Sampling in Motion Planning

Algorithms and Data Structures 2 – Motion Planning and its applications

University of Applied Sciences Stuttgart

Dr. Daniel Schneider

Random Samples

- Every motion planner is using random samples to generate configurations.
- In the original approaches, a uniform sampling is used.
- Uniform sampling works, but also has some challenges…

Algorithm 3: sPRM$(\{(c^i_{init}, c^i_{goal})\}, r, n)$

$E \leftarrow \emptyset,\ V \leftarrow \emptyset$

foreach $((c^i_{init}, c^i_{goal}) \in \{(c^i_{init}, c^i_{goal})\})$ **do**

  $V \leftarrow V \cup c^i_{init} \cup c^i_{goal}$

for $j \leftarrow 0$ **to** n **do**

  $V \leftarrow V \cup CFreeSample()$

foreach $v \in V$ **do**

  $U \leftarrow Neighbors(v, V, r)$

  foreach $u \in U$ **do**

    if $(edgeIsValid(u, v))$ **then**

      $E \leftarrow E \cup (u, v)$

foreach $((c^i_{init}, c^i_{goal}) \in \{(c^i_{init}, c^i_{goal})\})$ **do**

  if $connected(c^i_{init}, c^i_{goal}, V, E)$ **then**

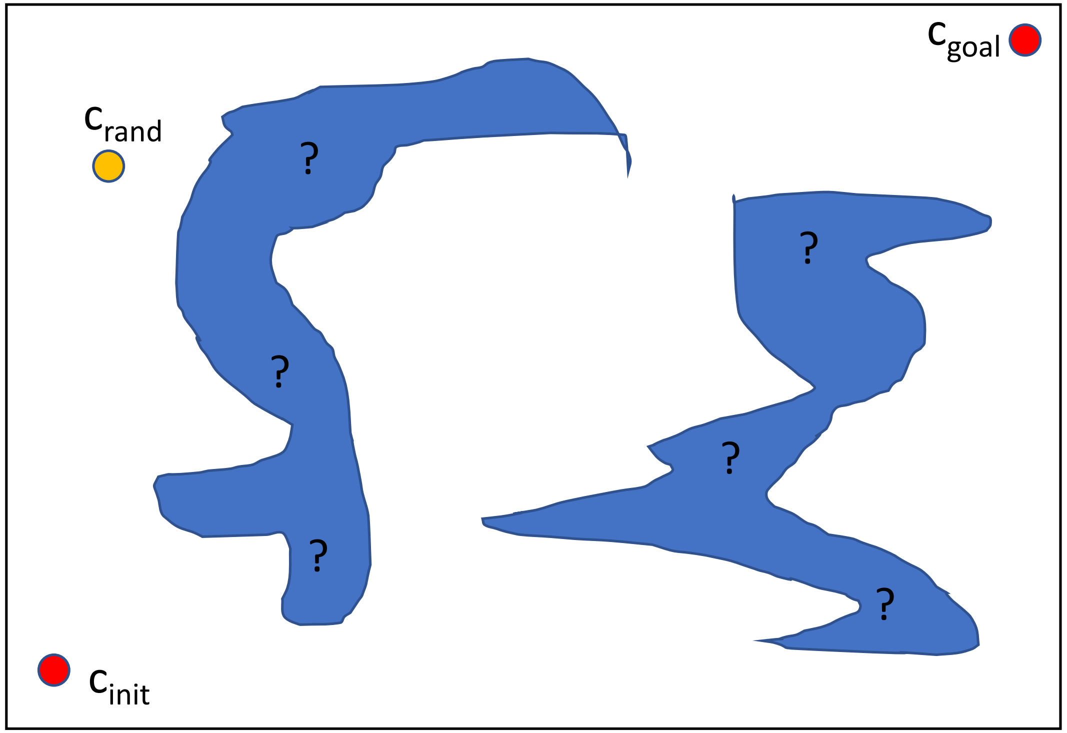
    $\sigma_i = shortestPath(c^i_{init}, c^i_{goal}, V, E)$

  else

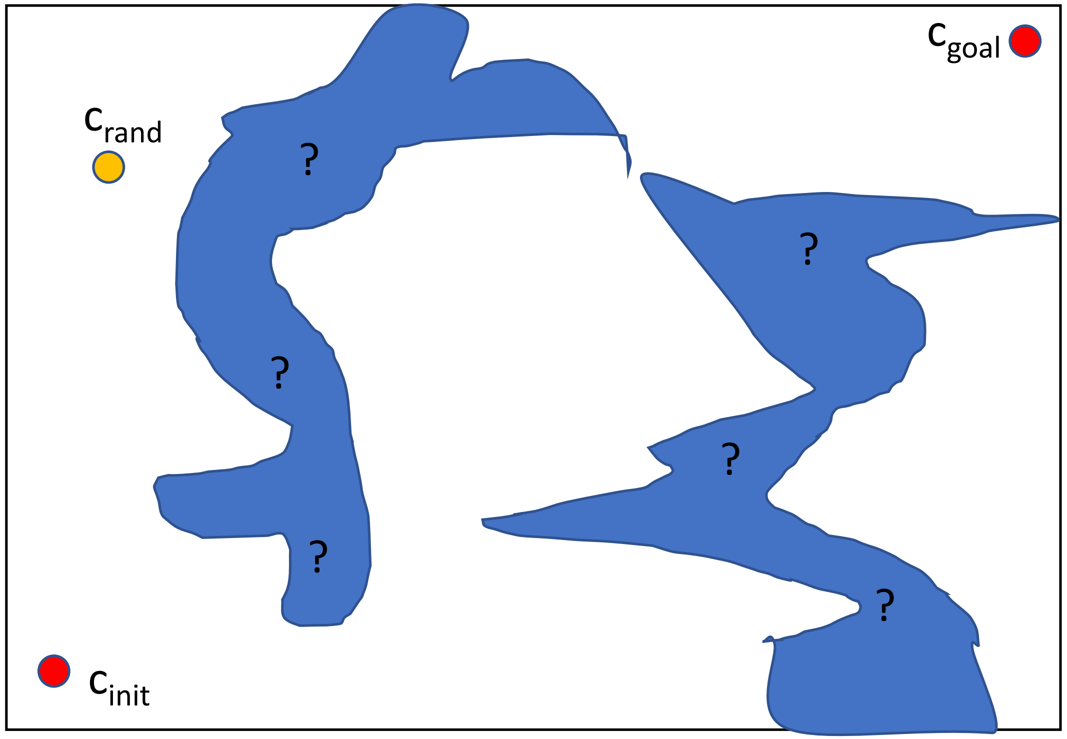
    $\sigma_i \leftarrow \emptyset$

return $\{\sigma_i\}$

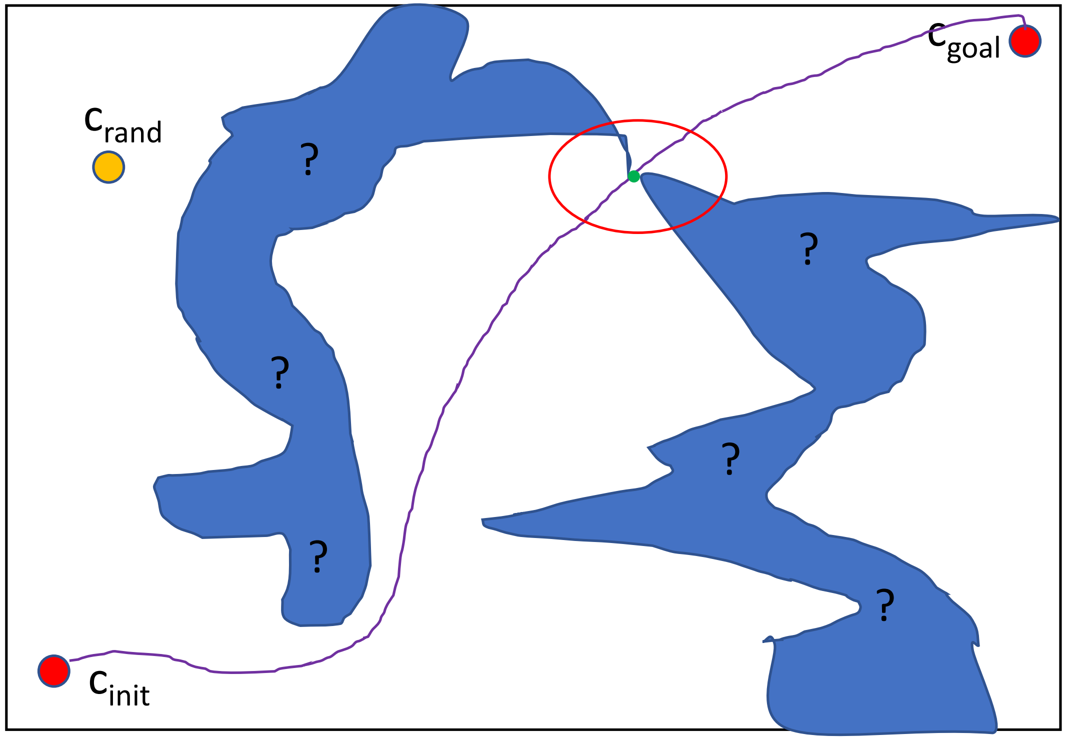
One major challenge: Narrow Passage



One major challenge: Narrow Passage



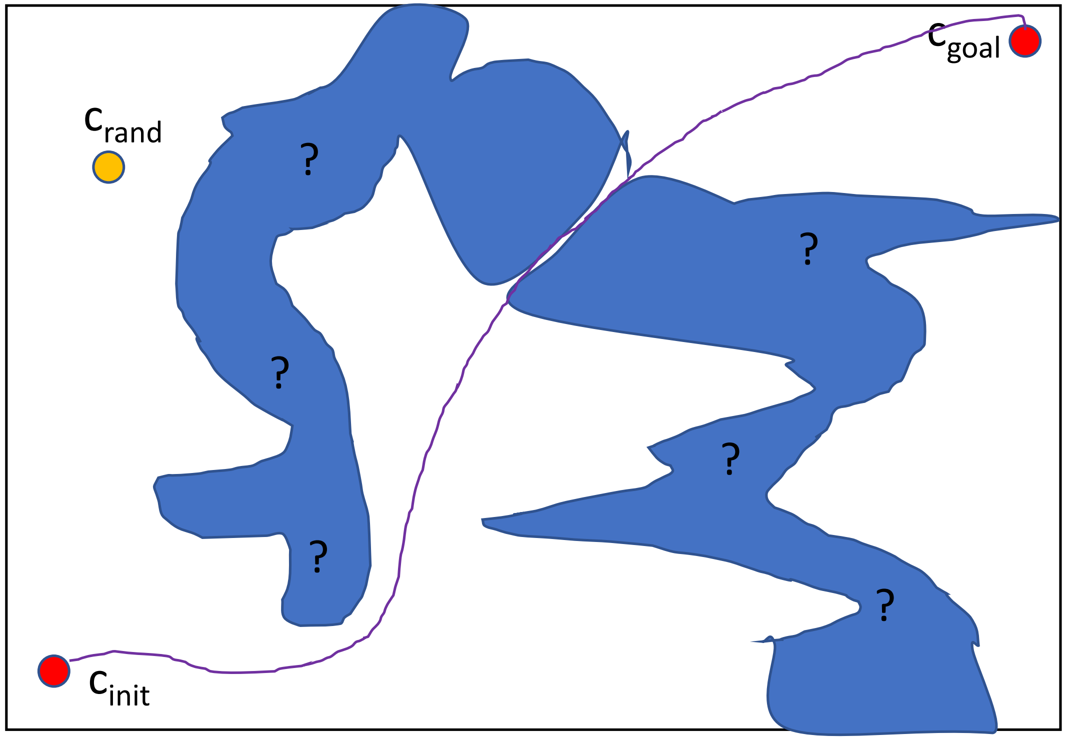
One major challenge: Narrow Passage



Narrow Passage:

Narrow passages are regions of the configuration space that are part of the solution path and the smallest sphere along this path is very small in relation to the free confiuguration space.

One major challenge: Narrow Passage



The challenge for uniform sampling:

The probability to sample configurations in the free space of the narrow passage is very low what leads to

- Long running times
- Data structures (e.g. nearest neighbour search) are getting to large.

Solution: Sampling Strategies

- Do not only use uniform sampled configurations but also include other sampling strategies.
- In this lecture we will have a look at the most common sampling strategies.

Some Notes:

1) For all sampling strategies we still need the characteristic that the sampling covers (with increasing amount of samples) the whole configuration space.

If this property is not given by the new strategy one common approach is to mix the chosen strategy with uniform samples. (e.g. 50% uniform Sampling, 50% Strategy XYZ)

2) Sampling Strategies are often applied to PRM algorithms and EST algorithms. Research has shown that the impact of sampling strategies is higher on these algorithms than on RRT-like approaches.

Categories of sampling strategies.

- **Configurationspace-based sampling**: These sampling strategies are, beside the simple collision check, only using information of the configuration space. Examples are:
  - Uniform Sampling
  - Gaussian Sampling
  - Bridge Test Sampling
  - …
- **Workspace-based sampling**: These sampling strategies are using information of the workspace to guide the sampling. Information like distance of the robot to its surrounding environment.
  - Sphere Sampling
  - Medial Axis Sampling
  - ….

Sampling Strategy: Gaussian Sampling

- This sampling strategy was one of the first approaches to the uniform sampling approach.
- Still today this approach is used in many industrial applications to plan motions and proves to be very efficient.
- The implementation is very simple and there is no additional data structure needed.
- This sampling approach also includes the C_{obs} for the algorithm and not only the C_{free} as in the original approach.


Gaussian Sampling for Probabilistic Roadmap Planners*

Valérie Boor, Mark H. Overmars, A. Frank van der Stappen
Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands
Email: {*valerie,markov,frankst*}*@cs.uu.nl*

Abstract

Probabilistic roadmap planners (PRMs) have become a popular technique for motion planning that has shown great potential. A critical aspect of a PRM is the probabilistic strategy used to sample the free configuration space. In this paper we present a new, simple sampling strategy, which we call the Gaussian sampler, that gives a much better coverage of the difficult parts of the free configuration space compared with the standard uniform sampler. This results in much smaller roadmaps that can be computed faster. The approach uses only elementary operations which makes it suitable for many different planning problems. Experiments indicate that the technique is indeed very efficient in practice.

1 Introduction

Automated motion planning is rapidly gaining importance in various fields. Beside the obvious use in robotics, new applications arise in fields such as animation, computer games, virtual medicine, and maintenance planning and training in industrial CAD systems. See Figure 1 for a typical example of such an industrial scene with over 4000 obstacles, containing a total of over 350000 triangles. Such applications put different demands on the motion planners. In particular, the scenes they have to work in are very complex and obstacles tend to be clustered in certain areas. As a result motion planning is easy at many places in the scenes but very difficult in such cluttered areas. This asks for planners that adapt their strategy to the local properties of the scene and for which the complexity of finding a path depends more on the difficulty of the path than on the complexity of the total scene.

Over the past two decades the motion planning problem has been studied extensively. Many different approaches have been proposed, including potential field techniques, roadmap methods, cell decomposition, neural networks, and genetic algorithms. (See the book of Latombe [15] for an extensive overview of the situation up to 1991 and e.g. the proceedings of the yearly IEEE International Conference on Robotics and Automation for many more recent results.) Unfortunately, many of these approaches are not very well suited for the applications described above. The complexity of the methods often depends on the complexity of the total scene (like the cell-decomposition methods) or the planners cannot deal adequately with cluttered parts of the environments (like potential field approaches). The *probabilistic*

*This research has been supported by the ESPRIT LTR project MOLOG. A preliminary version of this paper appeared in [6].

2

Gaussian Sampling for Probabilistic Roadmap Planners, V. Boor, H. Overmars, A. F. von der Stappen, 2001

Sampling Strategy: Gaussian Sampling

Algorithm 2 GaussianSampling

loop
  $c_1 \leftarrow$ a random forbidden configuration (c_t, c_r)
  $d \leftarrow$ a distance chosen according to a normal distribution
  $c_2 \leftarrow$ a configuration (c'_t, c_r) with c'_t at distance d from c_t
  if $c_2 \in \mathcal{C}_{\text{free}}$ **then**
    add c_2 to the graph

Algorithm 3: sPRM($\{(c^i_{init}, c^i_{goal})\}, r, n$)

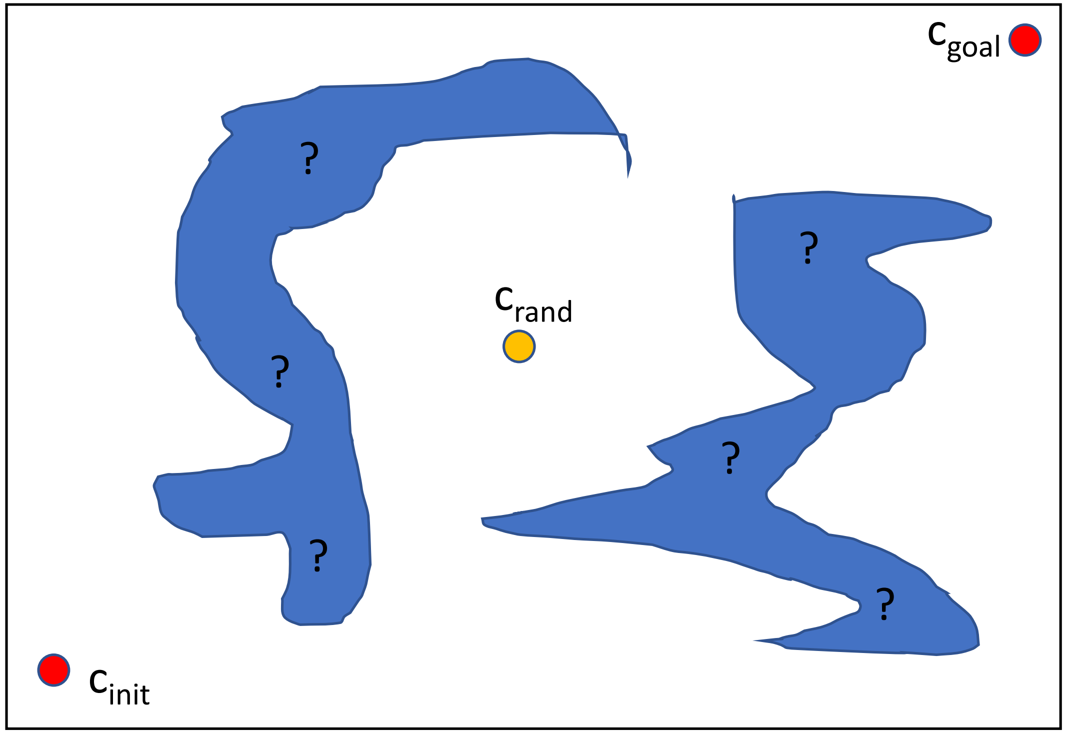
$E \leftarrow \emptyset$, $V \leftarrow \emptyset$
foreach $((c^i_{init}, c^i_{goal}) \in \{(c^i_{init}, c^i_{goal})\})$ **do**
  $V \leftarrow V \cup c^i_{init} \cup c^i_{goal}$
for $j \leftarrow 0$ **to** n **do**
  $V \leftarrow V \cup CFreeSample()$
foreach $v \in V$ **do**
  $U \leftarrow Neighbors(v, V, r)$
  foreach $u \in U$ **do**
    if $(edgeIsValid(u, v))$ **then**
      $E \leftarrow E \cup (u, v)$
foreach $((c^i_{init}, c^i_{goal}) \in \{(c^i_{init}, c^i_{goal})\})$ **do**
  if $connected(c^i_{init}, c^i_{goal}, V, E)$ **then**
    $\sigma_i = shortestPath(c^i_{init}, c^i_{goal}, V, E)$
  else
    $\sigma_i \leftarrow \emptyset$
return $\{\sigma_i\}$

Sampling Strategy: Gaussian Sampling

Algorithm 2 GaussianSampling

```
1: loop
2:   c_1 ← a random forbidden configuration (c_t, c_r)
3:   d ← a distance chosen according to a normal distribution
4:   c_2 ← a configuration (c'_t, c_r) with c'_t at distance d from c_t
5:   if c_2 ∈ C_free then
6:     add c_2 to the graph
```

- This configuration is in free of collision.
- Therefore is discarded

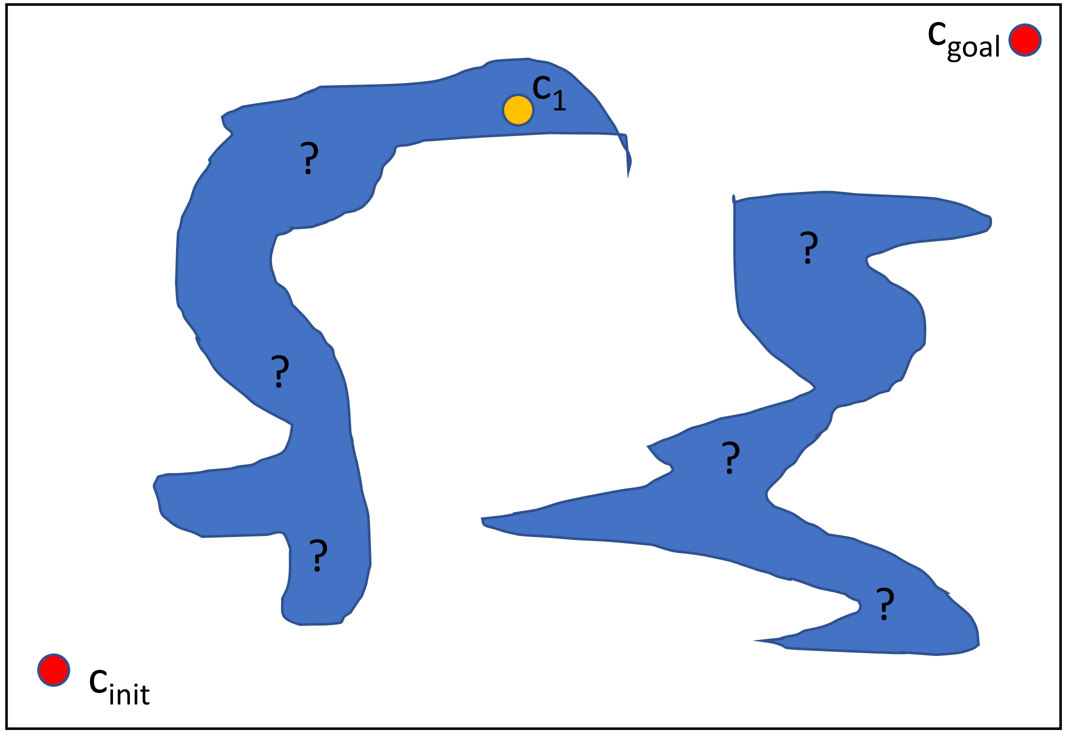


Sampling Strategy: Gaussian Sampling

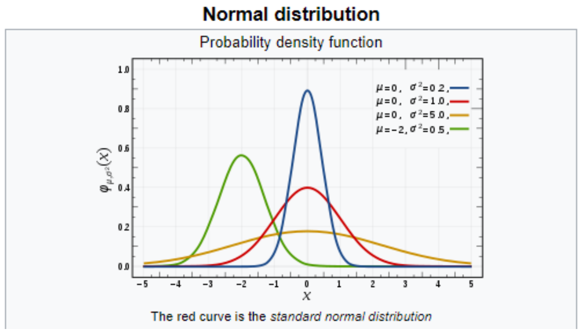
Algorithm 2 GaussianSampling

1: **loop**
2: $c_1 \leftarrow$ a random forbidden configuration (c_t, c_r)
3: $d \leftarrow$ a distance chosen according to a normal distribution
4: $c_2 \leftarrow$ a configuration (c'_t, c_r) with c'_t at distance d from c_t
5: **if** $c_2 \in \mathcal{C}_{\text{free}}$ **then**
6: add c_2 to the graph

- This configuration not free of collision and so we take this configuration as c_1.



Sampling Strategy: Gaussian Sampling

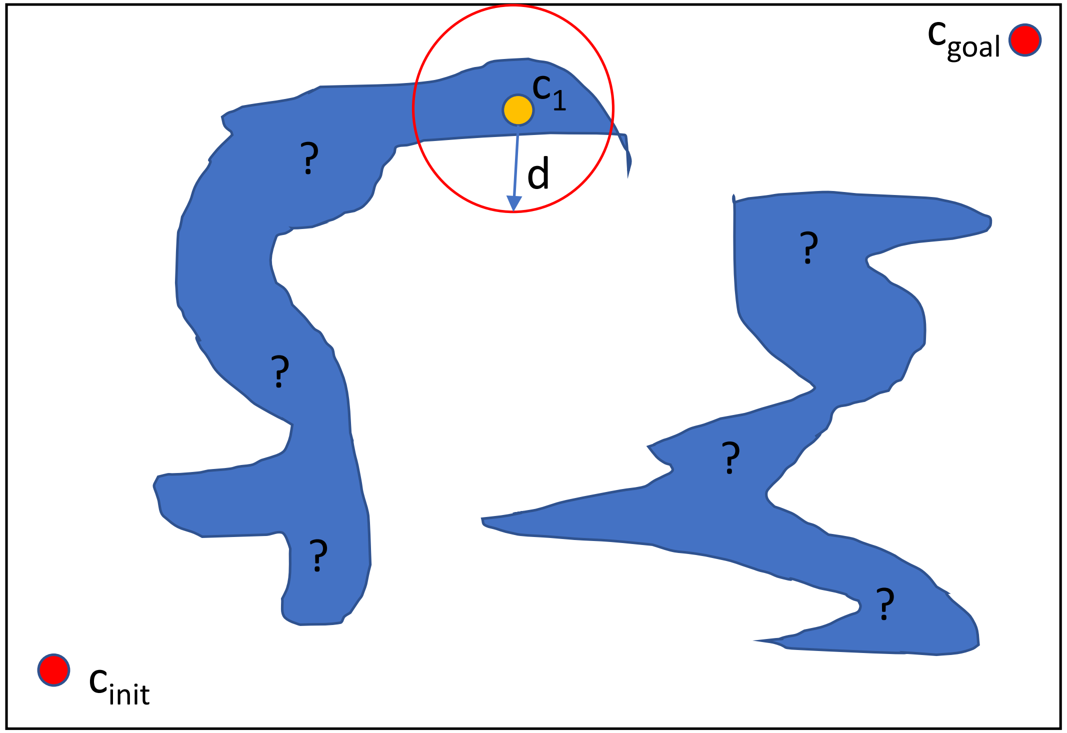


www.wikipedia.com

Algorithm 2 GaussianSampling

```
loop
    c_1 ← a random forbidden configuration (c_t, c_r)
    d ← a distance chosen according to a normal distribution
    c_2 ← a configuration (c'_t, c_r) with c'_t at distance d from c_t
    if c_2 ∈ C_free then
        add c_2 to the graph
```

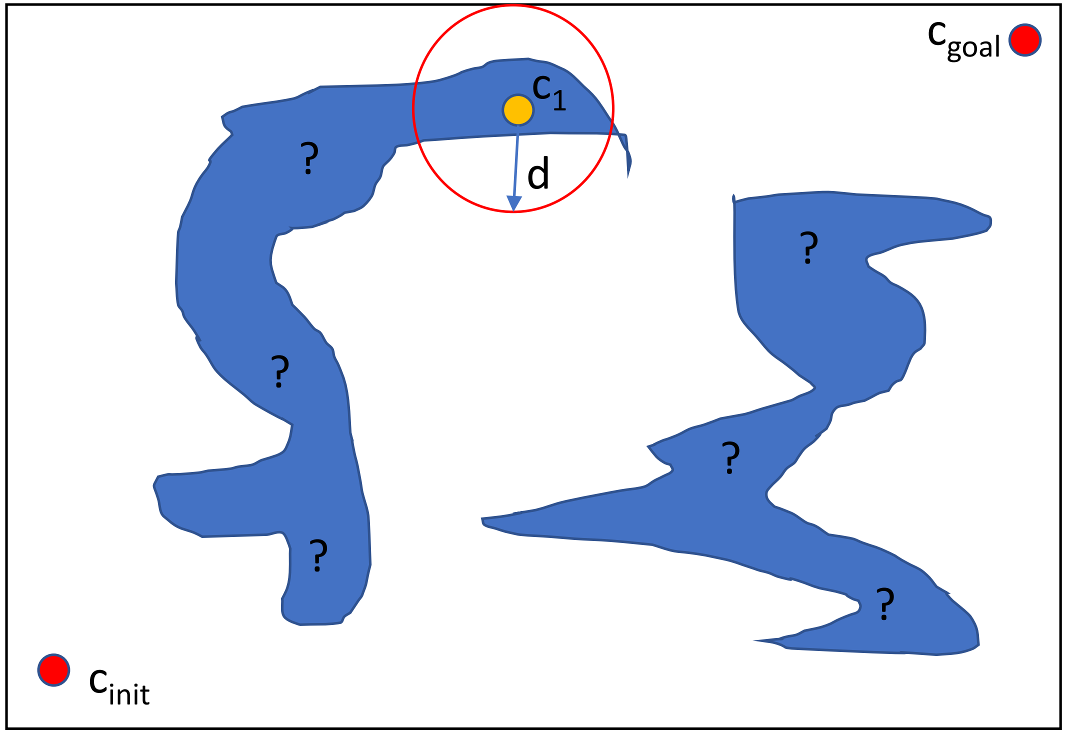
- Now we sample a random distance using a normal distribution with the mean of 0 and a standard deviation of σ.
- σ is a parameter of the sampling approach and has to be defined with experimental testing of the type of motion planning problem.
- You can also define σ based on heuristics that are based on the size of the configuration space. (e.g. σ is 1/1000 of the size of the configuration space)



Sampling Strategy: Gaussian Sampling

Algorithm 2 GAUSSIANSAMPLING

loop
  $c_1 \leftarrow$ a random forbidden configuration (c_t, c_r)
  $d \leftarrow$ a distance chosen according to a normal distribution
  $c_2 \leftarrow$ a configuration (c_t', c_r) with c_t' at distance d from c_t
  if $c_2 \in \mathcal{C}_{\text{free}}$ **then**
    add c_2 to the graph



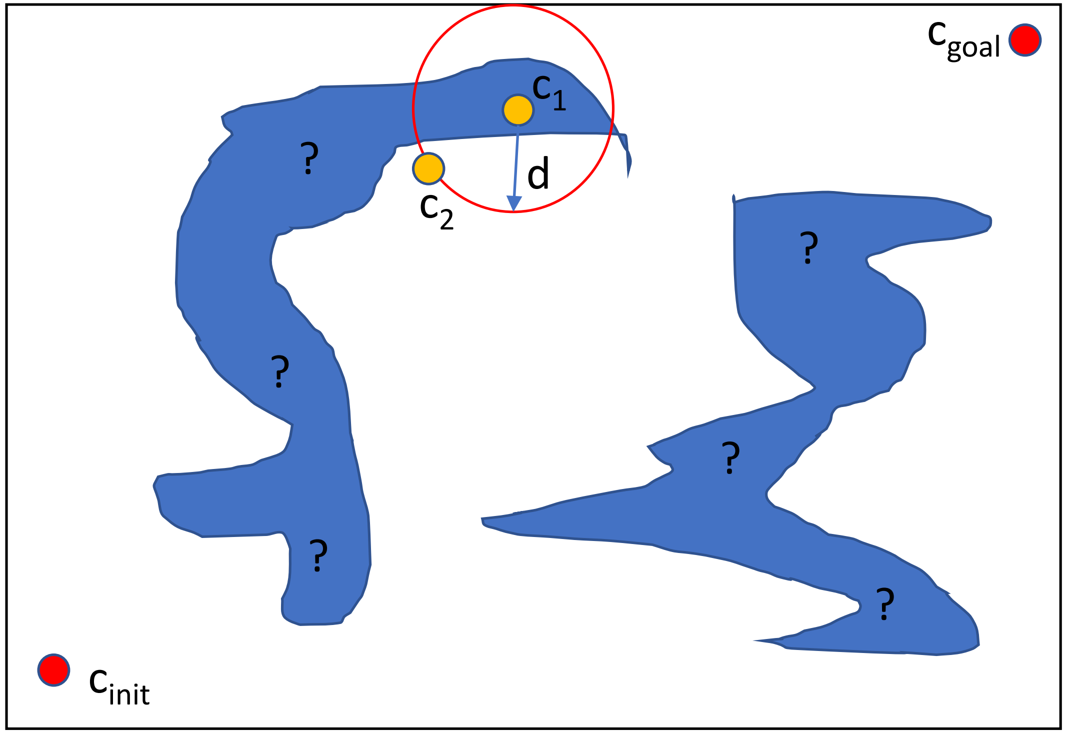
- We have to be careful here with the degrees of freedom. The distance is different for translational DOFs and rotataional DOFs.
- E.g you can define that σ is $1/1000$ of the size of the configuration space for the translational DOFs and for the rotational DOF you define σ as $\sigma = 360°/100 = 3{,}6°$
- Not in the paper the author only samples the distance of the translational DOFs. Later in the literature it was shown that sampling the rotational DOF as well is beneficial

Sampling Strategy: Gaussian Sampling

Algorithm 2 GAUSSIANSAMPLING

```
loop
    c_1 ← a random forbidden configuration (c_t, c_r)
    d ← a distance chosen according to a normal distribution
    c_2 ← a configuration (c'_t, c_r) with c'_t at distance d from c_t
    if c_2 ∈ C_free then
        add c_2 to the graph
```

- Here in this line the algorithm sample a configuration with the sampled distance.

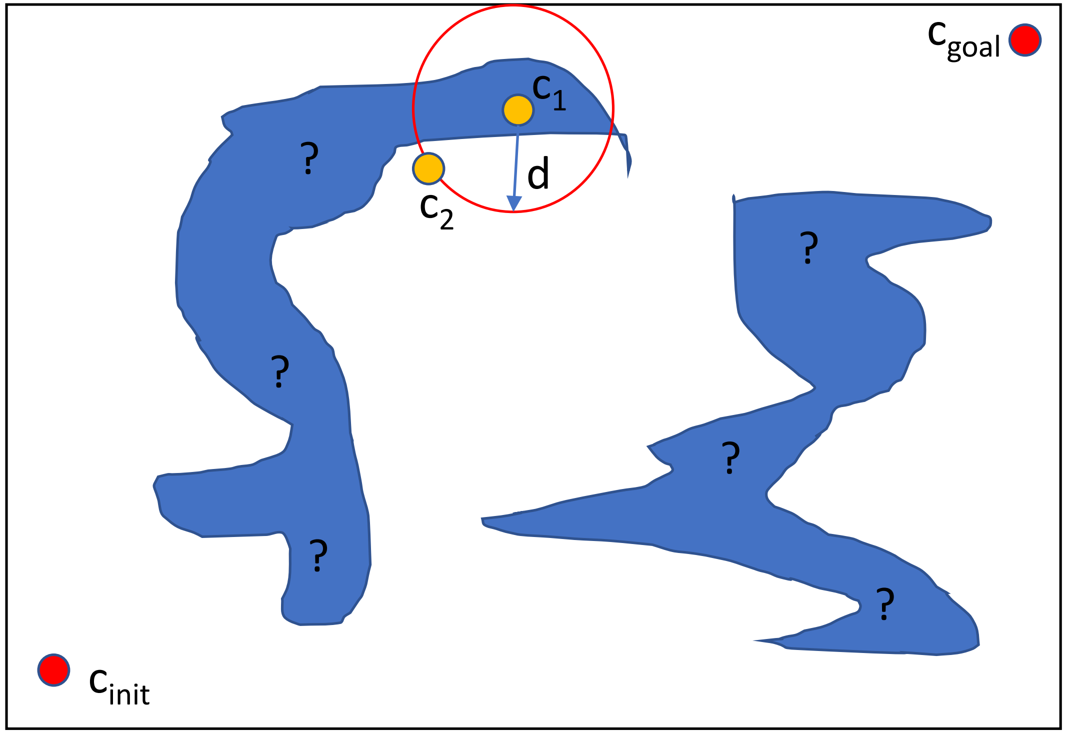


Sampling Strategy: Gaussian Sampling

Algorithm 2 GAUSSIANSAMPLING

loop
  $c_1 \leftarrow$ a random forbidden configuration (c_t, c_r)
  $d \leftarrow$ a distance chosen according to a normal distribution
  $c_2 \leftarrow$ a configuration (c_t', c_r) with c_t' at distance d from c_t
  if $c_2 \in \mathcal{C}_{\text{free}}$ **then**
    add c_2 to the graph

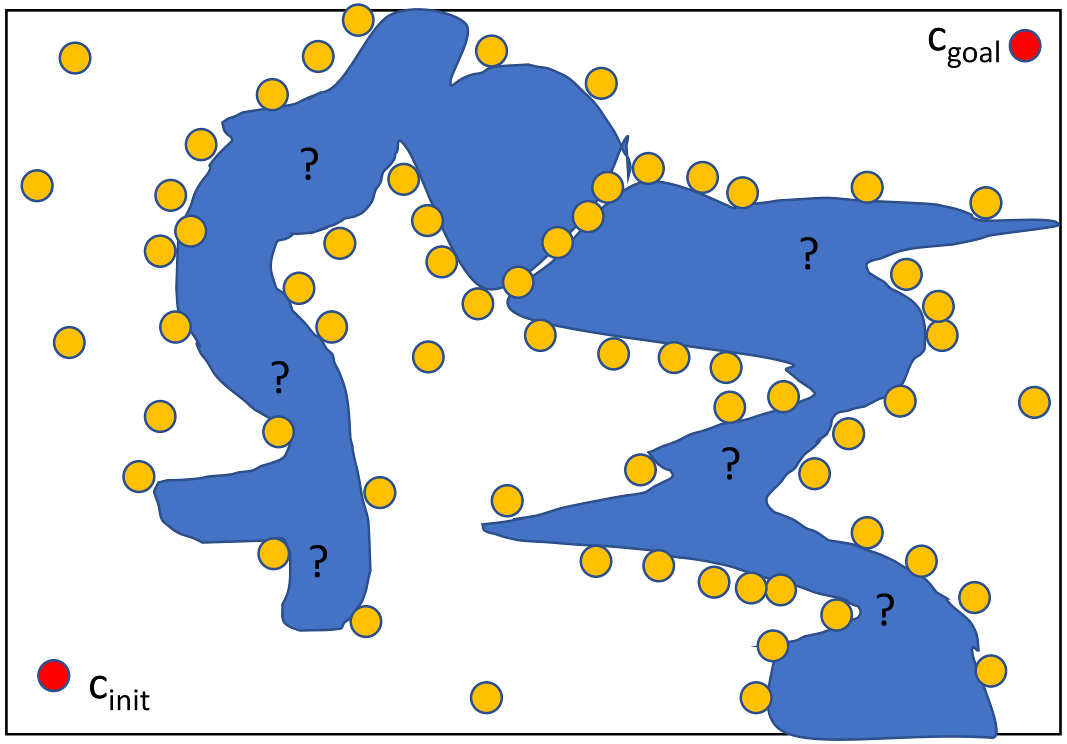
- If this configuration is free of collision (as in this example), the configuration is added to the roadmap.
- If the sampled configuration is not free of collision the loop start from the beginning and new C_1 is sampled.



Sampling Strategy: Gaussian Sampling

How will a sampled roadmap then look like?

- It will sample configurations at the border of C_{obs} with a higher probability.
- The likelyhood of sampling configurations in the narrow passage increased substantially.
- Still there is a probabilty (if sampled distance is high) that also configurations in the wide free space are samples.



Sampling Strategy: Gaussian Sampling

What is the problem with the approach?

- In order to get a configurations of C_{obs} we need to do a collision check.
- Collision detection take substantial running time.
- Many configurations are discarded and the runtime is „wasted“.
- Therefore there is are similar variants that improve this…

Algorithm 2 GAUSSIANSAMPLING

loop
    $c_1 \leftarrow$ a random forbidden configuration (c_t, c_r)
    $d \leftarrow$ a distance chosen according to a normal distribution
    $c_2 \leftarrow$ a configuration (c'_t, c_r) with c'_t at distance d from c_t
    if $c_2 \in \mathcal{C}_{\text{free}}$ **then**
        add c_2 to the graph

Sampling Strategy: Gaussian Sampling2

Idea:

- You postpone the collision check.
- You first sample the two configurations in the same way as before and then you check if one of the samples is in C_{obs} and the other in C_{free}.
- This leads to more configuratios that are accepted and less configurations that are discarded.
- Here the algorithms uses the symetric property of the distance.

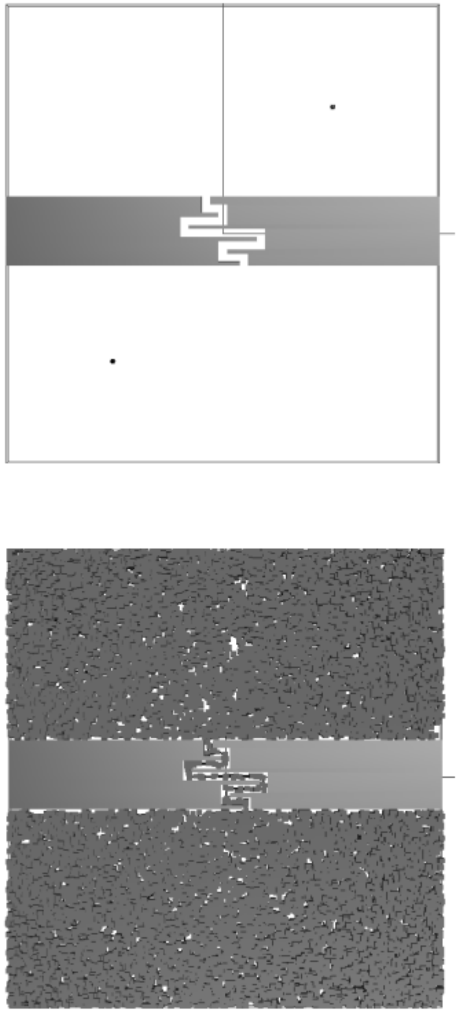
Algorithm 3 GAUSSIANSAMPLING2

loop
  $c_1 \leftarrow$ a random configuration (c_t, c_r)
  $d \leftarrow$ a distance chosen according to a normal distribution
  $c_2 \leftarrow$ a configuration (c'_t, c_r) with c'_t at distance d from c_t
  if $c_1 \in \mathcal{C}_{\text{free}}$ **and** $c_2 \notin \mathcal{C}_{\text{free}}$ **then**
    add c_1 to the graph
  else if $c_2 \in \mathcal{C}_{\text{free}}$ **and** $c_1 \notin \mathcal{C}_{\text{free}}$ **then**
    add c_2 to the graph
  else
    discard both

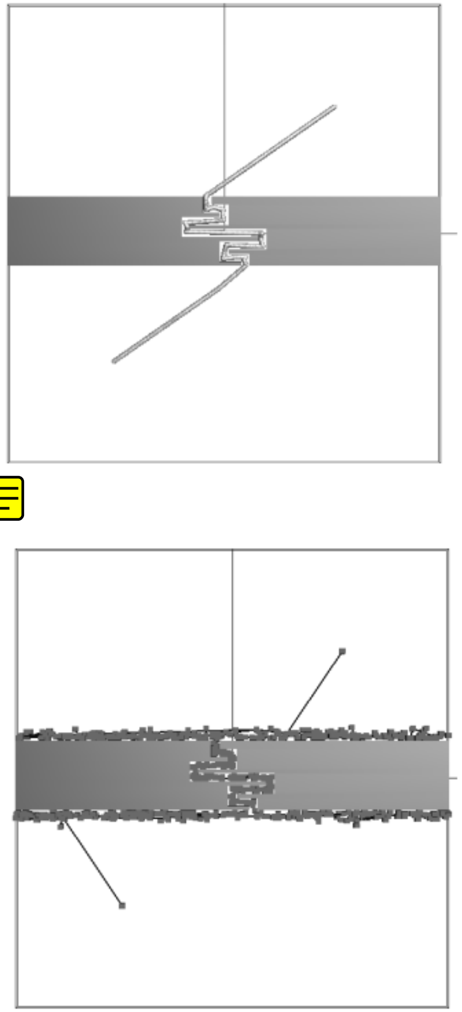
Sampling Strategy: Gaussian Sampling2

Results with PRM:

	Uniform	Gaussian
Total time:	920 s	39 s
Number of nodes:	18,000	460
Total number of samples:	22,000	86,000
Number of local planner calls:	74,000	1,800
Number of collision checks:	5,500,000	100,000



uniform samplig



gaussian sampling

Sampling Strategy: Bridge Test Sampling

Exercise:

- Read the paper on your own and try to understand the Bridge Test approach.

SUBMITTED TO
IEEE International Conference on Robotics & Automation, 2003

The Bridge Test for Sampling Narrow Passages with Probabilistic Roadmap Planners

David Hsu[*] Tingting Jiang[†] John Reif[†] Zheng Sun[†]

[*]*Department of Computer Science*
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599, USA
dyhsu@cs.unc.edu

[†]*Department of Computer Science*
Duke University
Durham, NC 27708, USA
{rucu, reif, sunz}@cs.duke.edu

Abstract

Probabilistic roadmap (PRM) planners have been successful in path planning of robots with many degrees of freedom, but narrow passages in a robot's configuration space create significant difficulty for PRM planners. This paper presents a new sampling strategy in the PRM framework for finding paths through narrow passages. A key ingredient of our strategy is the bridge test, *which boosts the sampling density inside narrow passages. The bridge test relies on simple tests of local geometry and can be implemented efficiently in high-dimensional configuration spaces. The strengths of the bridge test and uniform sampling complement each other naturally. We combine them to obtain the final hybrid sampling strategy. Our planner was tested on point robots and articulated robots in planar workspaces. Preliminary experiments show that the hybrid sampling strategy enables relatively small roadmaps to reliably capture the connectivity of configuration spaces with difficult narrow passages.*

1 Introduction

During the past decade, probabilistic roadmap (PRM) planners [ABD^{+}98, BK00, BOvdS99, HLM99, KŠLO96, NSL99, LK01] have emerged as a powerful framework for path planning of robots with many degrees of freedom (dofs). A classic PRM planner [KŠLO96] samples at random a robot's configuration space to construct a network, called a *roadmap*, that approximates the connectivity of the free space. It then searches the roadmap for a collision-free path between given initial and goal configurations. PRM planners are simple to implement and efficient in practice. As a result, they have found many important applications, including robotics, virtual prototyping, computer animation, and computational biology (see, *e.g.*, [ABG^{+}02, ADS02, HLM99, KL00, LK01, SLvGC01, SLB99]).

Despite the success of PRM planners, path planning for many-dof robots is difficult. Several instances of the problem have been proven to be PSPACE-hard [HJW84, Rei79, SS83]. It is unlikely that random sampling, the key idea be-

Figure 1. An example of samples generated with the bridge test. In this and all later figures, black dots indicate sampled milestones, and shaded regions indicate obstacles.

hind PRM planners, can overcome such difficulty entirely. Indeed, narrow passages in a robot's configuration space pose significant difficulty for PRM planners. Intuitively a narrow passage is a small region whose removal changes the connectivity of the free space. We can also give formal characterizations [BKL^{+}97, HLM99] using the notion of *visibility sets*. To capture the connectivity of the free space accurately, a PRM planner must sample configurations in the narrow passages. This is difficult, because narrow passages have small volumes, and the probability of drawing random samples from small sets is low.

In this paper, we propose a new sampling strategy in the PRM framework in order to find paths through narrow passages efficiently. Key to our new strategy is the *bridge test*, which boosts the sampling density inside narrow passages and thus improves the connectivity of roadmaps. In a bridge test, we check for collision at three sampled configurations: the two endpoints and the midpoint of a short line segment s. We accept the midpoint as a new node in the roadmap graph being constructed, if the two endpoints are in collision and the midpoint is collision-free. We call this a bridge test, because the line segment s resembles a bridge: the endpoints of s, located inside obstacles, act as piers, and the midpoint hovers over the free space. For a configuration inside a narrow passage, building *short* bridges through it is easy, due to the geometry of narrow passages; for a con-

1

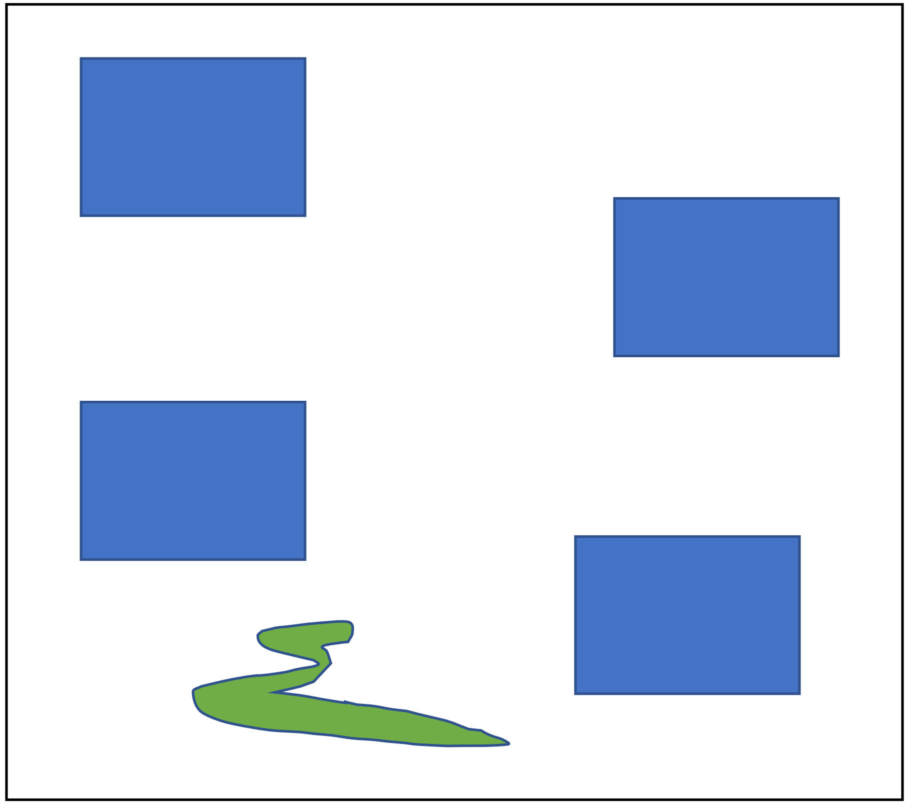
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.377&rep=rep1&type=pdf

Sampling Strategy: Sphere Sampling

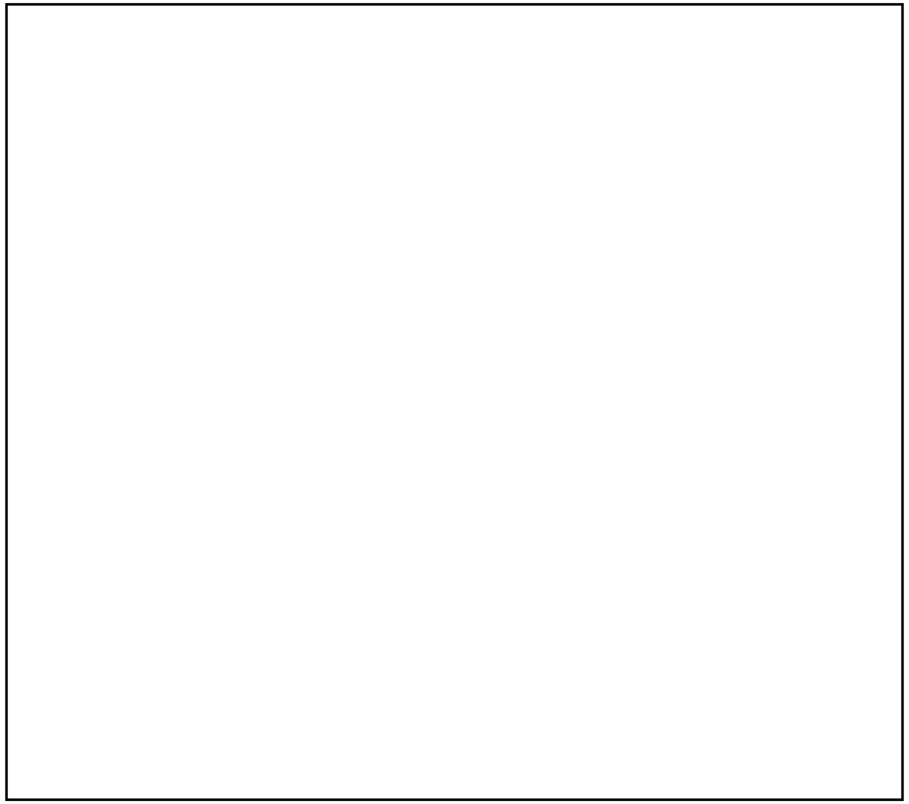
- This algorithm includes a distance computation in addition to a collision detection.
- The idea is to avoid collision detection for configurations that are for sure in free space.
- This algorithm is efficient for scenarios where the collision detection is exceptional expensive. (e.g. large 3D triangle set, force computation, Energy computation…).
- Not as simple as the other strategies. Especially for complex and high DOF configuration spaces.

Sampling Strategy: Sphere Sampling

Workspace

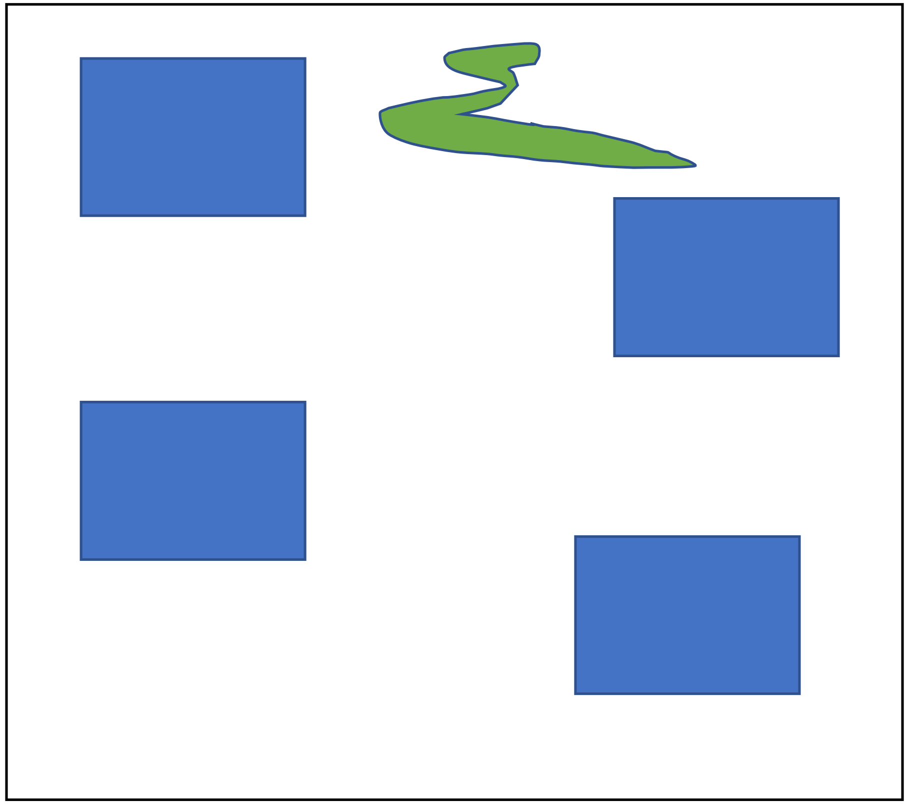


Configuration Space



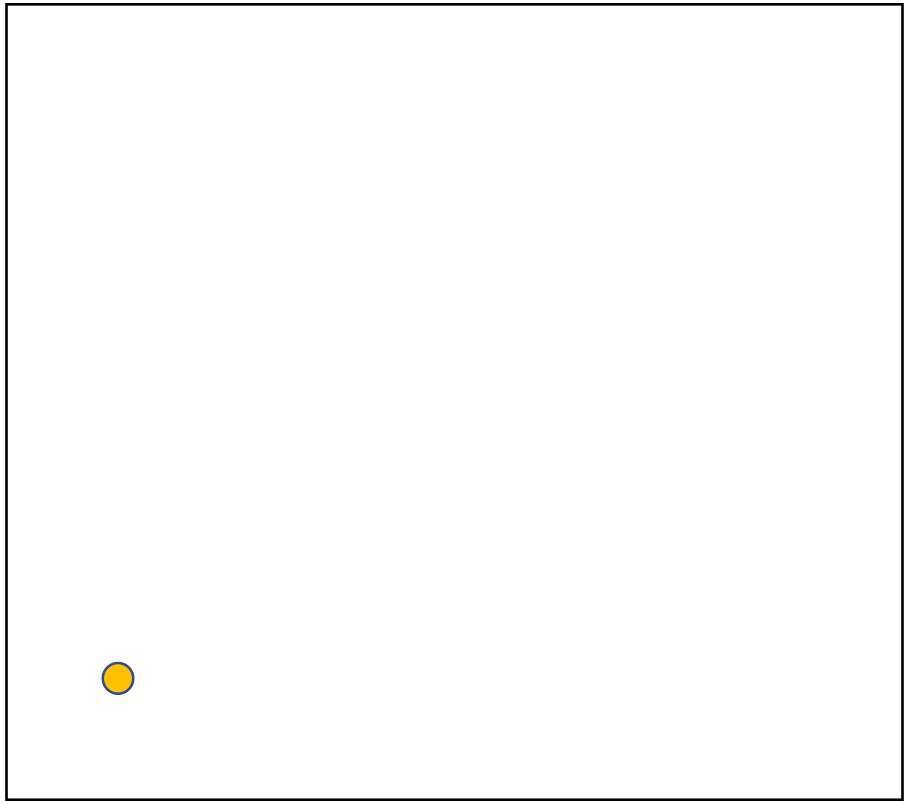
Sampling Strategy: Sphere Sampling

Workspace



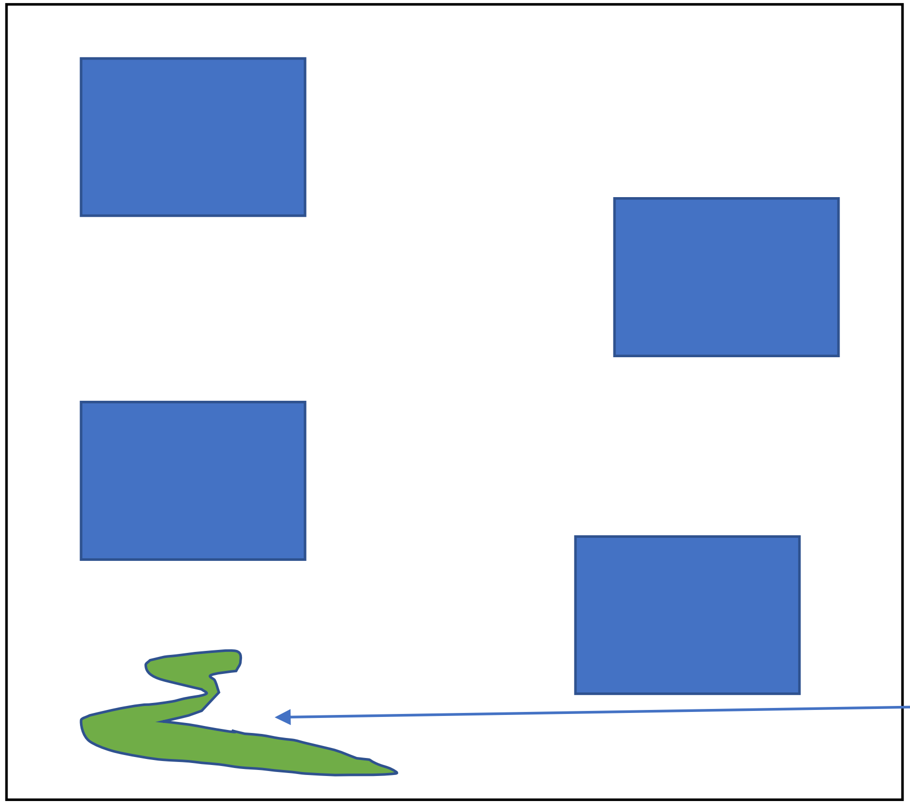
1. Sample a configuration.

Configuration Space



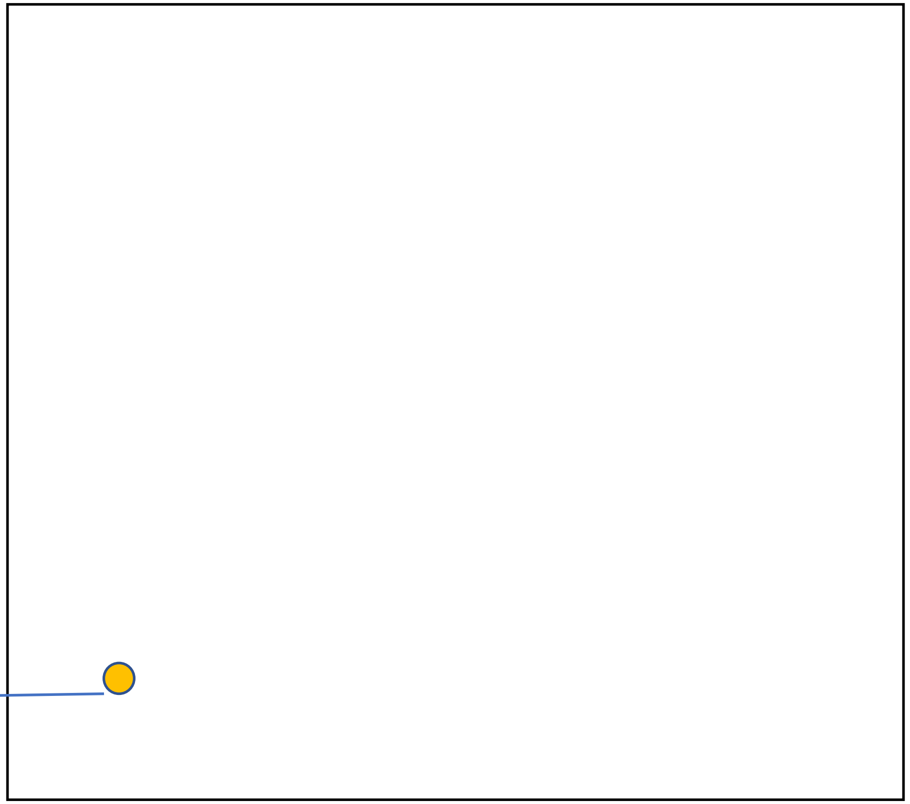
Sampling Strategy: Sphere Sampling

Workspace



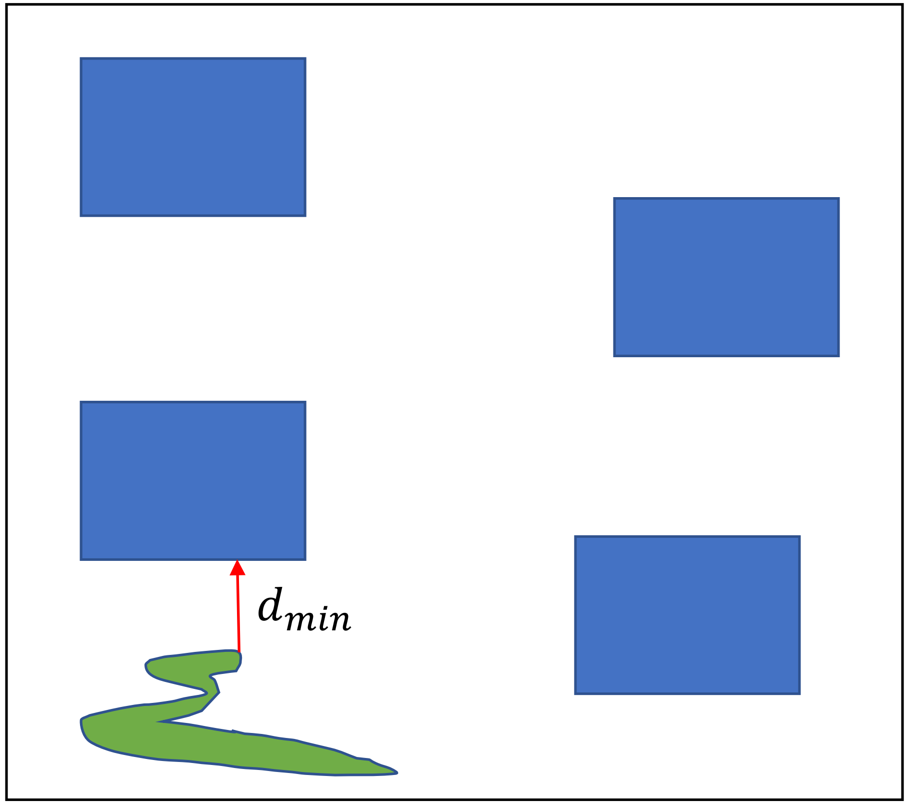
1. Sample a configuration.
2. Map it in the workspace.

Configuration Space



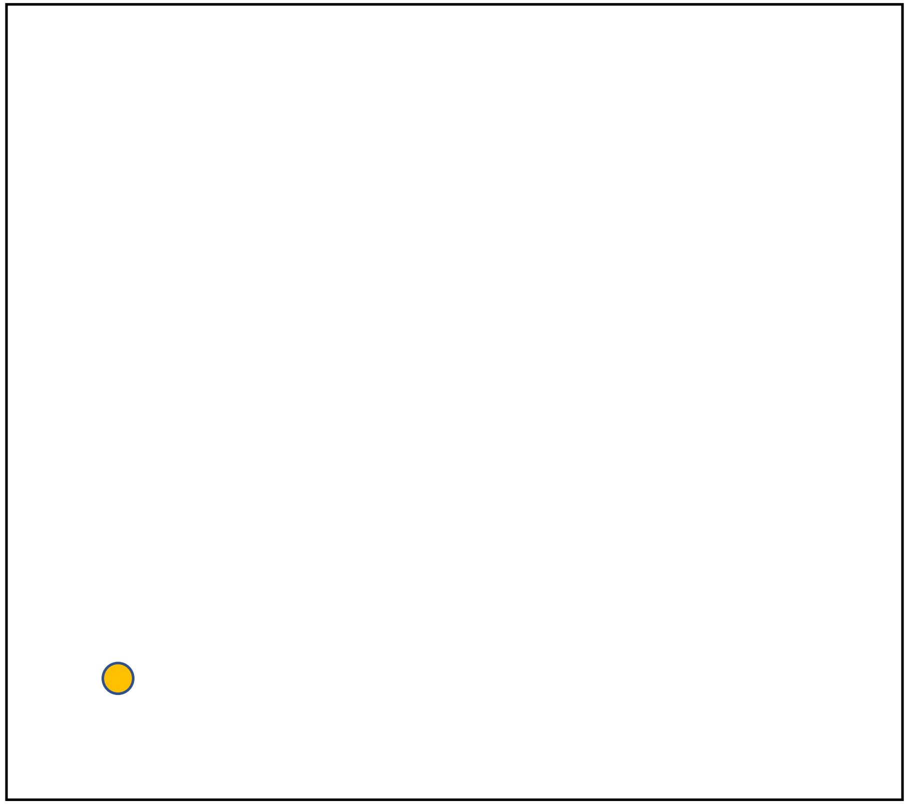
Sampling Strategy: Sphere Sampling

Workspace



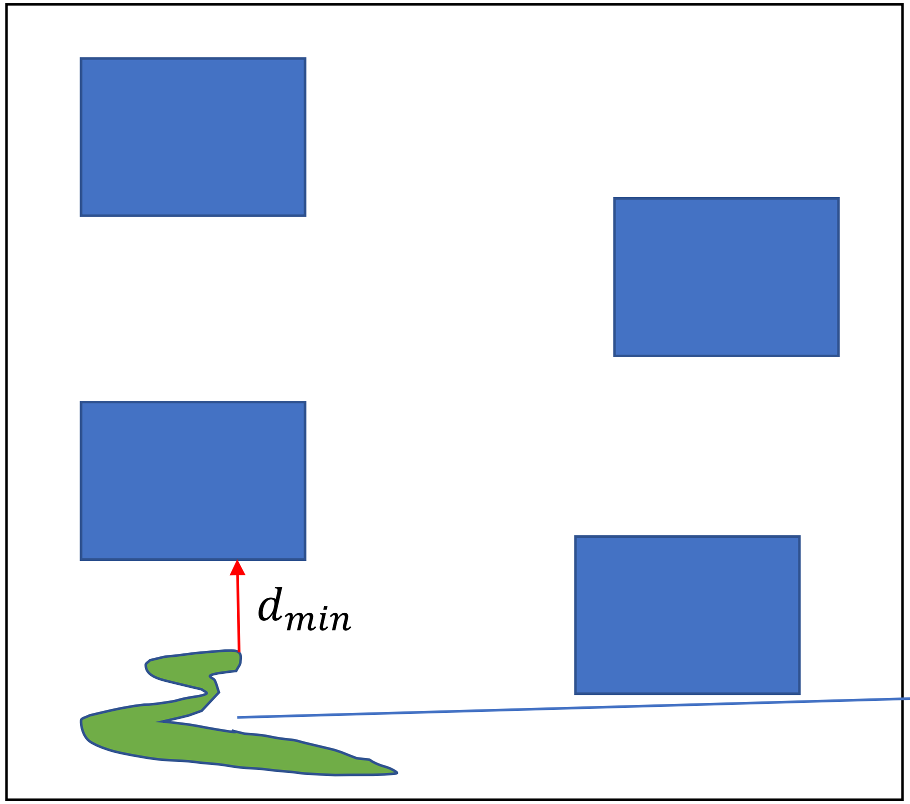
1. Sample a configuration
2. Map it in the workspace
3. Compute the Minimal Distance to the obstacles.

Configuration Space



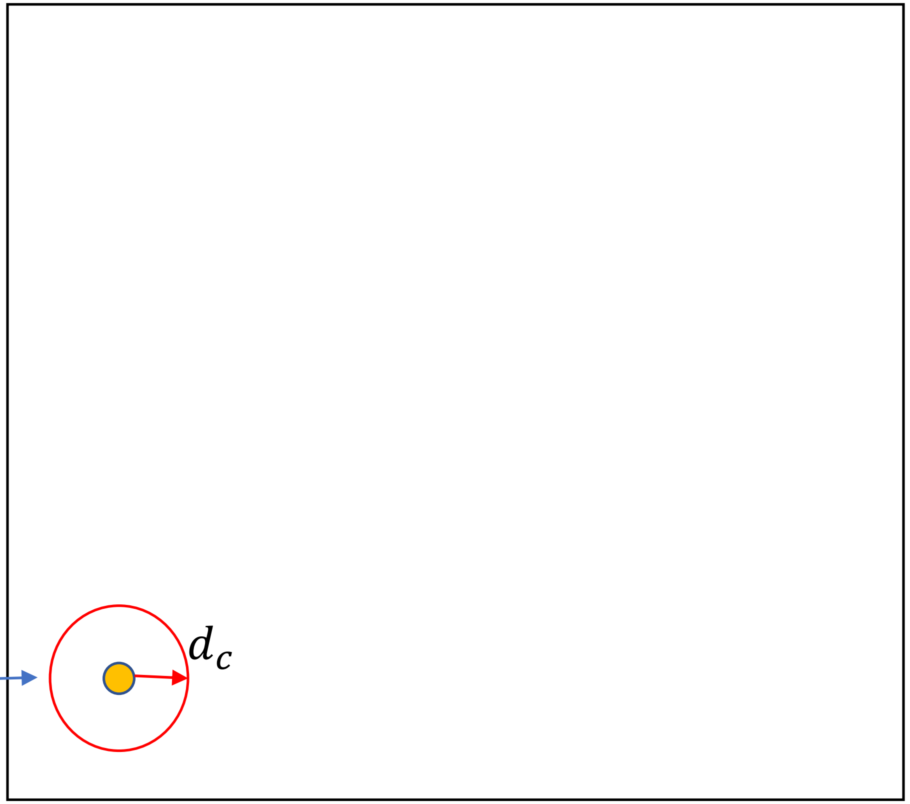
Sampling Strategy: Sphere Sampling

Workspace

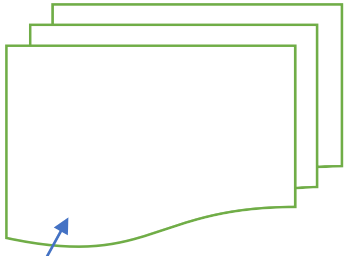


1. Sample a configuration
2. Map it in the workspace
3. Compute the Minimal Distance to the obstacles.
4. With distance you can conclude in the configuration space a sphere that is for sure free of collision.

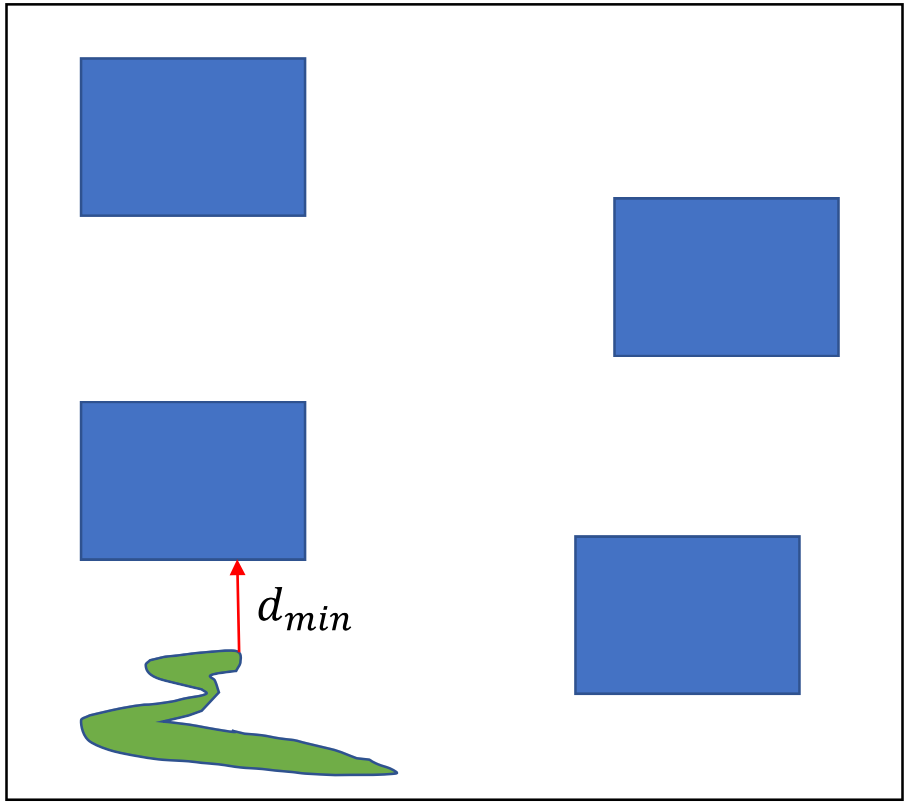
Configuration Space



Sampling Strategy: Sphere Sampling



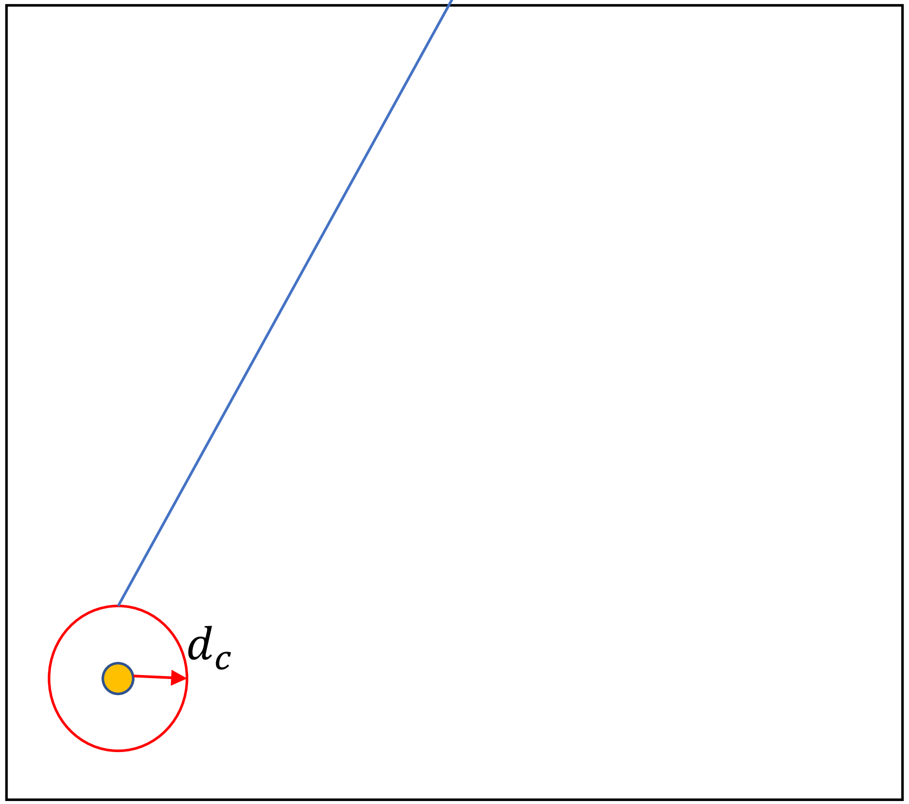
Workspace



Some Notes:

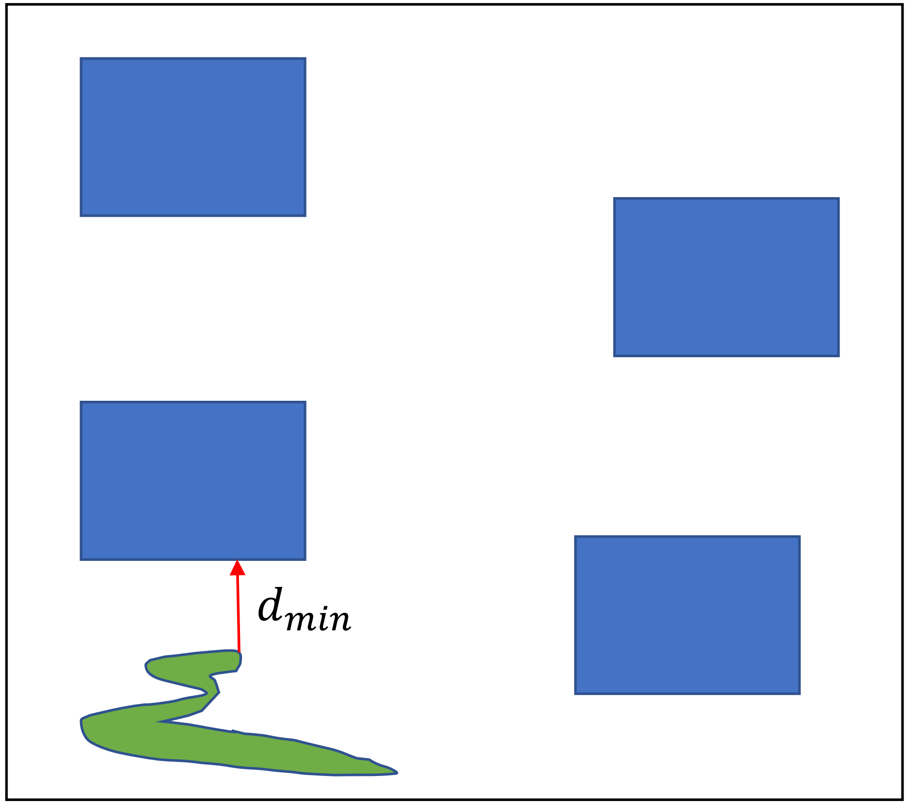
- As the robot can also rotate, the minimal distance d_{min} in the workspace is not the same as d_c in the configurations space.
- Moreover, for most of the application a estimation is used to estimate the size of the sphere in the configuration space.

Configuration Space

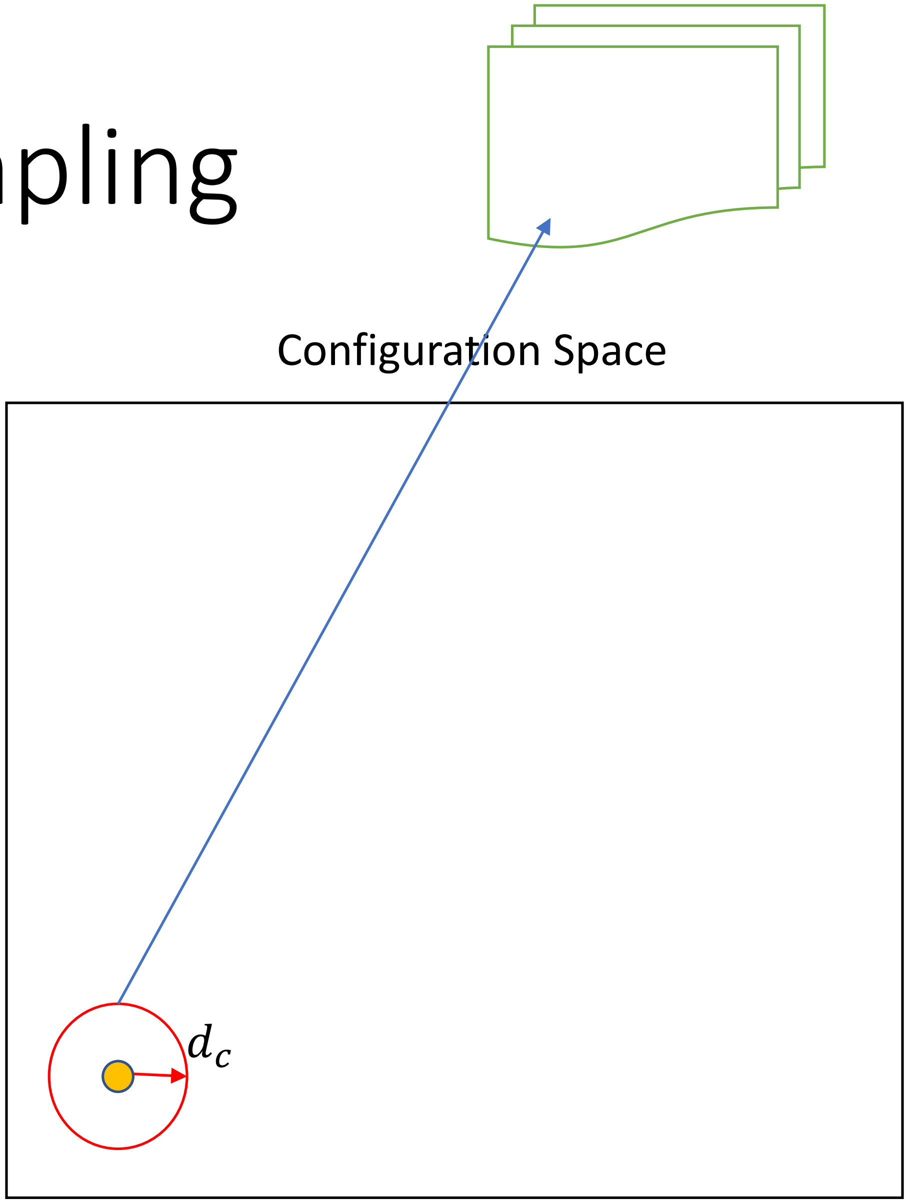


Sampling Strategy: Sphere Sampling

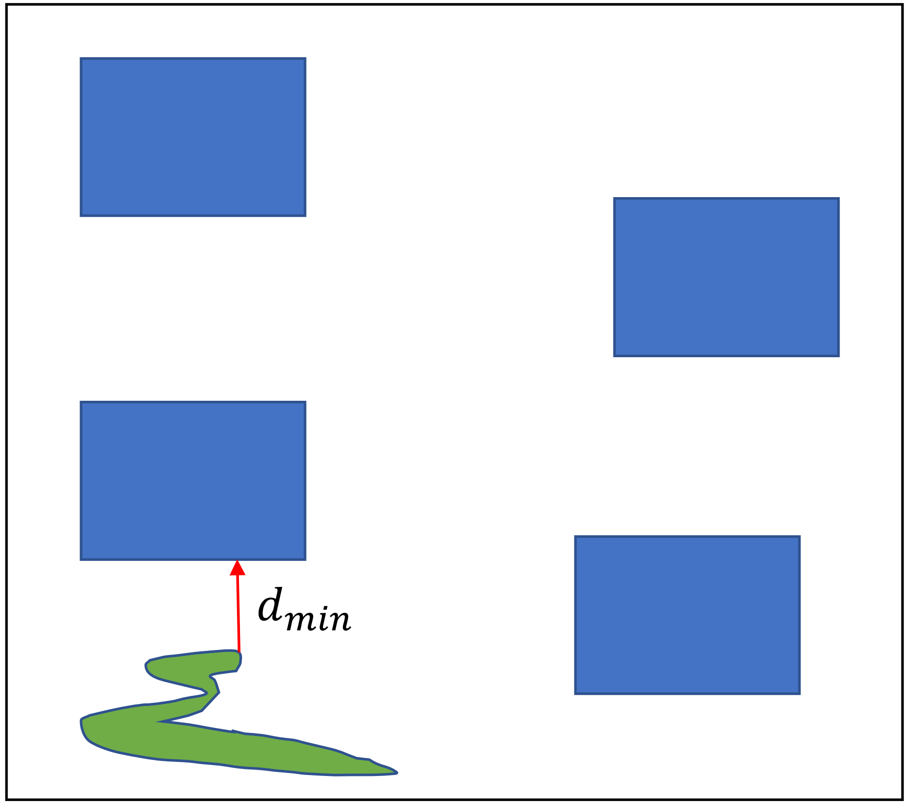
Workspace



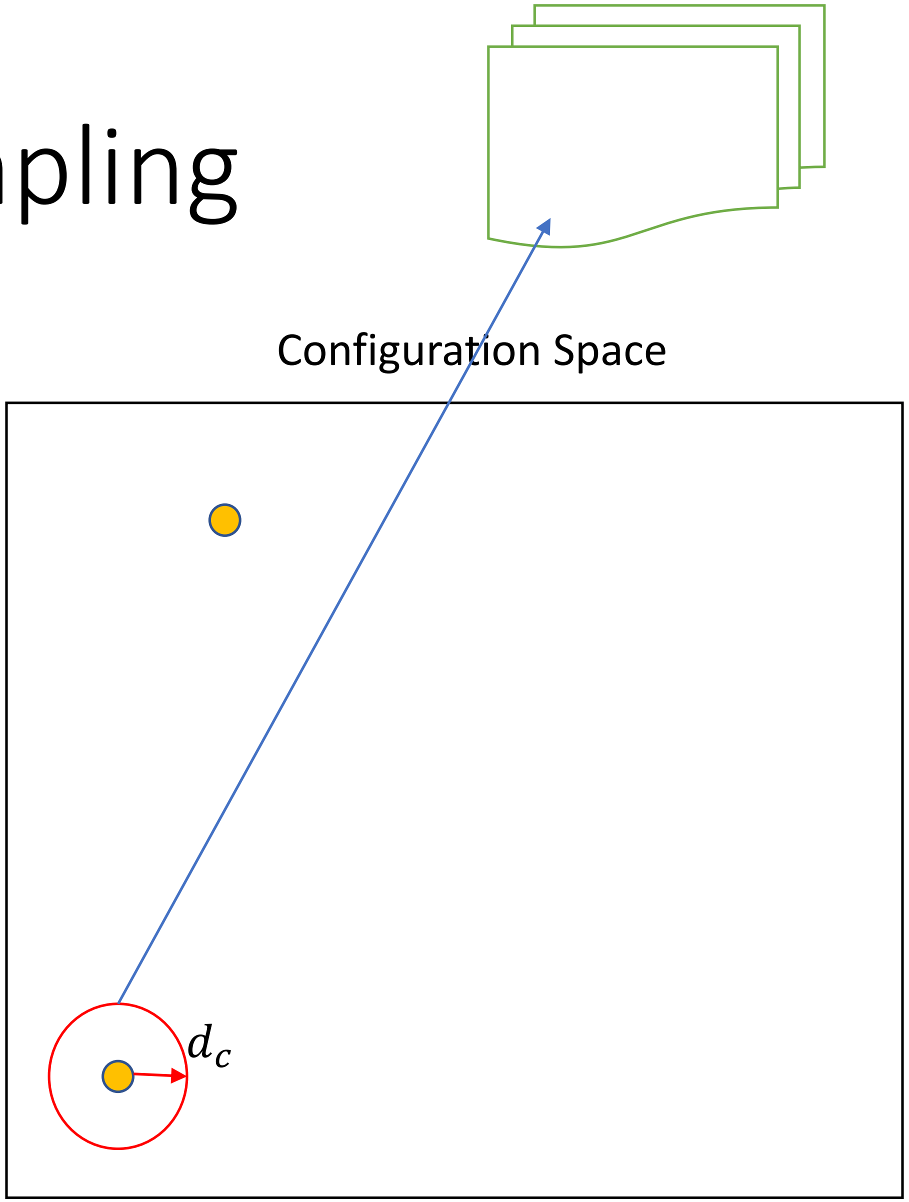
1. Sample a configuration
2. Map it in the workspace
3. Compute the Minimal Distance to the obstalces.
4. With distance you can conclude in the configuration space a sphere that is for sure free of collision.
5. You store this sphere in a list data structure.



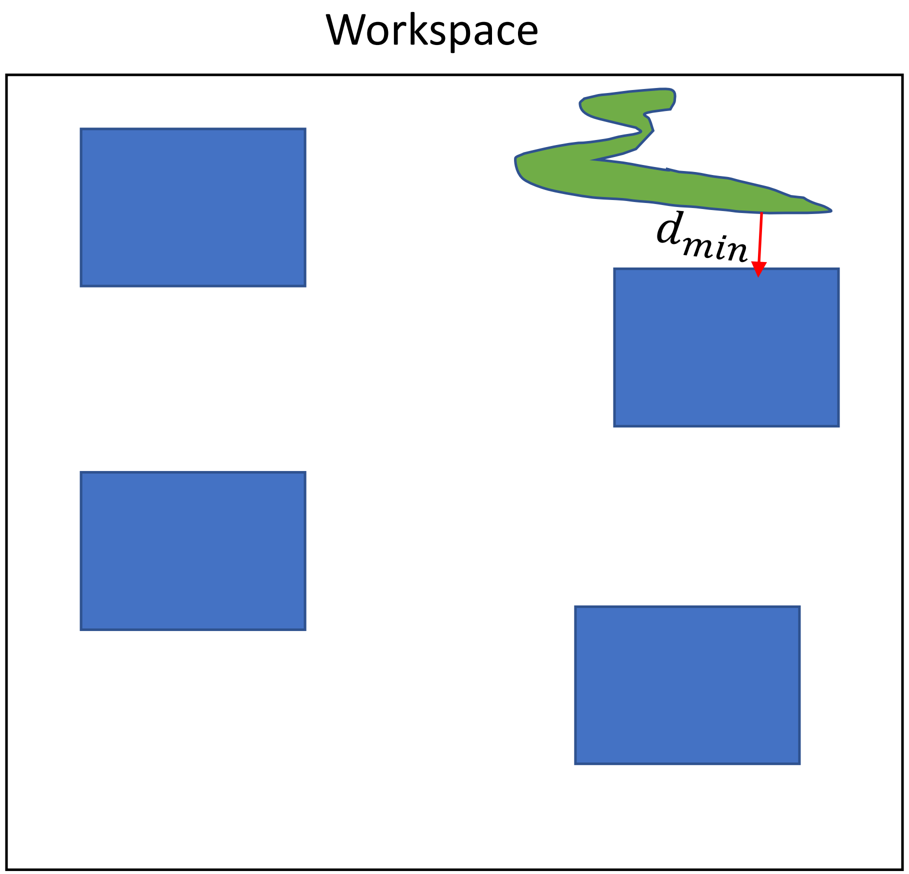
Sampling Strategy: Sphere Sampling



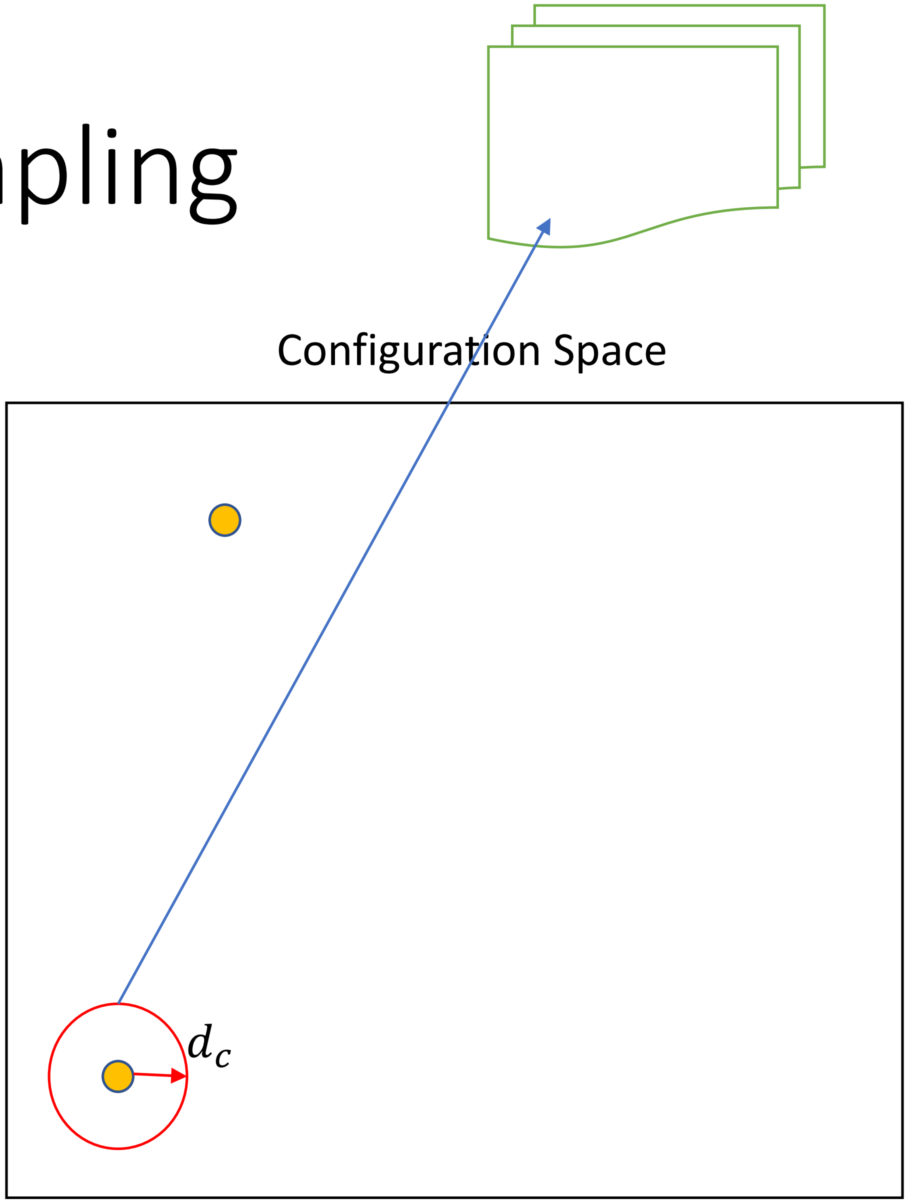
6. You now sample the next configuration.



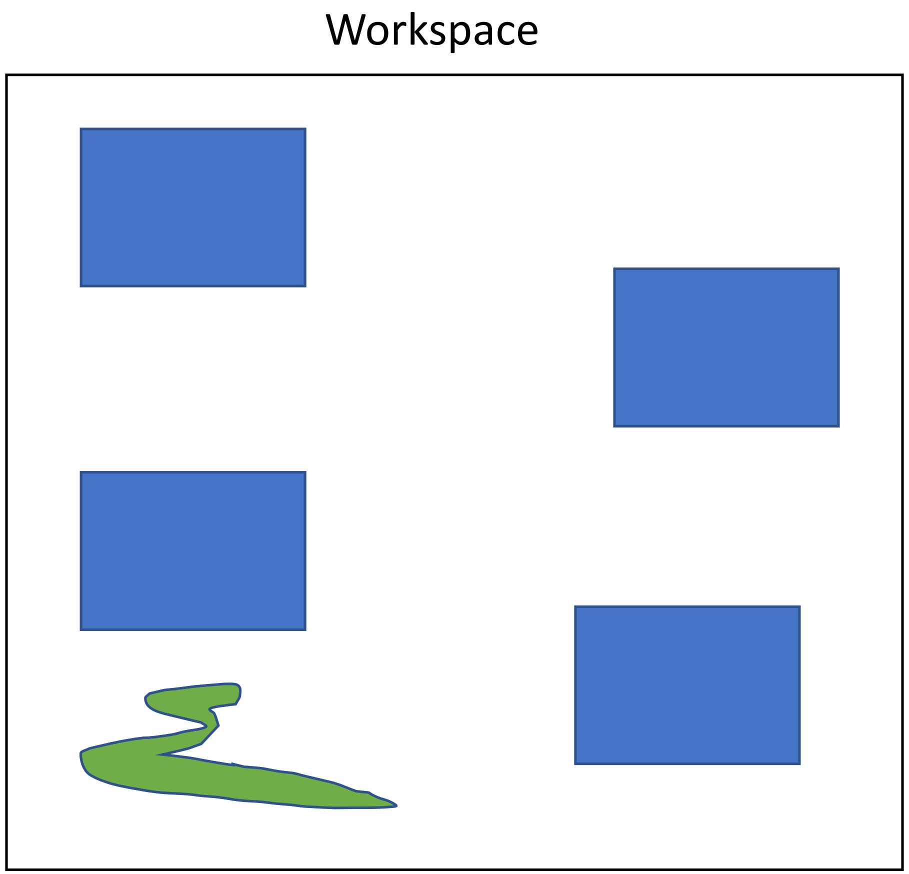
Sampling Strategy: Sphere Sampling



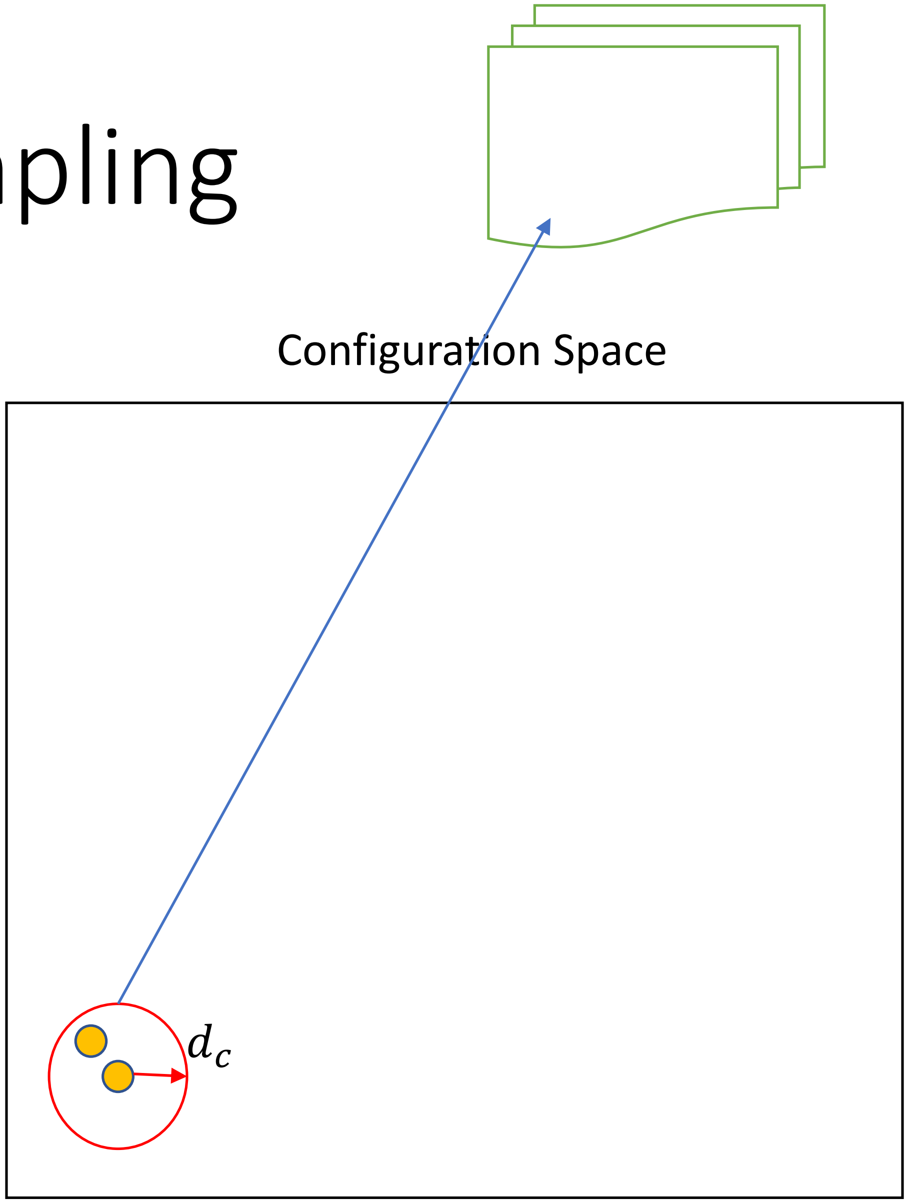
6. You now sample the next configuration.
7. If the configurations is not in one of the spheres you start again, computing the minimal distance.



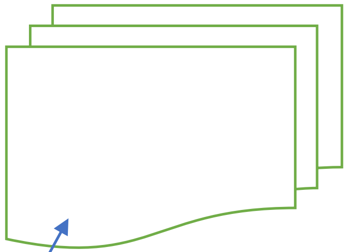
Sampling Strategy: Sphere Sampling



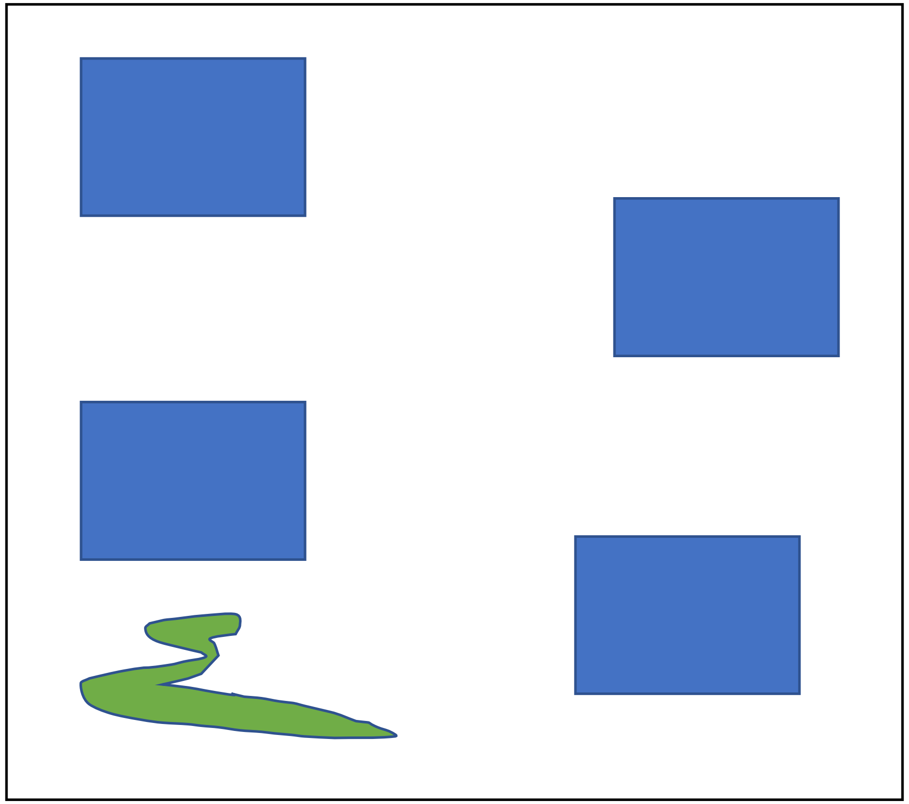
6. You now sample the next configuration.
7. If the configurations is not in one of the spheres you start again, computing the minimal distance.
8. But if the sphere is inside one of the sphere, a collision is not needed → as you already know it is free of collision.



Sampling Strategy: Sphere Sampling

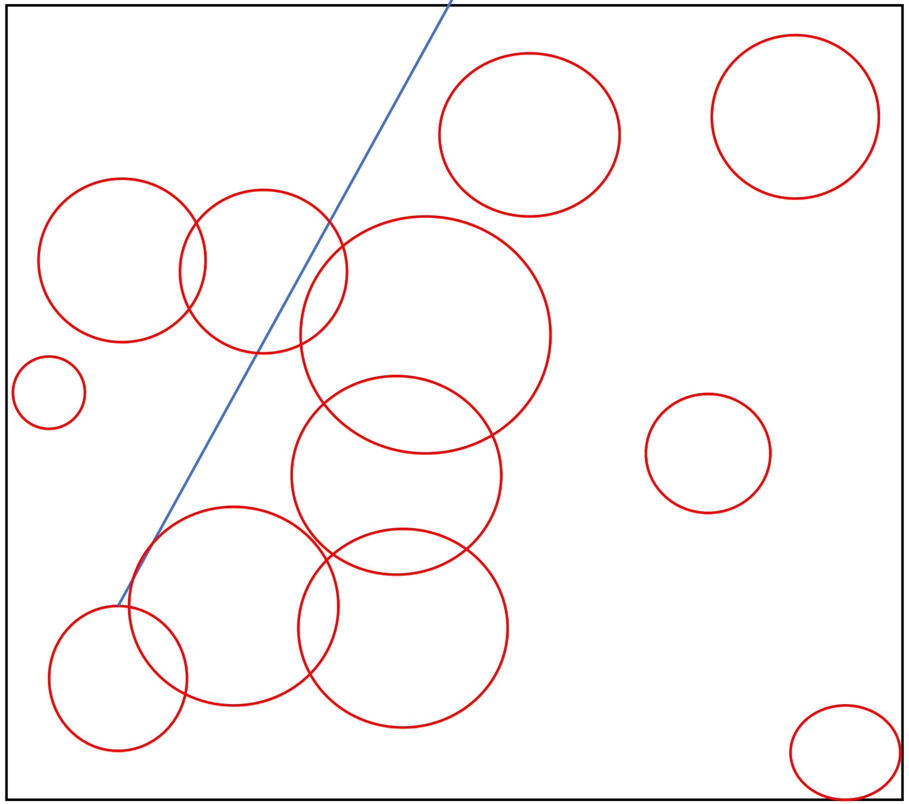


Workspace

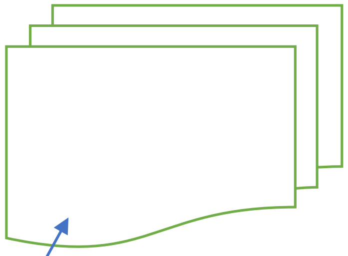


6. You now sample the next configuration.
7. If the configurations is not in one of the spheres you start again, computing the minimal distance.
8. But if the sphere is inside one of the sphere, a collision is not needed → as you already know it is free of collision.
9. You know repeat…

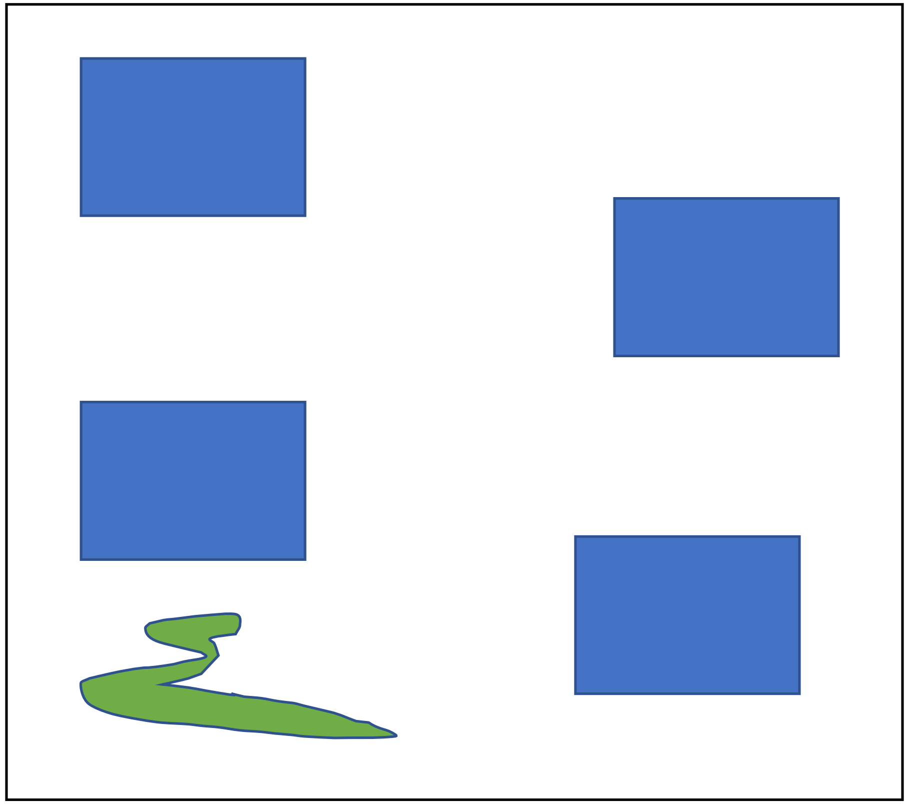
Configuration Space



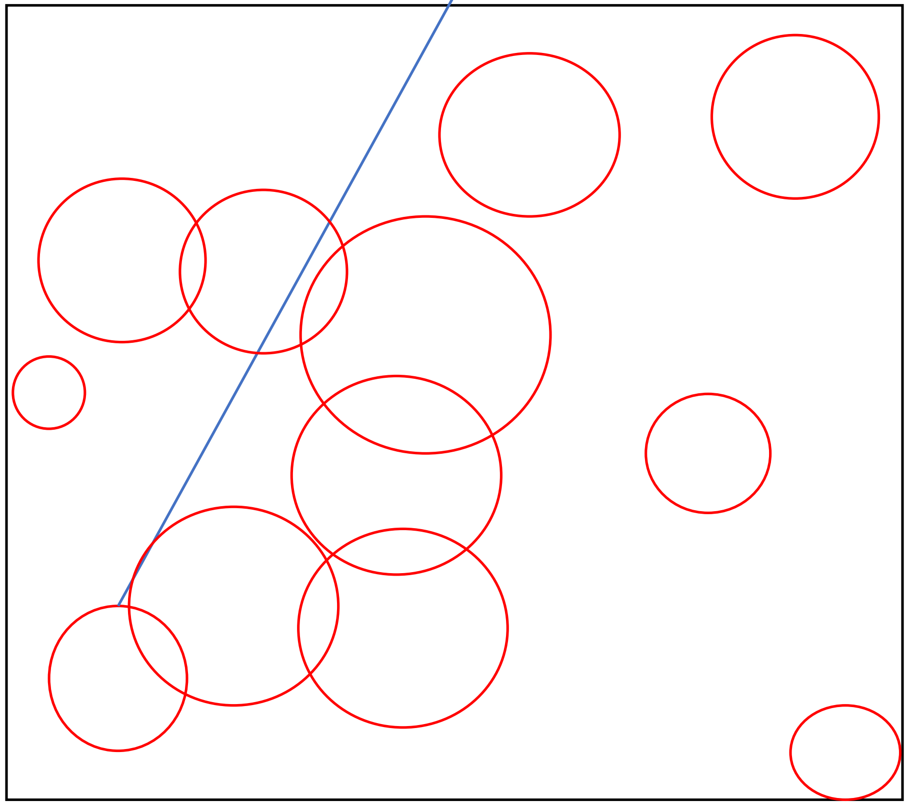
Sampling Strategy: Sphere Sampling



Workspace



Configuration Space

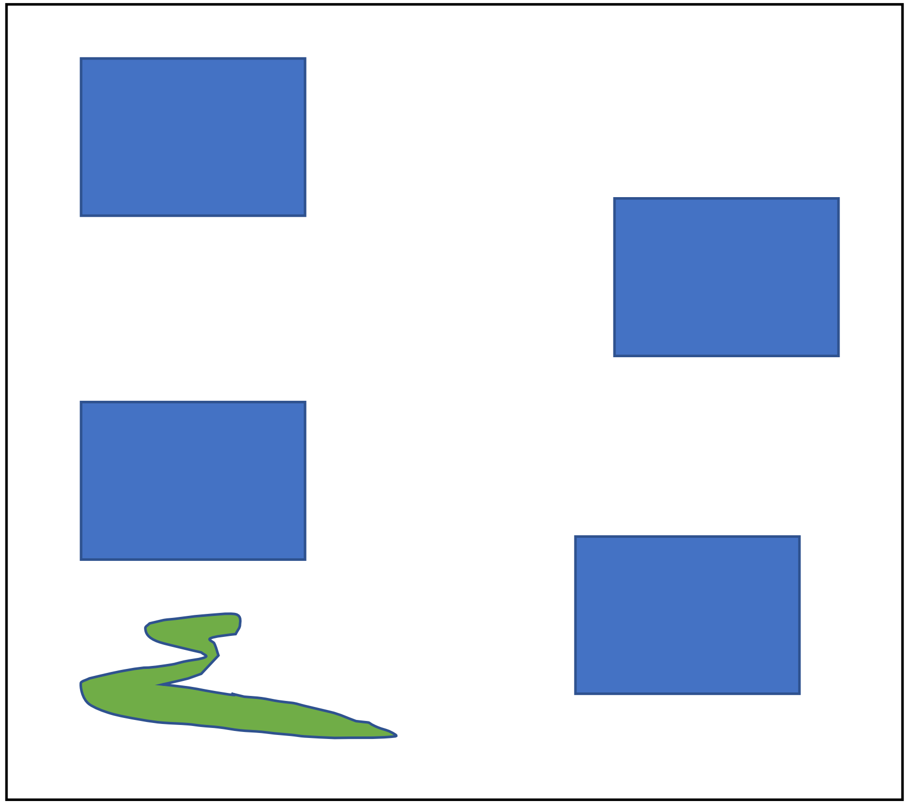


Some Notes:

- The amount of spheres is increasing and you need to make sure that check in the a configuration in the list of spheres is not more expensive than collision detection/distance computation. → A hashmap is needed for the list of spheres.

Sampling Strategy: Sphere Sampling

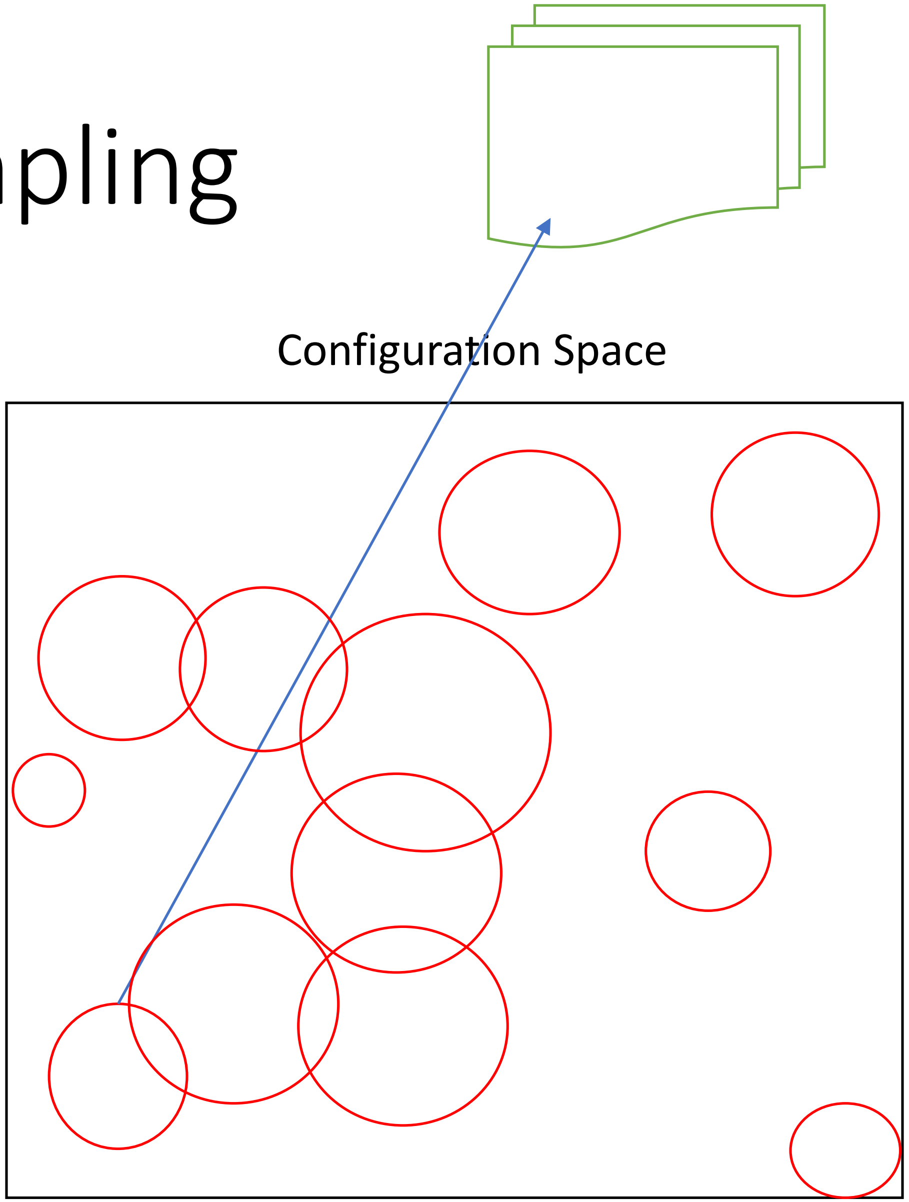
Workspace



Some Notes:

- There are configurations for which the sphere in the configuration space is very small. It is a good idea to ignore those spheres that are below a given threshold.→ experiments depending on the motion planning problems.

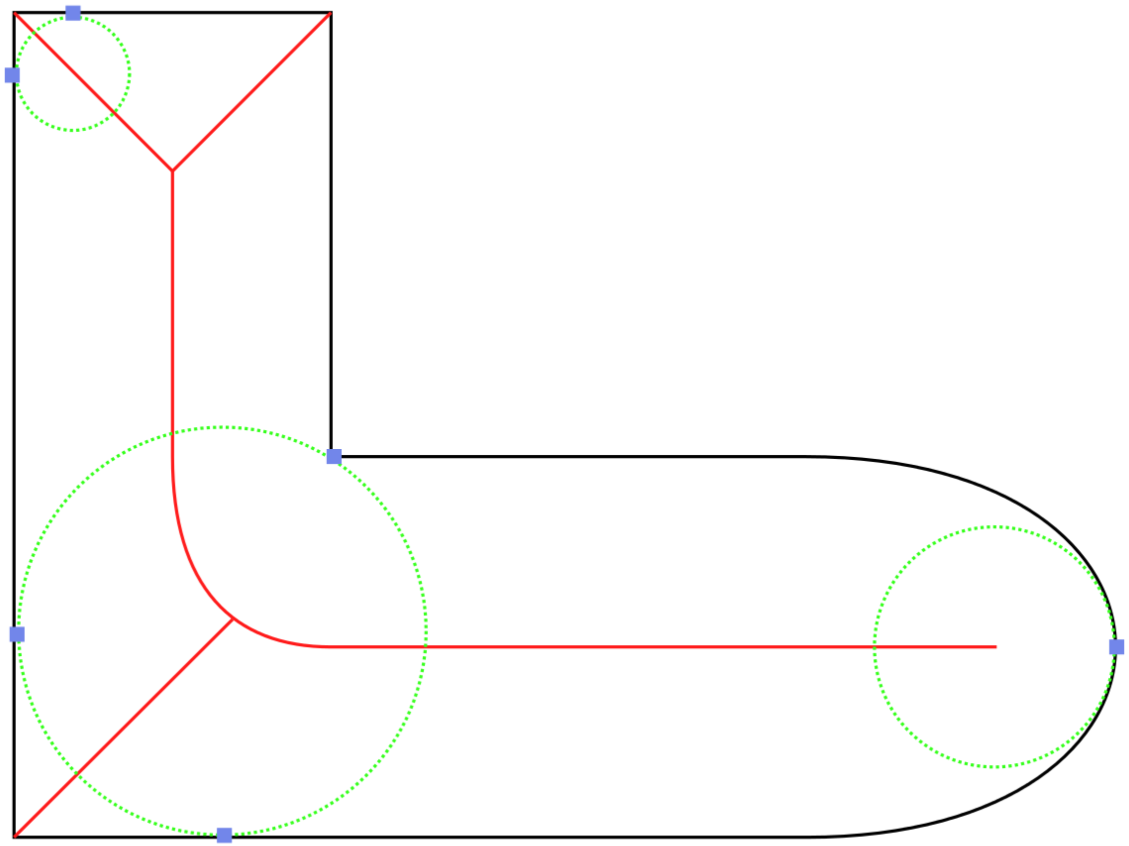
Configuration Space



Sampling Strategy: Medial Axis Sampling

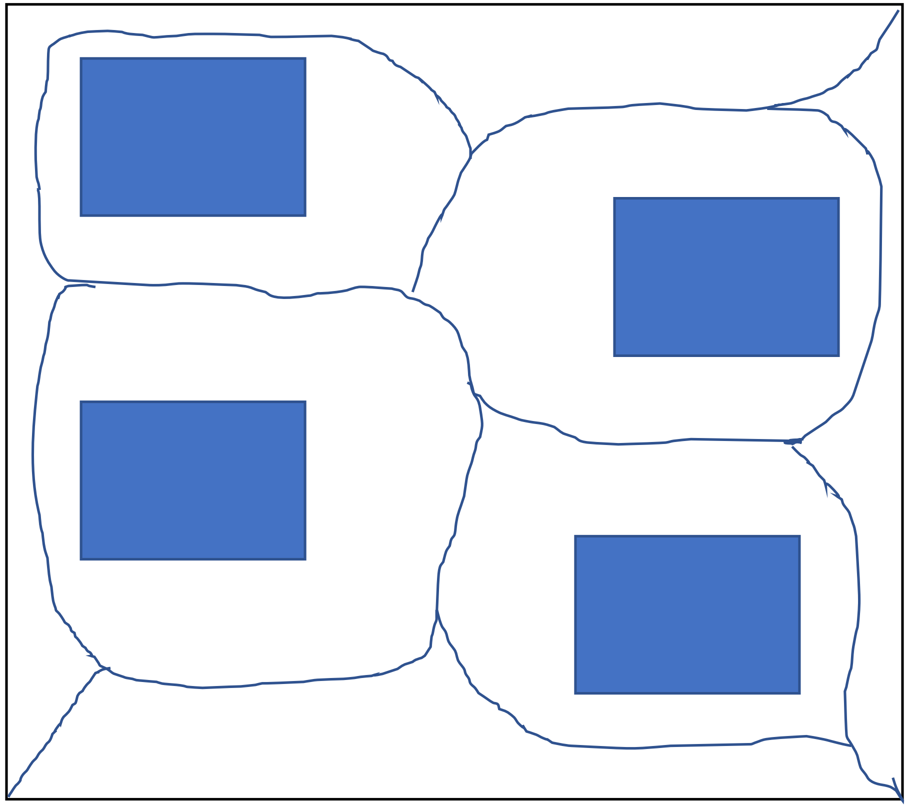
What is the medial axis?

The axis that is defined by the center of a maximum circle that is inside the space considered.

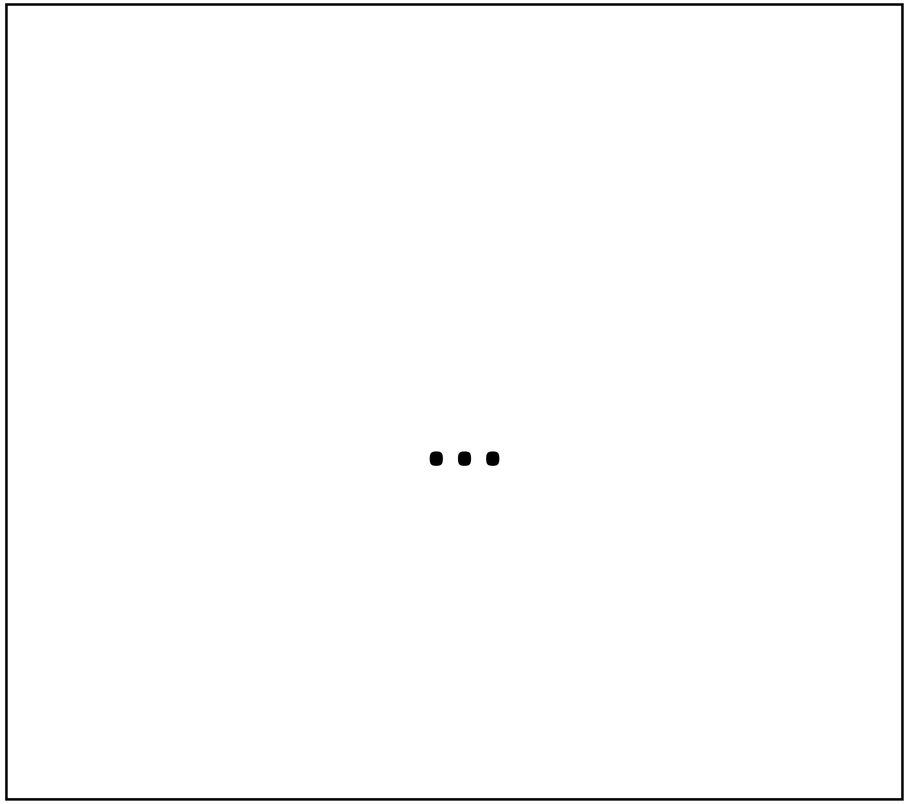


www.wikipedia.org

Sampling Strategy: Medial Axis Sampling



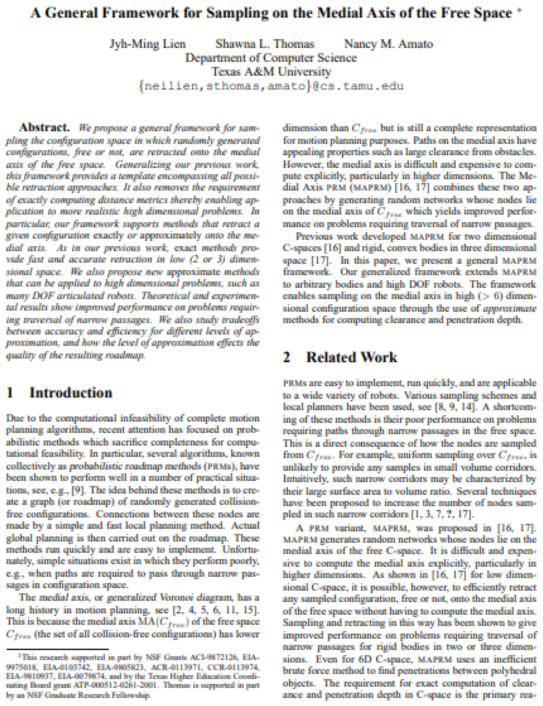
Configuration Space



- The medial axis can be computed in the configuration space as well as in the workspace.
- In the workspace the medial axis already provides a good estimation of usable path for the planning.
- Computing the exact medial axis for general space is complex. Therefore often the approximations are used.

Sampling Strategy: Medial Axis Sampling

- How can we now use the medial axis in sampling?



A General Framework for Sampling on the Medial Axis of the Free Space *

Jyh-Ming Lien Shawna L. Thomas Nancy M. Amato
Department of Computer Science
Texas A&M University
{neilien,sthomas,amato}@cs.tamu.edu

Abstract. *We propose a general framework for sampling the configuration space in which randomly generated configurations, free or not, are retracted onto the medial axis of the free space. Generalizing our previous work, this framework provides a template encompassing all possible retraction approaches. It also removes the requirement of exactly computing distance metrics thereby enabling application to more realistic high dimensional problems. In particular, our framework supports methods that retract a given configuration* exactly *or* approximately *onto the medial axis. As in our previous work, exact methods provide fast and accurate retraction in low (2 or 3) dimensional space. We also propose new* approximate *methods that can be applied to high dimensional problems, such as many DOF articulated robots. Theoretical and experimental results show improved performance on problems requiring traversal of narrow passages. We also study tradeoffs between accuracy and efficiency for different levels of approximation, and how the level of approximation effects the quality of the resulting roadmap.*

1 Introduction

Due to the computational infeasibility of complete motion planning algorithms, recent attention has focused on probabilistic methods which sacrifice completeness for computational feasibility. In particular, several algorithms, known collectively as *probabilistic roadmap methods* (PRMs), have been shown to perform well in a number of practical situations, see, e.g., [9]. The idea behind these methods is to create a graph (or roadmap) of randomly generated collision-free configurations. Connections between these nodes are made by a simple and fast local planning method. Actual global planning is then carried out on the roadmap. These methods run quickly and are easy to implement. Unfortunately, simple situations exist in which they perform poorly, e.g., when paths are required to pass through narrow passages in configuration space.

The *medial axis*, or *generalized Voronoi diagram*, has a long history in motion planning, see [2, 4, 5, 6, 11, 15]. This is because the medial axis $MA(C_{free})$ of the free space C_{free} (the set of all collision-free configurations) has lower dimension than C_{free} but is still a complete representation for motion planning purposes. Paths on the medial axis have appealing properties such as large clearance from obstacles. However, the medial axis is difficult and expensive to compute explicitly, particularly in higher dimensions. The Medial Axis PRM (MAPRM) [16, 17] combines these two approaches by generating random networks whose nodes lie on the medial axis of C_{free} which yields improved performance on problems requiring traversal of narrow passages.

Previous work developed MAPRM for two dimensional C-spaces [16] and rigid, convex bodies in three dimensional space [17]. In this paper, we present a general MAPRM framework. Our generalized framework extends MAPRM to arbitrary bodies and high DOF robots. The framework enables sampling on the medial axis in high (> 6) dimensional configuration space through the use of *approximate* methods for computing clearance and penetration depth.

2 Related Work

PRMs are easy to implement, run quickly, and are applicable to a wide variety of robots. Various sampling schemes and local planners have been used, see [8, 9, 14]. A shortcoming of these methods is their poor performance on problems requiring paths through narrow passages in the free space. This is a direct consequence of how the nodes are sampled from C_{free}. For example, uniform sampling over C_{free}, is unlikely to provide any samples in small volume corridors. Intuitively, such narrow corridors may be characterized by their large surface area to volume ratio. Several techniques have been proposed to increase the number of nodes sampled in such narrow corridors [1, 3, 7, ?, 17].

A PRM variant, MAPRM, was proposed in [16, 17]. MAPRM generates random networks whose nodes lie on the medial axis of the free C-space. It is difficult and expensive to compute the medial axis explicitly, particularly in higher dimensions. As shown in [16, 17] for low dimensional C-space, it is possible, however, to efficiently retract any sampled configuration, free or not, onto the medial axis of the free space without having to compute the medial axis. Sampling and retracting in this way has been shown to give improved performance on problems requiring traversal of narrow passages for rigid bodies in two or three dimensions. Even for 6D C-space, MAPRM uses an inefficient brute force method to find penetrations between polyhedral objects. The requirement for exact computation of clearance and penetration depth in C-space is the primary rea-

[1]This research supported in part by NSF Grants ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACR-0113971, CCR-0113974, EIA-9810937, EIA-0079874, and by the Texas Higher Education Coordinating Board grant ATP-000512-0261-2001. Thomas is supported in part by an NSF Graduate Research Fellowship.

A General Framework for Sampling on the Medial Axis of the Free Space, J-M. Lien, S. L. Thomas, N. M. Amato, Proceedings of the IEEE International Conference on Robotics and Automation, 2003

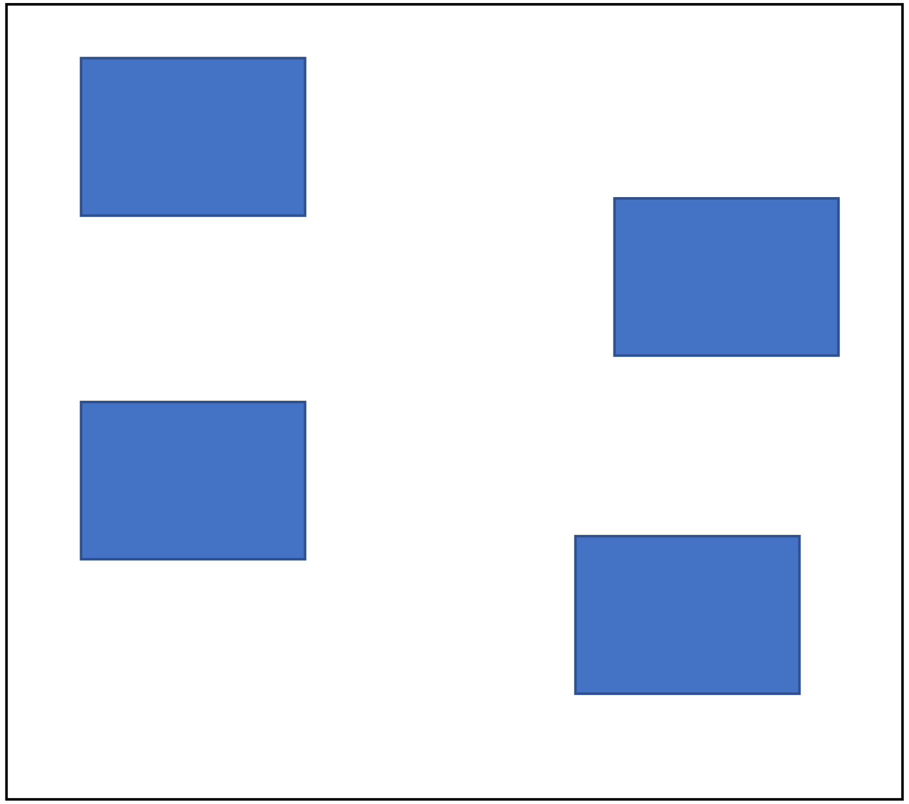
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



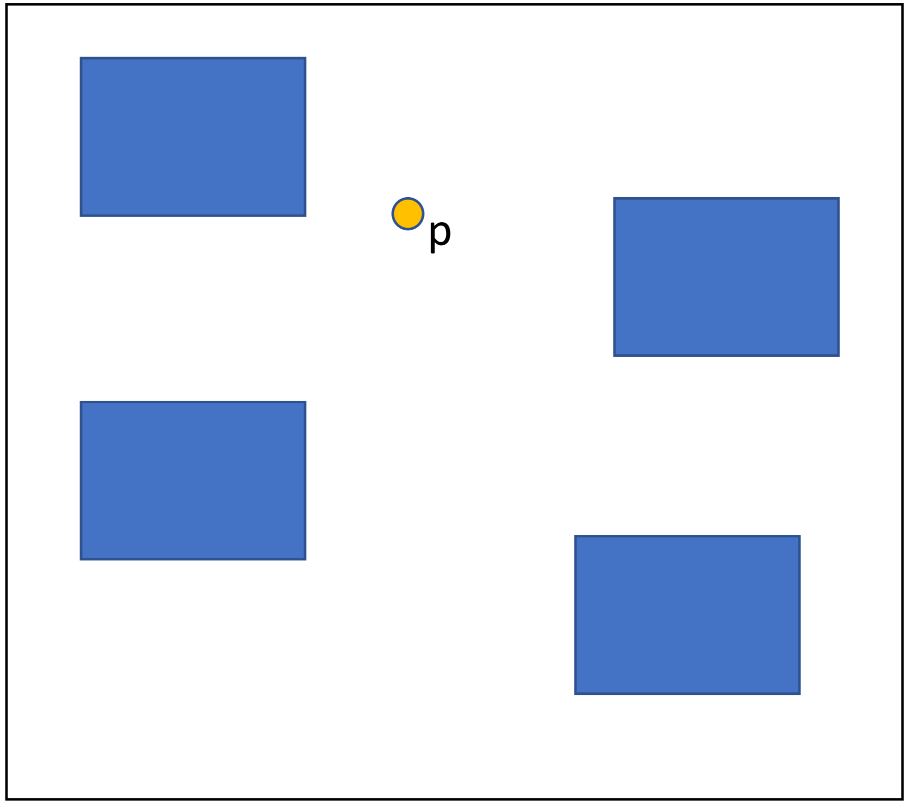
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

1: **repeat**
2: Sample a configuration p from C-space.
3: **if** $p \in C_{free}$ **then**
4: q = **NearestContactCfg_Clearance**(p)
5: Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
6: **else**
7: q = **NearestContactCfg_Penetration**(p)
8: Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
9: **end if**
10: Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
11: **until** N nodes have been generated
12: Build connections between nodes using local planners.



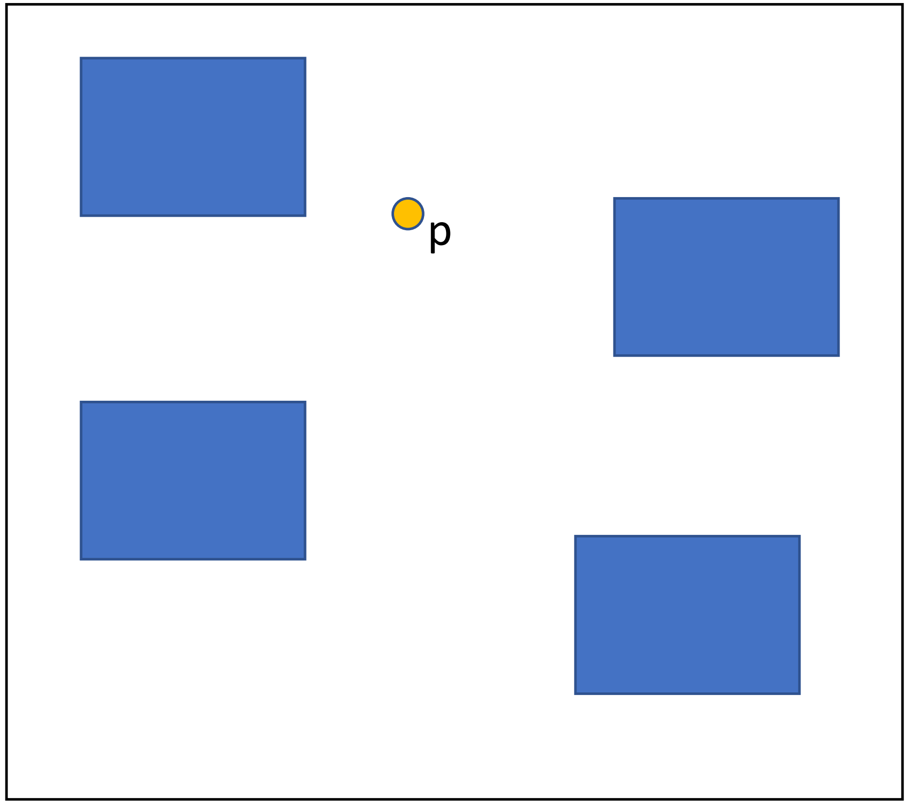
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

1: **repeat**
2: Sample a configuration p from C-space.
3: **if** $p \in C_{free}$ **then**
4: q = **NearestContactCfg_Clearance**(p)
5: Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
6: **else**
7: q = **NearestContactCfg_Penetration**(p)
8: Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
9: **end if**
10: Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
11: **until** N nodes have been generated
12: Build connections between nodes using local planners.



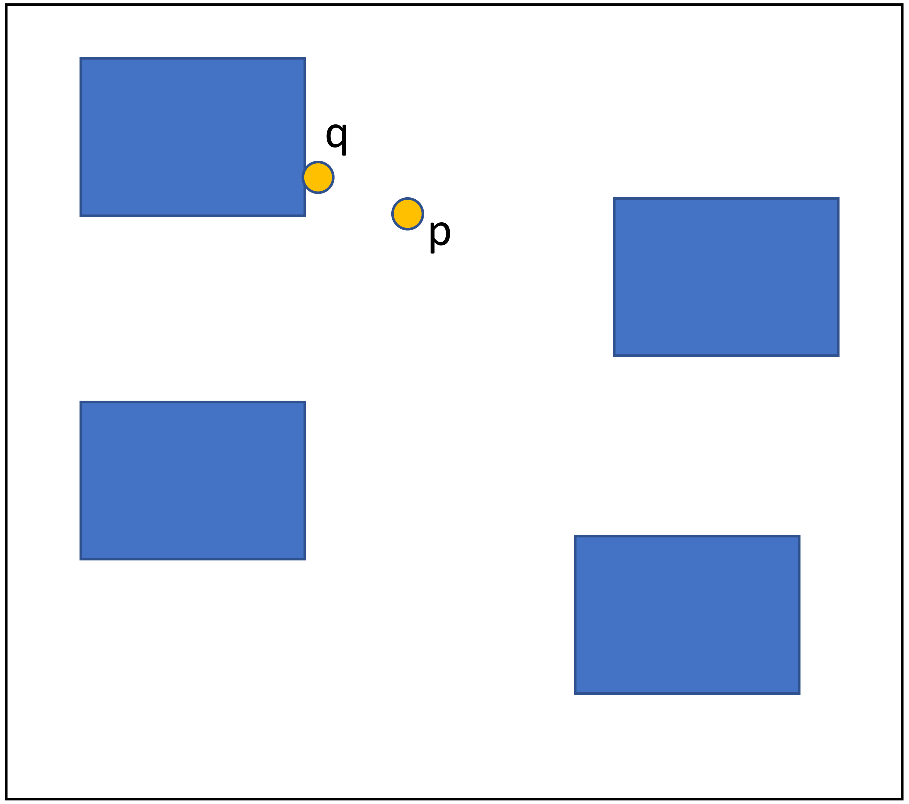
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

```
repeat
    Sample a configuration p from C-space.
    if p ∈ C_free then
        q = NearestContactCfg_Clearance(p)
        Set retraction direction v⃗ = q⃗p⃗ and start point s = p
    else
        q = NearestContactCfg_Penetration(p)
        Set retraction direction v⃗ = p⃗q⃗ and start point s = q
    end if
    Starting at configuration s, move robot in direction v⃗ until
    it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.
```



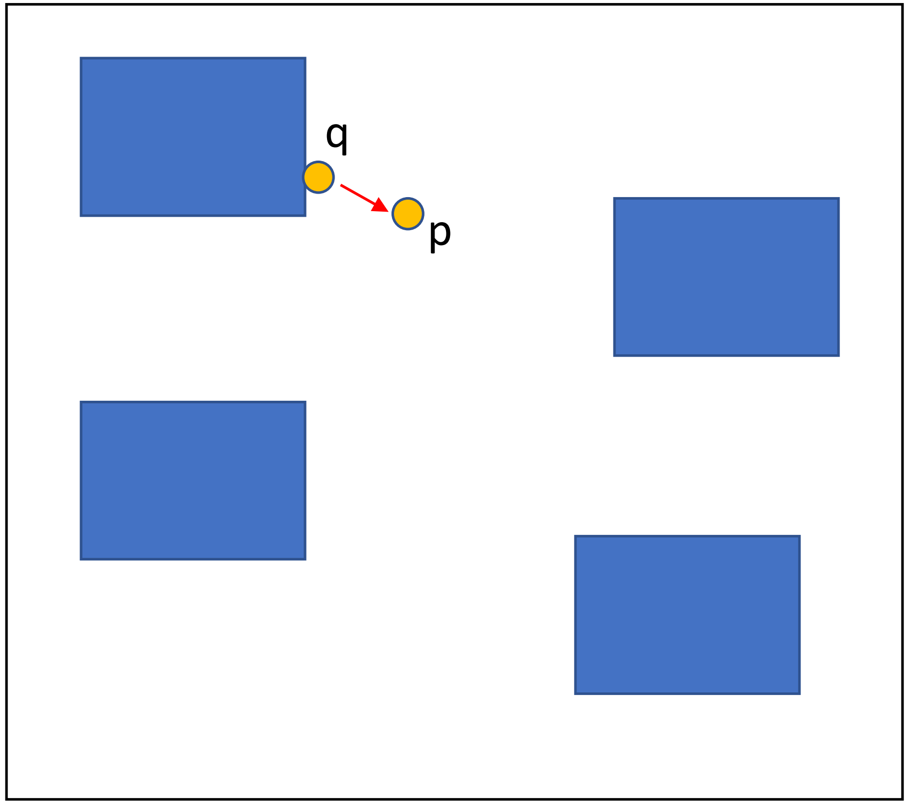
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



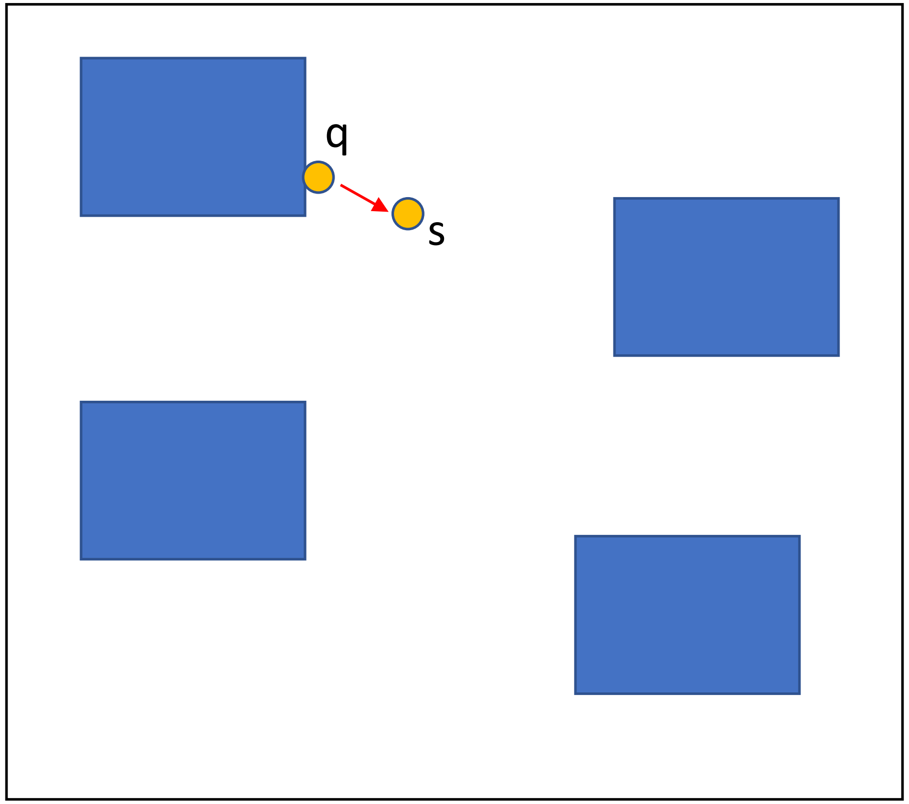
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

```
repeat
    Sample a configuration p from C-space.
    if p ∈ C_free then
        q = NearestContactCfg_Clearance(p)
        Set retraction direction v⃗ = q⃗p and start point s = p
    else
        q = NearestContactCfg_Penetration(p)
        Set retraction direction v⃗ = p⃗q and start point s = q
    end if
    Starting at configuration s, move robot in direction v⃗ until
    it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.
```



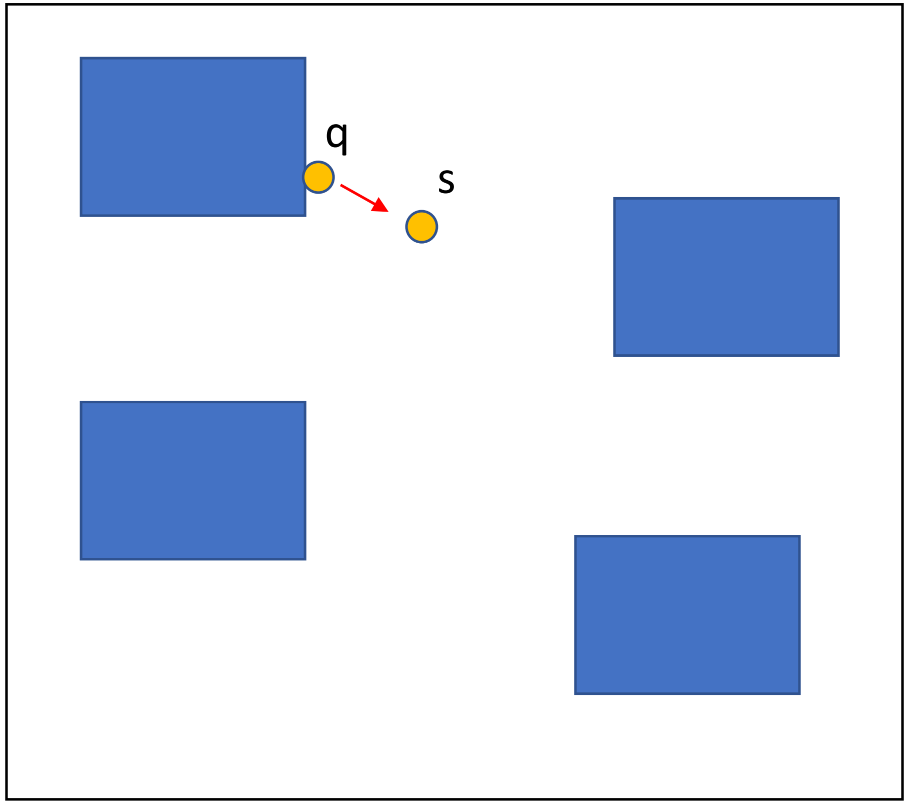
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

```
repeat
    Sample a configuration p from C-space.
    if p ∈ C_free then
        q = NearestContactCfg_Clearance(p)
        Set retraction direction v⃗ = qp⃗ and start point s = p
    else
        q = NearestContactCfg_Penetration(p)
        Set retraction direction v⃗ = pq⃗ and start point s = q
    end if
    Starting at configuration s, move robot in direction v⃗ until
    it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.
```



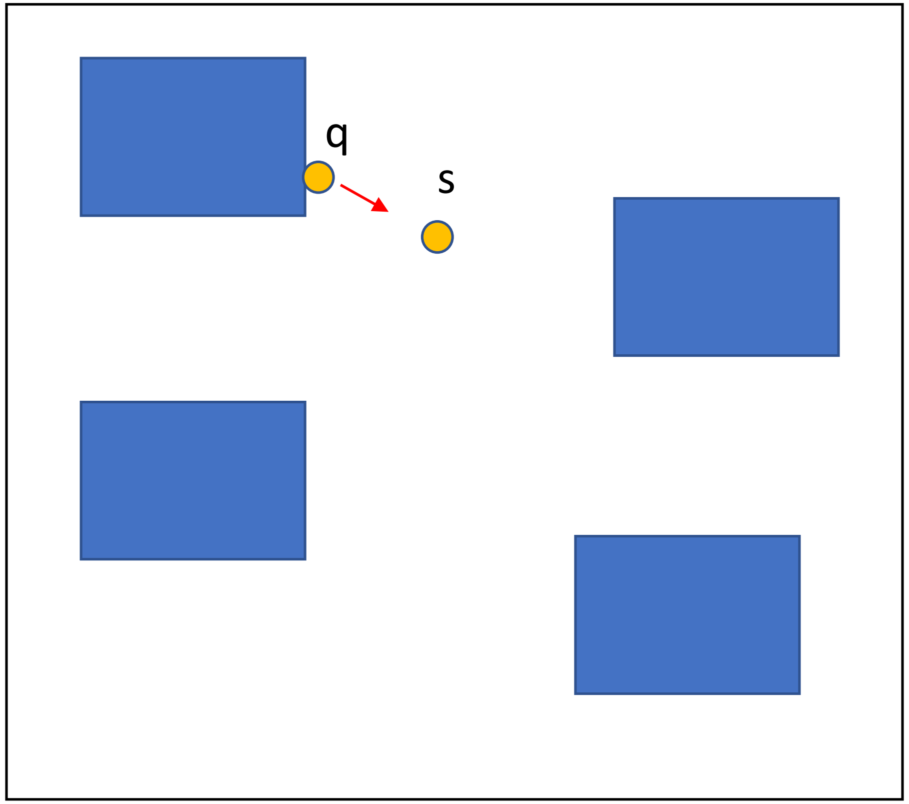
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



Sampling Strategy: Medial Axis Sampling

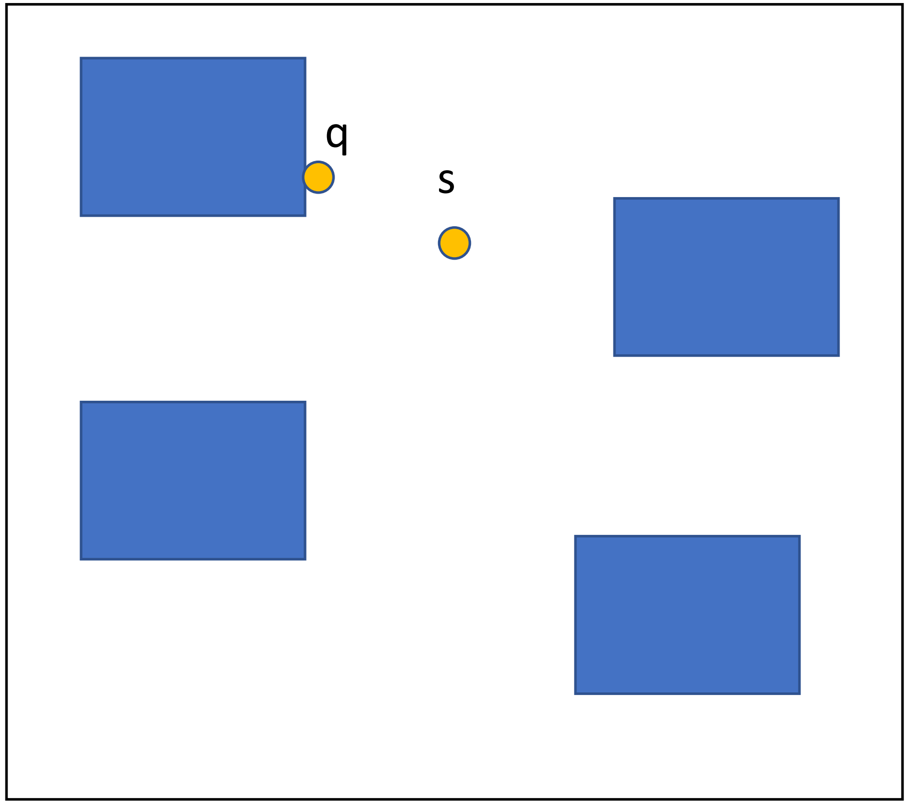
Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.

Output. N nodes in C_{free} connected into a roadmap.

repeat
    Sample a configuration p from C-space.
    if $p \in C_{free}$ **then**
        q = **NearestContactCfg_Clearance**(p)
        Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
    else
        q = **NearestContactCfg_Penetration**(p)
        Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
    end if
    Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



Sampling Strategy: Medial Axis Sampling

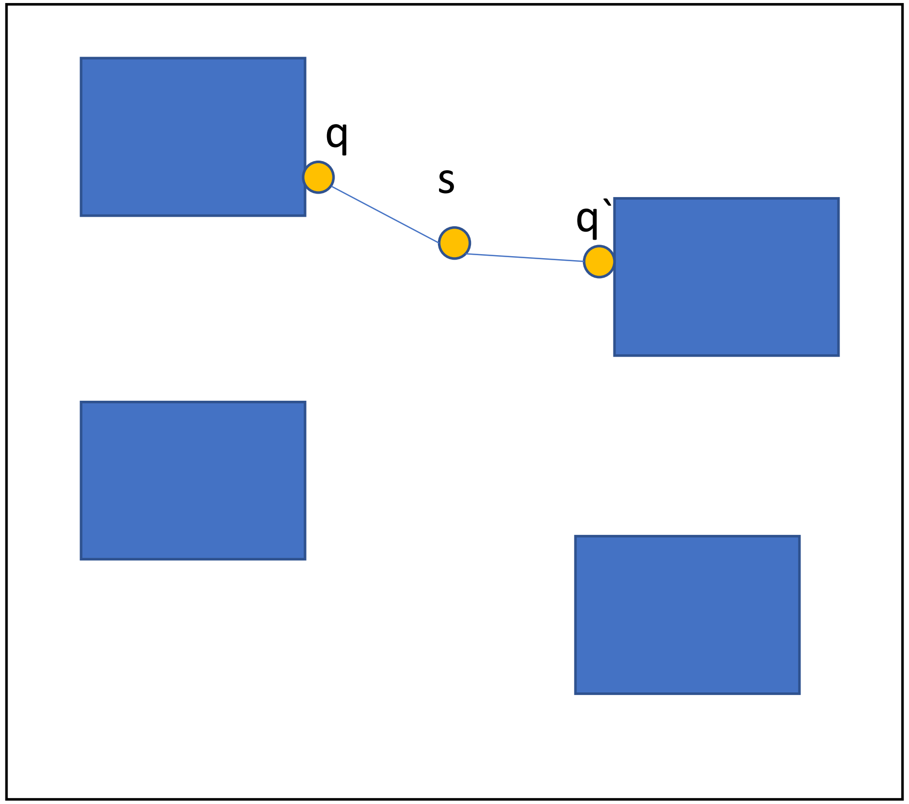
Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.

Output. N nodes in C_{free} connected into a roadmap.

1: **repeat**
2: Sample a configuration p from C-space.
3: **if** $p \in C_{free}$ **then**
4: q = **NearestContactCfg_Clearance**(p)
5: Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
6: **else**
7: q = **NearestContactCfg_Penetration**(p)
8: Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
9: **end if**
10: Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
11: **until** N nodes have been generated
12: Build connections between nodes using local planners.



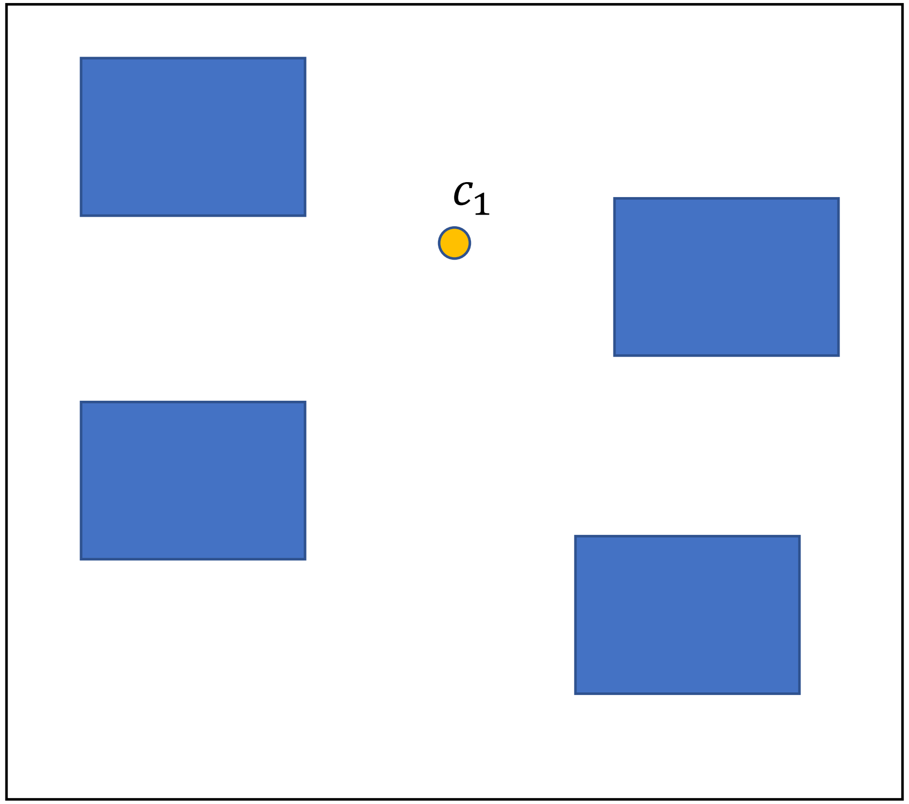
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

1: **repeat**
2: Sample a configuration p from C-space.
3: **if** $p \in C_{free}$ **then**
4: q = **NearestContactCfg_Clearance**(p)
5: Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
6: **else**
7: q = **NearestContactCfg_Penetration**(p)
8: Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
9: **end if**
10: Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
11: **until** N nodes have been generated
12: Build connections between nodes using local planners.



Sampling Strategy: Medial Axis Sampling

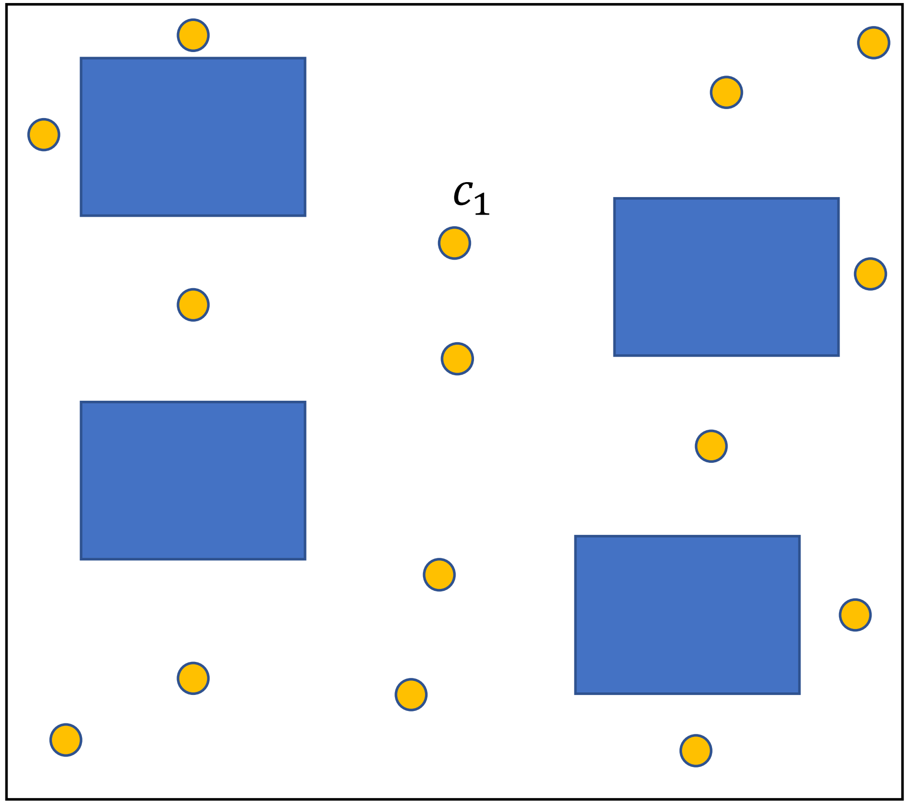
Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.

Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{q\,p}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{p\,q}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



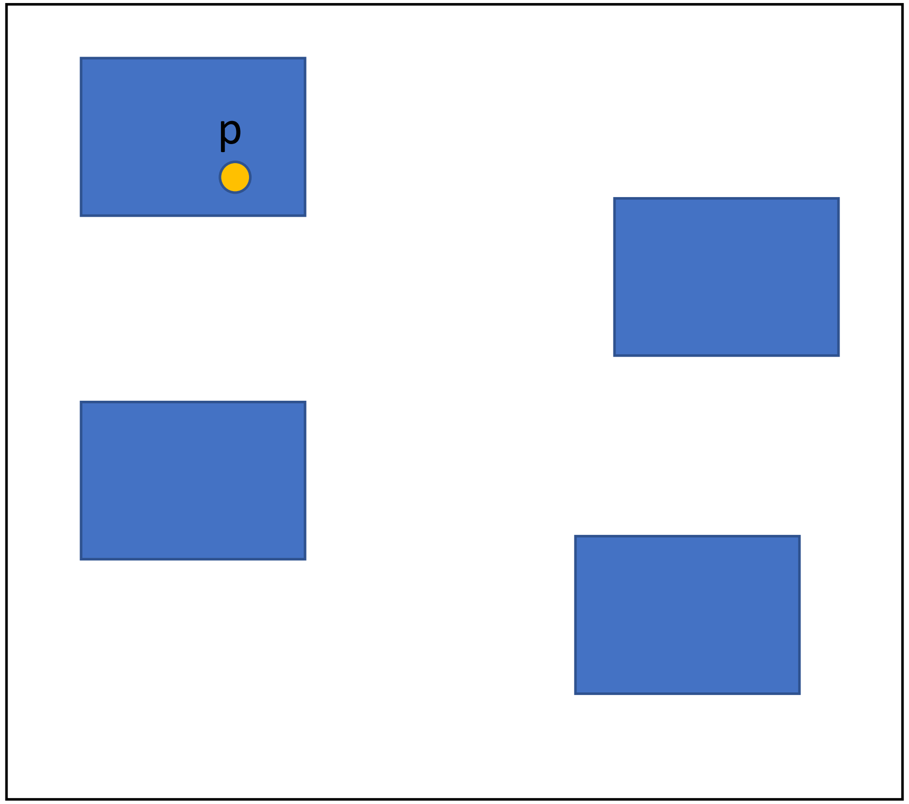
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
    Sample a configuration p from C-space.
    if $p \in C_{free}$ **then**
        q = **NearestContactCfg_Clearance**(p)
        Set retraction direction $\vec{v} = \overrightarrow{q\,p}$ and start point $s = p$
    else
        q = **NearestContactCfg_Penetration**(p)
        Set retraction direction $\vec{v} = \overrightarrow{p\,q}$ and start point $s = q$
    end if
    Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



Sampling Strategy: Medial Axis Sampling

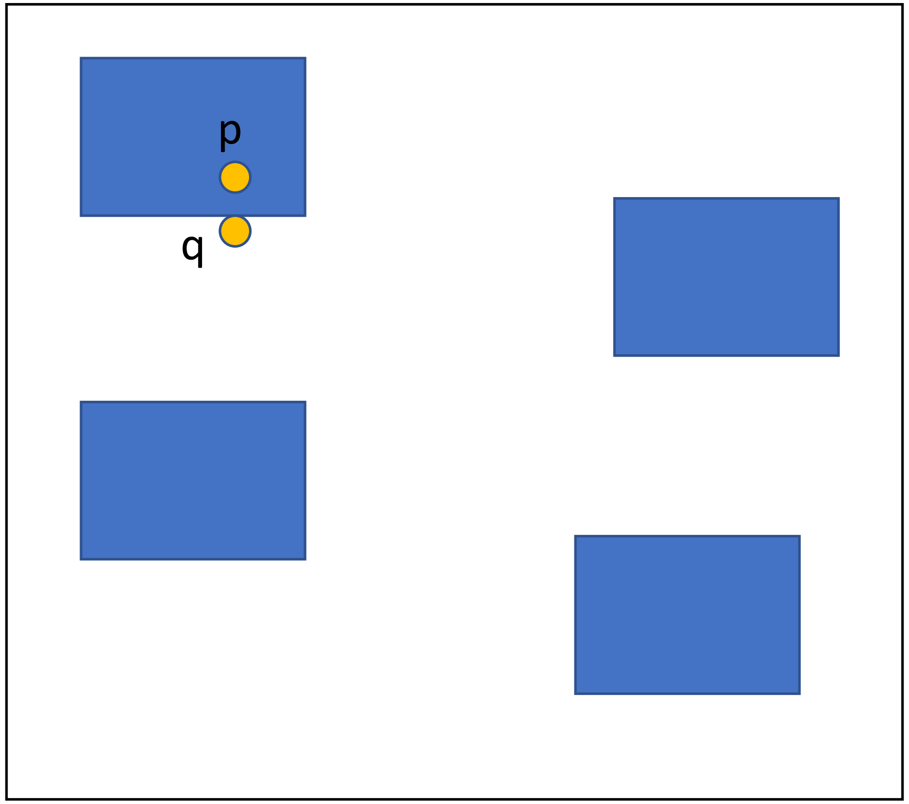
Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.

Output. N nodes in C_{free} connected into a roadmap.

1: **repeat**
2: Sample a configuration p from C-space.
3: **if** $p \in C_{free}$ **then**
4: q = **NearestContactCfg_Clearance**(p)
5: Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
6: **else**
7: q = **NearestContactCfg_Penetration**(p)
8: Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
9: **end if**
10: Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
11: **until** N nodes have been generated
12: Build connections between nodes using local planners.



Sampling Strategy: Medial Axis Sampling

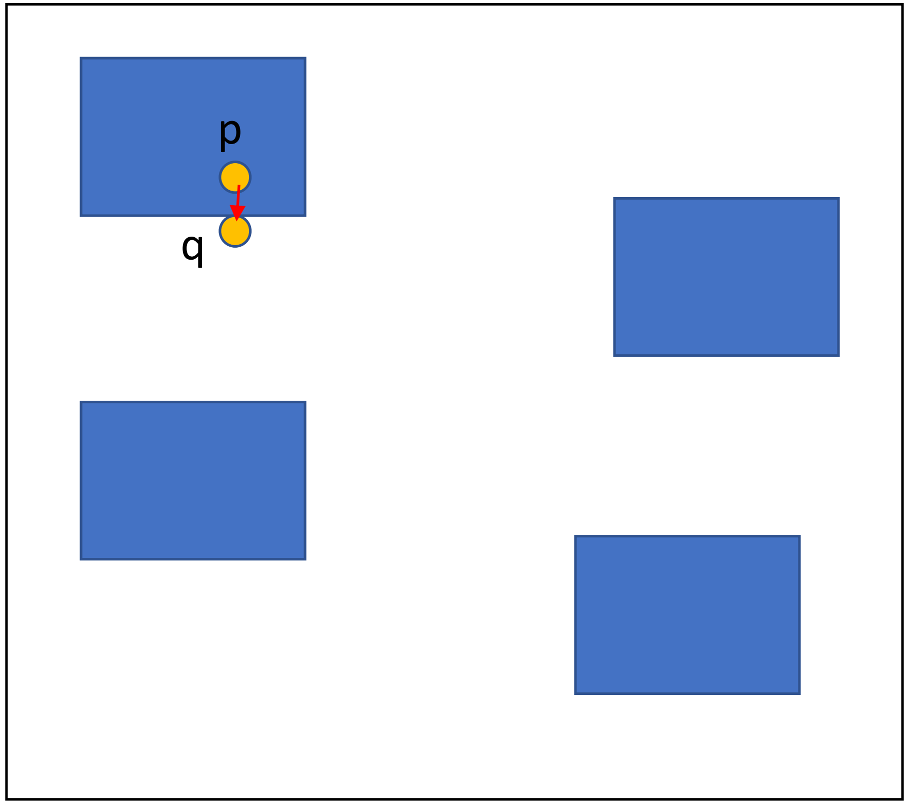
Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.

Output. N nodes in C_{free} connected into a roadmap.

```
repeat
    Sample a configuration p from C-space.
    if p ∈ C_free then
        q = NearestContactCfg_Clearance(p)
        Set retraction direction v⃗ = q⃗p⃗ and start point s = p
    else
        q = NearestContactCfg_Penetration(p)
        Set retraction direction v⃗ = p⃗q⃗ and start point s = q
    end if
    Starting at configuration s, move robot in direction v⃗ until
    it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.
```



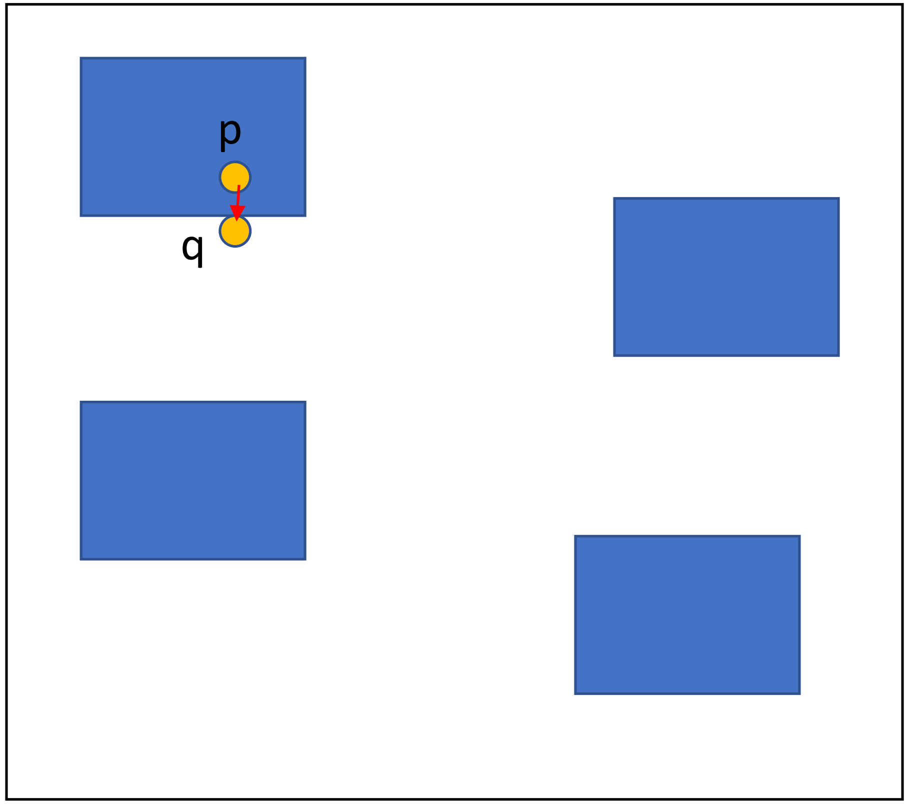
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
    Sample a configuration p from C-space.
    if $p \in C_{free}$ **then**
        q = **NearestContactCfg_Clearance**(p)
        Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
    else
        q = **NearestContactCfg_Penetration**(p)
        Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
    end if
    Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



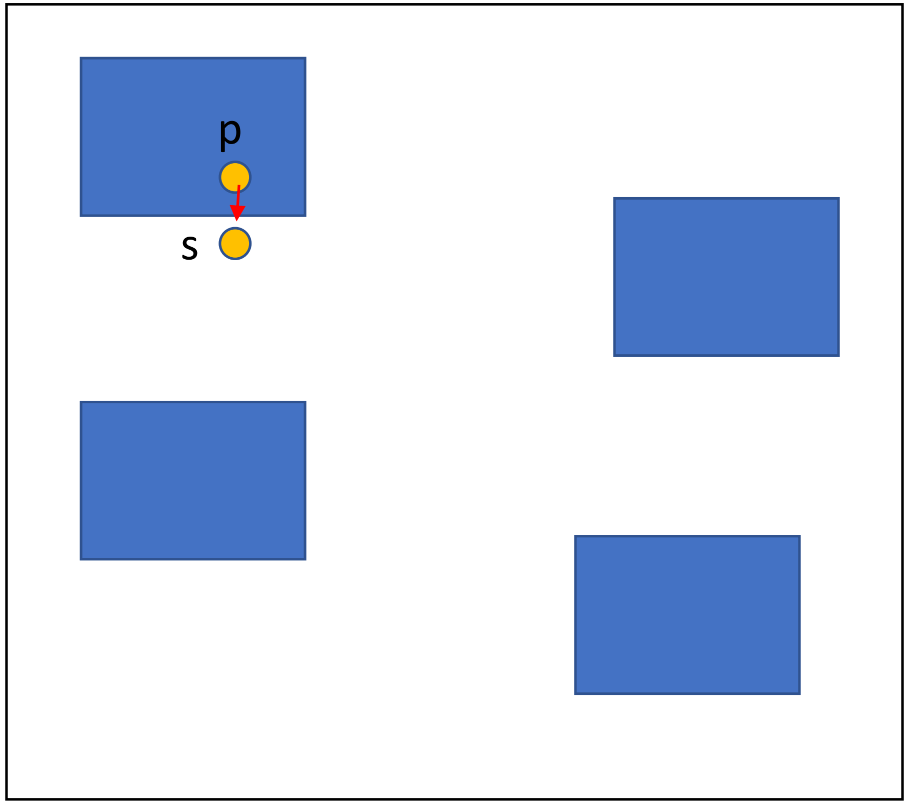
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



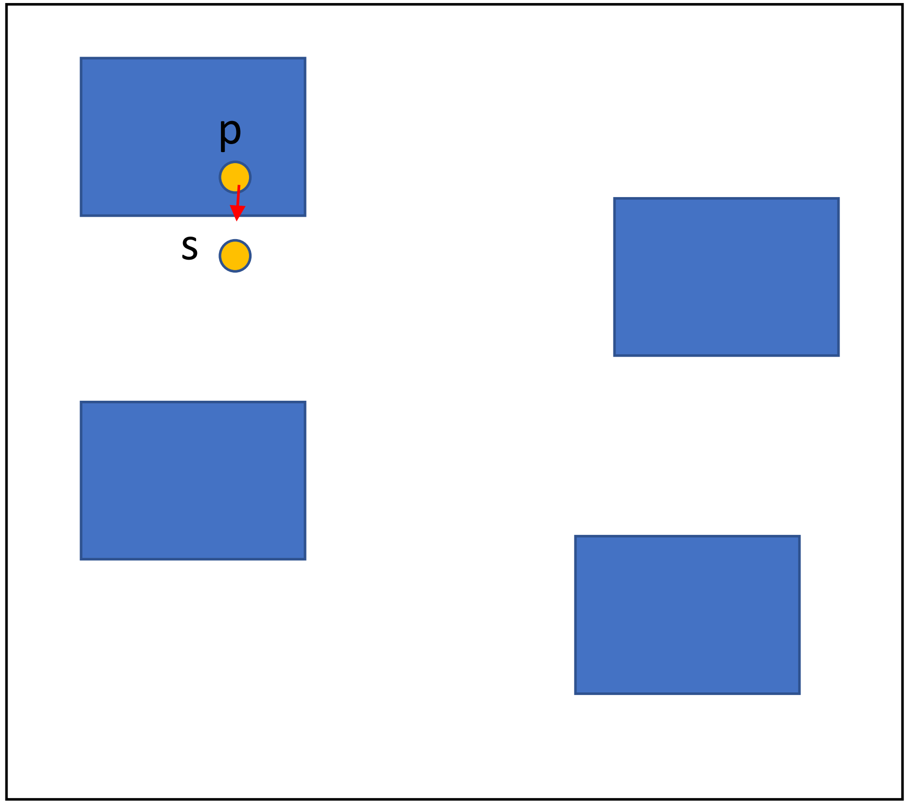
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



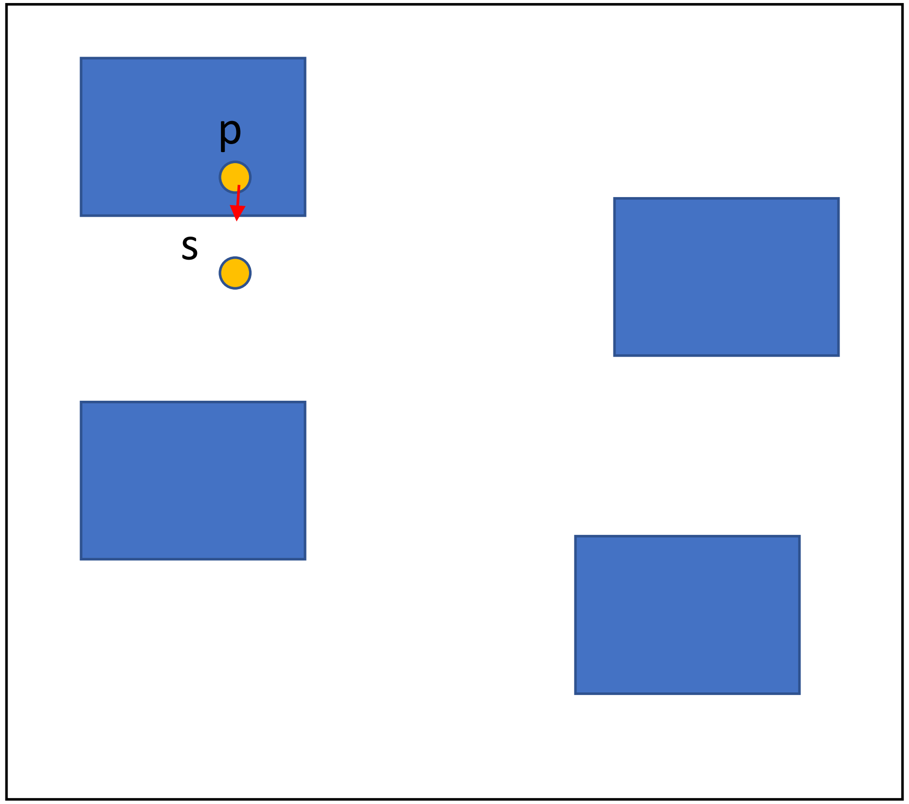
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



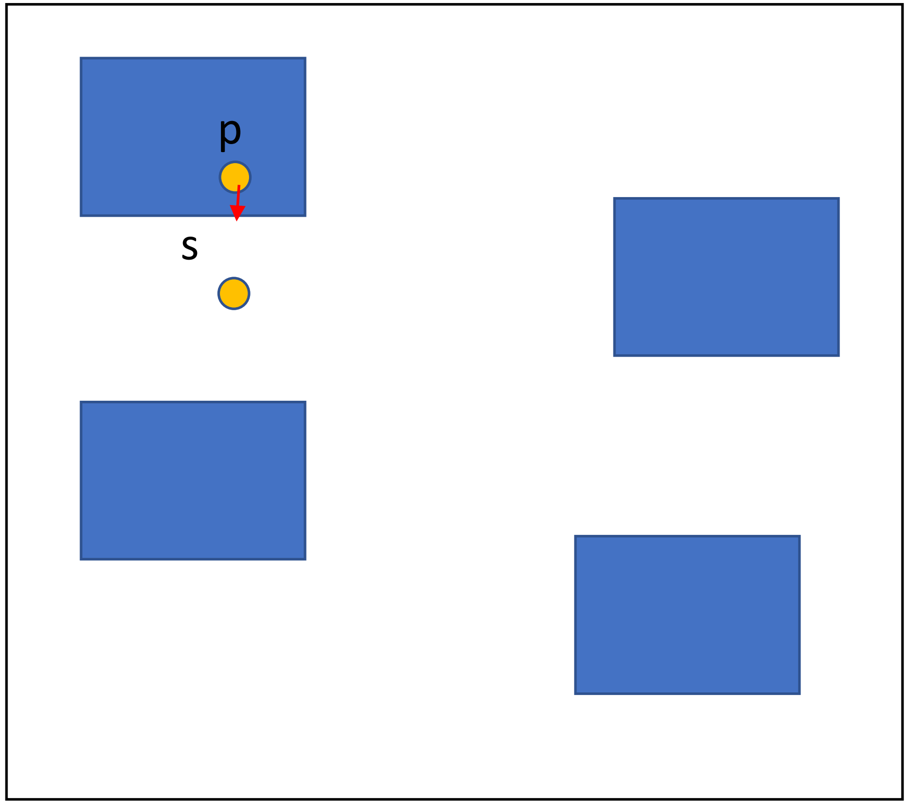
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

```
repeat
    Sample a configuration p from C-space.
    if p ∈ C_free then
        q = NearestContactCfg_Clearance(p)
        Set retraction direction v⃗ = q⃗p and start point s = p
    else
        q = NearestContactCfg_Penetration(p)
        Set retraction direction v⃗ = p⃗q and start point s = q
    end if
    Starting at configuration s, move robot in direction v⃗ until
    it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.
```



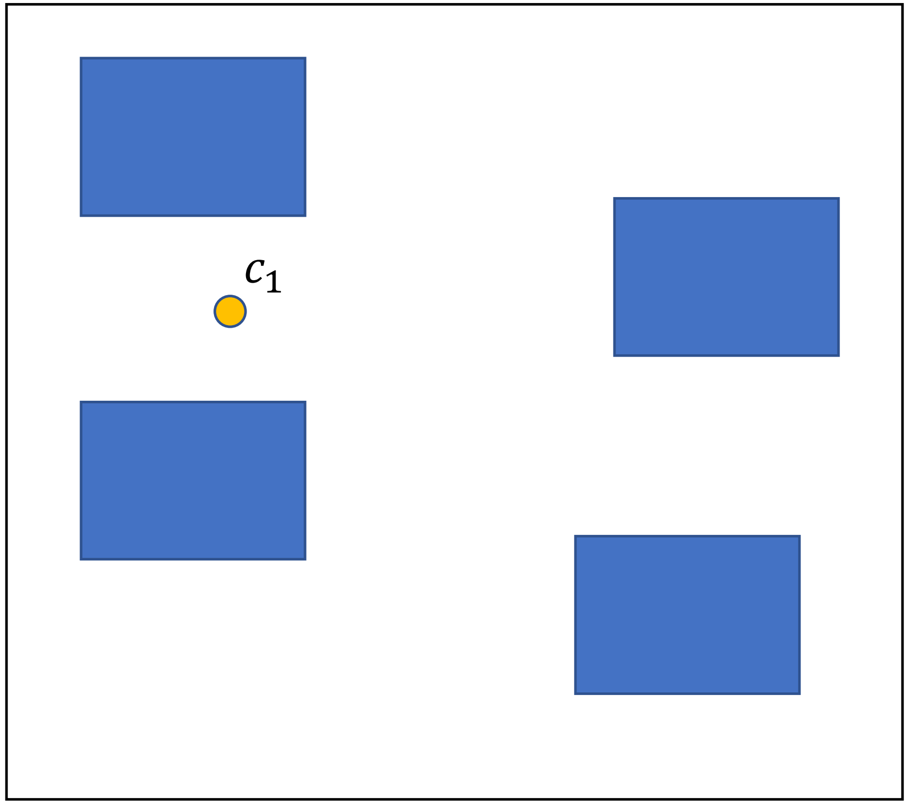
Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.
Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.



Sampling Strategy: Medial Axis Sampling

Algorithm 3.1 Basic MAPRM Framework

Preprocessing:

Input. N, the number of nodes to generate.

Output. N nodes in C_{free} connected into a roadmap.

repeat
  Sample a configuration p from C-space.
  if $p \in C_{free}$ **then**
    q = **NearestContactCfg_Clearance**(p)
    Set retraction direction $\vec{v} = \overrightarrow{qp}$ and start point $s = p$
  else
    q = **NearestContactCfg_Penetration**(p)
    Set retraction direction $\vec{v} = \overrightarrow{pq}$ and start point $s = q$
  end if
  Starting at configuration s, move robot in direction $\vec{v}$ until it has two nearest boundary points (i.e., is on medial axis).
until N nodes have been generated
Build connections between nodes using local planners.

- Computing the nearest contact configuration is hard and in most cases done by distance computation as well as penetration computation in the workspace.
- This sampling approach provides smoother paths in comparison to other sampling strategies.
- It is not as efficient in solving narrow passage problems as the other sampling strategies.

Conclusion

- We have seen that there are many ideas to sample the configuration space.
- They enable us to solve difficult motion planning problems.
- Some of them are very easy to implement.
- Recent research has stopped looking at more sampling strategies with the exception of…

Conclusion

Using AI and machine learning to teach the algorithms

2015: Machine learning guided exploration for sampling-based motion planning algorithms, *Oktay Arslan and Panagiotis Tsiotras*

2018: Deeply Informed Neural Sampling for Robot Motion Planning, *Ahmed H. Qureshi and Michael C. Yip*

2018: Learning Sampling Distributions for Robot Motion Planning, *Brian Ichter , James Harrison , Marco Pavone*