

Motion planning: sampling-based planners III

basic modifications

Vojtěch Vonásek

Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

- One may consider sampling-based planning as a “magic” tool
... but that’s not true at all!

Sampling-based planners have many issues

- Narrow passage problem
 - Difficulty of sampling small region in $\mathcal{C}_{\text{free}}$ surrounded by \mathcal{C}_{obs}
 - Problematic if (all) solutions have to pass that region
- Sensitivity to metric & parameters
 - How to measure distance in \mathcal{C} ?
 - Selecting a good metric is as difficult as motion planning!
 - Many methods have “too many” parameters
 - Some parameters are hidden (or not well described)
 - How to tune the parameters?
- Supporting functions
 - Collision detection & nearest-neighbor search
 - Fast and reliable implementation

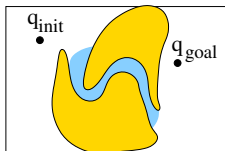
How do we recognize the issue? → performance measurement!

Narrow passage (NP)

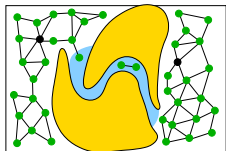
- A region $\mathcal{R} \subseteq \mathcal{C}_{\text{free}}$ with a small volume $\text{vol}(\mathcal{R}) < \text{vol}(\mathcal{C})$
- Probability that a random sample falls to \mathcal{R} is $\sim \text{vol}(\mathcal{R})/\text{vol}(\mathcal{C})$
- NP are problematic if their removal changes connectivity of $\mathcal{C}_{\text{free}}$
- NP are regions in $\mathcal{C} \rightarrow$ they are given implicitly
- Location/size/volume/shape of NPs is not known!

Consequences of having NP

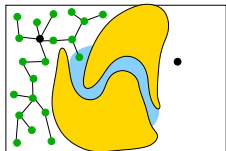
- PRM builds unconnected roadmaps \rightarrow no solution
- RRT/EST cannot enter NP \rightarrow no solution
- Number of samples must be significantly increased
- Runtime is increased



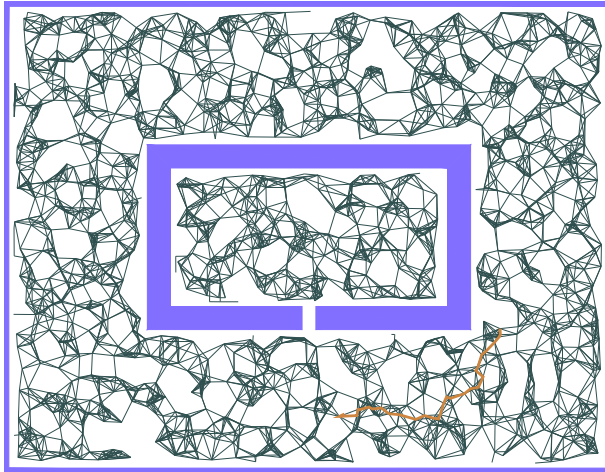
narrow passage (NP)



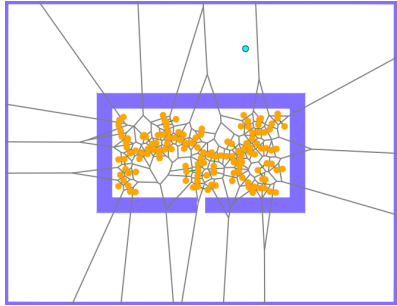
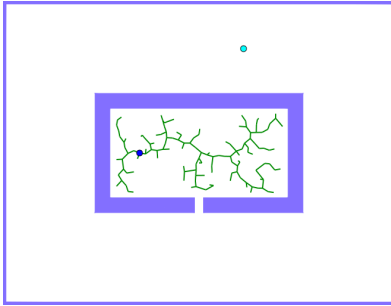
PRM & NP



RRT/EST & NP



Narrow passage & RRT

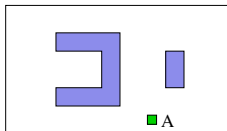




- Narrow passages are in \mathcal{C}
- Sometimes, we cannot (easily) see/estimate them from workspace!
- What makes the narrow passage in the Alpha-puzzle benchmark?

How does \mathcal{C}_{obs} appears?

- Can we guess shape of \mathcal{C}_{obs} based on workspace?
- $\text{vol}(\mathcal{A}) \ll \text{vol}(\mathcal{O})$

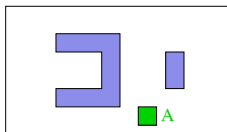


Workspace

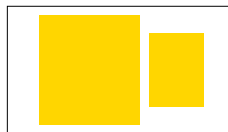


Configuration space

- $\text{vol}(\mathcal{A}) < \text{vol}(\mathcal{O})$



Workspace



Configuration space

- When obstacles \mathcal{O} dominate, they mostly influence the shape of \mathcal{C}_{obs}

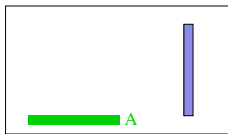
How does \mathcal{C}_{obs} appear?

- Let $X, Y \subset \mathbb{R}^n$, X and Y are nonempty
- Brunn-Minkowski theorem:

$$\text{vol}(X \oplus Y) \geq (\text{vol}(X)^{\frac{1}{n}} + \text{vol}(Y)^{\frac{1}{n}})^n$$

- $\text{vol}(\mathcal{C}_{\text{obs}})$ is larger than $\min(\text{vol}(\mathcal{A}), \text{vol}(\mathcal{O}))$
- $\text{vol}(\mathcal{C}_{\text{obs}})$ can be much larger!

Example: $\text{vol}(\mathcal{A}) = \text{vol}(\mathcal{O})$



Workspace



Configuration space

Why improvements of PRM/RRT/EST?

- To cope with the narrow passage problem, improve path quality, speed-up planning, to enable planning in specific cases

Main tricks

- Change distribution of random samples
- Dedicated metrics
- Improved nearest-neighbor search
- Use suitable local planners
- Improve collision-detection

```
1 initialize tree  $\mathcal{T}$  with  $q_{init}$ 
2 for  $i = 1, \dots, l_{max}$  do
3    $q_{rand} = \text{generate randomly in } \mathcal{C}$ 
4    $q_{near} = \text{find nearest node in } \mathcal{T} \text{ towards}$ 
    $q_{rand}$ 
5    $q_{new} = \text{localPlanner from } q_{near} \text{ towards}$ 
    $q_{rand}$ 
6   if  $\text{canConnect}(q_{near}, q_{new})$  then
7      $\mathcal{T}.\text{addNode}(q_{new})$ 
8      $\mathcal{T}.\text{addEdge}(q_{near}, q_{new})$ 
9     if  $\rho(q_{new}, q_{goal}) < d_{goal}$  then
10      return path from  $q_{init}$  to  $q_{new}$ 
```

- Many existing modifications of sampling-based planners, look at surveys
- Next slides present the basic principle of improvements

☛ Elbanhawi, M., & Simic, M. (2014). Sampling-based robot motion planning: A review. *IEEE access*, 2, 56-77.

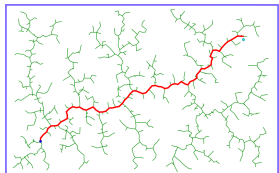
☛ Veras, Luiz Gustavo D. O., Felipe L. L. Medeiros, and Lamartine N. F. Guimaraes. Systematic Literature Review of Sampling Process in Rapidly-Exploring Random Trees. *IEEE Access* 7 (2019)

Observation

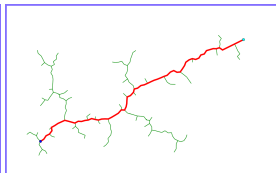
- RRT tree grows towards random samples
- If we sample some region more dense, the tree is “attracted” to grow there

Goal-bias

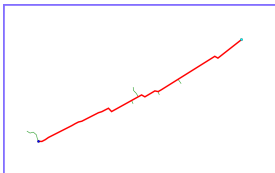
- Random sample q_{rand} is generated in \mathcal{C} with probability $(1 - p_{\text{goal}})$, otherwise it is set to $q_{\text{rand}} = q_{\text{goal}}$
- The rest of RRT algorithm is the same
- Improves the performance if the tree can directly reach the goal
- Decreases the performance if the tree is hindered by obstacles



$p_{\text{goal}} = 0$



$p_{\text{goal}} = 0.1$



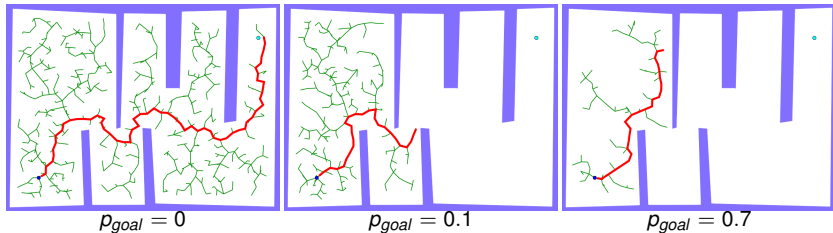
$p_{\text{goal}} = 0.7$

Observation

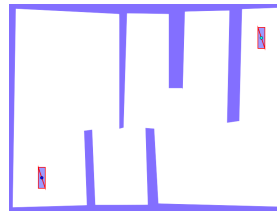
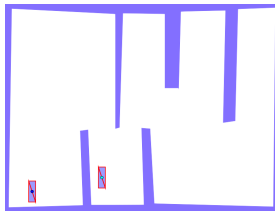
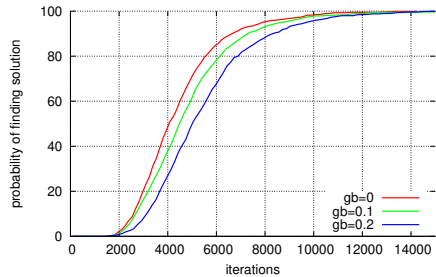
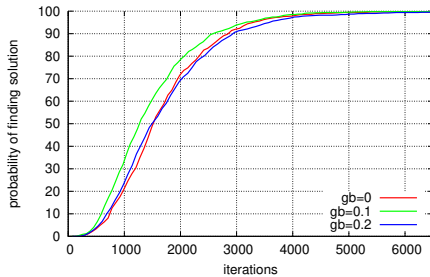
- RRT tree grows towards random samples
- If we sample some region more dense, the tree is “attracted” to grow there

Goal-bias

- Random sample q_{rand} is generated in \mathcal{C} with probability $(1 - p_{\text{goal}})$, otherwise it is set to $q_{\text{rand}} = q_{\text{goal}}$
- The rest of RRT algorithm is the same
- Improves the performance if the tree can directly reach the goal
- Decreases the performance if the tree is hindered by obstacles

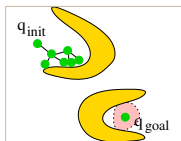


- Goal-bias may improve or even worsen the performance!



Observation

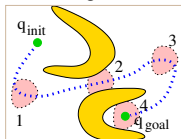
- Goal-bias attracts the tree towards q_{goal} , but the tree may be blocked by obstacles
- Generalization: we can attract the tree toward any region $\mathcal{R} \subseteq \mathcal{C}$ if we sample \mathcal{R} densely



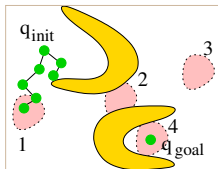
RRT + goal-bias

Guided-based sampling

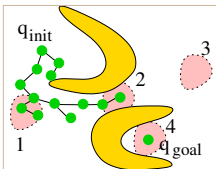
- Estimate a path that can “guide” the tree in the \mathcal{C} -space
- Generate q_{rand} around the path-waypoints (starting from first waypoint) until the tree reaches the waypoint
- Then generate q_{rand} around the next waypoint



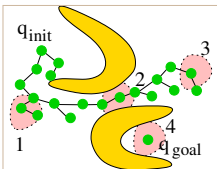
Guiding path



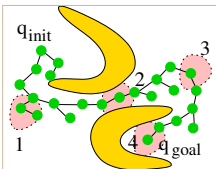
Sampling at 1



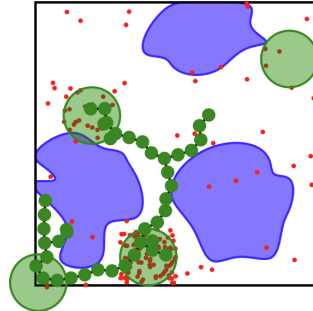
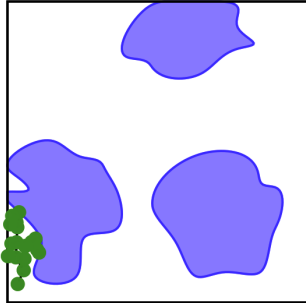
Sampling at 2



Sampling at 3



Sampling at 4



How to compute the guiding path?

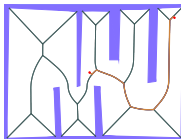
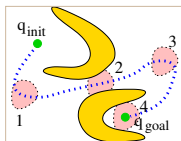
- Generally, the guiding path has to be located in \mathcal{C} !!
- Finding a good guiding path has the same complexity as the original planning problem!
- (i.e., guiding sampling is 'planning solved by planning')
- Practically, we have two options

Guiding path in \mathcal{W}

- Path is computed in workspace — geometric planning (Voronoi diagram, Visibility graph, etc.)
- Suitable for low-dimensional problems
- The remaining dimensions are sampled uniformly

Guiding path in \mathcal{C}

- Path is computed in \mathcal{C} by a simplified search

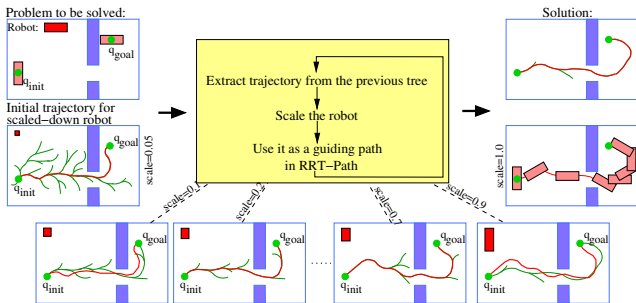


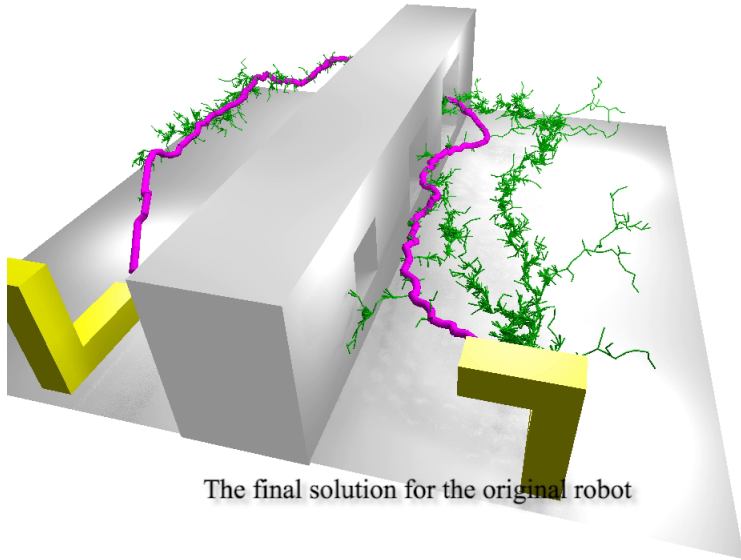
Guiding path in \mathcal{W}

$q = (x, y, \varphi)$
 (x, y) from the
path
 φ randomly

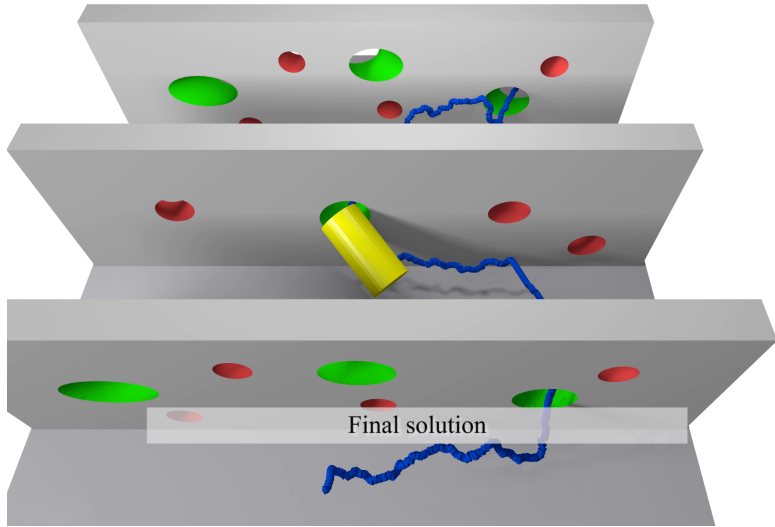
Guiding path in \mathcal{C}

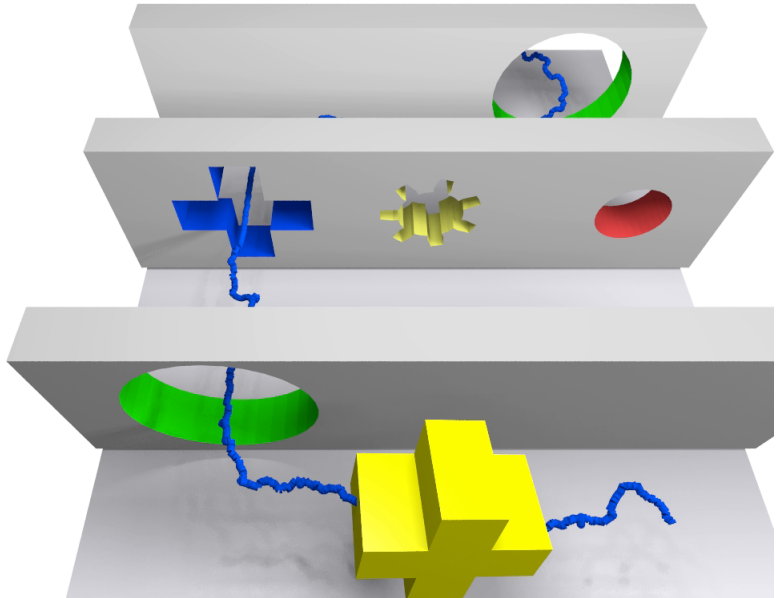
- Problem is simplified — relaxation of constraints
- For example, robot is scaled-down
- Solve simplified planning problem
- Use the solution to generate random samples along it
- The process can be iterative





The final solution for the original robot

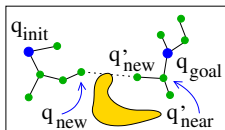
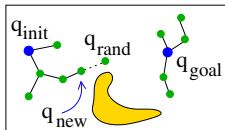
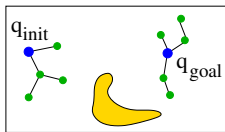




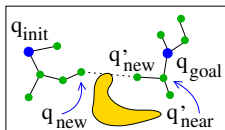
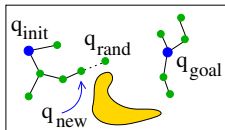
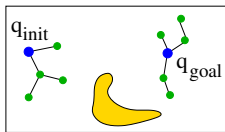
RRT improvement III: bidirectional search

- Use two trees: \mathcal{T}_i rooted at q_{init} , \mathcal{T}_g rooted q_{goal}
- One tree expands towards q_{rand} , second tree expands towards q_{new} of the first tree

```
1  $\mathcal{T}_i.addNode(q_{init})$ 
2  $\mathcal{T}_g.addNode(q_{goal})$ 
3 for  $i = 1, \dots, l_{max}$  do
4    $q_{rand} = \text{generate randomly in } \mathcal{C}$ 
5    $q_{near} = \text{find nearest node in } \mathcal{T}_i \text{ towards } q_{rand}$ 
6    $q_{new} = \text{localPlanner from } q_{near} \text{ towards } q_{rand}$ 
7   if  $canConnect(q_{near}, q_{new})$  then
8      $\mathcal{T}_i.addNode(q_{new})$ 
9      $\mathcal{T}_i.addEdge(q_{near}, q_{new})$ 
10     $q'_{near} = \text{find nearest node in } \mathcal{T}_g \text{ towards } q_{new}$ 
11     $q'_{new} = \text{localPlanner from } q_{near} \text{ towards } q_{rand}$ 
12    if  $canConnect(q'_{near}, q'_{new})$  then
13       $\mathcal{T}_g.addNode(q_{new})$ 
14       $\mathcal{T}_g.addEdge(q_{near}, q_{new})$ 
15      if  $canConnect(q'_{new}, q_{new})$  then
16        joint trees
17        return path from  $q_{init}$  to  $q_{goal}$ 
18   $\mathcal{T}_i, \mathcal{T}_g = \mathcal{T}_g, \mathcal{T}_i$  // swap trees
```



- Use two trees: \mathcal{T}_i rooted at q_{init} , \mathcal{T}_g rooted q_{goal}
- One tree expands towards q_{rand} , second tree expands towards q_{new} of the first tree
- Helps to enter narrow passages (sometimes)
- Connection of two trees
 - Computationally intensive
 - To speed up, performs only if $\varrho(q_{new}, q'_{new})$ is small enough
 - Difficult if motion model/constraints have to be considered
- Balanced trees: swap trees if $|\mathcal{T}_i| > |\mathcal{T}_g|$



Original PRM/sPRM

- Uniform sampling $q \sim U(\mathcal{C})$

Gaussian sampling: two-samples

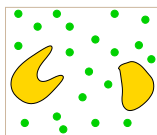
- Uniform sample $q_1 \sim U(\mathcal{C})$, then another sample $q_2 \sim N(q, \Sigma)$ (around q_1 from Gaussian distribution)
- Ignore if $q_1, q_2 \in \mathcal{C}_{\text{free}}$ or $q_1, q_2 \in \mathcal{C}_{\text{obs}}$, otherwise
- add the collision-free one to the roadmap
- Generates the random samples near \mathcal{C}_{obs} only!

Gaussian + uniform

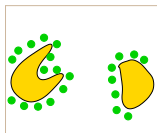
- Combination of two previous methods
- More dense sampling around \mathcal{C}_{obs} than basic PRM

Bridge test

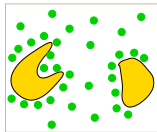
- Generate q_1 and q_2 using the Gaussian method
- Determine the midpoint q' on the line segment $|q_1, q_2|$
- Use q' if $q' \in \mathcal{C}_{\text{free}}$ and $q_1, q_2 \in \mathcal{C}_{\text{obs}}$



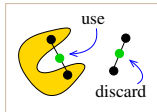
Uniform



Gaussian

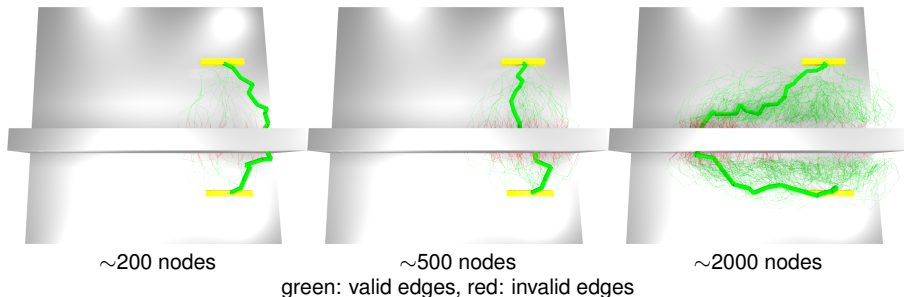


Gaussian + Uniform



Bridge-Test

- Build PRM roadmap, but without collision detection of edges
- After a path is found, edges are checked for collision and the path is recalculated
- If no path is found, extend the roadmap by new samples/edges
- Otherwise, the path is collision-free



- Faster planning in certain scenarios, but not always!

