{S}_{\text{free}} \cup \boldsymbol{S}_{\text{obs}}$, the corresponding gradient $\boldsymbol{g}$ to the obstacle, and define the shortest and the longest moving step length $s_{\min}$ and $s_{\max}$, respectively. The function GradientSample: $(\boldsymbol{x}, \boldsymbol{g}, s_{\min}, s_{\max}) \rightarrow \boldsymbol{x}'$ returns a new point $\boldsymbol{x}'$ in the configuration space:

$$\begin{cases} s = \text{rand}(s_{\min}, s_{\max}) \\ \boldsymbol{x}' = \boldsymbol{x} + s * \text{SampleDirection}(\boldsymbol{g}) \end{cases} \qquad (3)$$

where s is a random variable uniformly distributed between $[s_{\min}, s_{\max}]$. The determination of $s_{\min}, s_{\max}$ can be: (1) if $\boldsymbol{x} \in \boldsymbol{S}_{\text{free}}$, $s_{\min} = -0.9 * d, s_{\max} = d$; (2) if $\boldsymbol{x} \in \boldsymbol{S}_{\text{obs}}$, $s_{\min} = 1.1 * d$, $s_{\max} = 2 * d$, where $d = \text{Dist}(\boldsymbol{x})$. Obviously, if $\boldsymbol{x} \in \boldsymbol{S}_{\text{free}}$ the new generated sample is collision-free. While if $\boldsymbol{x} \in \boldsymbol{S}_{\text{obs}}$ the new sample's collision status needs to be checked exactly. Fig. 2 shows the gradient sample region.

6) *Marching Gradient Sampling*: The marching gradient sampling sub-algorithm is shown in **Algorithm 1**, where the input i_f and i_o are the weighting factors used in (1). The subroutine Sample_i is the traditional sampling process. **Algorithm 1** is a combination of the above subroutines. The random indicator $I(i)$, the random direction generator and the random gradient bias function (3) maintain the exploration capability of the motion planning algorithms. And the marching seed (2) makes full use of the exploitation ability of current obtained information. This sampling process reduces the collision checking numbers evidently.

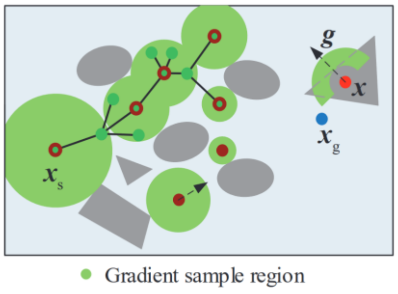


Fig. 2: Given a robot configuration $\boldsymbol{x}$, the minimum distance d and the gradient $\boldsymbol{g}$ to the nearest obstacle. The random state $\boldsymbol{x}'$ is generated by moving $\boldsymbol{x}$ along a random direction $\boldsymbol{g}'$ with random step length s, where $\boldsymbol{g}^{\text{T}}\boldsymbol{g}' \geq 0$ and the step length s is restricted in $[s_{\min}, s_{\max}]$.

```
Algorithm 1: MarchingGradientSample_i(i_f, i_o)
1   $I \leftarrow I(i)$;
2   if $I = 0$ then return $\text{Sample}_i$.
3   else if $I = 1$ then
4       $\boldsymbol{x}_{\text{nearest}} = \text{Nearest}(\boldsymbol{S}_{\text{free}}, \boldsymbol{X}_{\text{goal}})$;
5       $d \leftarrow \text{Dist}(\boldsymbol{x}_{\text{nearest}})$;  $\boldsymbol{g} \leftarrow \text{Grad}(\boldsymbol{x}_{\text{nearest}})$;
6       $s_{\min} = -0.9 * d$,  $s_{\max} = d$;
7       return $\text{GradientSample}(\boldsymbol{x}_{\text{nearest}}, \boldsymbol{g}, s_{\min}, s_{\max})$.
8   else if $I = 2$ then
9       $\boldsymbol{x}_{\text{nearest}} = \text{Nearest}(\boldsymbol{S}_{\text{obs}}, \boldsymbol{X}_{\text{goal}})$;
10      $d \leftarrow \text{Dist}(\boldsymbol{x}_{\text{nearest}})$;  $\boldsymbol{g} \leftarrow \text{Grad}(\boldsymbol{x}_{\text{nearest}})$;
11      $s_{\min} = 1.1 * d$,  $s_{\max} = 2 * d$;
12      return $\text{GradientSample}(\boldsymbol{x}_{\text{nearest}}, \boldsymbol{g}, s_{\min}, s_{\max})$.
```

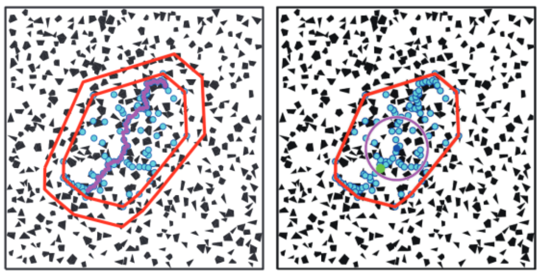
C. *Marching Gradient Informed Optimal Sampling*

If a feasible path has been found, i.e., a sample $\boldsymbol{x}_{\text{g}} \in \boldsymbol{X}_{\text{goal}}$ has been added to V, let c denote the path cost. Given $\boldsymbol{x} \in \boldsymbol{X}_{\text{free}}$, let $f(\boldsymbol{x})$ denote the cost of the optimal path from $\boldsymbol{x}_{\text{s}}$ to $\boldsymbol{X}_{\text{goal}}$ that passes $\boldsymbol{x}$, then the subset $\boldsymbol{X}_f \in \boldsymbol{X}_{\text{free}}$ that can improve current found path can be expressed in terms of c:

$$\boldsymbol{X}_f = \{\boldsymbol{x} \in \boldsymbol{X}_{\text{free}} \mid f(\boldsymbol{x}) < c\} \qquad (4)$$

The expression of $f(\boldsymbol{x})$ is usually unknown for most motion planning problems, hence a lower bound approximation is a common used technique. If the planning space is the Euclidean space, the approximation $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_s\| + \|\boldsymbol{x} - \boldsymbol{x}_g\|$ is used in [7] to construct the informed hyper-ellipsoid subset. That is, a direct line is connected between two configurations without considering the obstacles. As stated in [7], the probability of sampling a point in X_f approaches zero as the found path approaches the optimal path, hence the traditional optimal sampling-based motion planning algorithms have slow convergence speed. In this subsection, we use $(\boldsymbol{S}_{\text{free}}, \boldsymbol{S}_{\text{obs}}, \text{Dist}, \text{Grad})$ to improve the sampling property, extend the informed subset method to non-Euclidean space and increase the convergence speed to the optimal path.

The certificate region constructed by $(\boldsymbol{S}_{\text{free}}, \text{Dist})$ almost covers the free space as the nodes number approaches infinity.



(a) Random sample (b) Marching sample

Fig. 3: After the Branch-and-Bound algorithm the certificate region of set S_{free} covers most of the space that can improve current path quality. (a) Calculating the convex hull of S_{free} and expanding it by an adaptive factor, then the generated convex set has high probability to improve current path quality. (b) Random sampling a point (the blue point) in S_{free}, and using the best quality point (the green point) near it as the marching seed to guide the exploration.

Algorithm 2: Branch-and-Bound $(\boldsymbol{S}_i, c)$

```
1 while $\boldsymbol{x} \leftarrow \text{PickNext}(\boldsymbol{S}_i)$ do
2   if $x \in V$ then
3     if $\text{Cost}(\boldsymbol{x}_s, \boldsymbol{x}) + \text{LineCost}(\boldsymbol{x}, \boldsymbol{x}_g) > c$ then
4       $\boldsymbol{S}_i \leftarrow \boldsymbol{S}_i \setminus \boldsymbol{x}$;
5   else if $\text{LineCost}(\boldsymbol{x}_s, \boldsymbol{x}) + \text{LineCost}(\boldsymbol{x}, \boldsymbol{x}_g) > c$ then
6     $\boldsymbol{S}_i \leftarrow \boldsymbol{S}_i \setminus \boldsymbol{x}$.
7 end
```

Algorithm 3: $\boldsymbol{x}_{\text{best}} = \text{InformedBest}(\boldsymbol{S})$

```
1  $\boldsymbol{S}_1 \leftarrow \varnothing$;
2  $\text{ratio} \leftarrow 0, \quad \text{num} \leftarrow 0$;
3  $c \leftarrow \infty, \quad \boldsymbol{x}_{\text{best}} \leftarrow \varnothing$;
4  while $\boldsymbol{x} \leftarrow \text{PickNext}(\boldsymbol{S})$ do
5    if $x \in V$ then
6      $\text{ratio} \leftarrow \text{ratio} + \text{Cost}(\boldsymbol{x}_s, \boldsymbol{x}) / \text{LineCost}(\boldsymbol{x}_s, \boldsymbol{x})$;
7      $\text{num} \leftarrow \text{num} + 1$;
8      if $\text{Cost}(\boldsymbol{x}_s, \boldsymbol{x}) + \text{LineCost}(\boldsymbol{x}, \boldsymbol{x}_g) < c$ then
9        $c \leftarrow \text{Cost}(\boldsymbol{x}_s, \boldsymbol{x}) + \text{LineCost}(\boldsymbol{x}, \boldsymbol{x}_g)$;
10       $\boldsymbol{x}_{\text{best}} \leftarrow \boldsymbol{x}$;
11     end
12   else
13     $\boldsymbol{S}_1 \leftarrow \boldsymbol{S}_1 \cup \{\boldsymbol{x}\}$;
14 end
15 if $\text{num} > 0$ then
16   $\text{ratio} \leftarrow \text{ratio} / \text{num}$;
17 else
18   $\text{ratio} \leftarrow 1$;
19 while $\boldsymbol{x} \leftarrow \text{PickNext}(\boldsymbol{S}_1)$ do
20   if $\text{ratio} * \text{LineCost}(\boldsymbol{x}_s, \boldsymbol{x}) + \text{LineCost}(\boldsymbol{x}, \boldsymbol{x}_g) < c$ then
21     $c \leftarrow \text{ratio} * \text{LineCost}(\boldsymbol{x}_s, \boldsymbol{x}) + \text{LineCost}(\boldsymbol{x}, \boldsymbol{x}_g)$;
22     $\boldsymbol{x}_{\text{best}} \leftarrow \boldsymbol{x}$.
23   end
24 end
```

Algorithm 4: $\text{MGInformedSample}_i(i_f, i_o, r)$

```
1  $I \leftarrow I(i)$;
2  if $I = 0$ then
3    return $\text{UniformeSample}_i(\exp - \text{conv}(\boldsymbol{S}_{\text{free}}))$.
4  else if $I = 1$ then
5    $\boldsymbol{x} \leftarrow \text{UniformeSample}(\boldsymbol{S}_{\text{free}})$;
6    $\boldsymbol{S}_{\text{near}} \leftarrow \text{Near}(\boldsymbol{S}_{\text{free}}, \boldsymbol{x}, \boldsymbol{r})$;
7    $\boldsymbol{x}_{\text{best}} = \text{InformedBest}(\boldsymbol{S}_{\text{near}})$;
8    $d \leftarrow \text{Dist}(\boldsymbol{x}_{\text{best}}), \quad \boldsymbol{g} \leftarrow \text{Grad}(\boldsymbol{x}_{\text{best}})$;
9    $s_{\min} = -0.9 * d, \quad s_{\max} = d$;
10   return $\text{GradientSample}(\boldsymbol{x}_{\text{best}}, \boldsymbol{g}, s_{\min}, s_{\max})$.
11 else if $I = 2$ then
12   $\boldsymbol{x} \leftarrow \text{UniformeSample}(\boldsymbol{S}_{\text{obs}})$;
13   $d \leftarrow \text{Dist}(\boldsymbol{x}); \quad \boldsymbol{g} \leftarrow \text{Grad}(\boldsymbol{x})$;
14   $s_{\min} = 1.1 * d, \quad s_{\max} = 2 * d$;
15   return $\text{GradientSample}(\boldsymbol{x}_{\text{nearest}}, \boldsymbol{g}, s_{\min}, s_{\max})$.
```

And the left un-covered region is around the boundaries of the obstacles [9]. Therefore, the set S_{free} contains much more exploration information than V, and the cardinality of S_{free} is much less than that of V as the planner tries to find the optimal path. The branch and bound algorithm shown in **Algorithm 2** can increase the superiority of S_{free} furtherly. In **Algorithm 2**, S_i denotes S_{free} or S_{obs}, c is the cost of current found path, PickNext ( ) takes out a point from the set in order. As we can see, the **Algorithm 2** deletes the unnecessary nodes in S_{free} for finding the optimal path while maintaining the exploration ability. And the steps 2.2-2.5 never occurs for S_{obs}.

After the Branch-and-Bound algorithm, the points in set S_{free} and S_{obs} contain the information that can almost improve current path quality. Therefore, the marching gradient informed optimal sampling strategy proposed in this paper is also a combination of traditional random sampling method, sampling according to S_{free} and sampling according to S_{obs}. The traditional sampling method fully explores the space, increases the nodes number in S_{free} and makes S_{free} more knowledgeable. While sampling according to S_{obs} explores the information around the obstacles in the sense that the optimal path always has small clearance to the obstacles. At last, sampling according to S_{free} makes the algorithm informed to reduce the path cost quickly.

Similar to the informed hyper-ellipsoid subset, the random sampling process is restricted to a subspace deduced by S_{free}. In this paper, an enlarged convex hull of S_{free} is selected as the random sampling space as shown in Fig. 3(a) and Algorithm line 4.3, where the expansion factor is an adaptive variable. The sampling process according to S_{obs} has a small change, that instead of using the marching seed (2), a uniform sampling from S_{obs} (Algorithm line 4.12) is utilized to increase the probability of improving the current path. Since the uniform sampling can improve the whole path quality not just a small segment near X_{goal}. In the following, we will show the method in detail how to generate a sample according to S_{free} that can improve the path quality in a marching way.

Algorithm 5: Modified RRT

```
1  $V \leftarrow \{\boldsymbol{x}_s\}; E \leftarrow \varnothing; \boldsymbol{S}_{\text{free}} \leftarrow \varnothing; \boldsymbol{S}_{\text{obs}} \leftarrow \varnothing; \boldsymbol{S}_{\text{goal}} \leftarrow \varnothing$;
2  for $i = 1, 2, \cdots, n$ do
3      $(\boldsymbol{x}_{\text{rand}}, I) \leftarrow \text{MarchingGradientSample}_i$;
4      if $I = 1 \ \vee$ CollisionFreePoint($\boldsymbol{x}_{\text{rand}}$) then
5          if $\boldsymbol{x}_{\text{rand}} \in \boldsymbol{X}_{\text{goal}}$ then
6              $\boldsymbol{S}_{\text{goal}} \leftarrow \boldsymbol{S}_{\text{goal}} \bigcup \{\boldsymbol{x}_{\text{rand}}\}$;
7          $\boldsymbol{x}_{\text{nearest}} = \text{Nearest}((V, E), \boldsymbol{x}_{\text{rand}})$;
8          $\boldsymbol{x}_{\text{new}} = \text{Steer}(\boldsymbol{x}_{\text{nearest}}, \boldsymbol{x}_{\text{rand}})$;
9          if CollisionFreePath($\boldsymbol{x}_{\text{nearest}}, \boldsymbol{x}_{\text{new}}$) then
10             $V \leftarrow V \bigcup \boldsymbol{x}_{\text{new}}$;   $E \leftarrow E \bigcup \{\boldsymbol{x}_{\text{nearest}}, \boldsymbol{x}_{\text{new}}\}$;
11             $(c, \boldsymbol{x}_g) \leftarrow \text{MinLineCost}(\boldsymbol{x}_{\text{new}}, \boldsymbol{X}_{\text{goal}})$;
12             if $c \le \delta \ \wedge$ CollisionFreePath($\boldsymbol{x}_{\text{new}}, \boldsymbol{x}_g$) then
13                 $V \leftarrow V \bigcup \boldsymbol{x}_g$;   $E \leftarrow E \bigcup \{\boldsymbol{x}_{\text{new}}, \boldsymbol{x}_g\}$;
14                 break.
15     end
16 end
```

We will first describe the **Algorithm 3**: InformedBest. Given a subset $\boldsymbol{S} \subset \boldsymbol{S}_{\text{free}}$, InformedBest finds the most informed point $\boldsymbol{x}_{\text{best}}$ in $\boldsymbol{S}$ that can improve the current path quality in the fastest way. Specifically, $\boldsymbol{x}_{\text{best}}$ is the point that has the smallest cost value in terms of $\text{Cost}(\boldsymbol{x}_s, \boldsymbol{x}) + \text{Line-}$ Cost $(\boldsymbol{x}, \boldsymbol{x}_g)$ for all $\boldsymbol{x} \in \boldsymbol{S}$, as shown in Algorithm lines 3.8-3.11. Nevertheless, for a point $\boldsymbol{x} \notin \boldsymbol{S} \bigcap V$, there is not the cost function $\text{Cost}(\boldsymbol{x}_s, \boldsymbol{x})$. Hence, an approximation (Algorithm lines 3.6-3.7, 3.15-3.24) is calculated according to the main idea that the points in a small neighborhood always have similar characteristics. And this approximation is based the ratio of $\text{Cost}(\boldsymbol{x}_s, \boldsymbol{x})$ and $\text{LineCost}(\boldsymbol{x}_s, \boldsymbol{x})$. Then using the InformedBest function, the sampling strategy according to $\boldsymbol{S}_{\text{free}}$ can be constructed in Algorithm lines 4.5-4.7 and is shown in Fig. 3(b). A uniform random sample is picked out from $\boldsymbol{S}_{\text{free}}$ to enhance the algorithm's exploitation capability. And the most informed point is selected from its neighborhood to increase the algorithm's marching convergence speed.

*D. Examples with RRT and RRT**

We now show how the popular sampling-based motion planning algorithms, RRT and RRT*, can be modified to use the marching gradient sampling strategy. The modified versions of RRT and RRT* are shown in **Algorithm 5** and **Algorithm 6**, respectively. The **Algorithm 1** is used in both modified algorithms to find out a feasible path quickly, then the **Algorithm 4** is applied to the modified RRT* algorithm to increase the convergence speed to the optimal solution. In both algorithms, δ at line 5.12 and line 6.20 is the largest moving step length between two configurations. In **Algorithm 6**, ε at line 6.28 denotes the optimal tolerance, n_o at line 6.26 is a positive integer that denotes the period of optimal solution checking, the function Mod calculates the remainder of a number, and δ_b at line 6.30 is a lower bound of cost decreasing that triggers the branch and bound process.

Algorithm 6: Modified RRT*

```
1  $V \leftarrow \{\boldsymbol{x}_s\}; E \leftarrow \varnothing; \boldsymbol{S}_{\text{free}} \leftarrow \varnothing; \boldsymbol{S}_{\text{obs}} \leftarrow \varnothing; \boldsymbol{S}_{\text{goal}} \leftarrow \varnothing$;
2  issolved $\leftarrow$ False;   isoptimal $\leftarrow$ False;   cgoal $\leftarrow \infty$;
3  for $i = 1, 2, \cdots, n$ do
4      if $\neg$issolved then
5          $(\boldsymbol{x}_{\text{rand}}, I) \leftarrow \text{MarchingGradientSample}_i$;
6      else
7          $(\boldsymbol{x}_{\text{rand}}, I) \leftarrow \text{MGInformedSample}_i$;
8          $(c, \boldsymbol{x}_g) \leftarrow \text{MinLineCost}(\boldsymbol{x}_{\text{rand}}, \boldsymbol{S}_{\text{goal}})$;
9          if $\text{LineCost}(\boldsymbol{x}_s, \boldsymbol{x}_{\text{rand}}) + c > \text{cgoal}$ then
10             Continue;
11     if $I = 1 \ \vee$ CollisionFreePoint($\boldsymbol{x}_{\text{rand}}$) then
12         if $\boldsymbol{x}_{\text{rand}} \in \boldsymbol{X}_{\text{goal}}$ then
13             $\boldsymbol{S}_{\text{goal}} \leftarrow \boldsymbol{S}_{\text{goal}} \bigcup \{\boldsymbol{x}_{\text{rand}}\}$;
14         $\boldsymbol{x}_{\text{nearest}} = \text{Nearest}((V, E), \boldsymbol{x}_{\text{rand}})$;
15         $\boldsymbol{x}_{\text{new}} = \text{Steer}(\boldsymbol{x}_{\text{nearest}}, \boldsymbol{x}_{\text{rand}})$;
16         if CollisionFreePath($\boldsymbol{x}_{\text{nearest}}, \boldsymbol{x}_{\text{new}}$) then
17             MinCostConnection($\boldsymbol{x}_{\text{new}}$);   Rewire($\boldsymbol{x}_{\text{new}}$);
18             if $\neg$issolved then
19                 $(c, \boldsymbol{x}_g) \leftarrow \text{MinLineCost}(\boldsymbol{x}_{\text{new}}, \boldsymbol{S}_{\text{goal}})$;
20                 if $c \le \delta \ \wedge$ CollisionFreePath($\boldsymbol{x}_{\text{new}}, \boldsymbol{x}_g$) then
21                     $V \leftarrow V \bigcup \boldsymbol{x}_g$;   $E \leftarrow E \bigcup \{\boldsymbol{x}_{\text{new}}, \boldsymbol{x}_g\}$;
22                     issolved $\leftarrow$ True;
23                     cgoal $\leftarrow \text{Cost}(\boldsymbol{x}_s, \boldsymbol{x}_g)$;
24                     Branch-and-Bound ($\boldsymbol{S}_{\text{free}}$, cgoal);
25                     Branch-and-Bound ($\boldsymbol{S}_{\text{obs}}$, cgoal);
26             else if $\text{Mod}(i, n_o) = 0$ then
27                 cgoaln $\leftarrow \text{MinCost}(\boldsymbol{x}_s, \boldsymbol{S}_{goal})$;
28                 if cgoal $-$ cgoaln $\le \varepsilon$ then
29                     isoptimal $\leftarrow$ True; break;
30                 else if cgoal $-$ cgoal $> \delta_b$ then
31                     cgoal $\leftarrow$ cgoaln
32                     Branch-and-Bound ($\boldsymbol{S}_{\text{free}}$, cgoal);
33                     Branch-and-Bound ($\boldsymbol{S}_{\text{obs}}$, cgoal).
34 end
```

IV. Simulation Experiments

In this section we perform experiments in a simulated environment to evaluate the performance of the marching gradient sampling strategy guided RRT and RRT* algorithms. Let ALG be a label indicating one of the algorithms, the convergence speed to a feasible path or to the optimal path is compared between the algorithm ALG without and with this strategy. We denote the algorithms as ALGc and ALGcmg, respectively, where the notation c is the safety certificate method and the notation mg is the marching gradient sampling strategy. The simulation environment consists of a unit-square workspace with 470 randomly placed convex polygonal obstacles (see Fig. 4), and a point robot moves in it without kinematics constraint. One hundred collision-free start state and goal state pairs are randomly generated in this space to conduct the simulations, and the start-goal states maintain the constraint $\|\boldsymbol{x}_s - \boldsymbol{x}_g\| \ge 0.71$. For each problem fifty trials are performed for both ALGc and ALGcmg algorithms (upon the

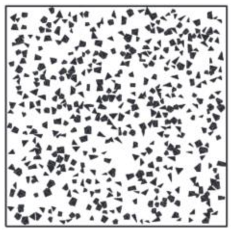


Fig. 4: A unit-square with 470 randomly placed obstacles are designed as the experiment scenario.

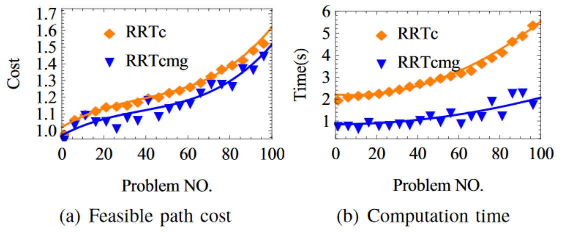


Fig. 5: The solution computed by RRTc and RRTcmg algorithms. (a) The path cost of the 100 randomly generated problems. (b) The computation time used by the algorithms.

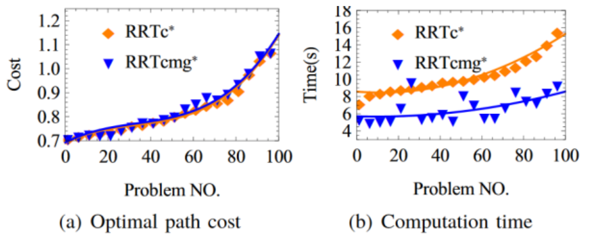


Fig. 6: The solution computed by RRTc* and RRTcmg* algorithms. (a) The path cost of the 100 randomly generated problems. (b) The computation time used by the algorithms

hundred pre-defined problems), and the average value denotes the solution. Simulations are conducted on a 4.20GHz Intel Core i7-7700K CPU with 8GBRAM in MATLAB.

The feasible/optimal path cost and the computation time of the one hundred problems computed by ALGc and ALGcmg algorithms are shown in Fig. 5 and Fig. 6, respectively. The problem NO. is sorted by ALGc's corresponding value in ascending order. That is, the problem NO. in different figures may be different. Figs. 5(a) and 6(a) show that both algorithms achieve the same performance in solution quality, and Figs. 5(b) and 6(b) demonstrate that ALGcmg's computation speed is near 2x speedup than that of the ALGc algorithm.

V. Conclusion

In this paper, we present a marching gradient sampling strategy to accelerate the convergence speed of sampling-based motion planning algorithms. This strategy uses an informed certificate set to solve both the feasible path planning problem and the optimal path planning problem. The informed certificate data set contains exact collision status checked robot states together with the minimum distance and gradient to the nearest obstacle. This data set assists the sampling subroutine in a marching way to converge to a feasible path quickly. Then the algorithm utilizes the data set to generate samples in the subspace that can truly improve current path quality and to converge informedly to the optimal path. The distance and gradient information helps generate samples with almost sure collision status and reduces the collision checking numbers. We used Monte Carlo simulations to evaluate the performance and convergence speed of this sampling strategy upon RRT and RRT* algorithms and compared it to the safety certificate strategy. The results demonstrate that our sampling strategy is twice faster to converge to the feasible path or the optimal path while maintaining the same solution quality.

VI. Acknowledgment

The authors wish to acknowledge the support from the National Natural Science Foundation of China (Grant NO. 51975236) and the Program for Guangdong Key Areas Research and Development (Grant No. 2019B010924005).

References

[1] S. M. LaValle, Planning algorithms. Cambridge university press,2006.

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Prob-abilistic roadmaps for path planning in high-dimensional configurationspaces,” IEEE transactions on Robotics and Automation, vol. 12, no. 4,pp. 566–580, 1996.

[3] S. M. LaValle and J. J. Kuffner Jr, “Randomized kinodynamic plan-ning,”The international journal of robotics research, vol. 20, no. 5,pp. 378–400, 2001.

[4] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimalmotion planning,”Int. J. Robot. Res., vol. 30, no. 7, pp. 846–894,2011.

[5] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast marchingtree: A fast marching sampling-based method for optimal motionplanning in many dimensions,”The International journal of roboticsresearch, vol. 34, no. 7, pp. 883–921, 2015.

[6] O. Salzman and D. Halperin, “Asymptotically near-optimal rrt forfast, high-quality motion planning,”IEEE Transactions on Robotics,vol. 32, no. 3, pp. 473–483, 2016.

[7] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, “Informed samplingfor asymptotically optimal path planning,”IEEE Transactions onRobotics, vol. 34, no. 4, pp. 966–984, 2018.

[8] J. Pan and D. Manocha, “Fast probabilistic collision checking forsampling-based motion planning using locality-sensitive hashing,”TheInternational Journal of Robotics Research, vol. 35, no. 12, pp. 1477–1496, 2016.

[9] J. Bialkowski, M. Otte, S. Karaman, and E. Frazzoli, “Efficientcollision checking in sampling-based motion planning via safetycertificates,”The International Journal of Robotics Research, vol. 35,no. 7, pp. 767–796, 2016.

[10] M. Otte and N. Correll, “C-forest: Parallel shortest path planning withsuperlinear speedup,”IEEE Transactions on Robotics, vol. 29, no. 3,pp. 798–806, 2013.

[11] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller,“Anytime motion planning using the rrt,” in2011 IEEE InternationalConference on Robotics and Automation. IEEE, 2011, pp. 1478–1483.

[12] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith,C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, “Chomp: Covarianthamiltonian optimization for motion planning,”The InternationalJournal of Robotics Research, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[13] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan,S. Patil, K. Goldberg, and P. Abbeel, “Motion planning with sequentialconvex optimization and convex collision checking,”The InternationalJournal of Robotics Research, vol. 33, no. 9, pp. 1251–1270, 2014.