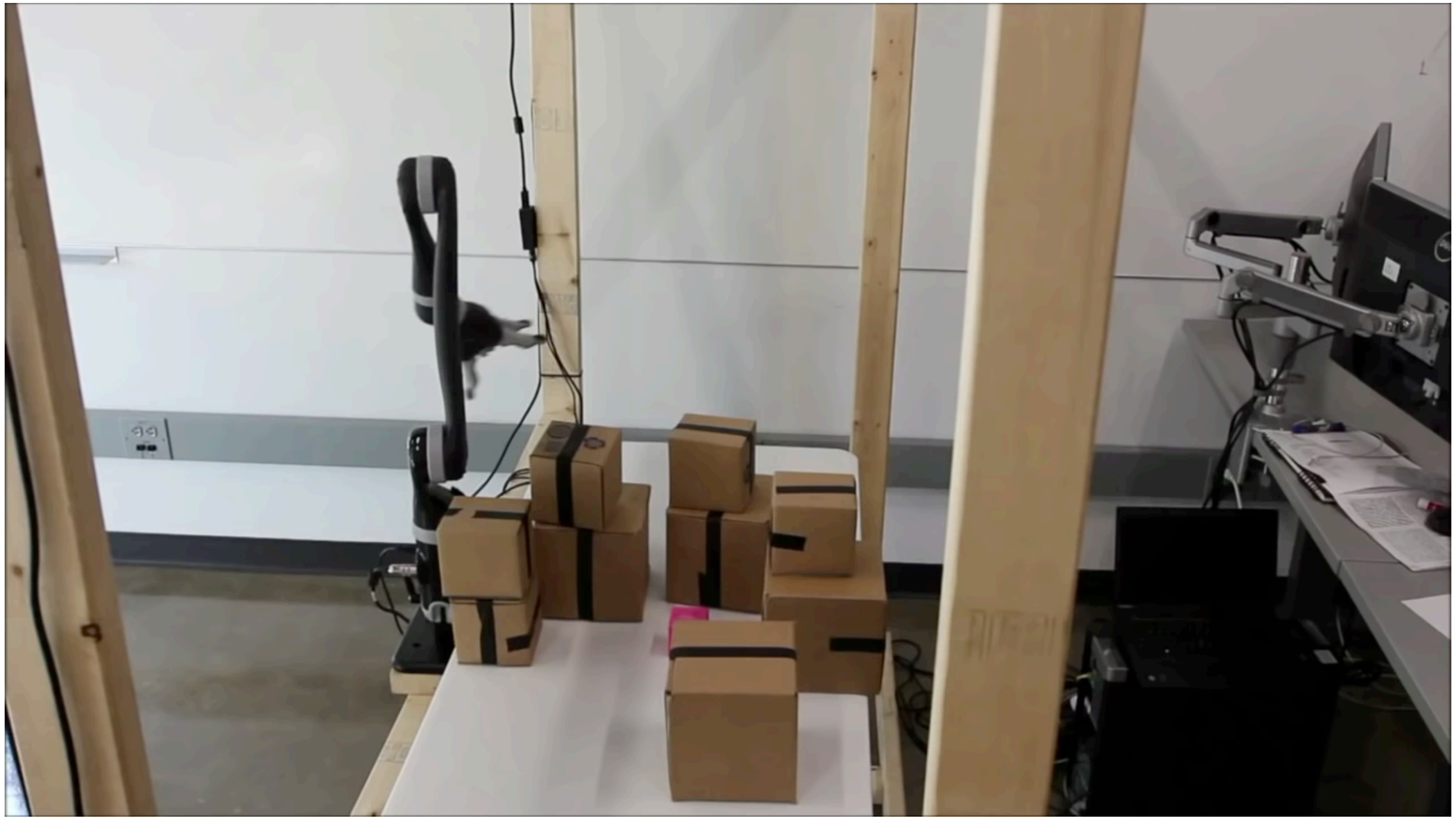
Human-Robot Interaction

Motion Planning

What we want to accomplish

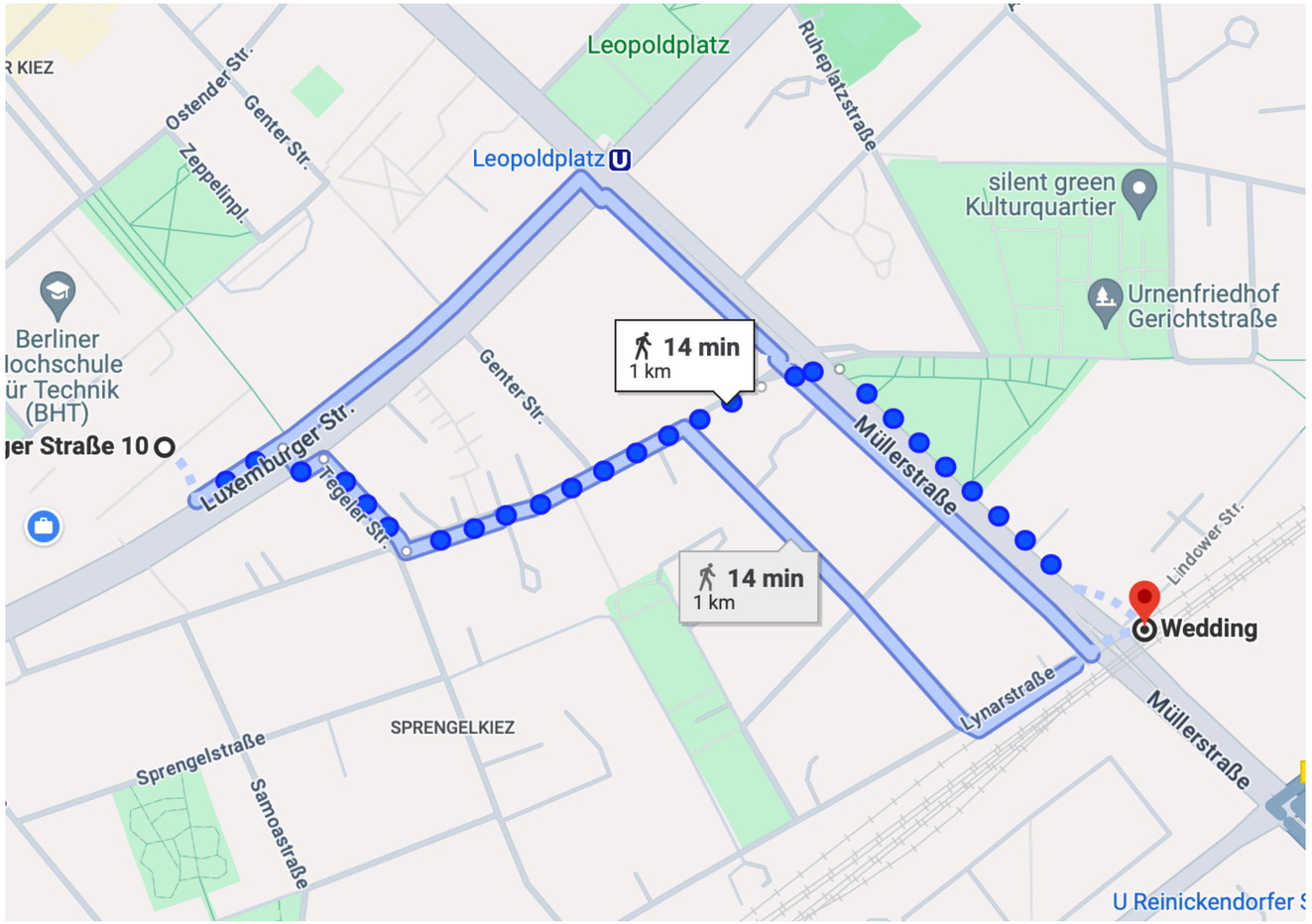


[Sean Murray et al., Duke University, 2016]

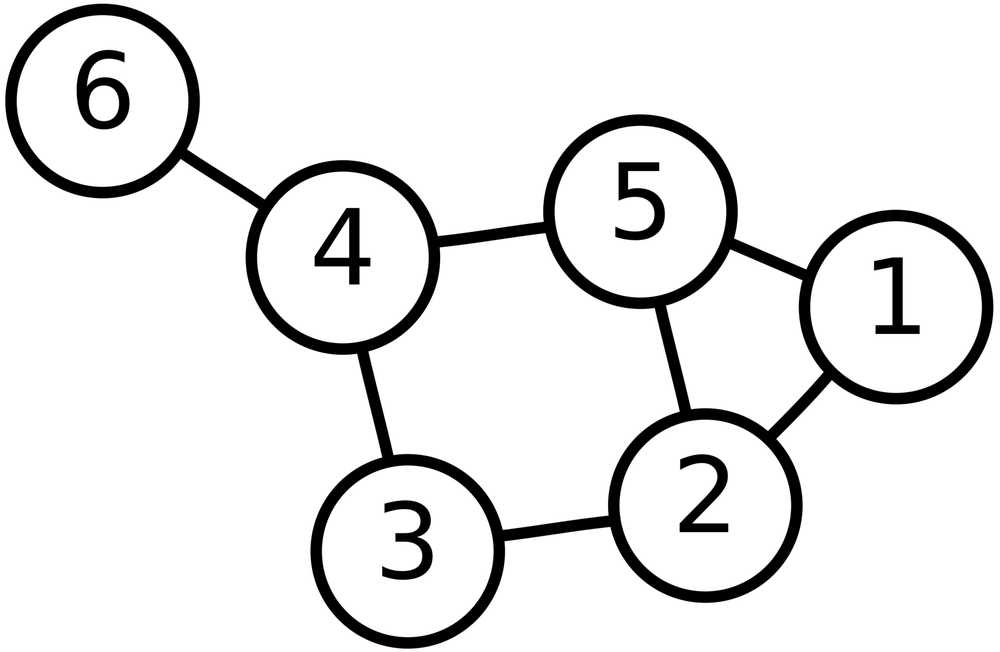
What we want to accomplish

Automatic generation of a **collision-free trajectory in configuration space** for moving the robot towards a **target configuration** or towards a **target pose**.

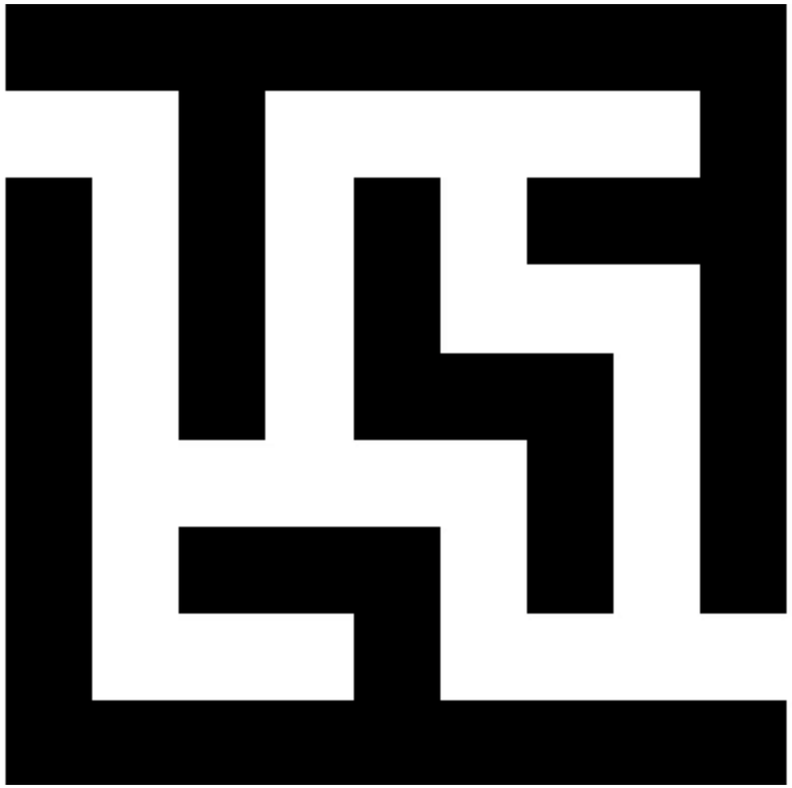
Seach-Algorithm



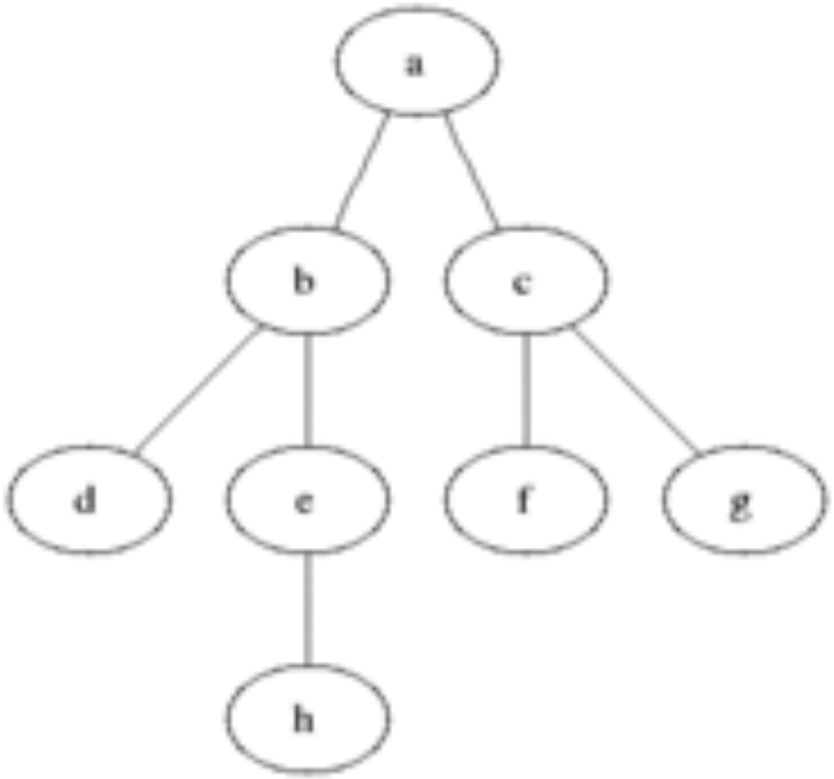
Graph Theory



Seach-Algorithm



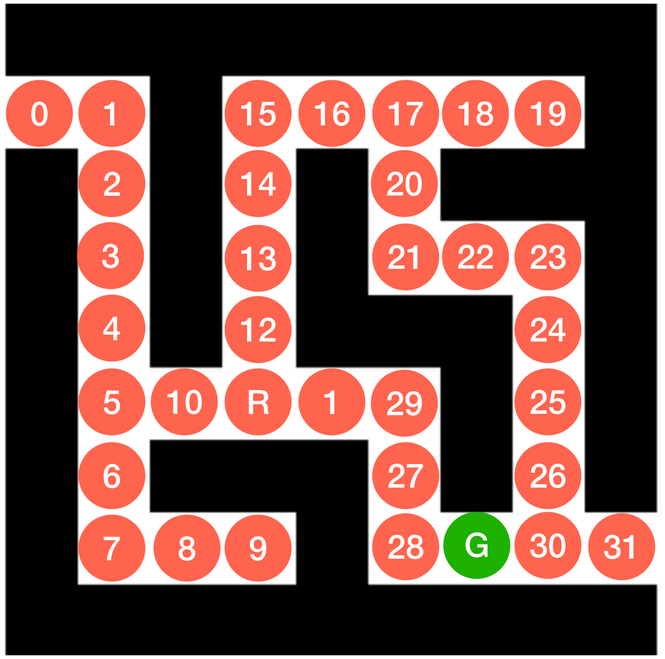
Seach-Algorithm



Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

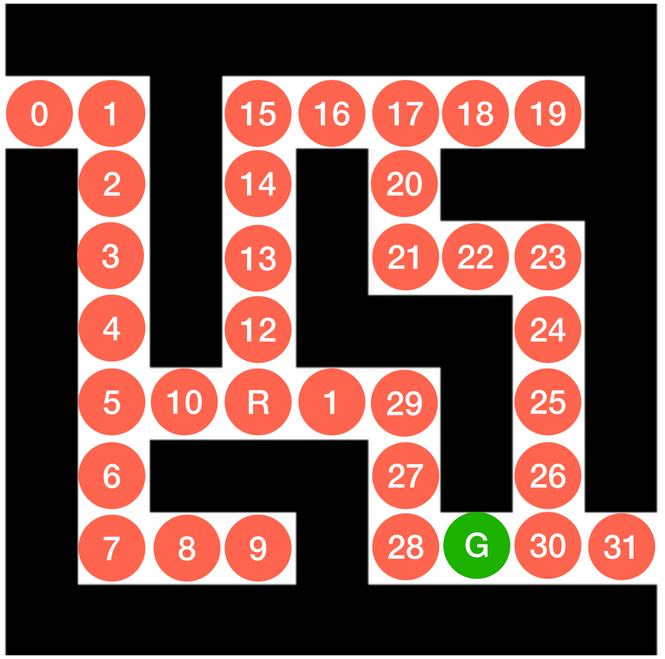


Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

Q:

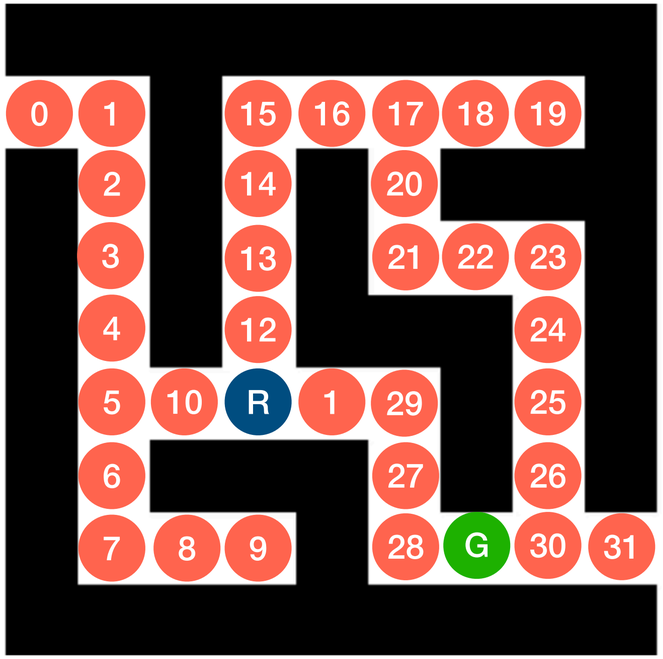


Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

Q:

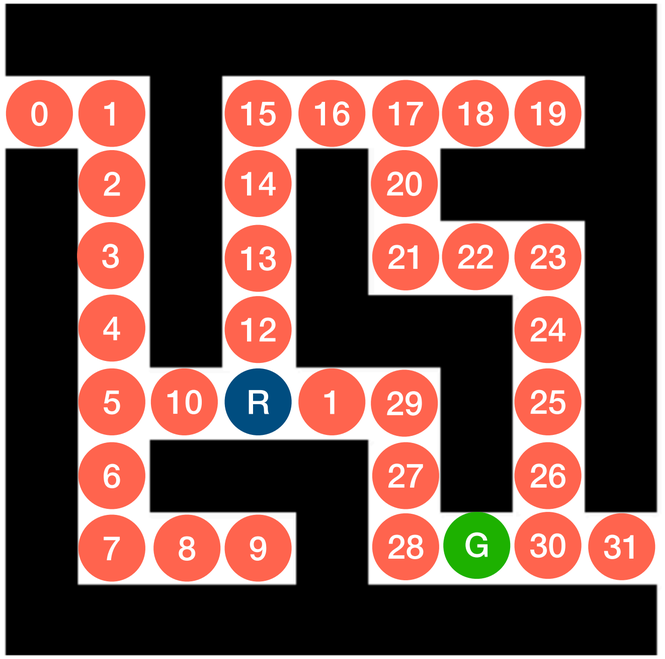


Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

Q: R



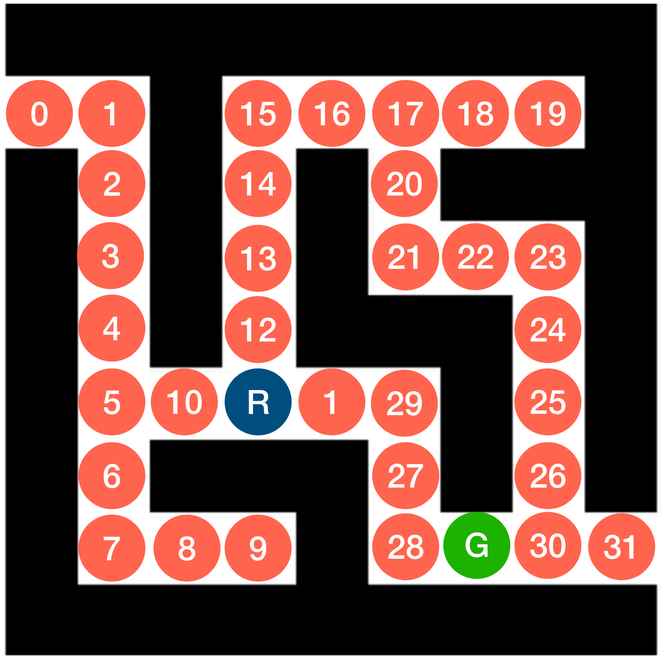
Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

Q:

v: R



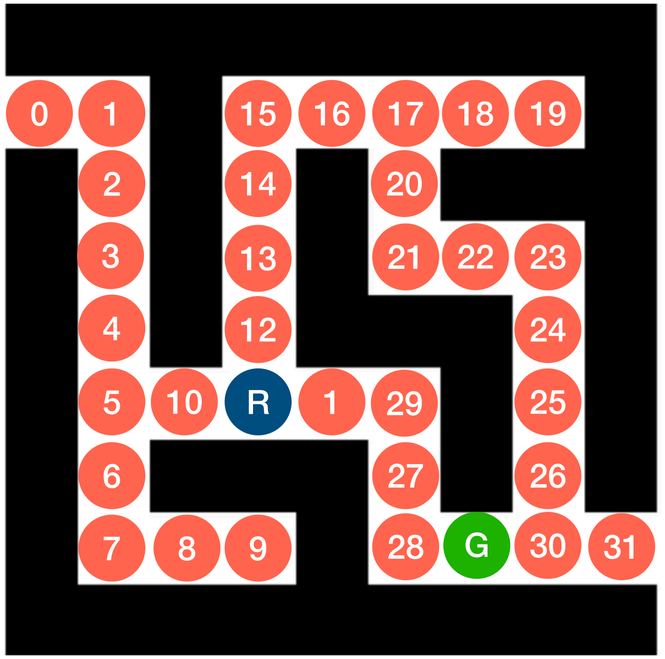
Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

Q:

v: R



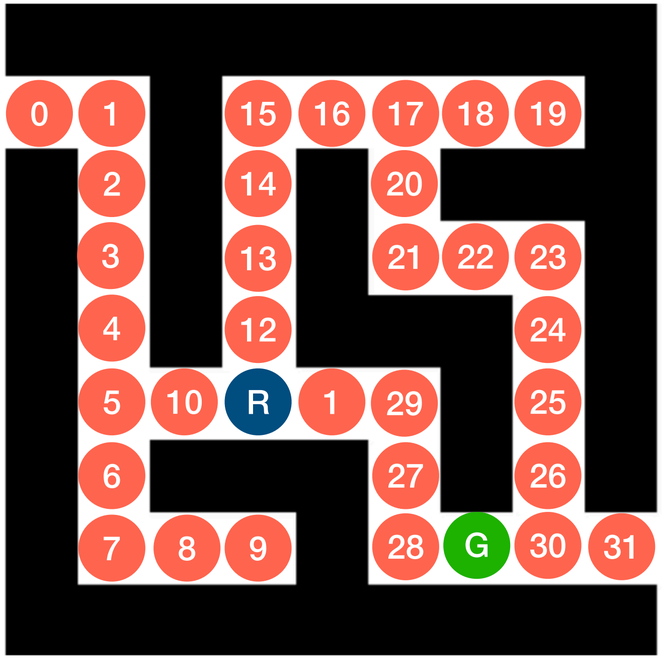
Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

Q:

v: R



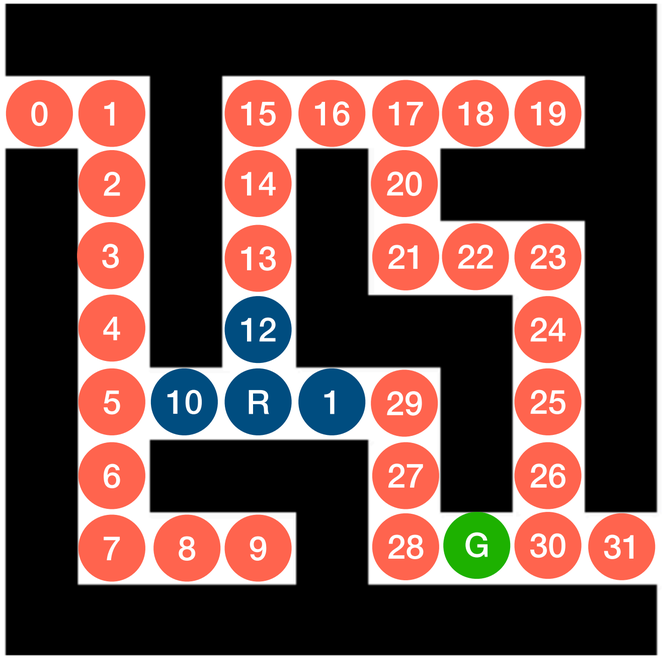
Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

Q:

v: R



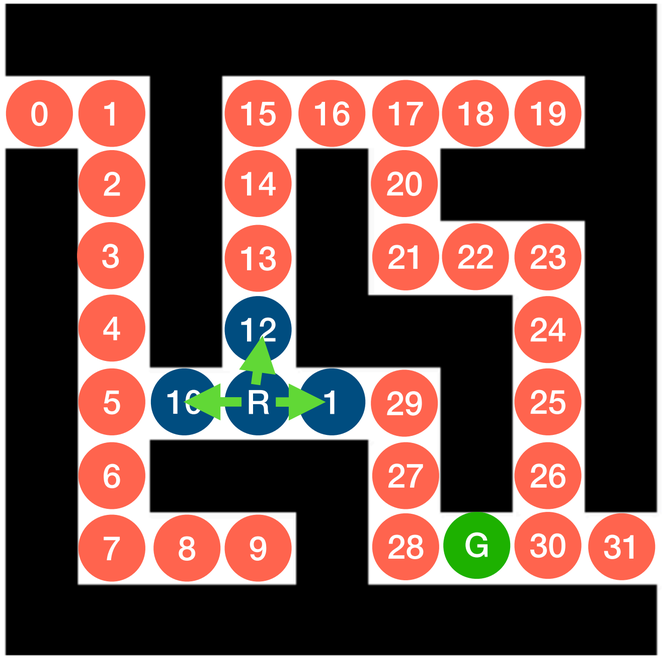
Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```

Q:

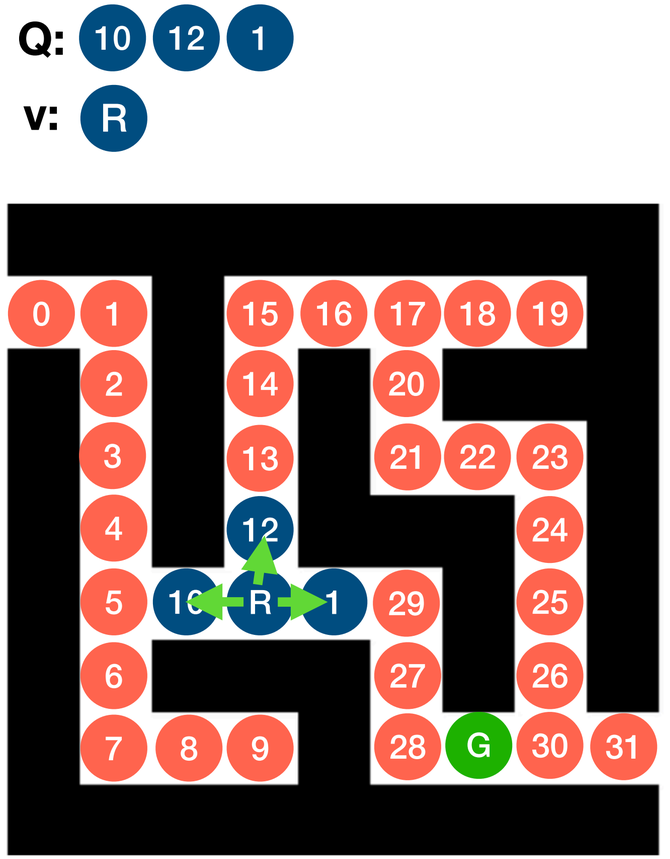
v: R



Seach-Algorithm

Breadth First Search (BFS)

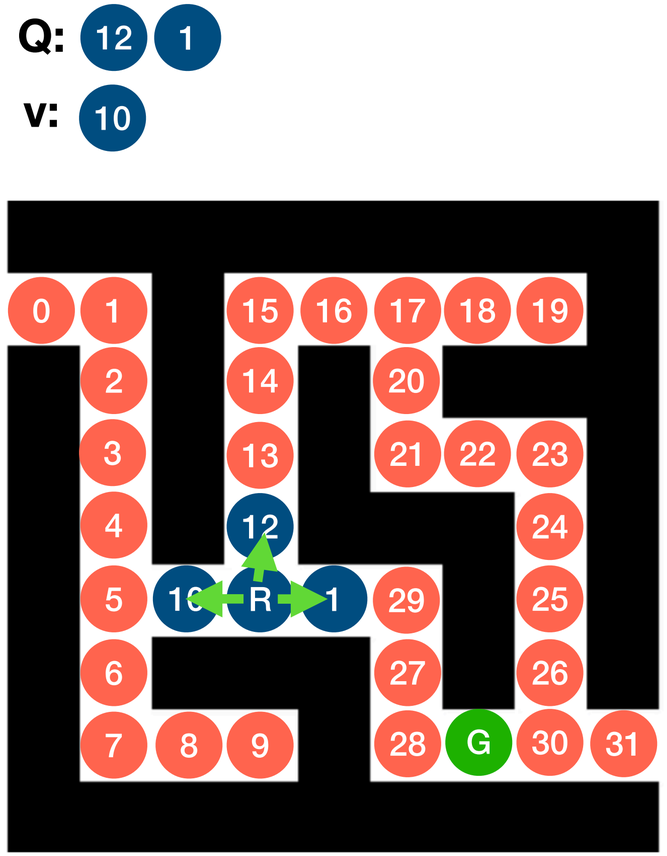
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

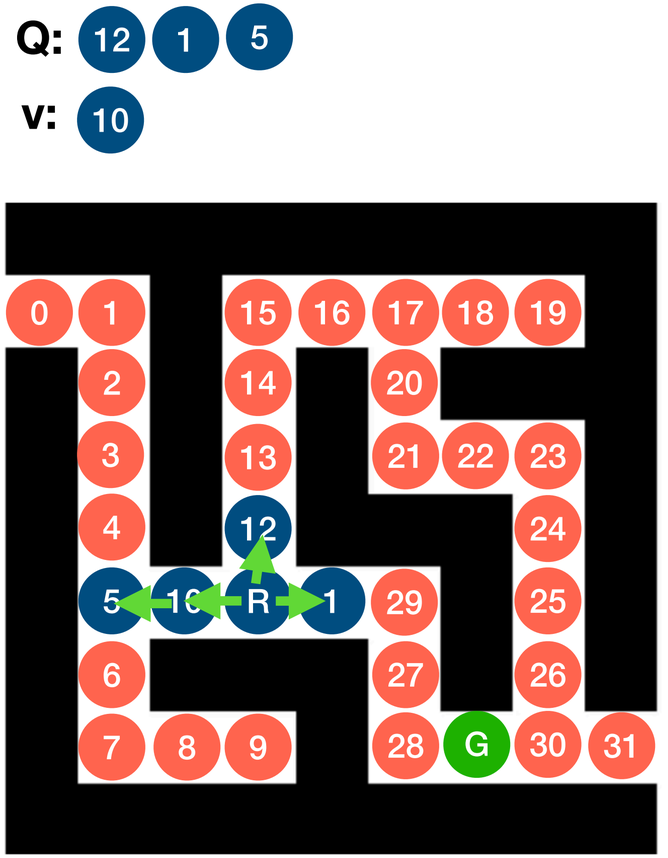
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

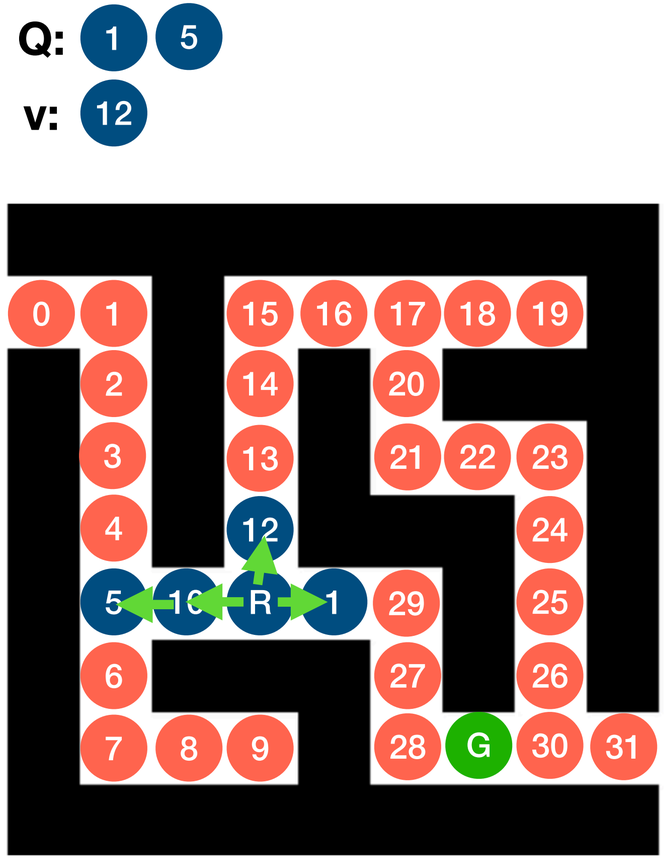
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

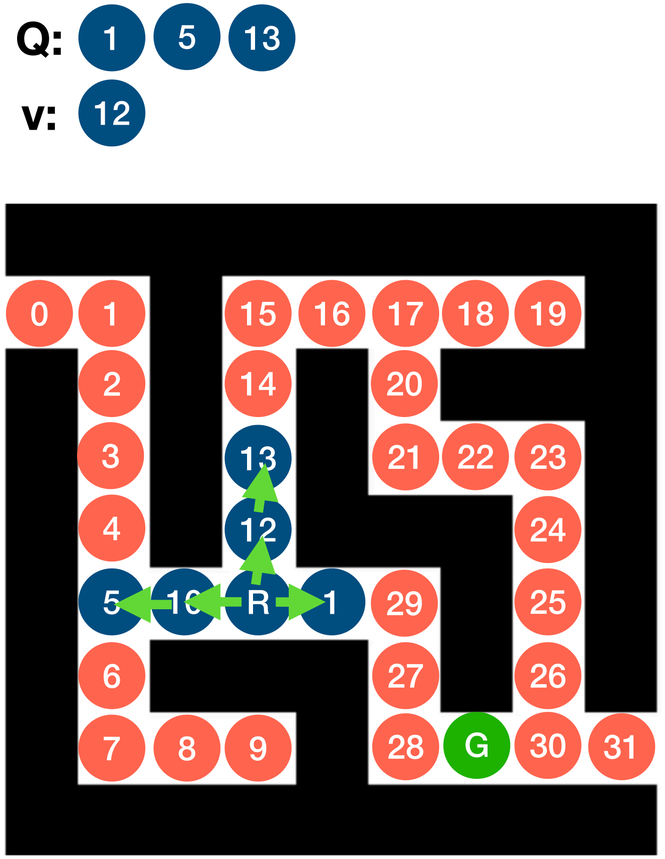
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

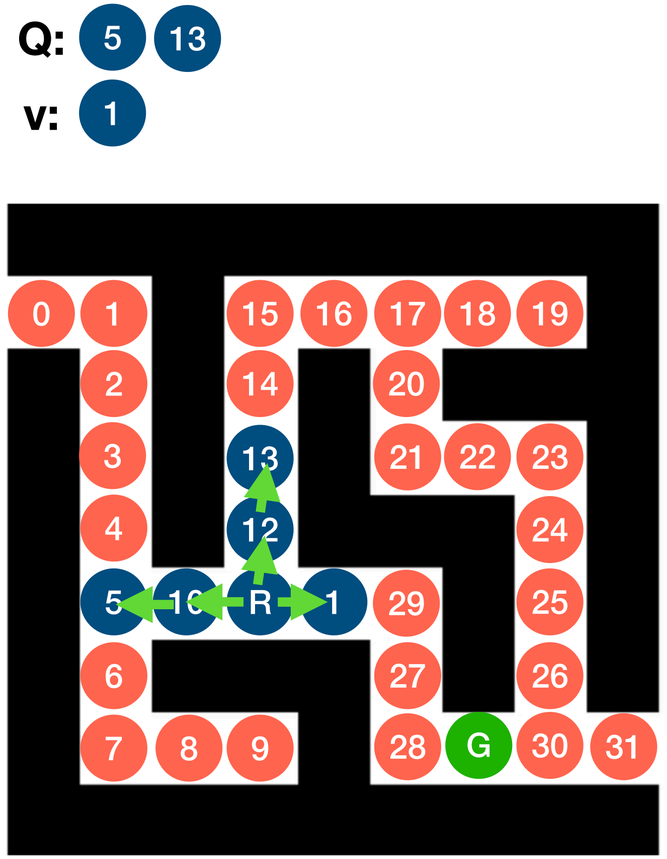
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

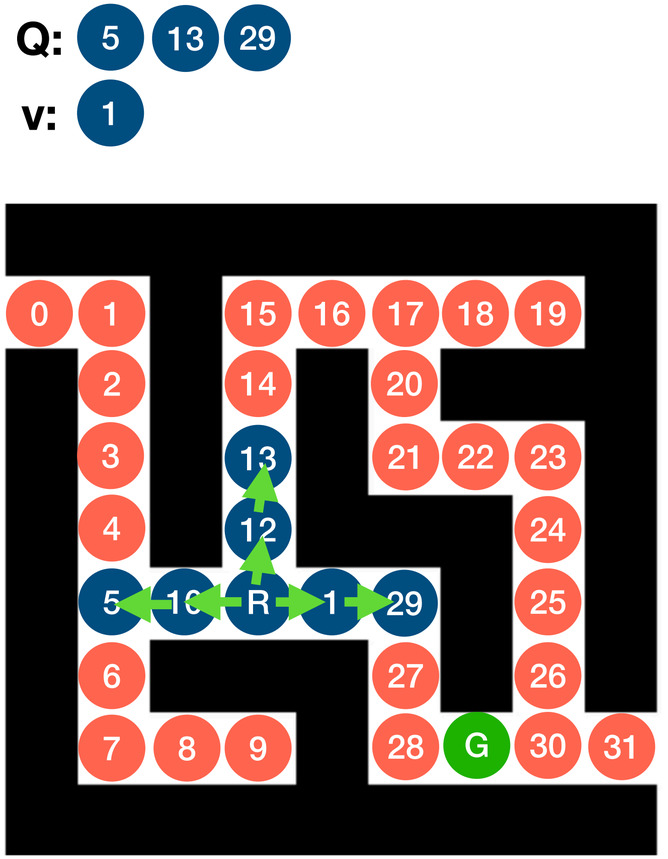
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

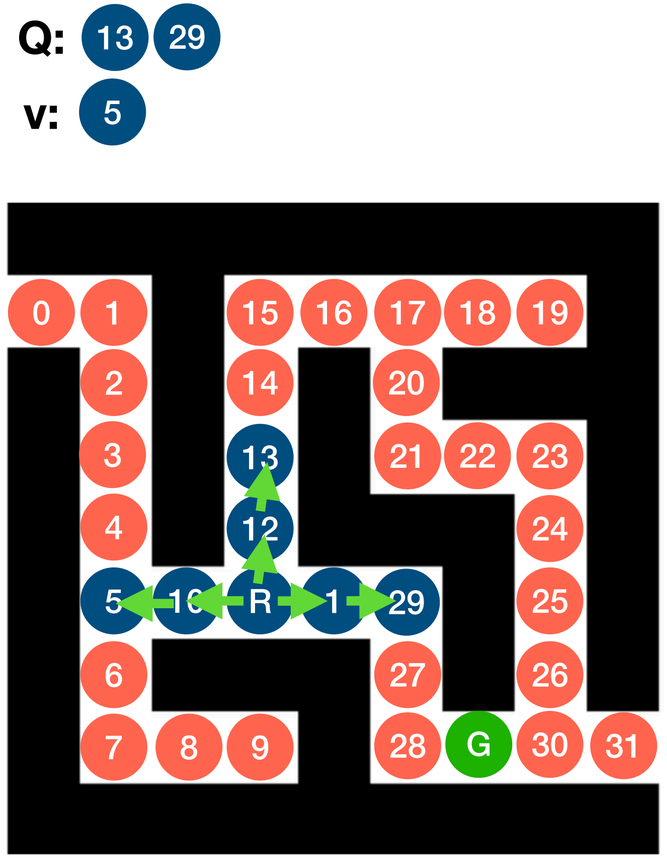
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

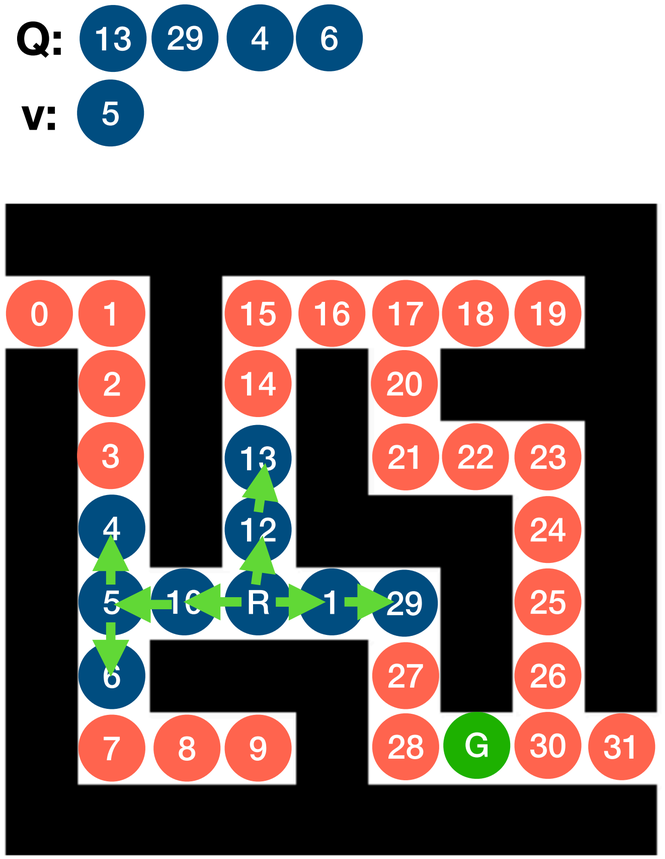
```
 1  procedure BFS(G, root) is
 2      let Q be a queue
 3      label root as explored
 4      Q.enqueue(root)
 5      while Q is not empty do
 6          v := Q.dequeue()
 7          if v is the goal then
 8              return v
 9          for all edges from v to w in G.adjacentEdges(v) do
10              if w is not labeled as explored then
11                  label w as explored
12                  w.parent := v
13                  Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

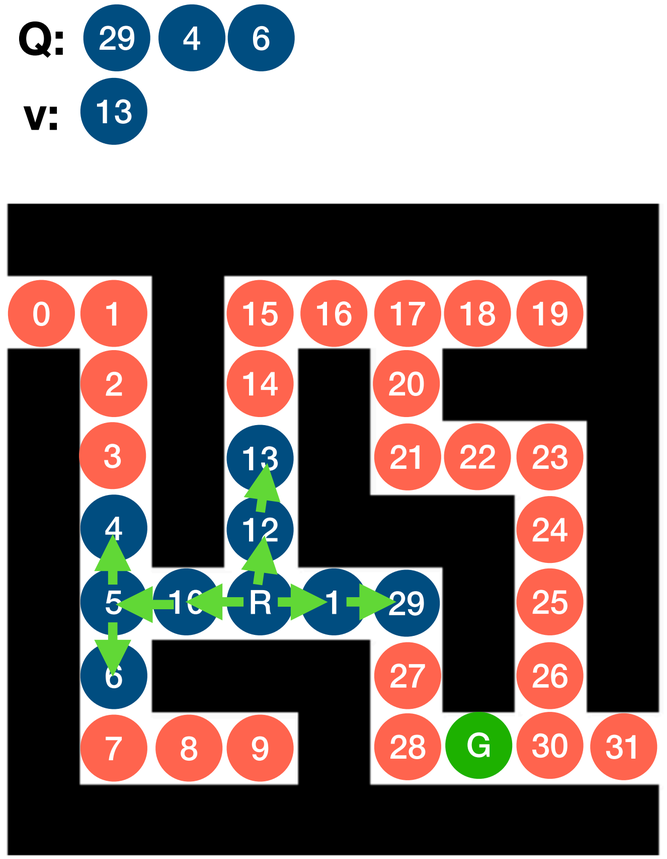
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

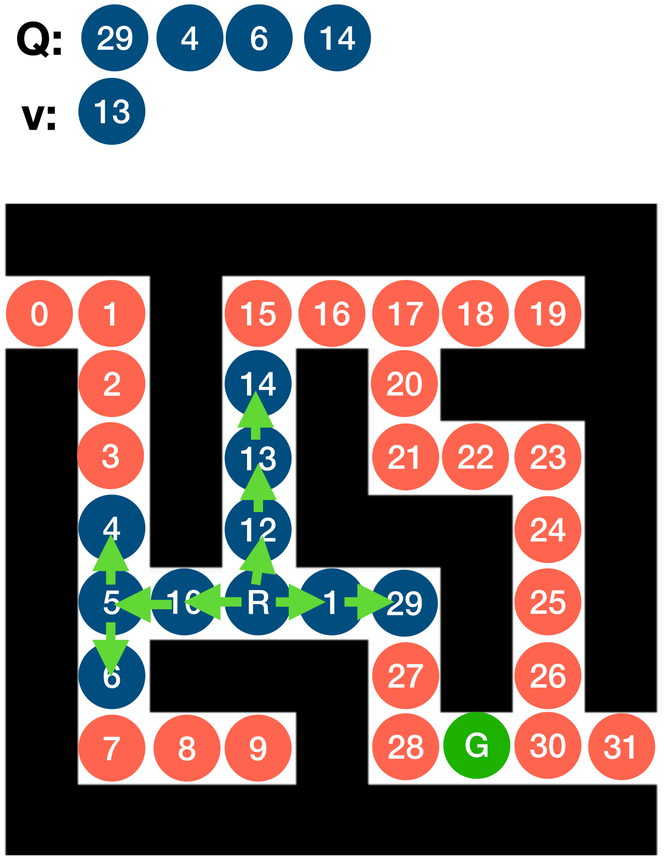
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

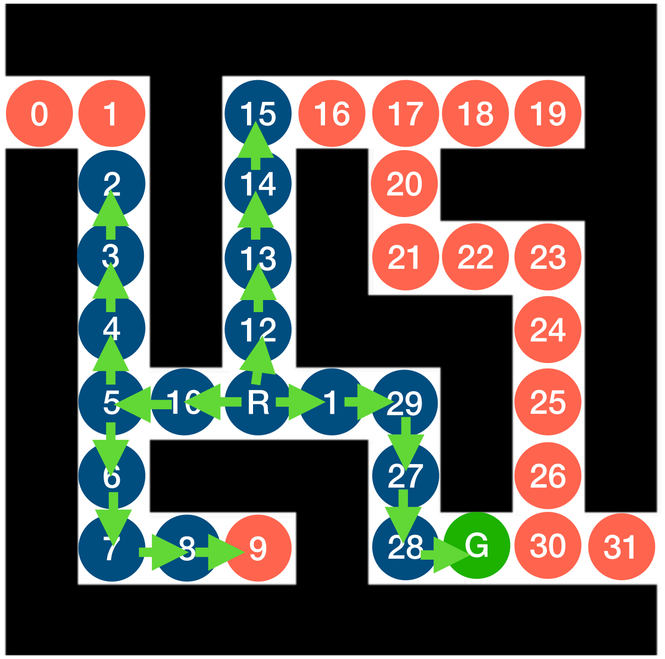
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

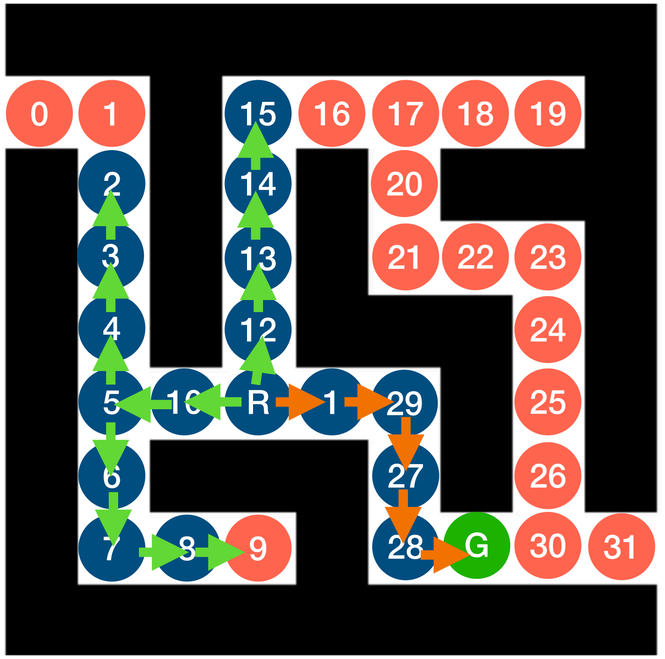
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

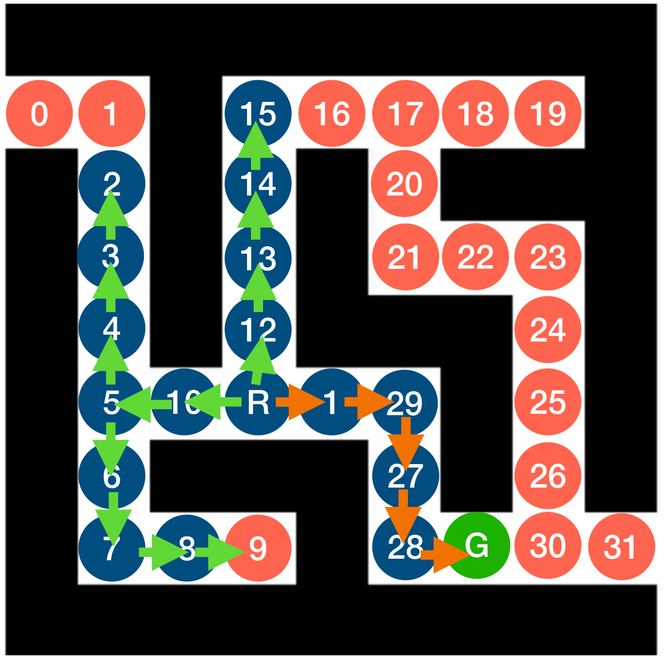
```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



Seach-Algorithm

Breadth First Search (BFS)

```
procedure BFS(G, root) is
 2       let Q be a queue
 3       label root as explored
 4       Q.enqueue(root)
 5       while Q is not empty do
 6           v := Q.dequeue()
 7           if v is the goal then
 8               return v
 9           for all edges from v to w in G.adjacentEdges(v) do
10               if w is not labeled as explored then
11                   label w as explored
12                   w.parent := v
13                   Q.enqueue(w)
```



BFS requires **Discrete** state space!

Mathematic Definition of the Problem

C : Configuration space

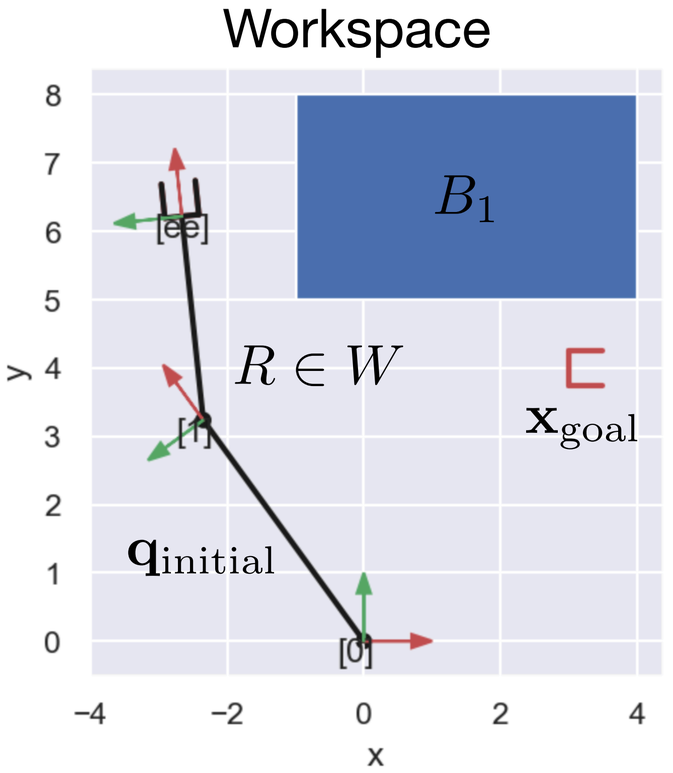
W : Workspace (end effector space)

$R \in W$: Robot

$B_1, B_2, .. \in W$: Obstacles

$\mathbf{q}_{\text{initial}}$: Robot's start configuration

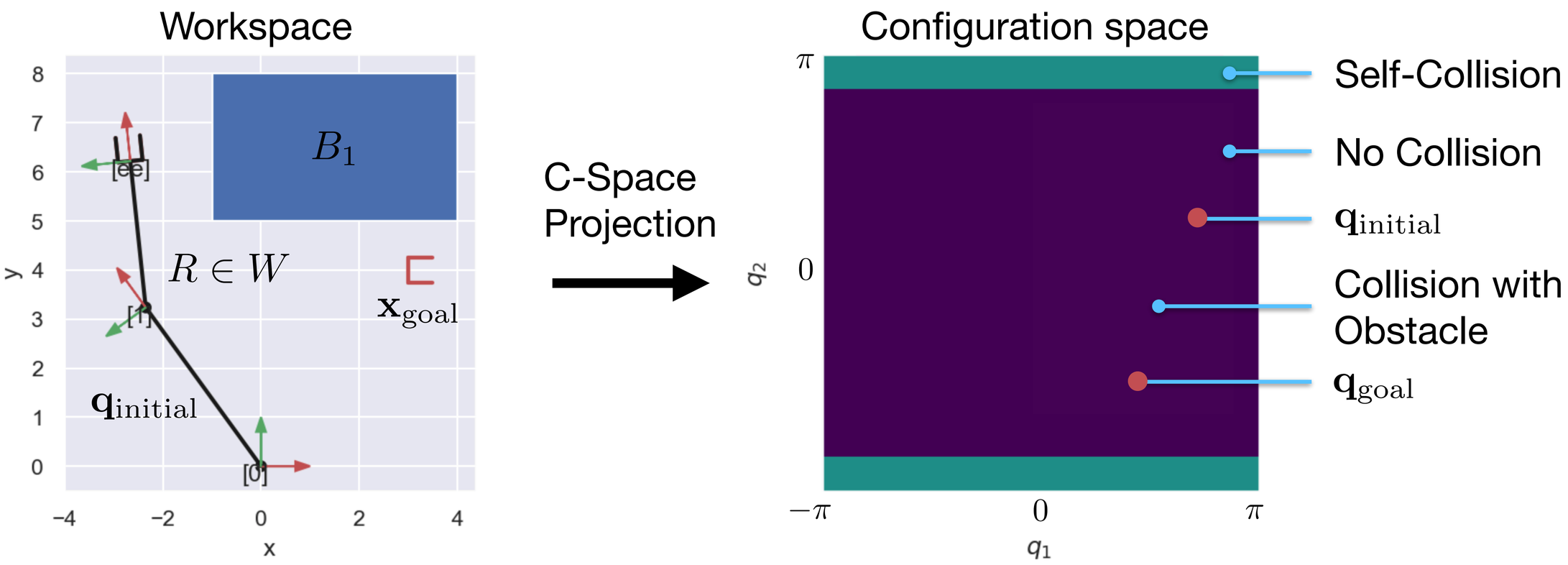
$\mathbf{q}_{\text{goal}}$, $\mathbf{x}_{\text{goal}}$: Goal configuration / goal ee-pose



Find a path π so that R moves from $\mathbf{q}_{\text{initial}}$ to either $\mathbf{q}_{\text{goal}}$ or $\mathbf{x}_{\text{goal}}$ without intersecting with any B_i or colliding with itself.

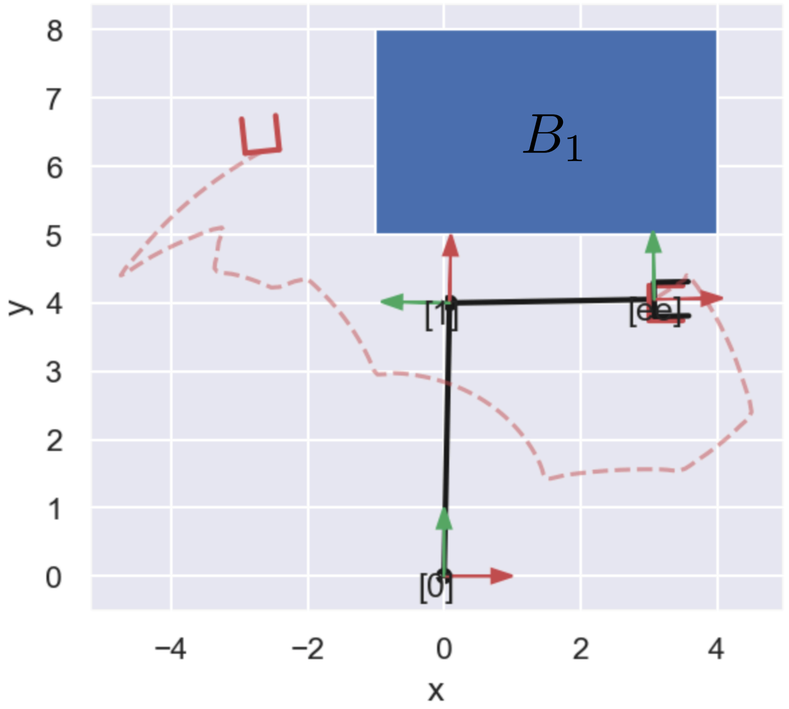
We have a **continuous** state space!

The World in Configuration Space



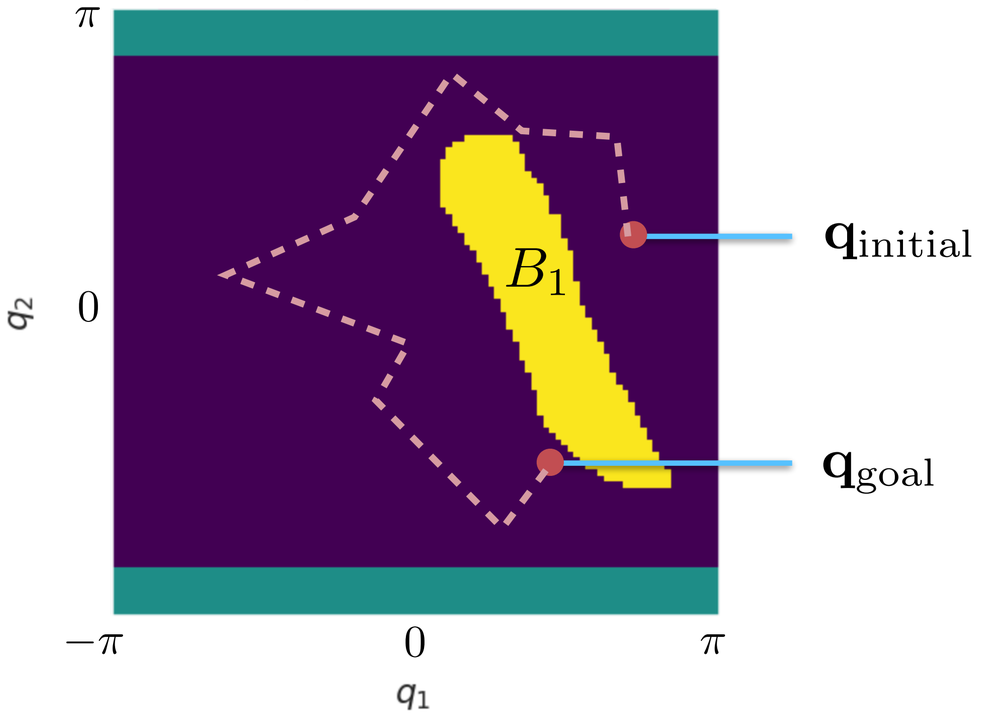
General Approach:
Path Planning in Configuration Space

Workspace



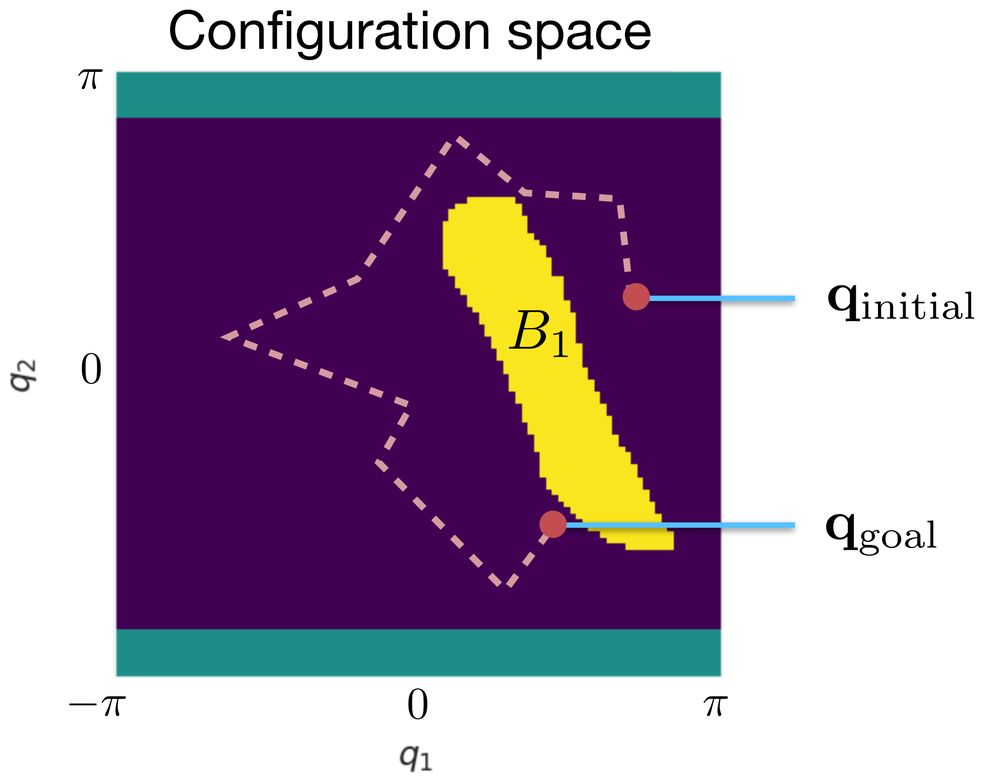
OP-Space Projection

Configuration space



This is what we want to solve

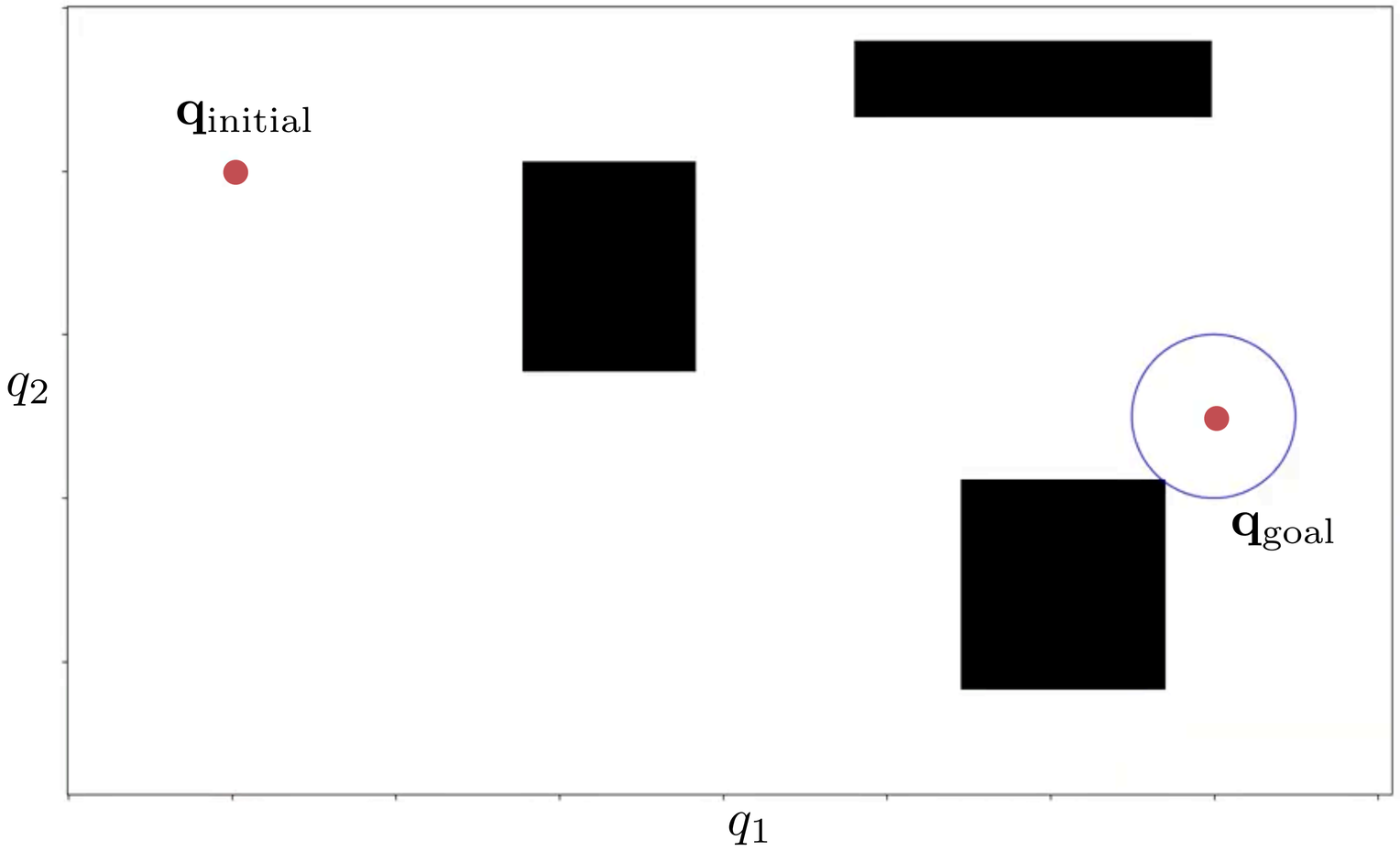
Problems with
Path Planning in Configuration Space



This is what we want to solve

Rapidly Exploring Random Trees (RRT)

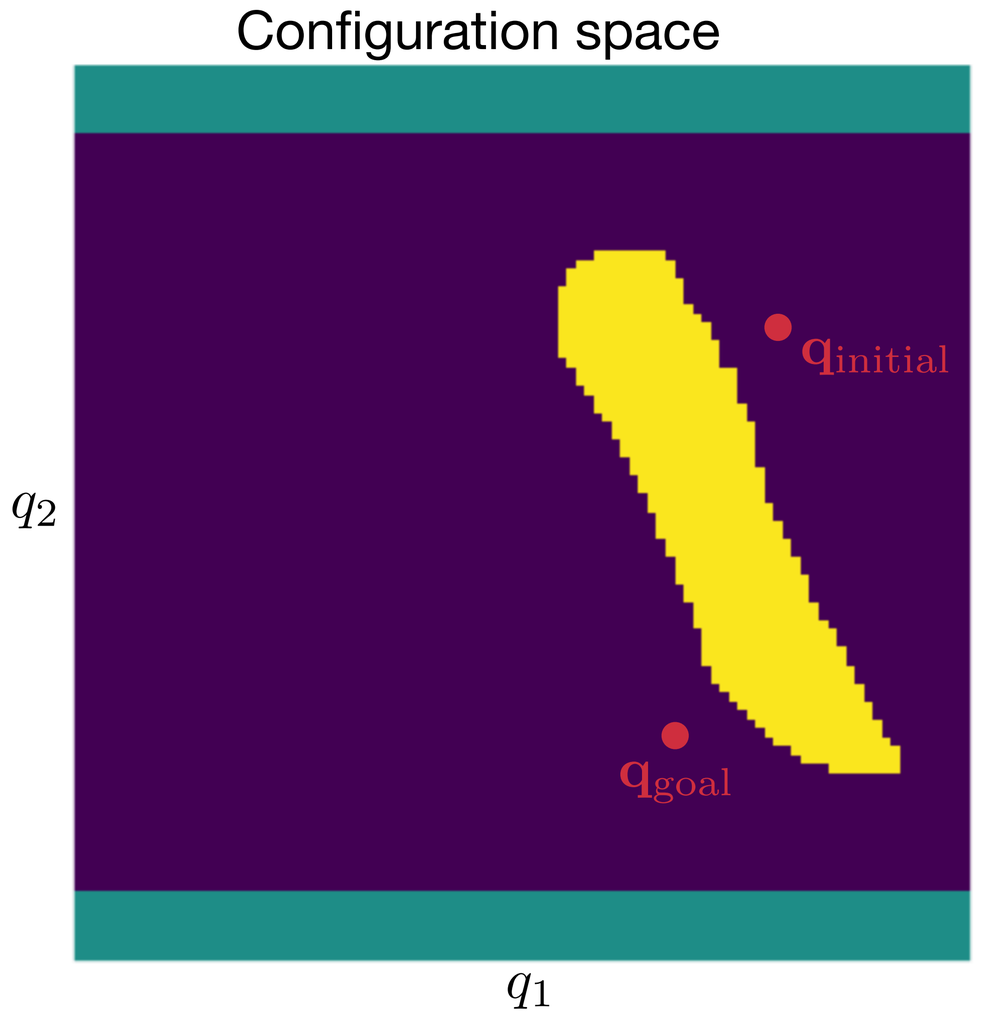
Rapidly Exploring Random Trees (RRT)



(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

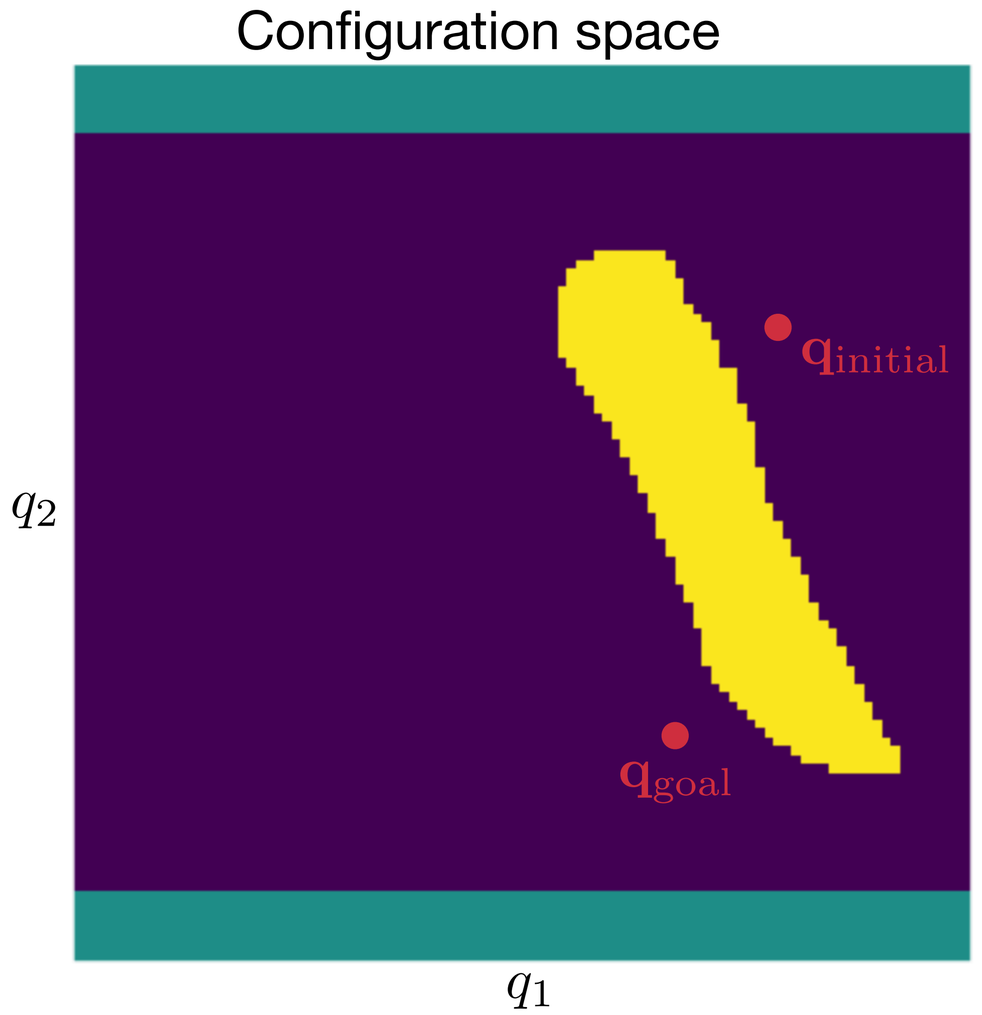


(Let’s assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:



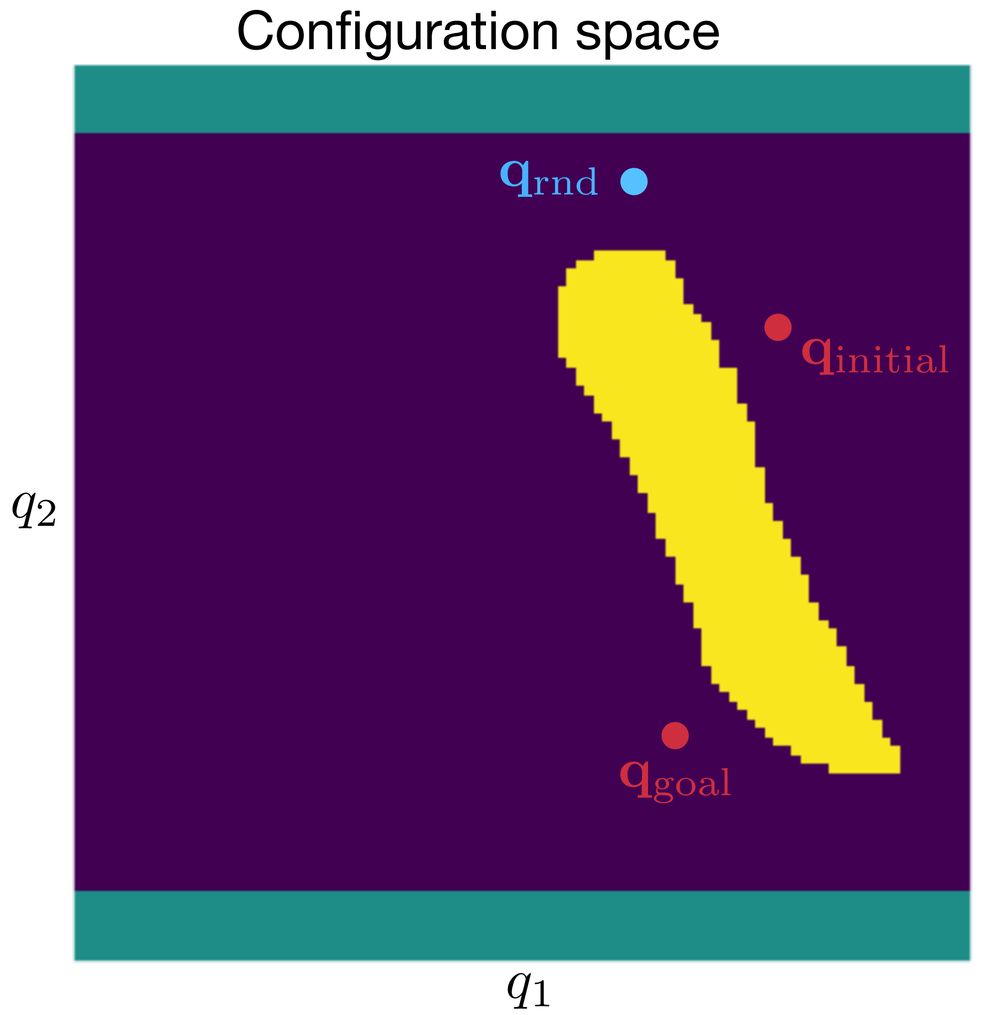
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$



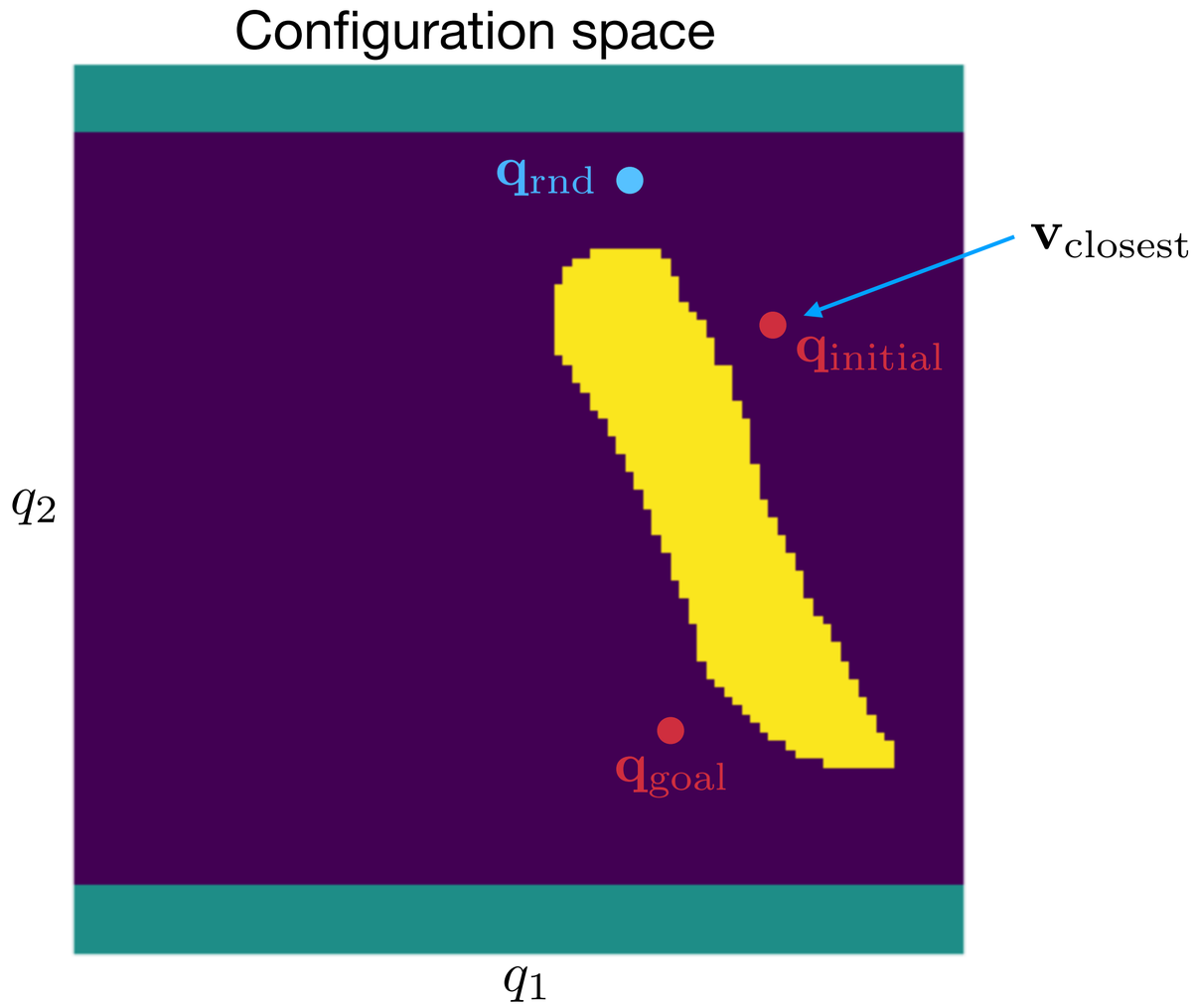
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree



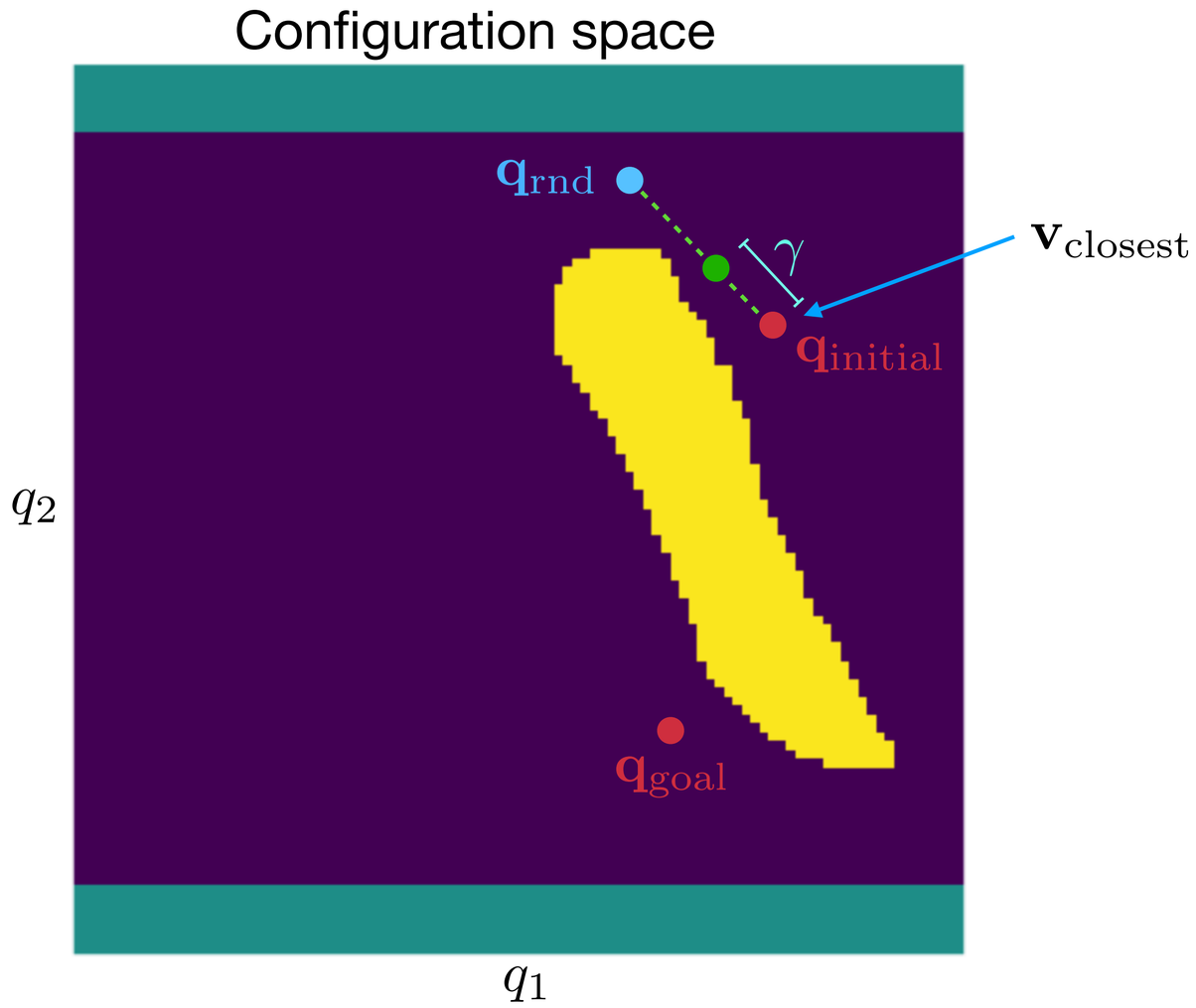
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ



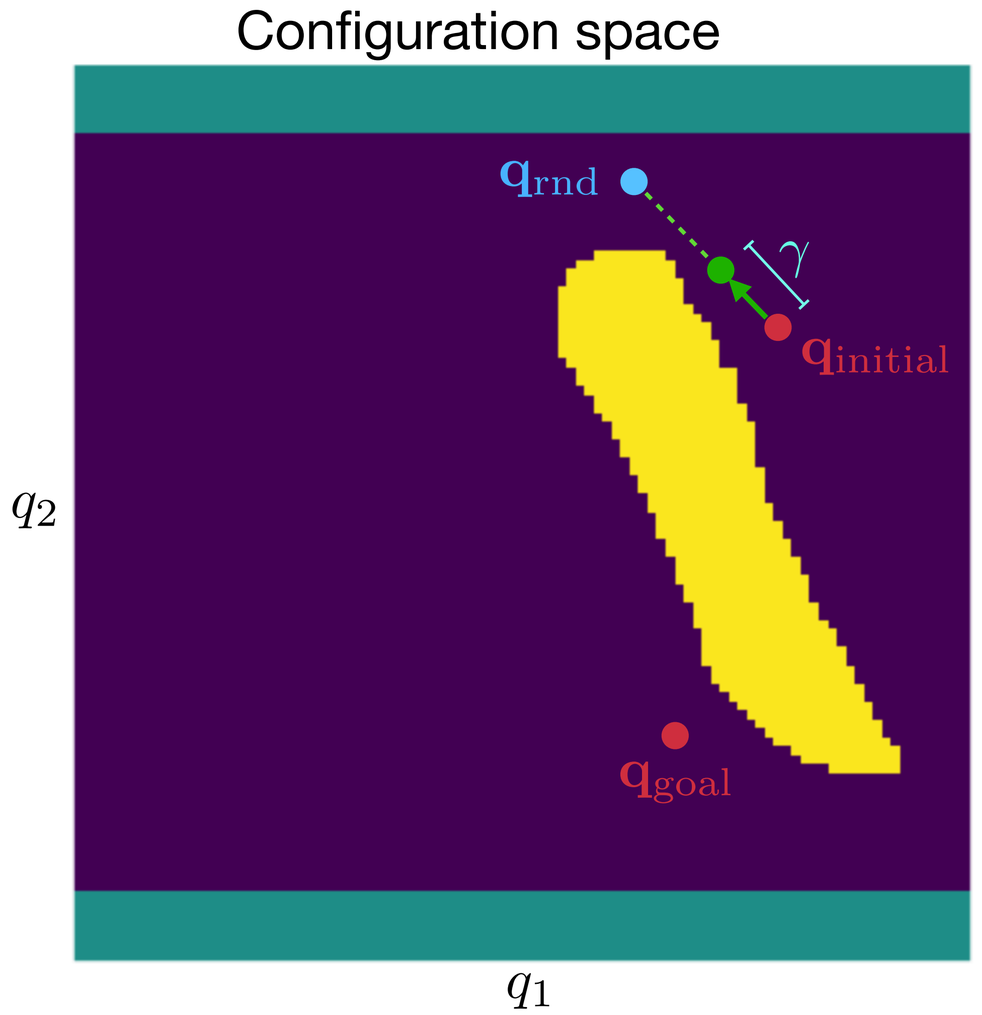
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree



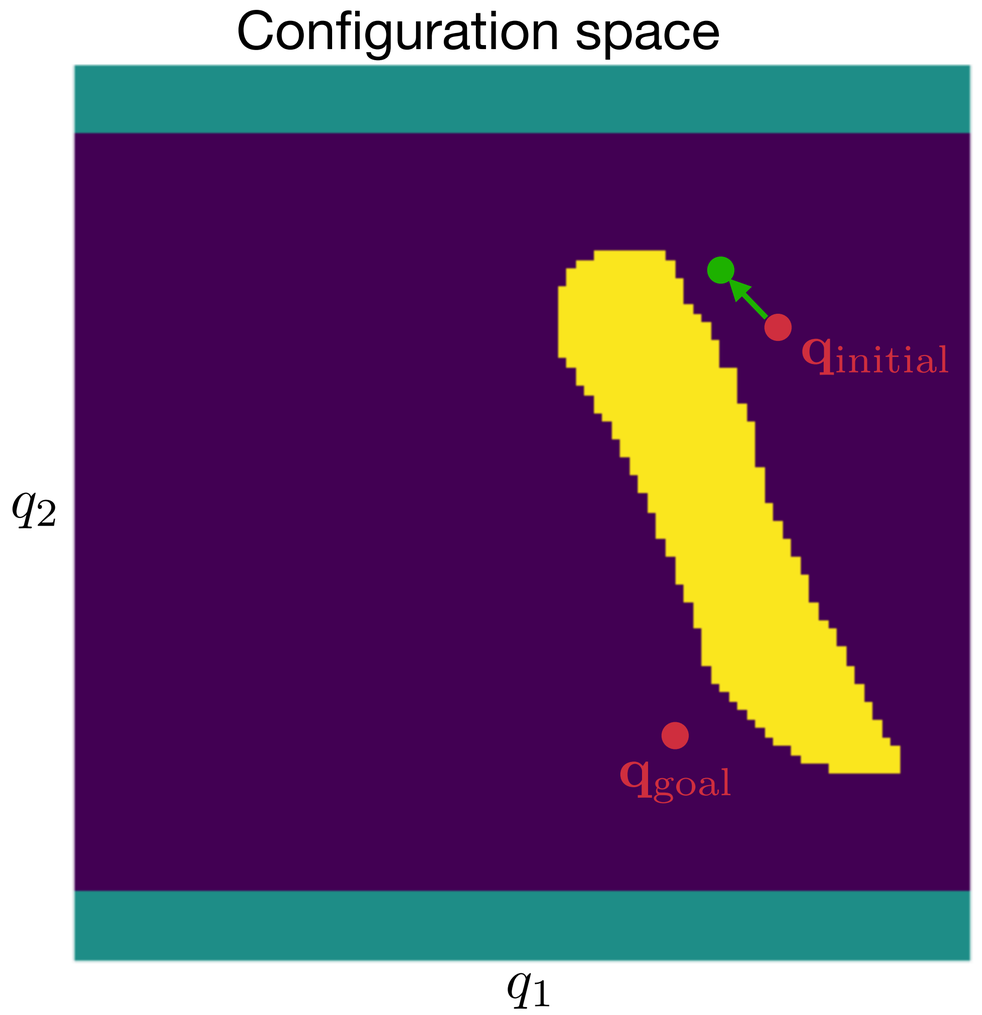
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



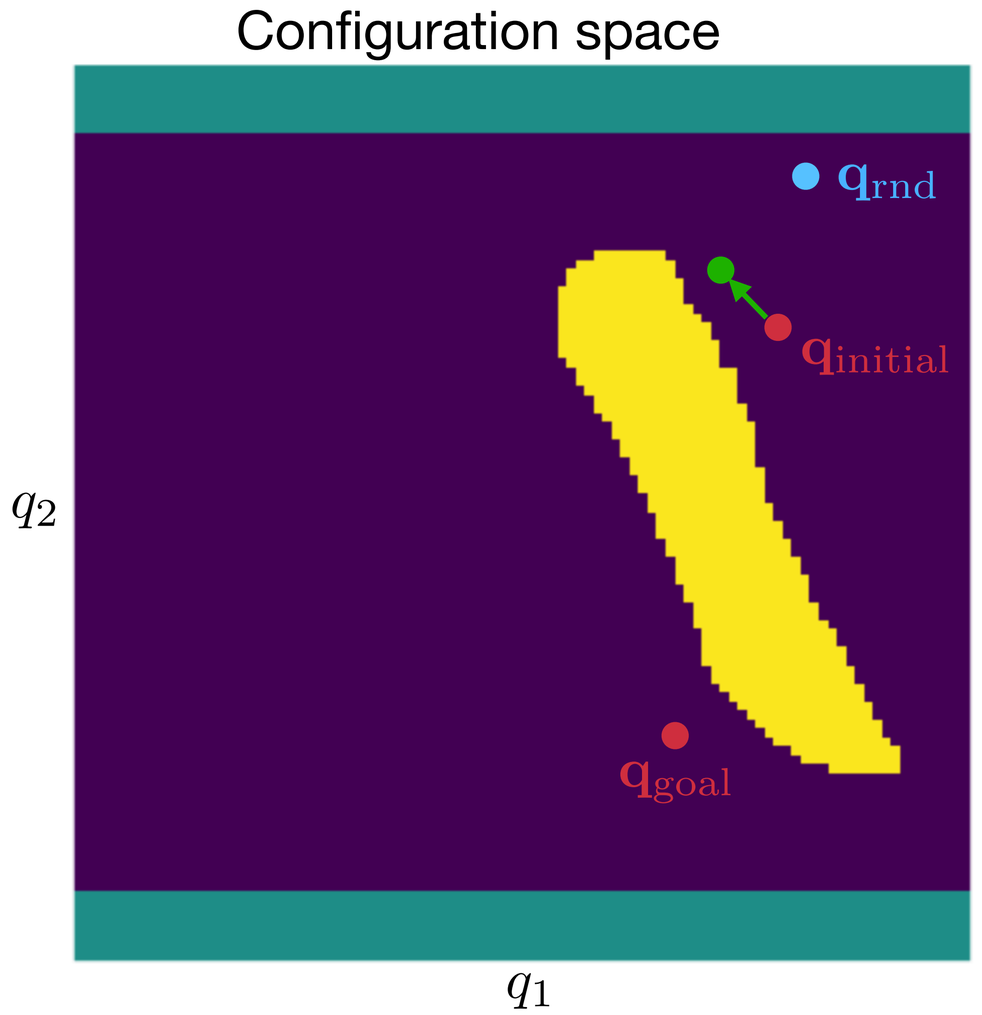
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



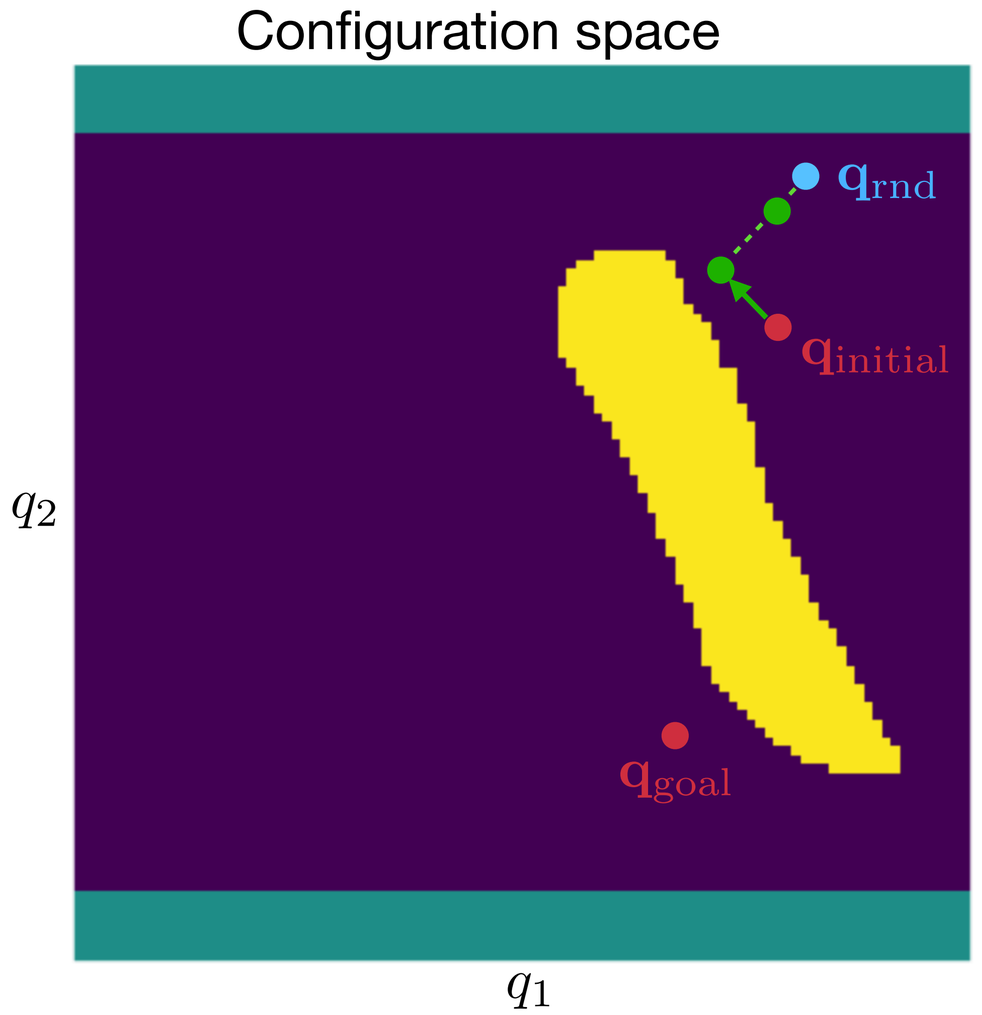
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



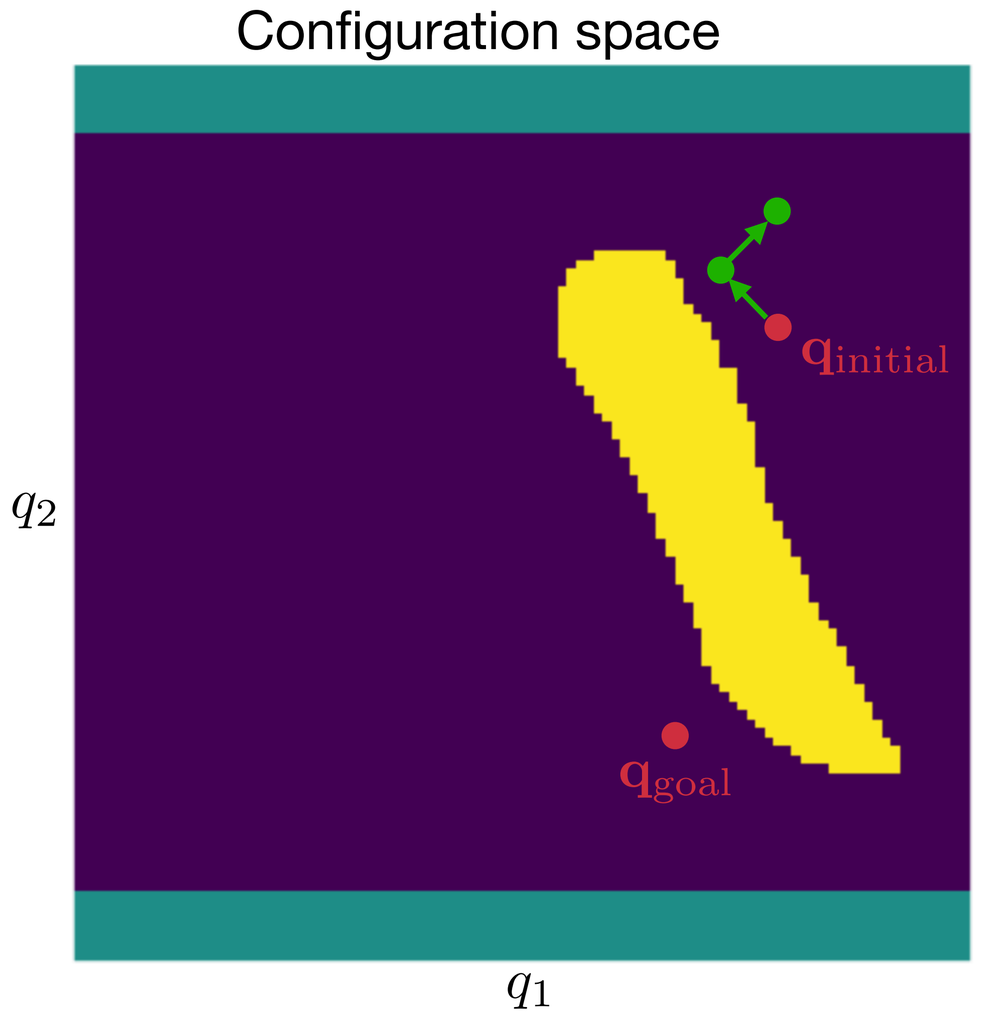
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



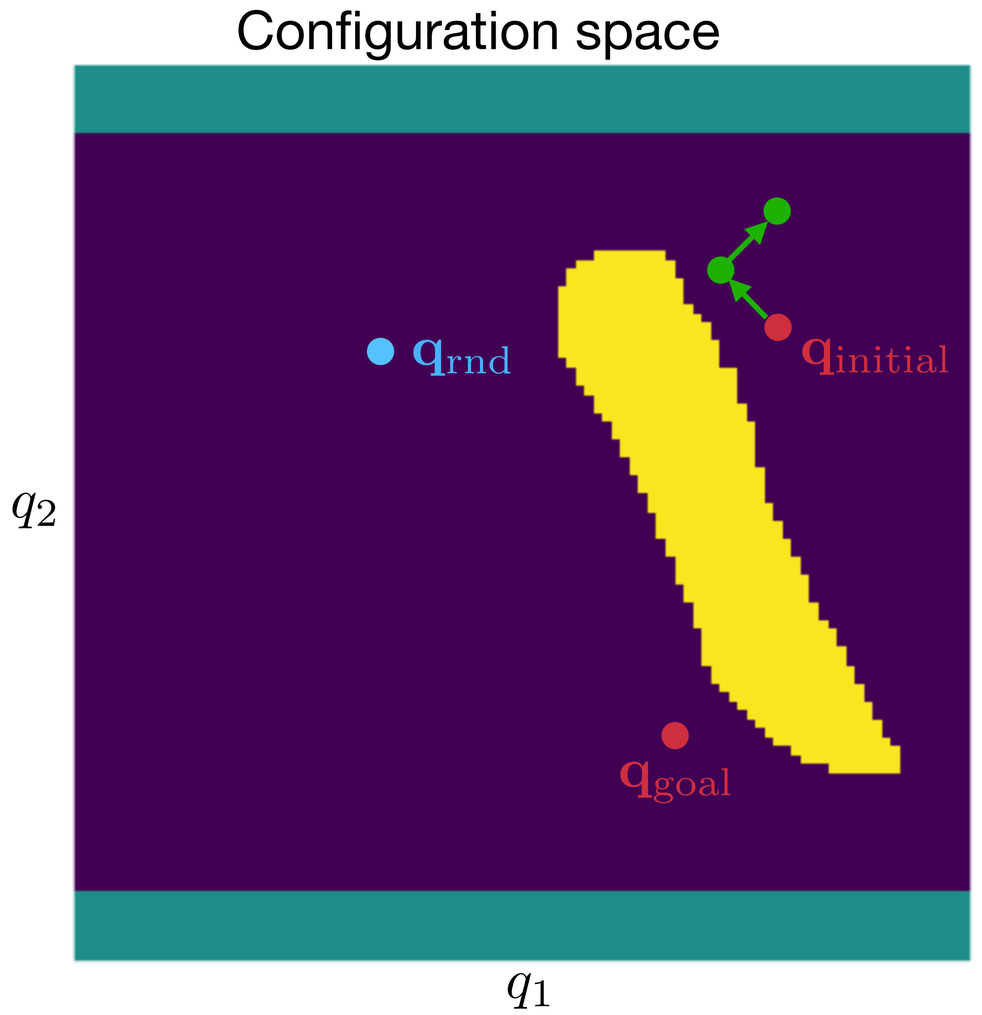
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



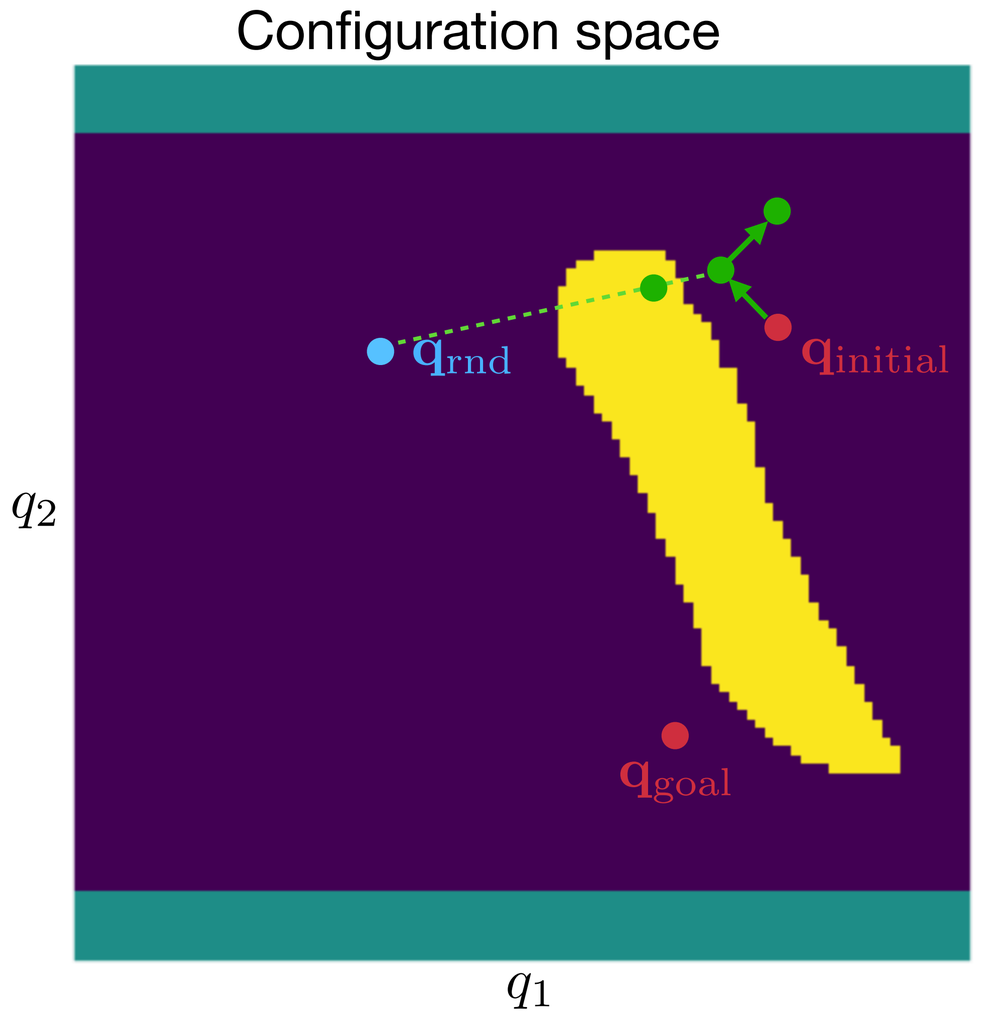
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



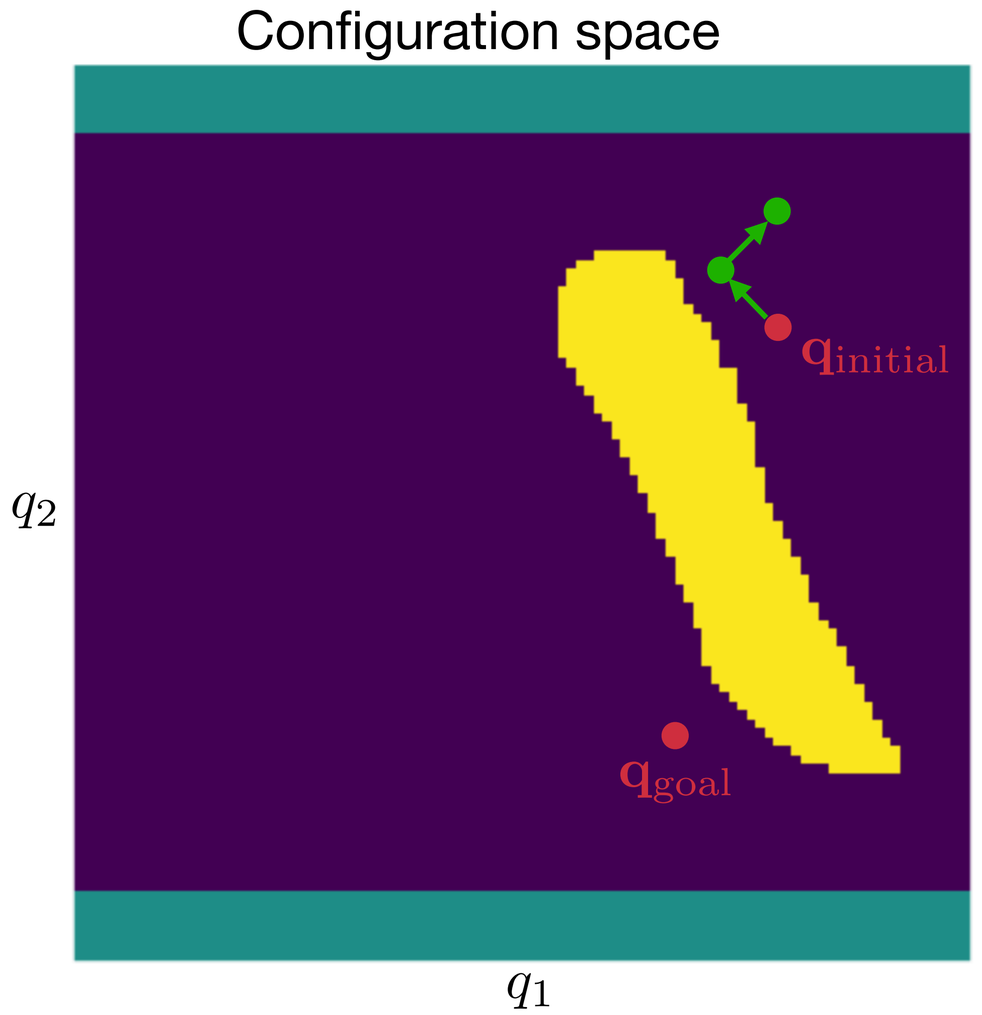
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



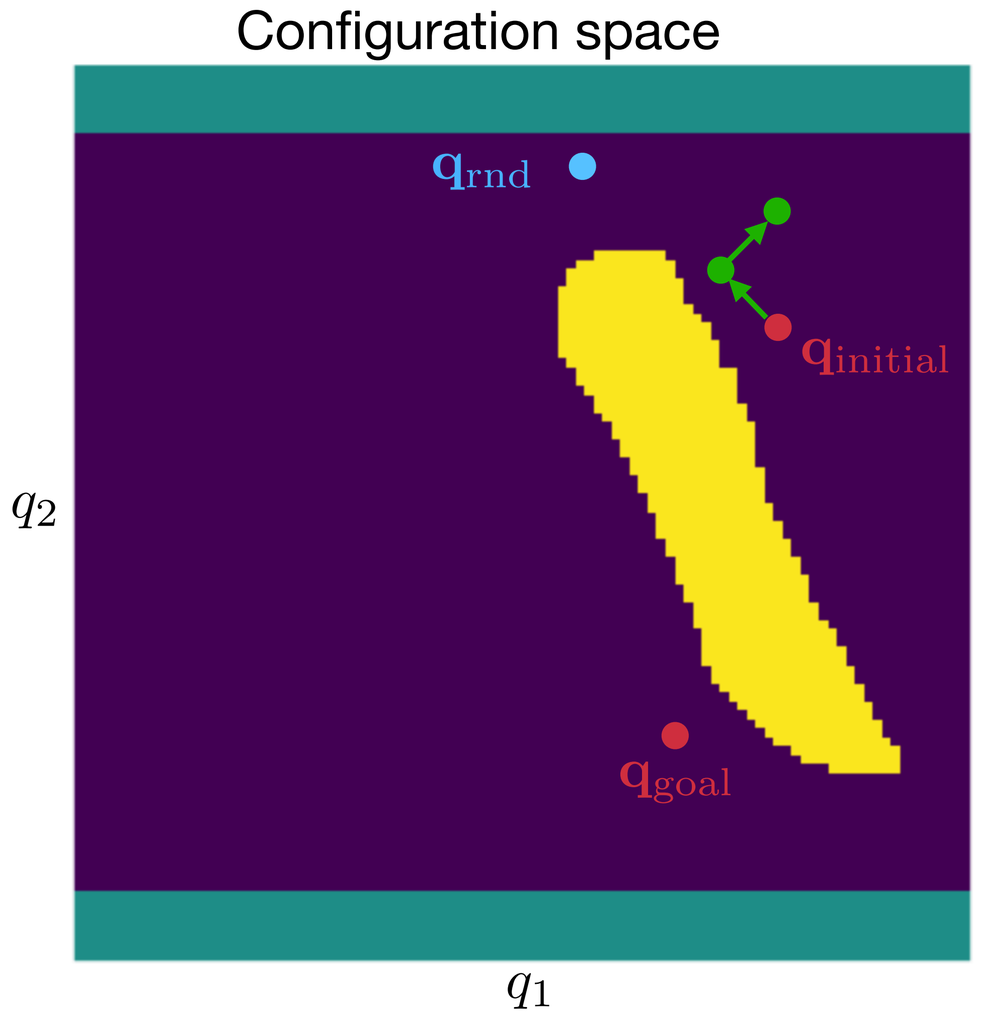
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



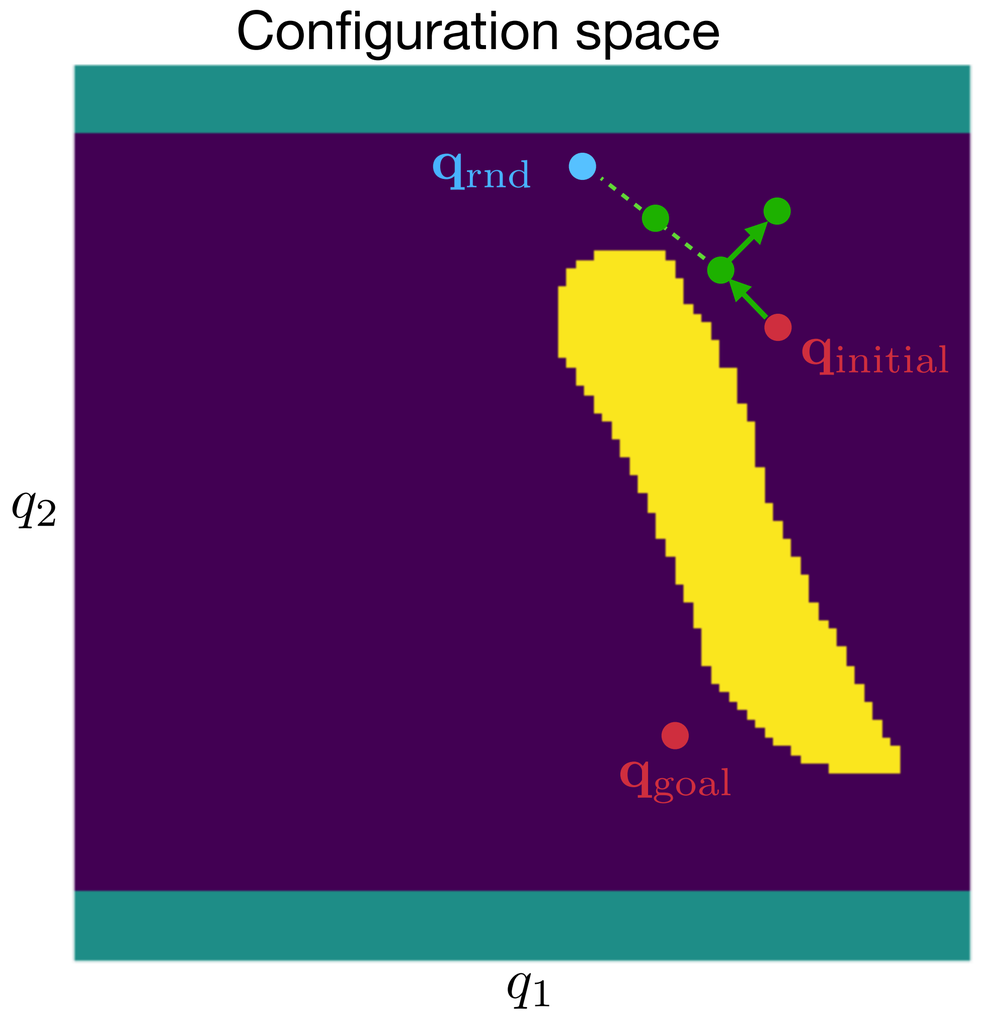
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



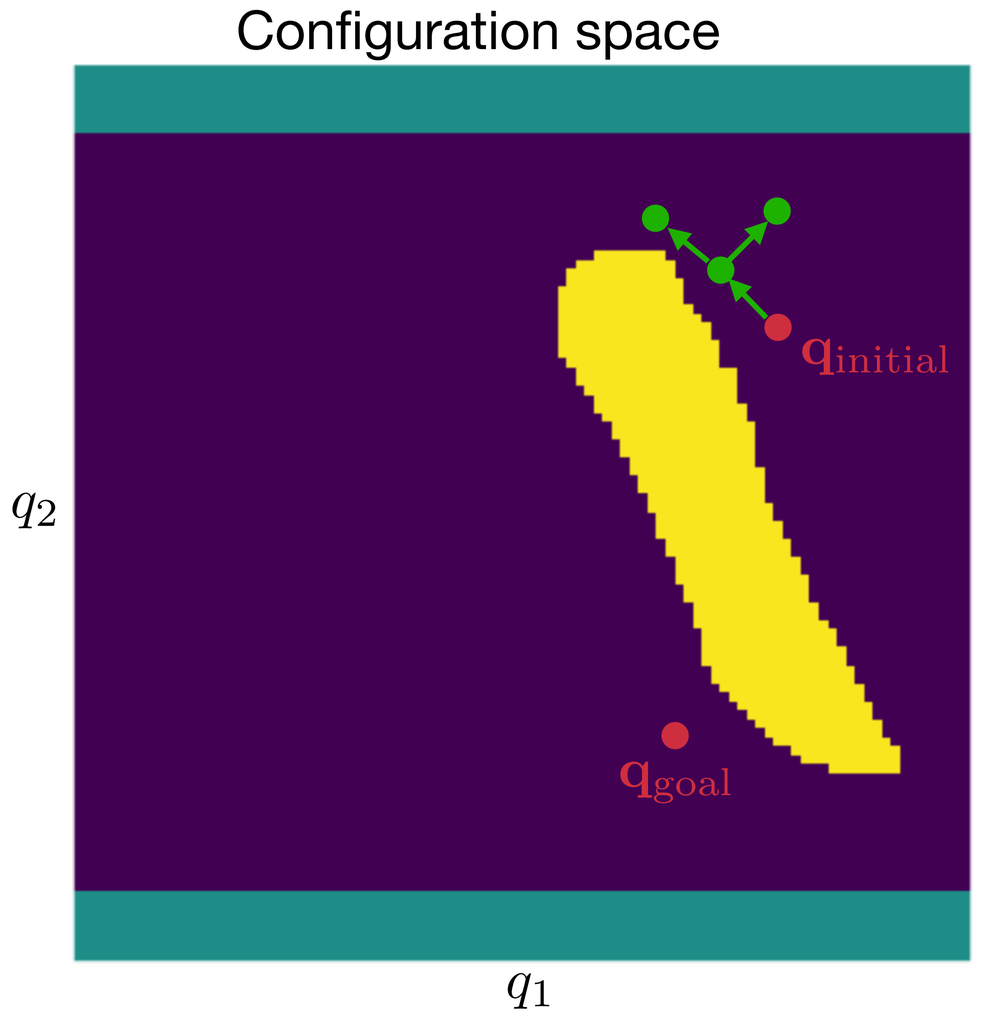
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



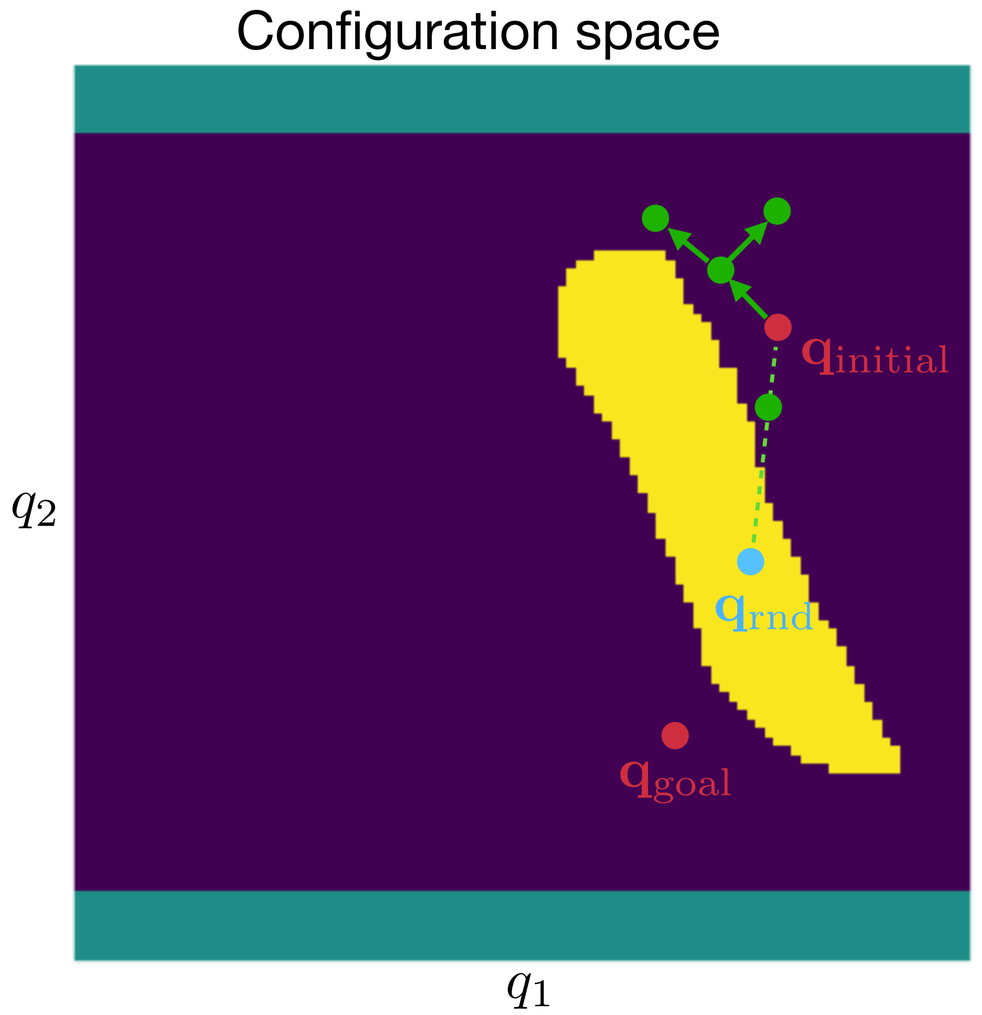
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



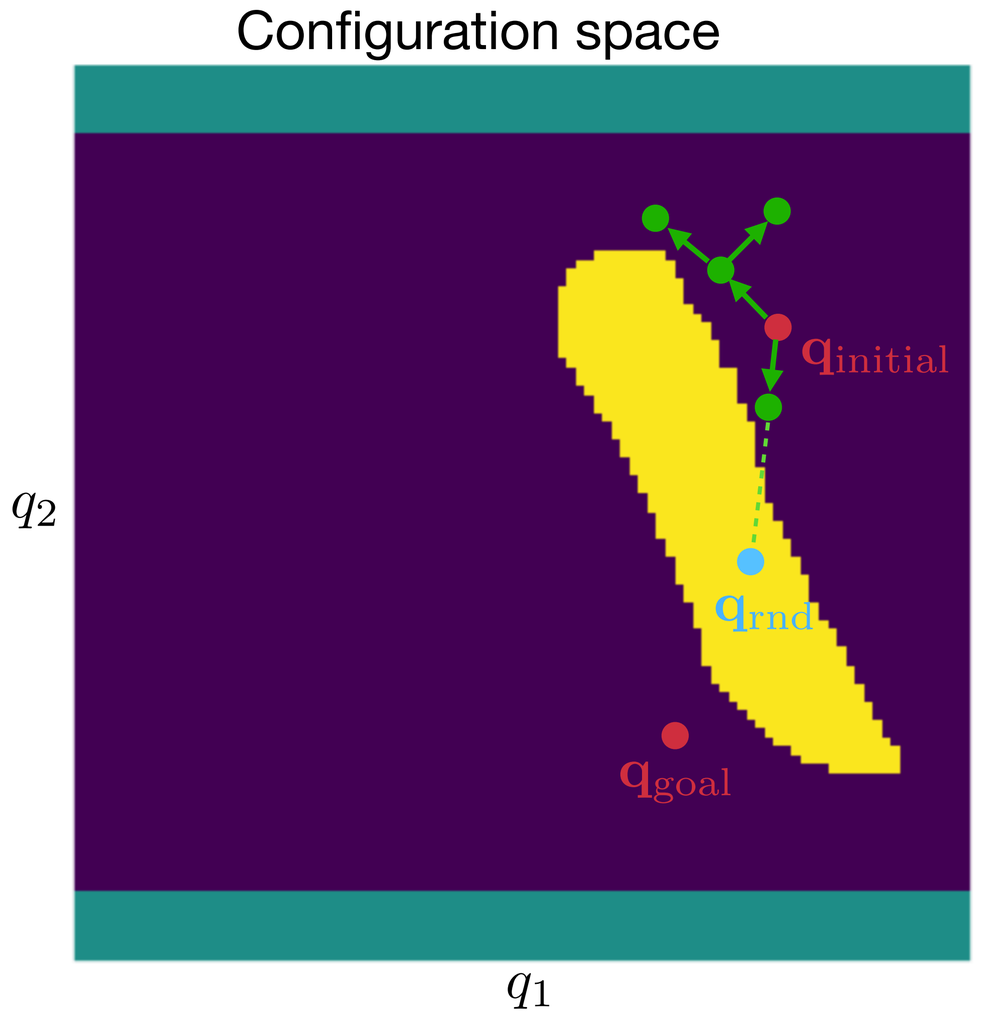
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



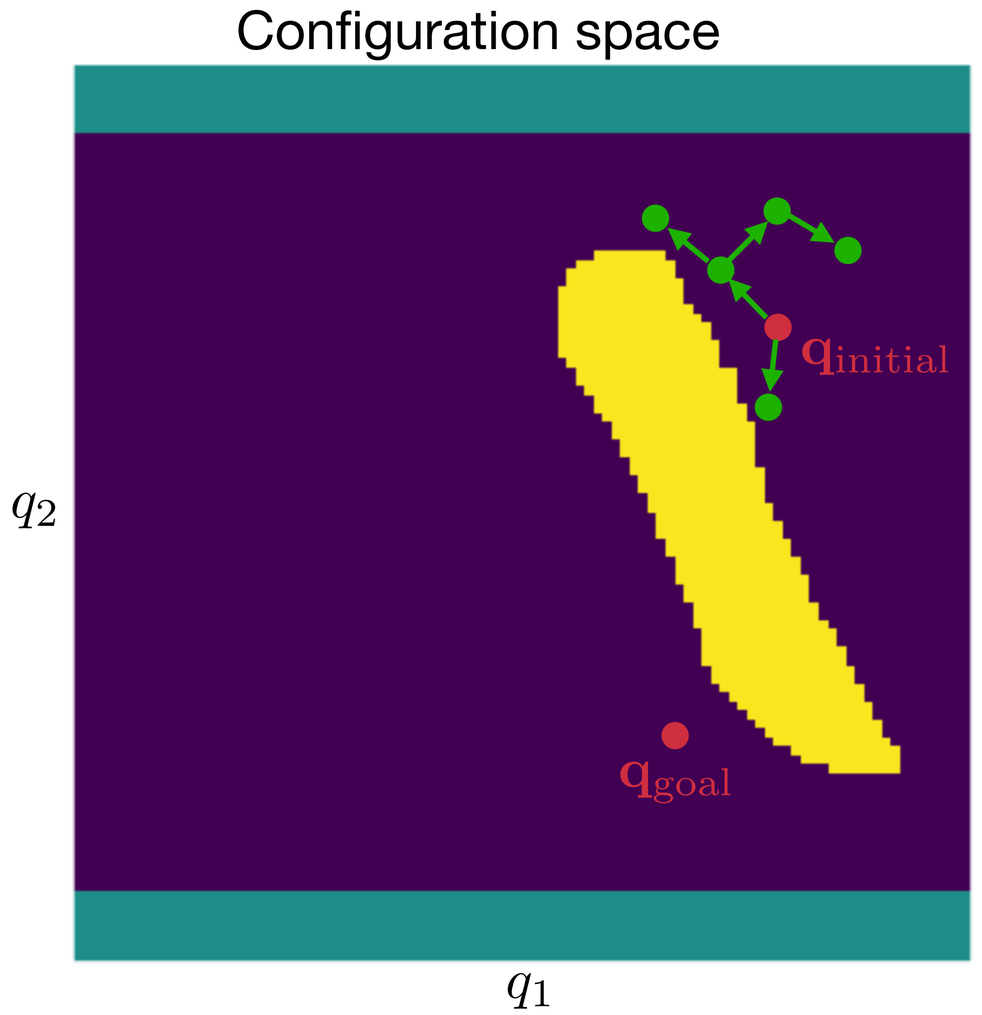
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



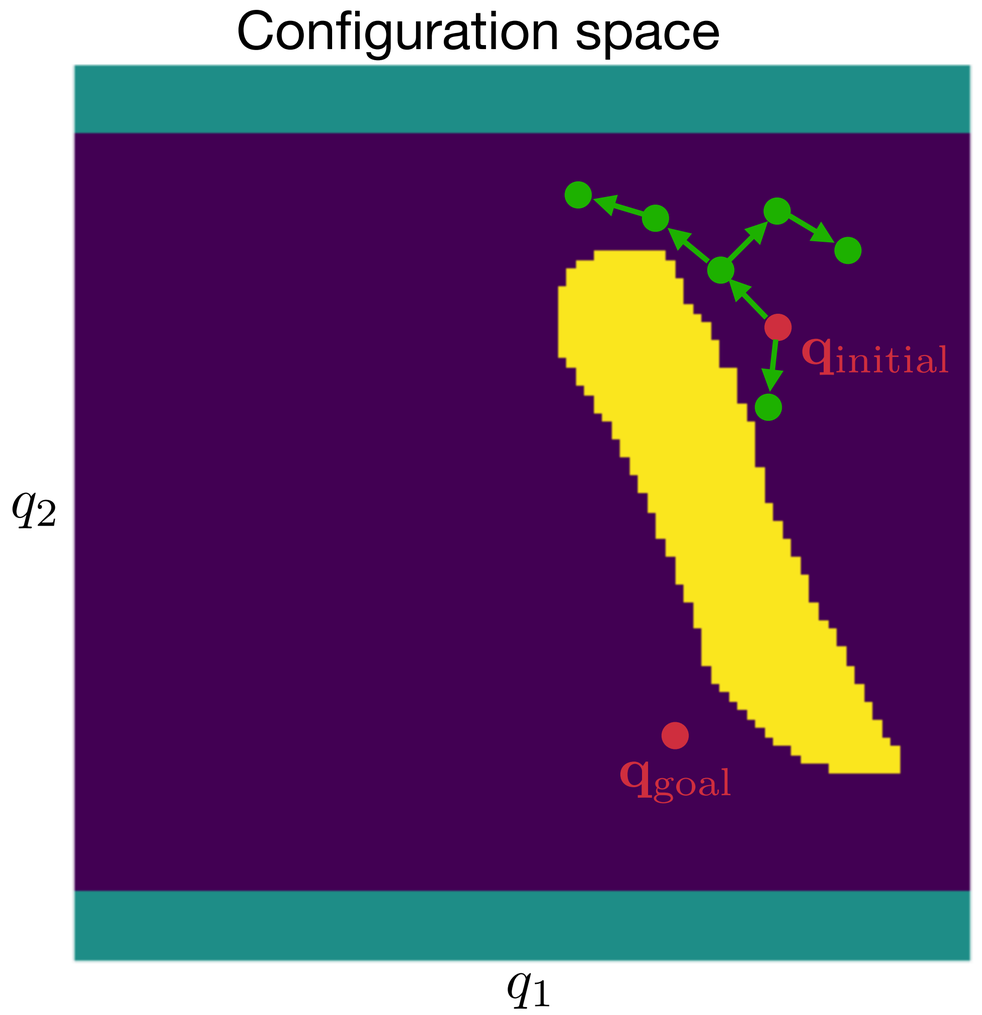
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



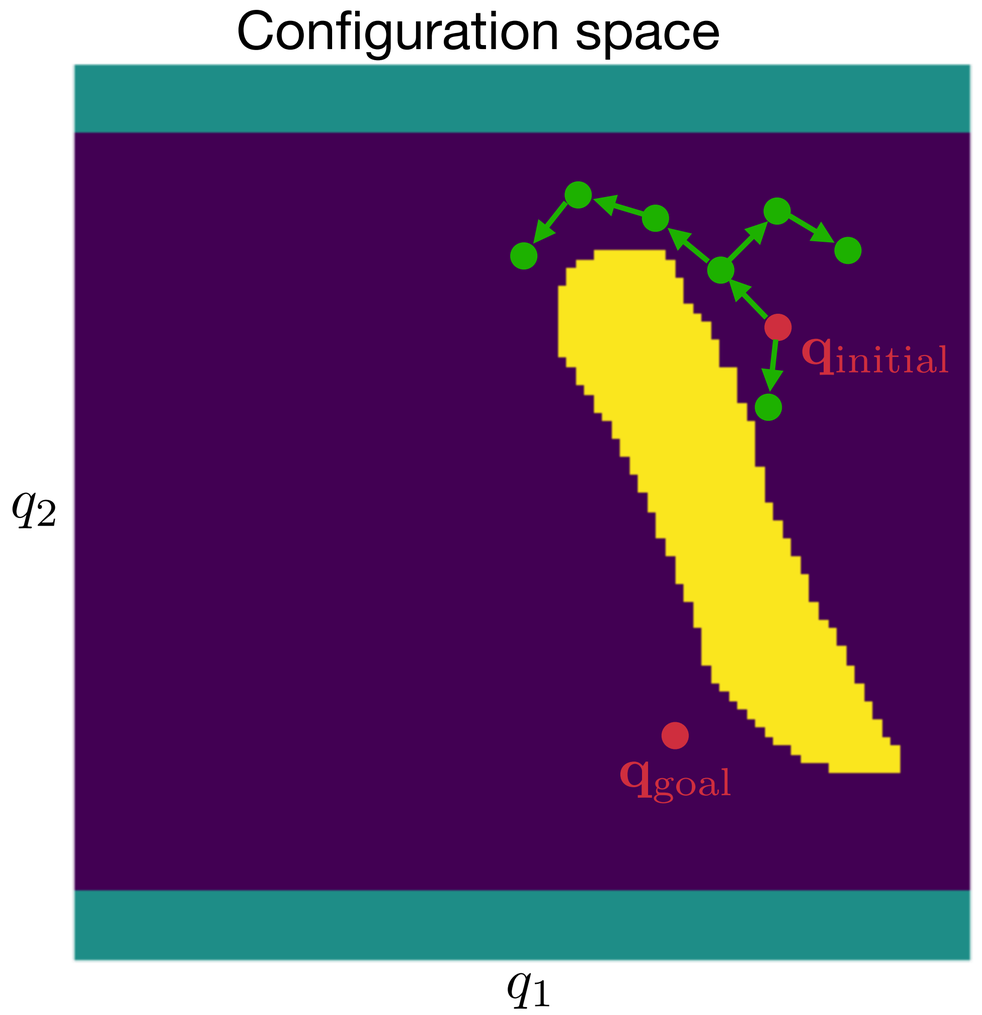
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



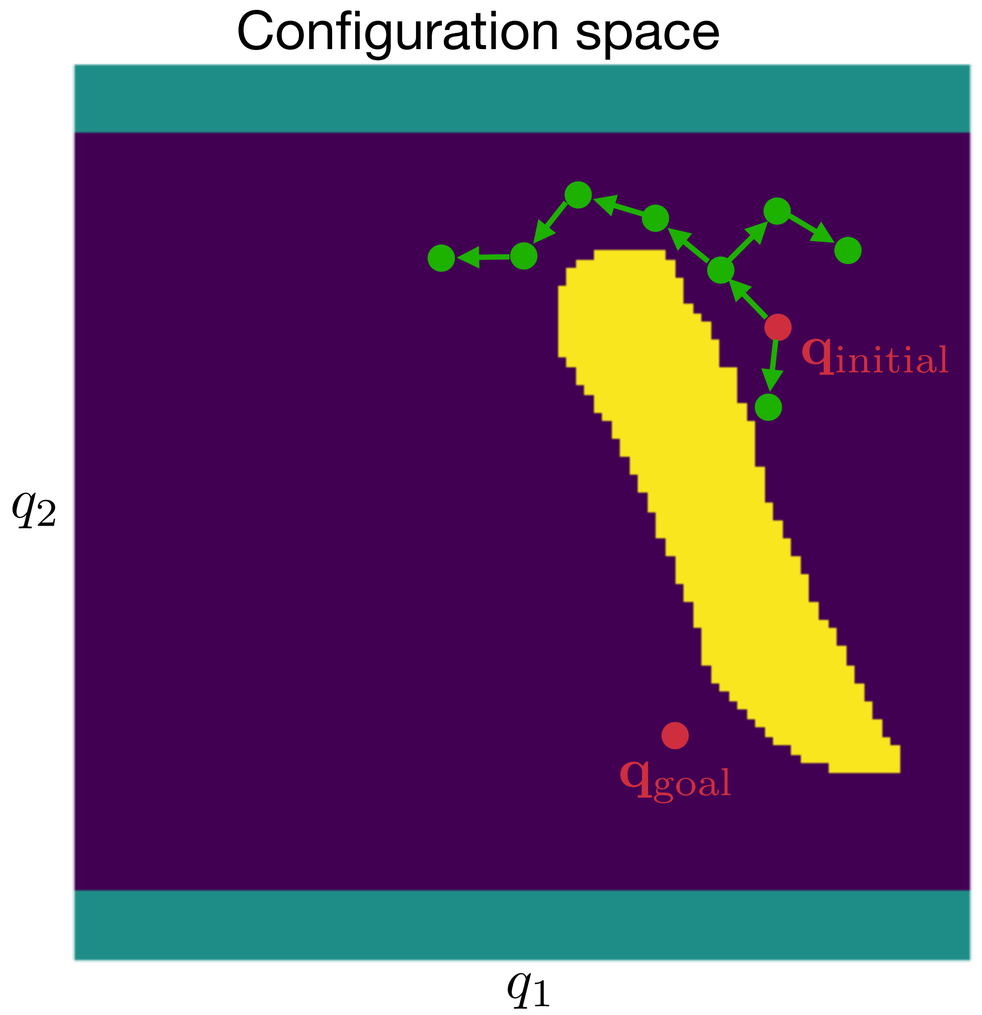
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



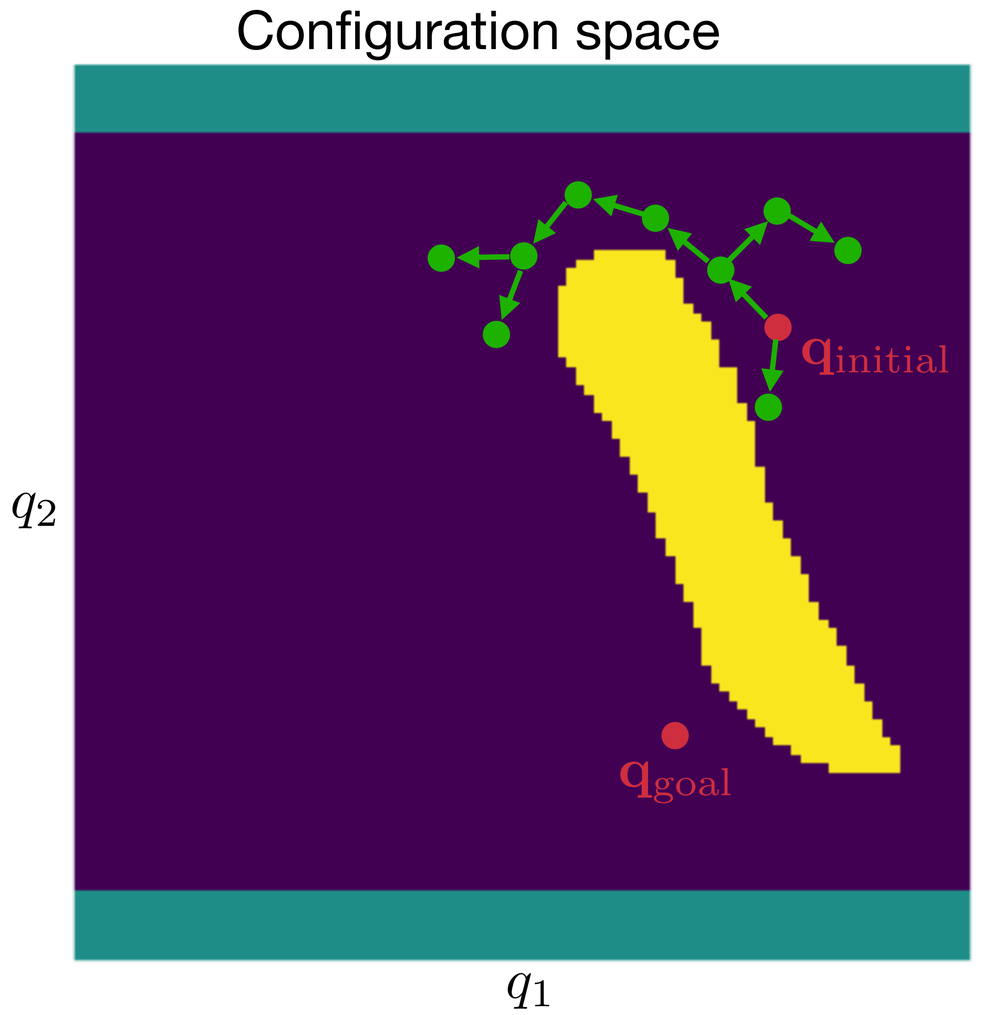
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



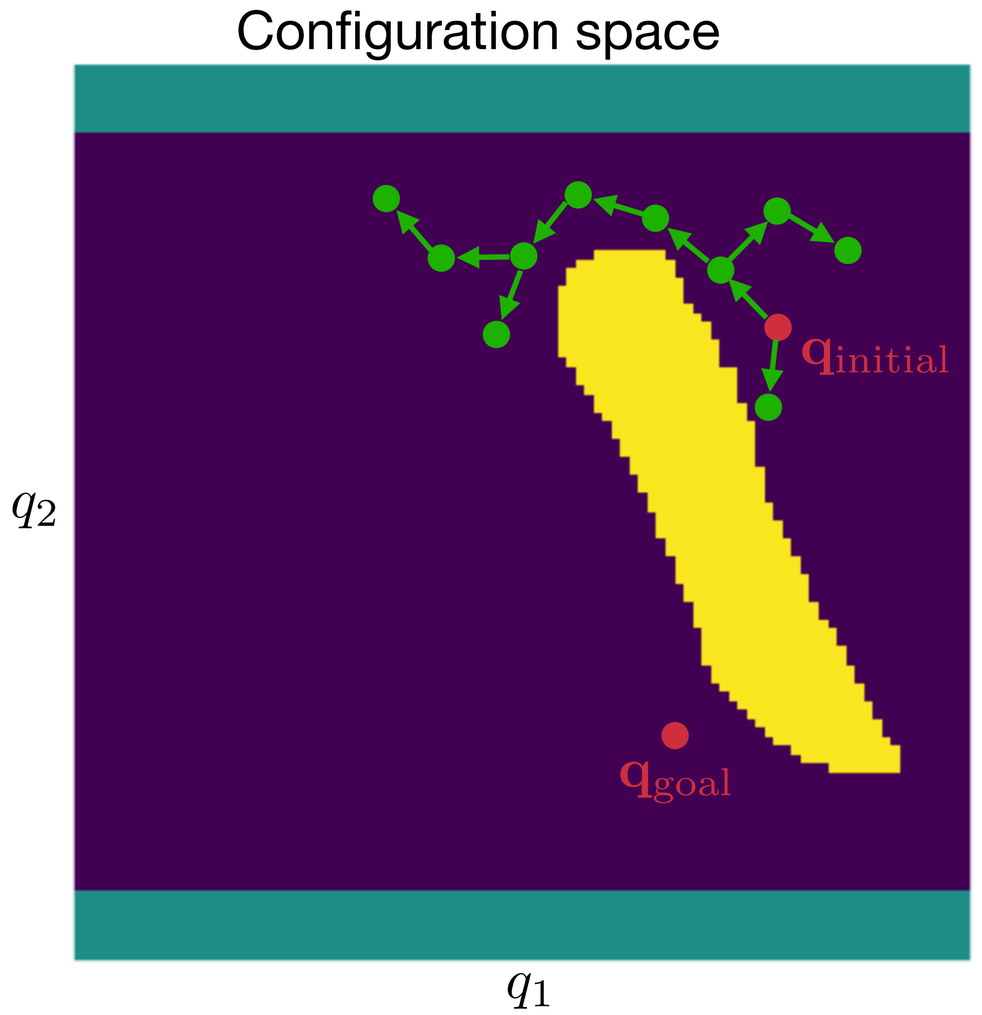
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



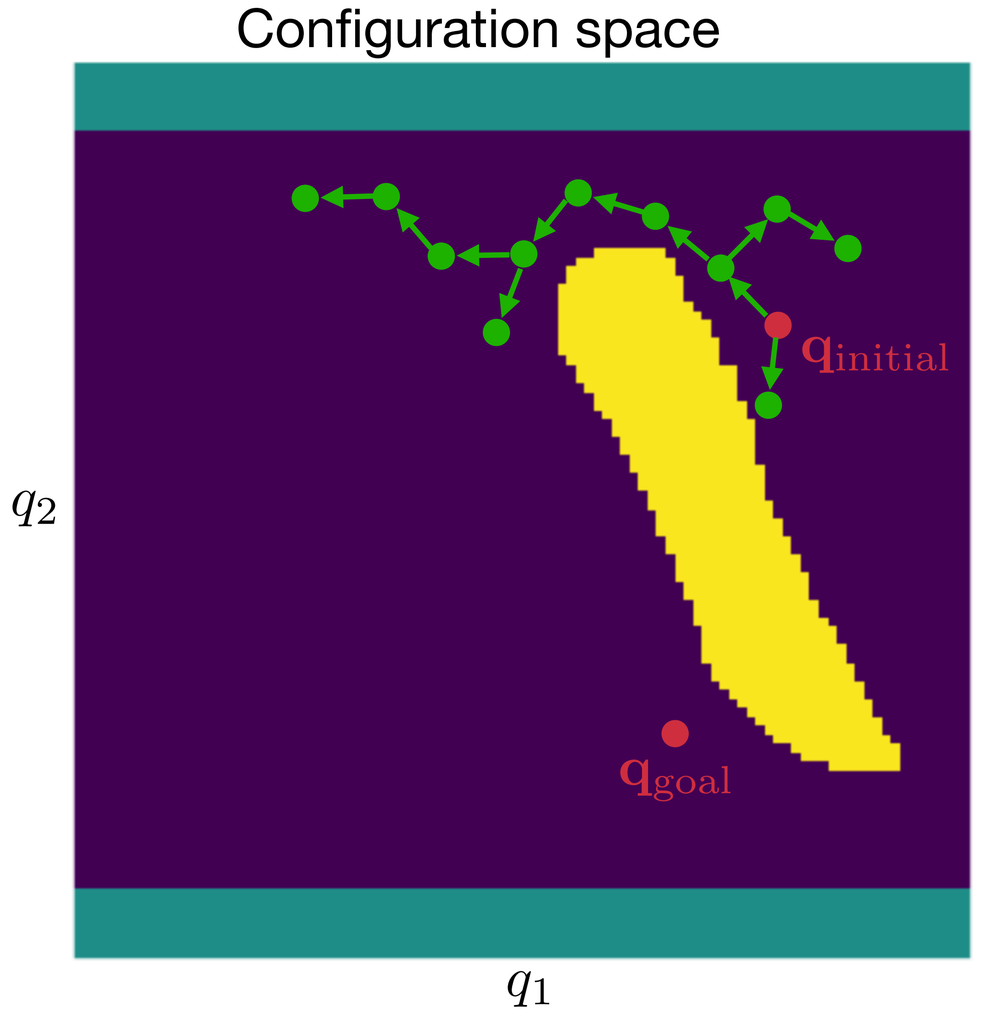
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



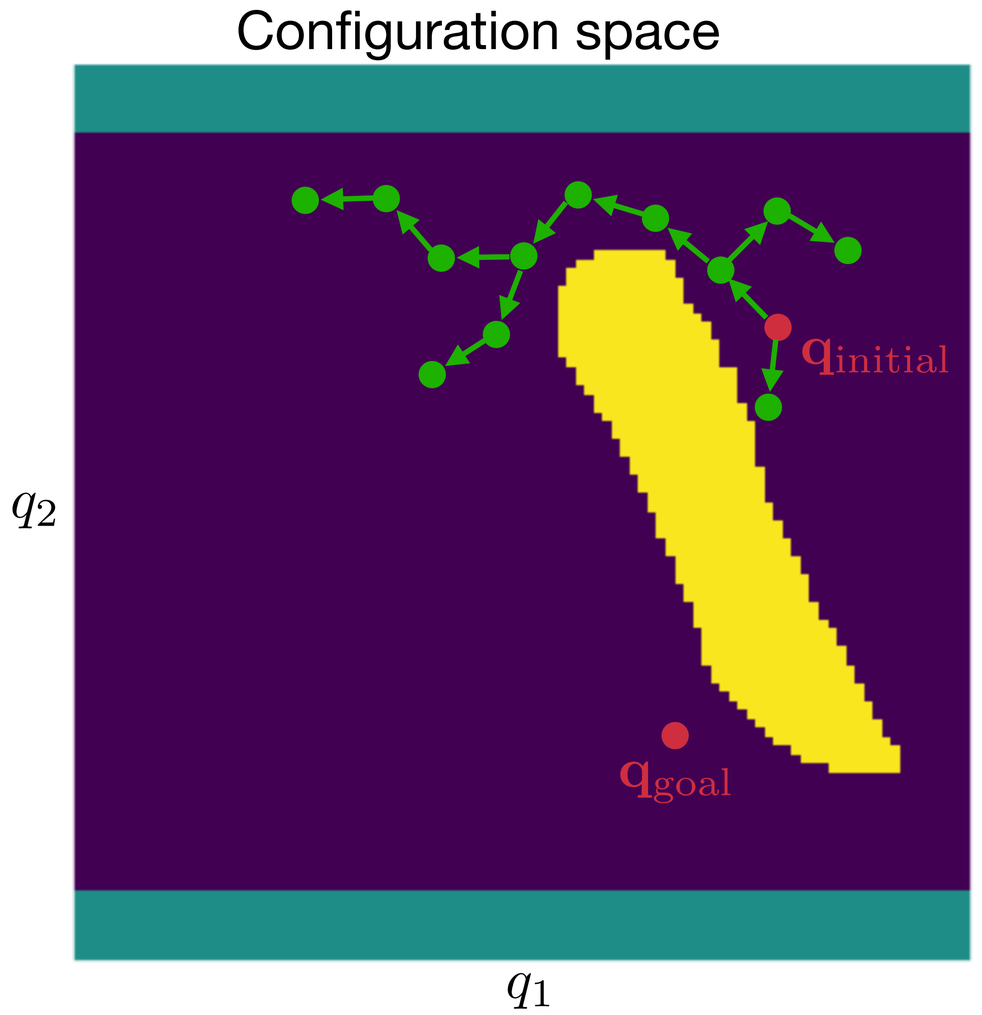
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



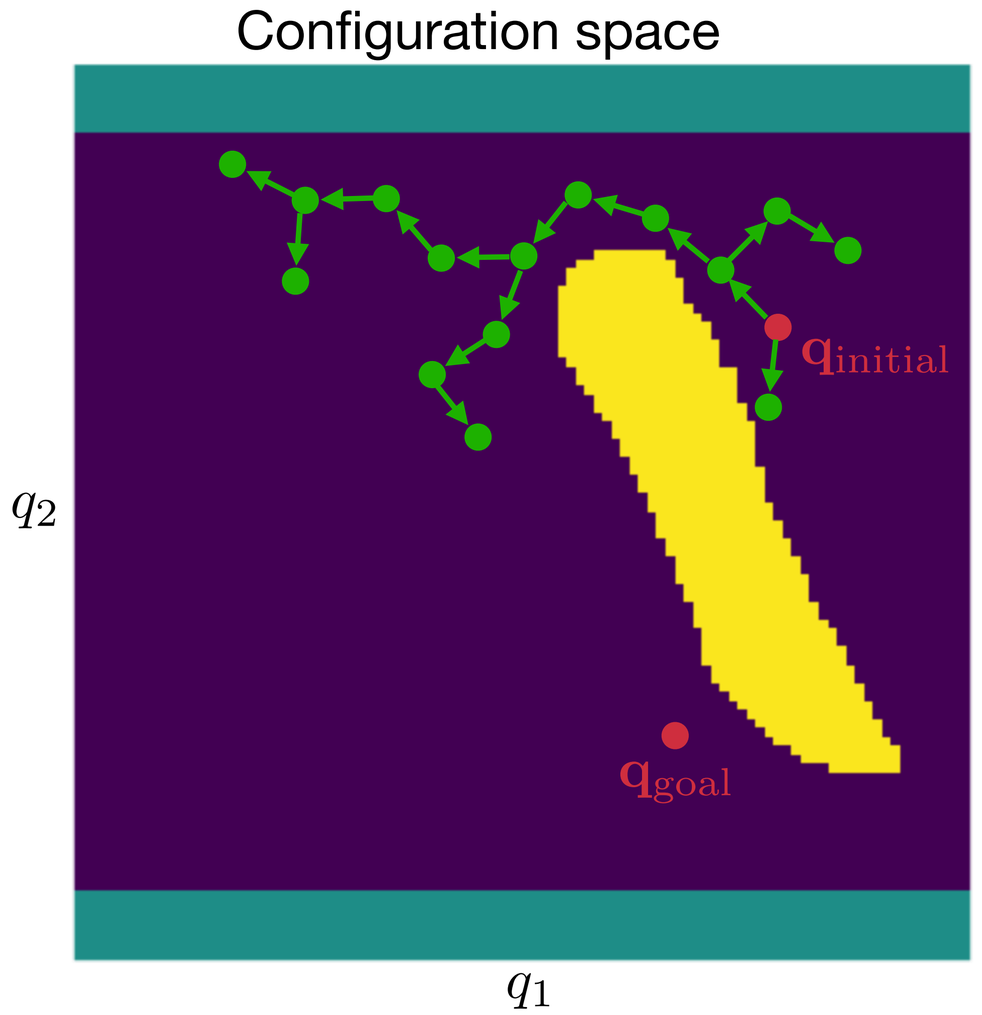
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



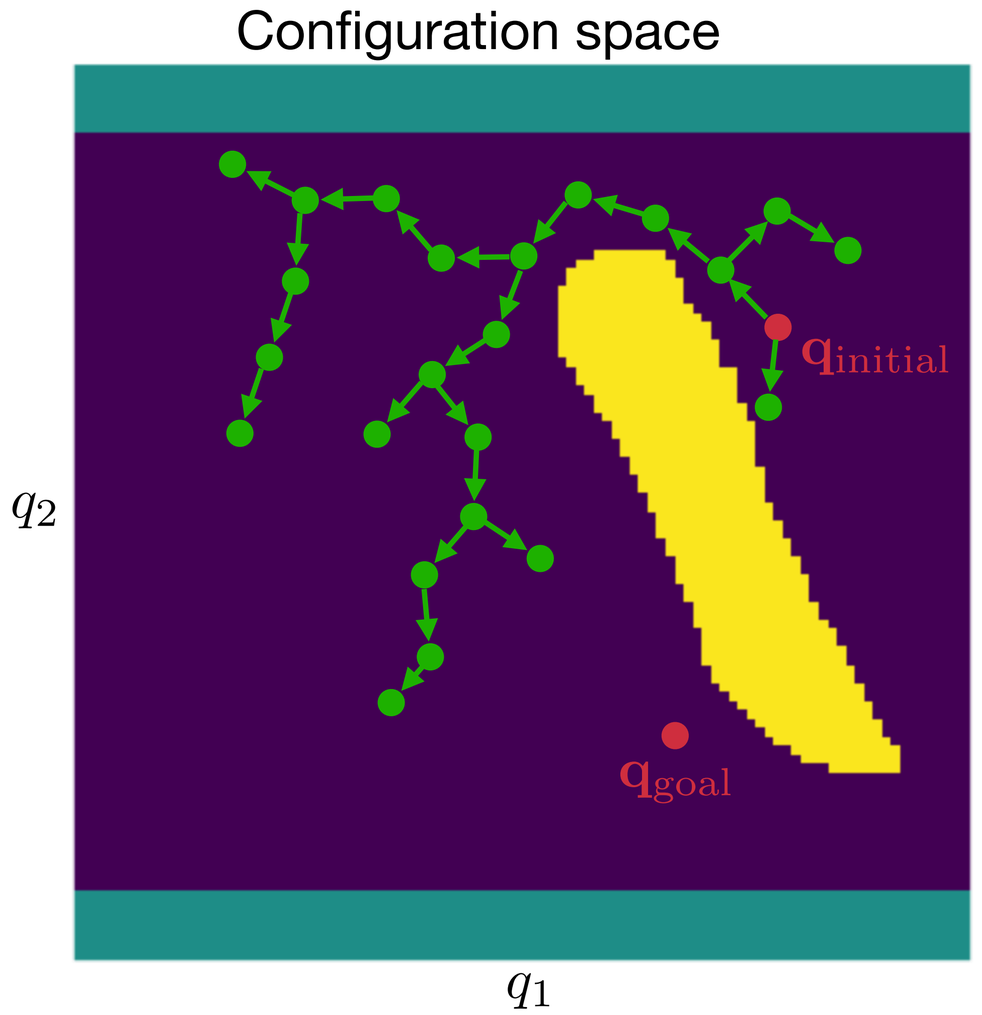
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



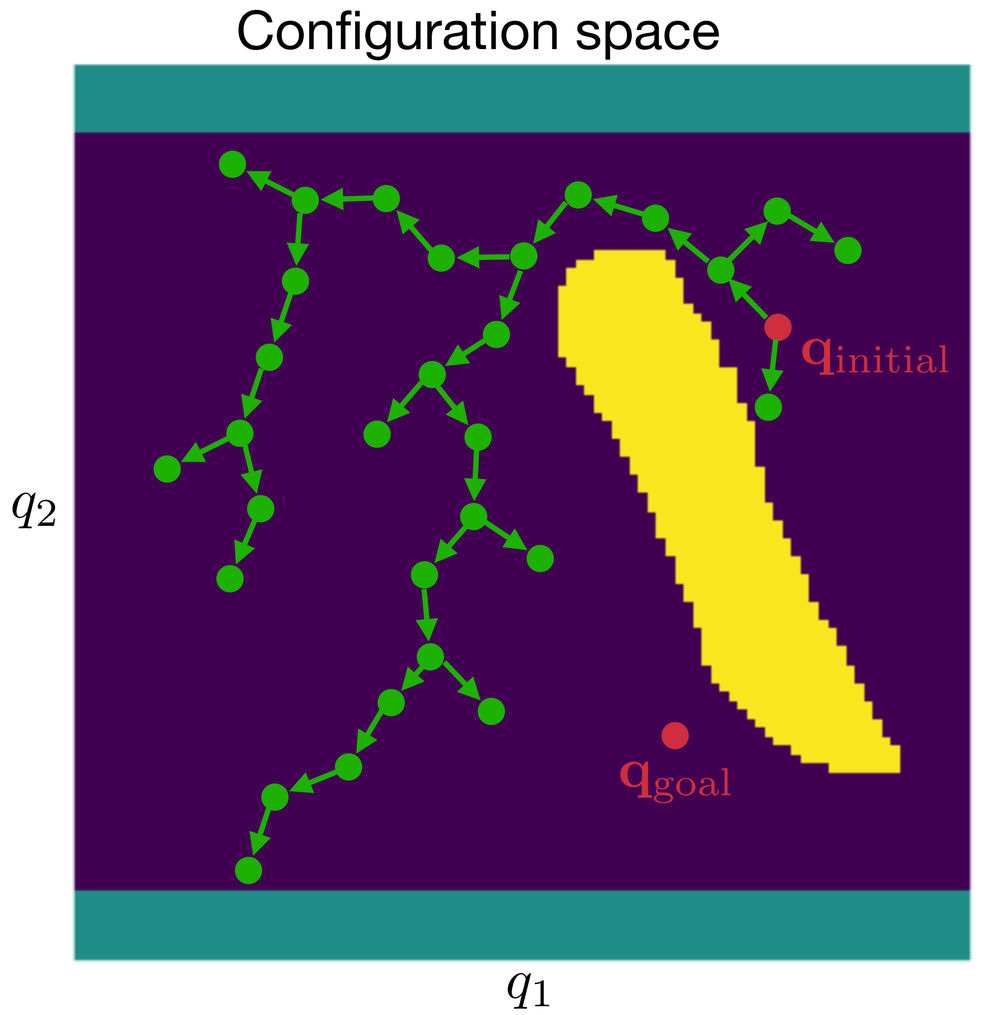
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



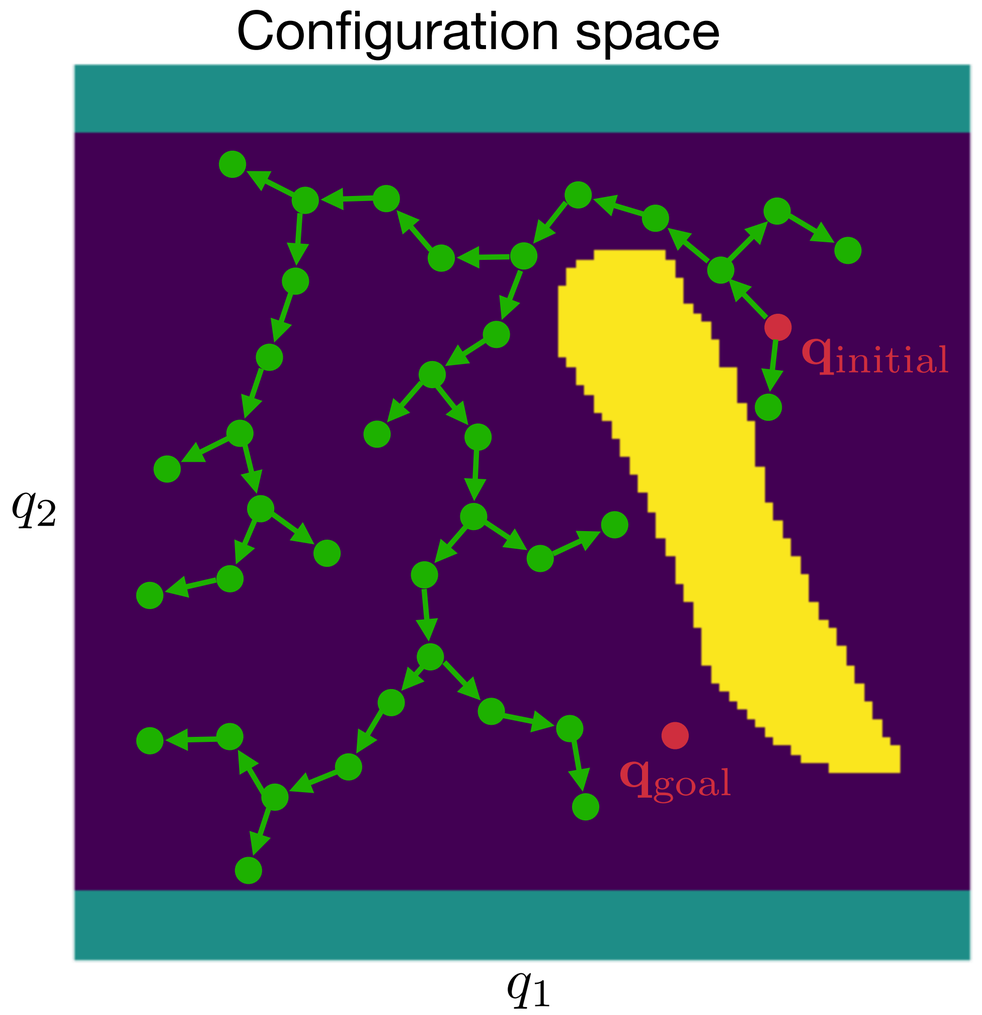
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



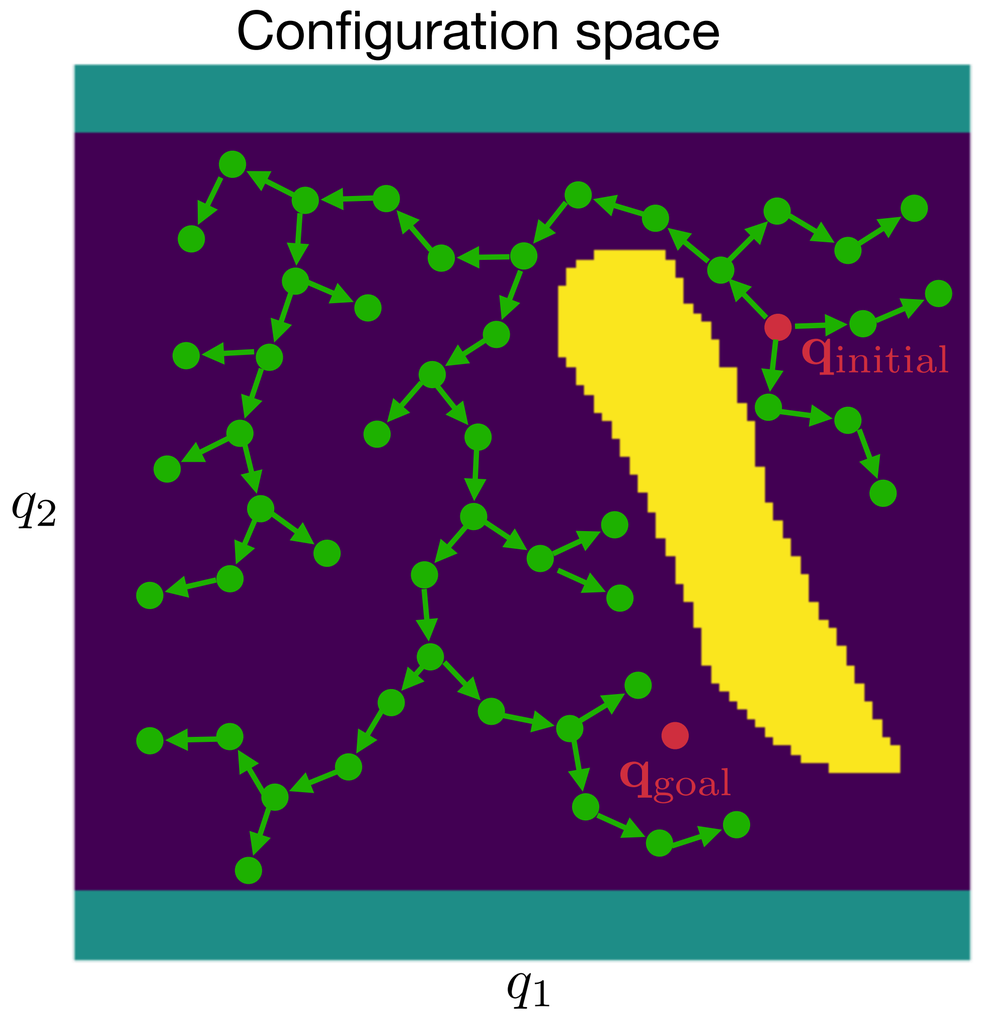
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



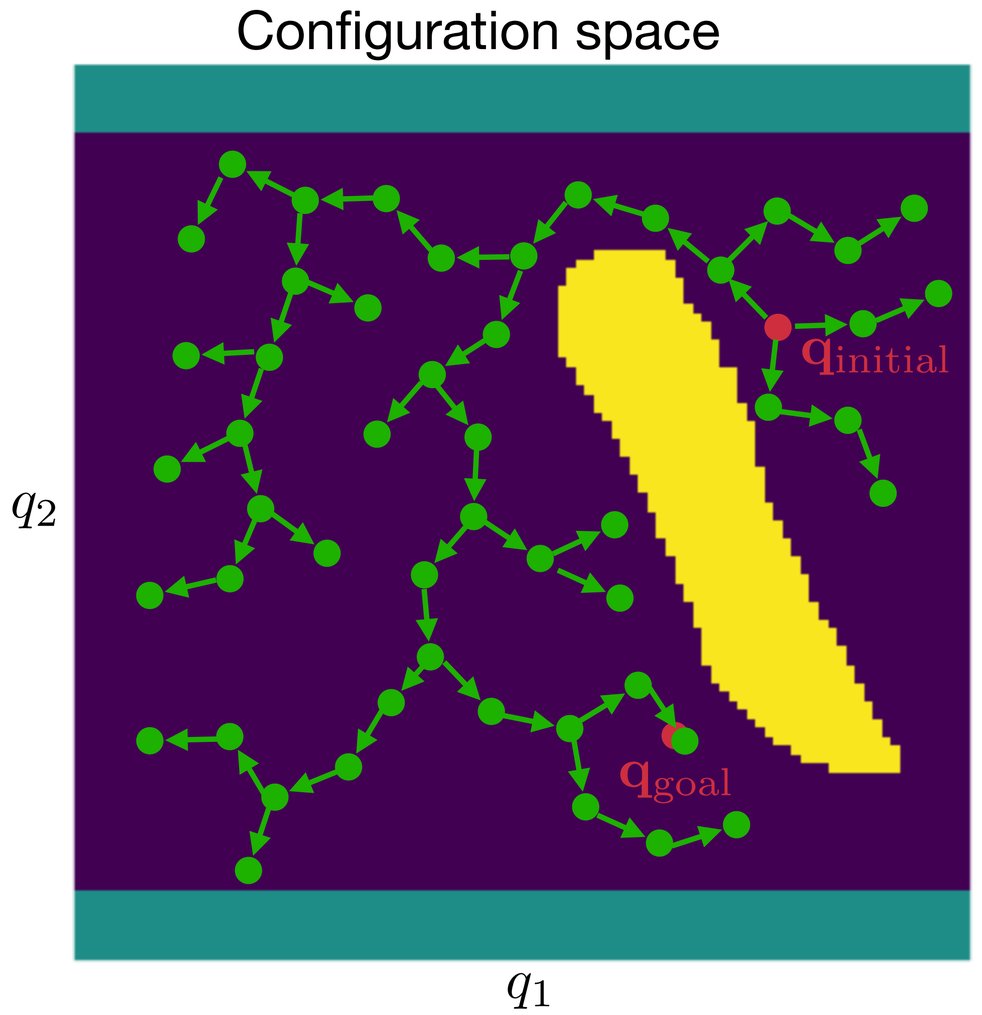
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



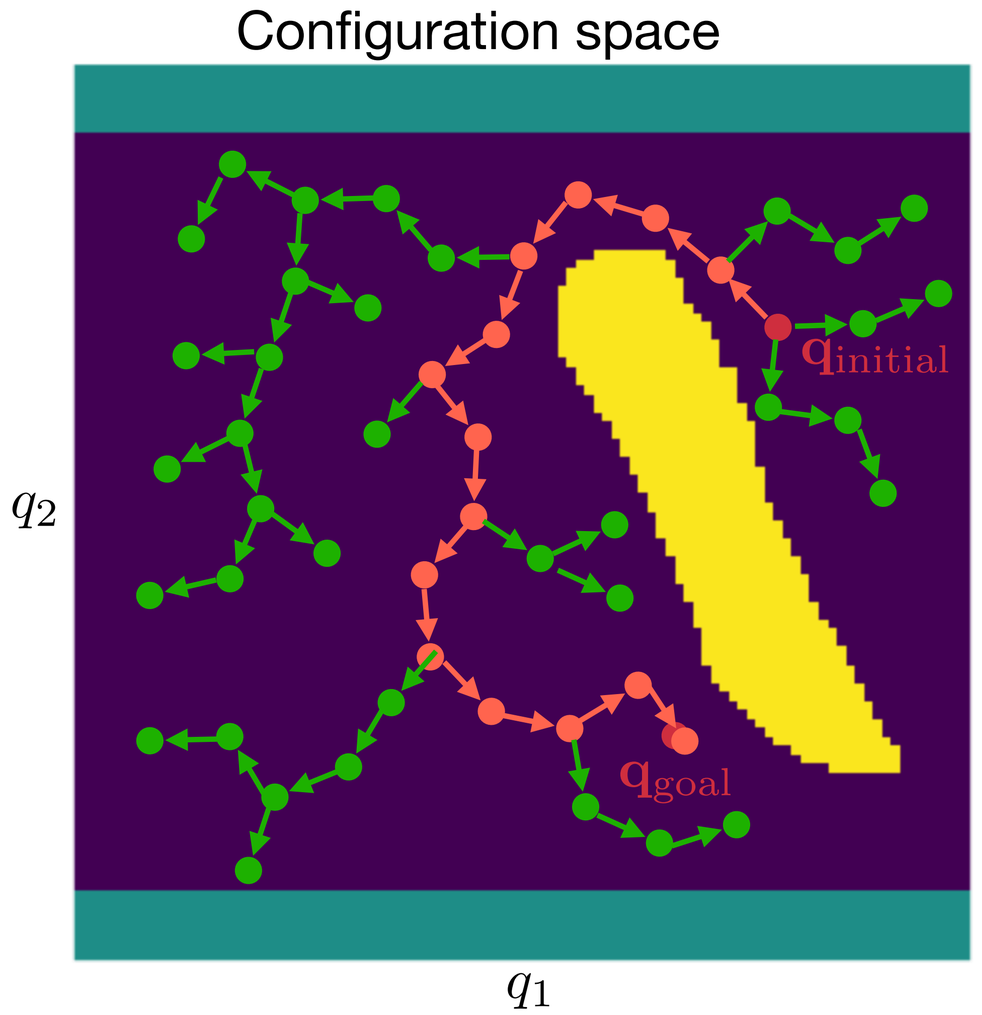
(Let's assume, we know the goal configuration)

Rapidly Exploring Random Trees (RRT)

How it works:

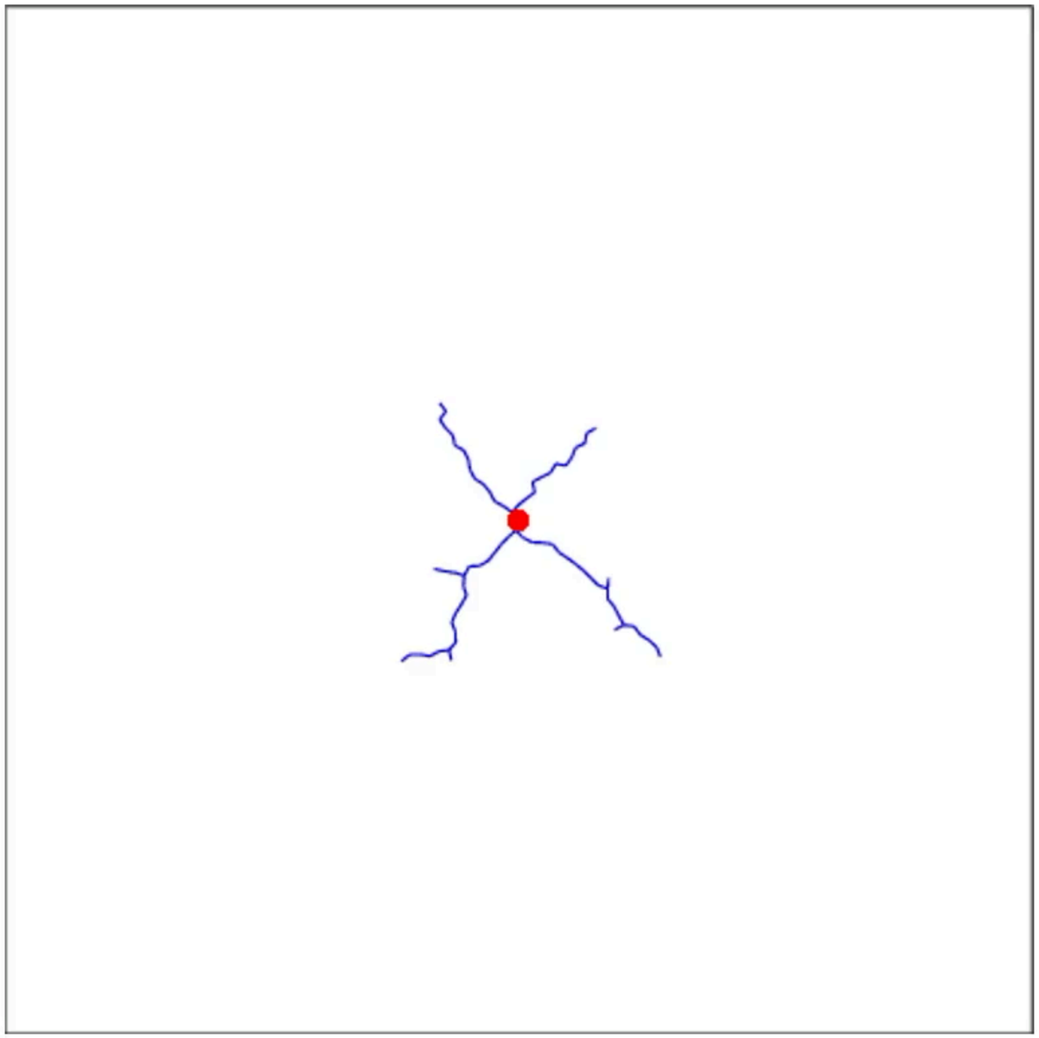
Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- Sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards $\mathbf{q}_{\text{rnd}}$ with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

What do they mean by “rapidly exploring” ?



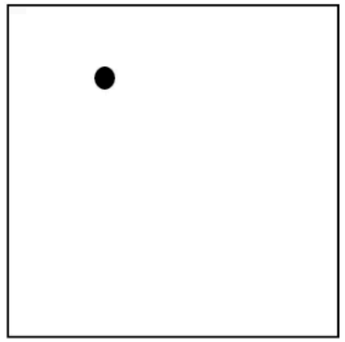
Steven LaValle

Two Phases:

1. Expansion Phase
2. Refinement Phase

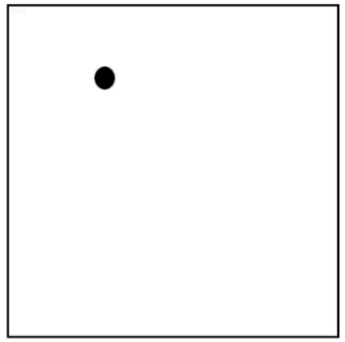
Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias



Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias

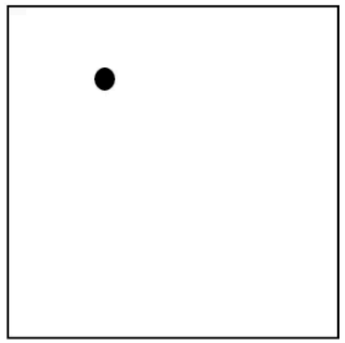


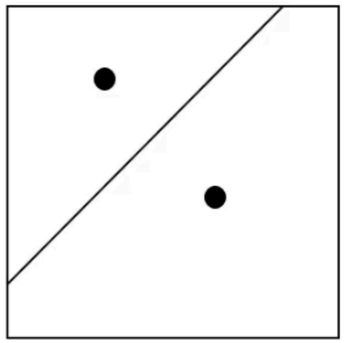
Voronoi Region

- Space in which a node would be the closest neighbor

Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias



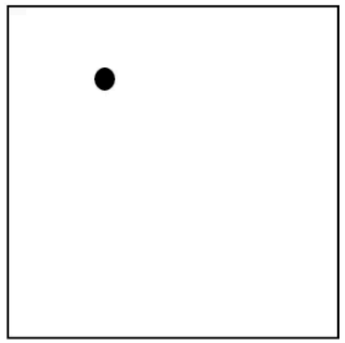


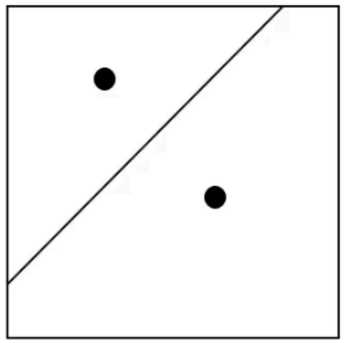
Voronoi Region

- Space in which a node would be the closest neighbor

Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias



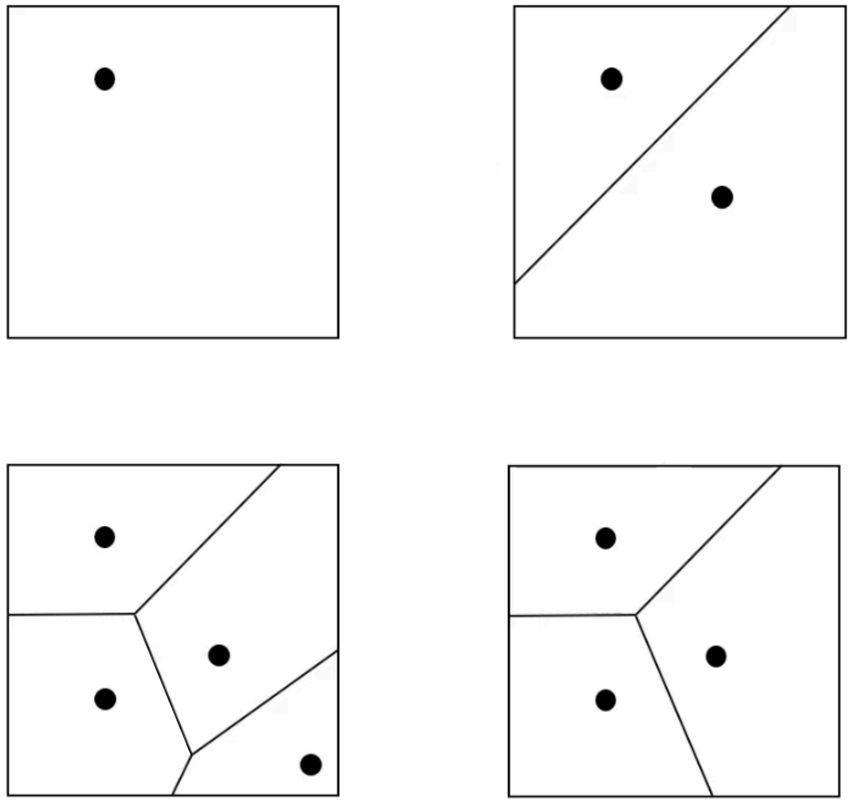


Voronoi Region

- Space in which a node would be the closest neighbor
- Its volume is proportional to the probability of being the closest

Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias

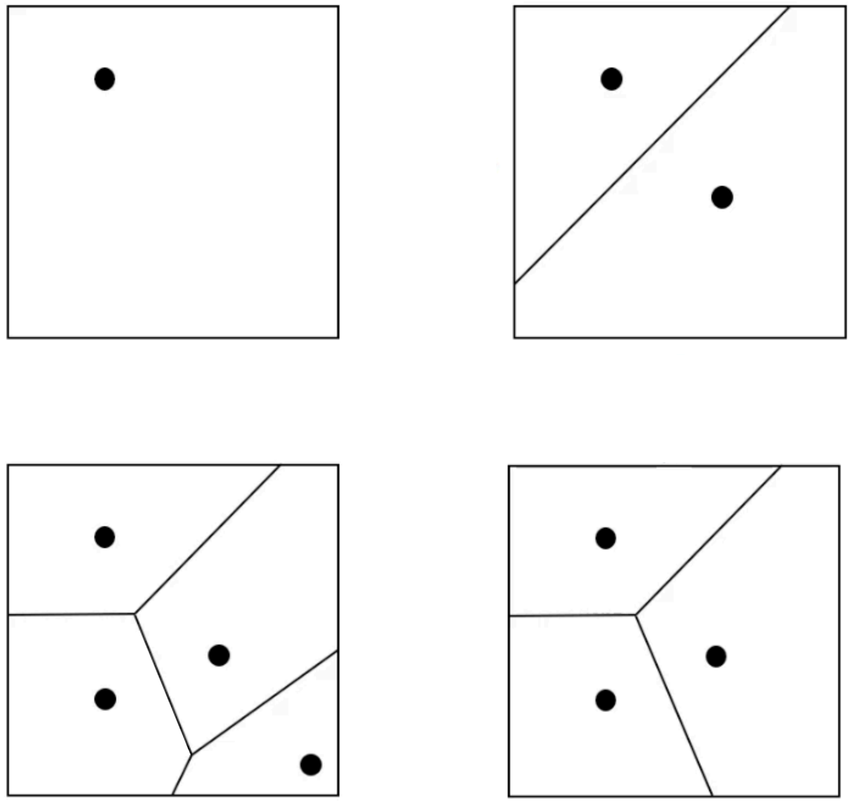


Voronoi Region

- Space in which a node would be the closest neighbor
- Its volume is proportional to the probability of being the closest

Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias

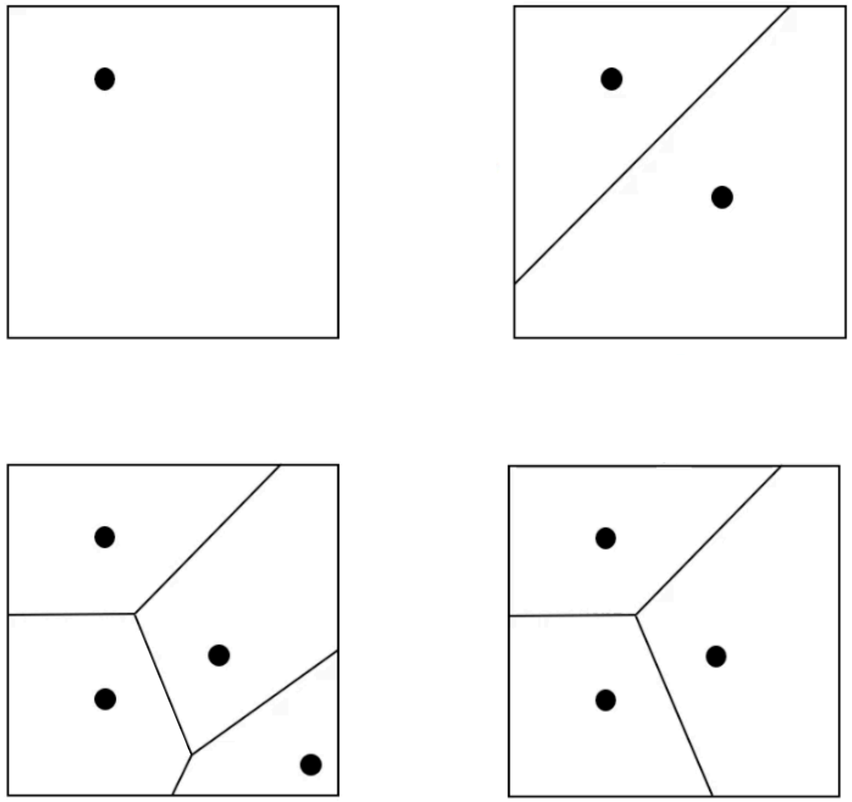


Voronoi Region

- Space in which a node would be the closest neighbor
- Its volume is proportional to the probability of being the closest
- The larger the volume, the higher the probability of being the closest

Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias

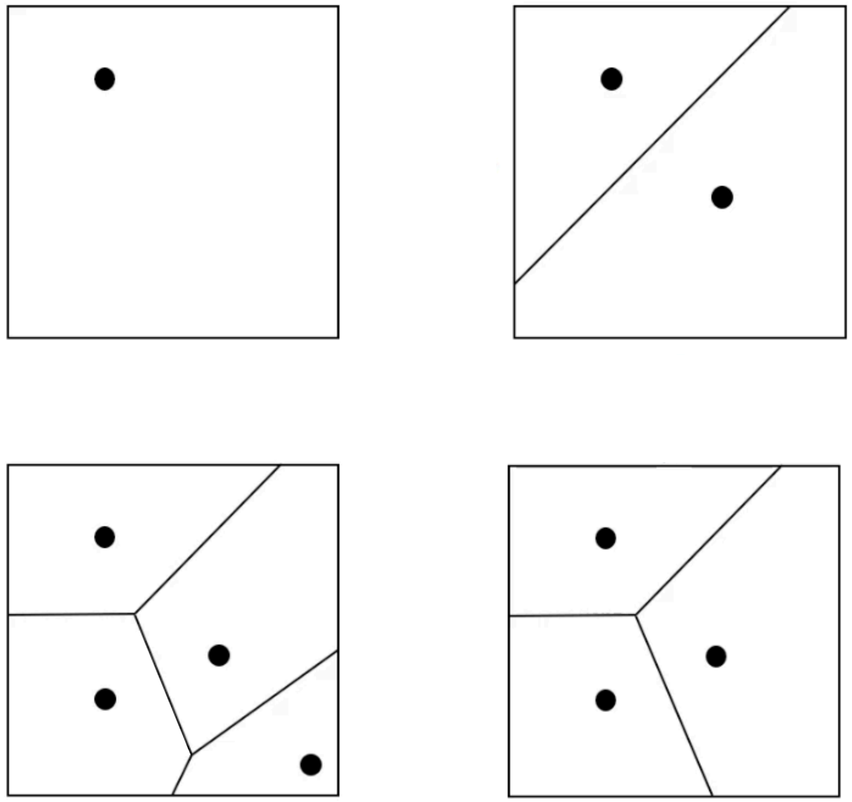


Voronoi Region

- Space in which a node would be the closest neighbor
- Its volume is proportional to the probability of being the closest
- The larger the volume, the higher the probability of being the closest
- Therefore, sparsely populated regions are most likely to be explored in the next step (“Voronoi Bias”)

Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias



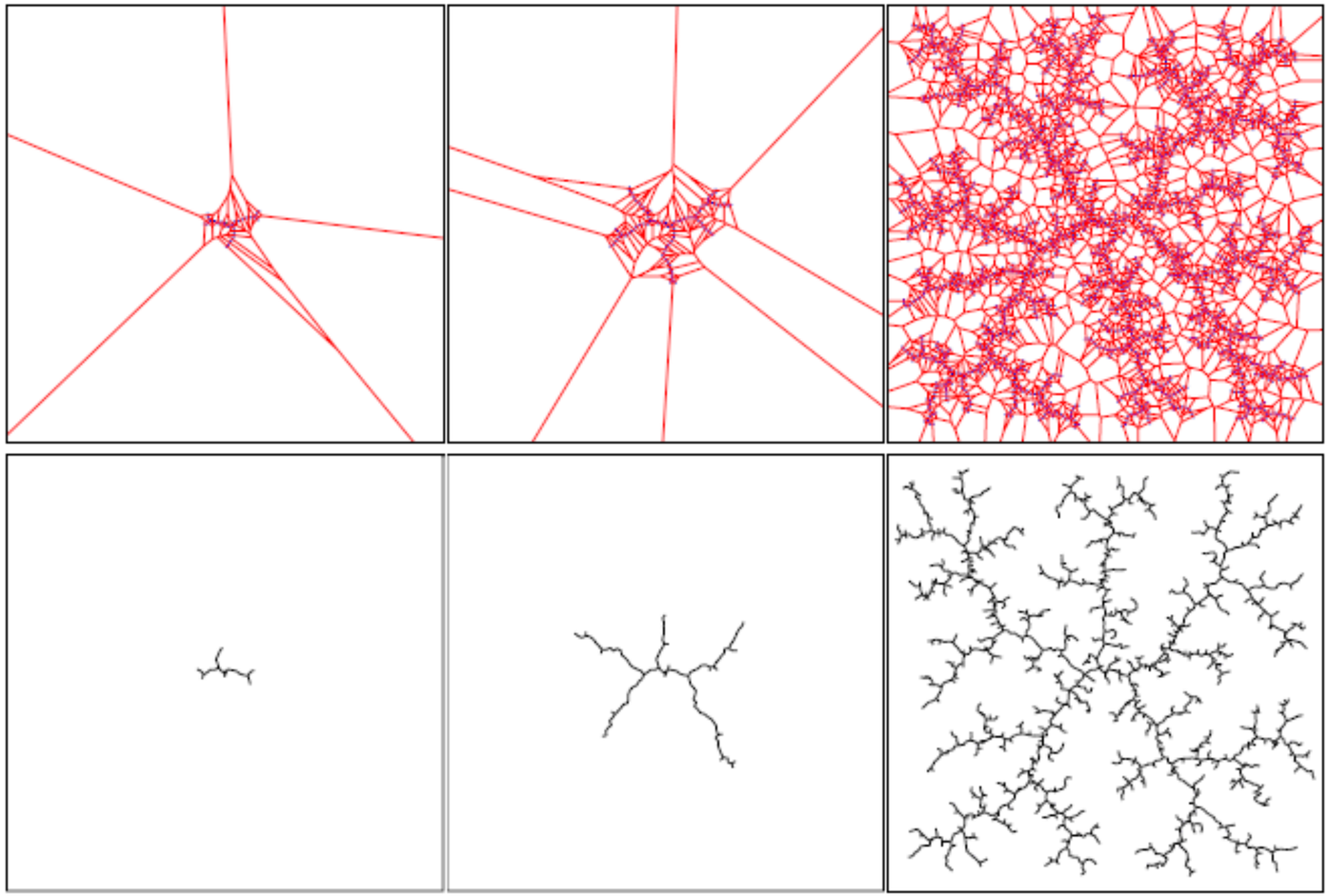
Voronoi Diagram

Voronoi Region

- Space in which a node would be the closest neighbor
- Its volume is proportional to the probability of being the closest
- The larger the volume, the higher the probability of being the closest
- Therefore, sparsely populated regions are most likely to be explored in the next step (“Voronoi Bias”)

Why is it rapidly exploring?

It’s about finding the closest node: The Voronoi Bias

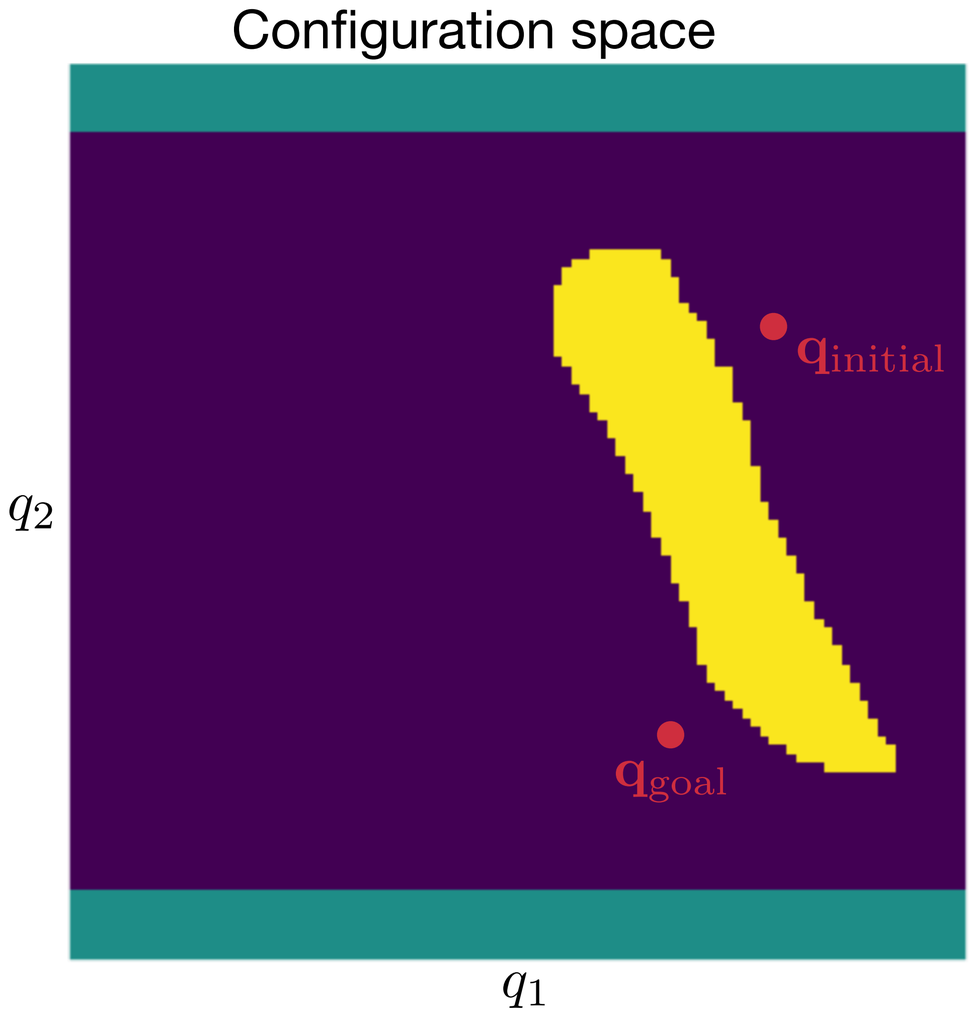


Can we make RRT better?

1.) Goal Bias

2.) RRT-Connect

1.) Goal Bias

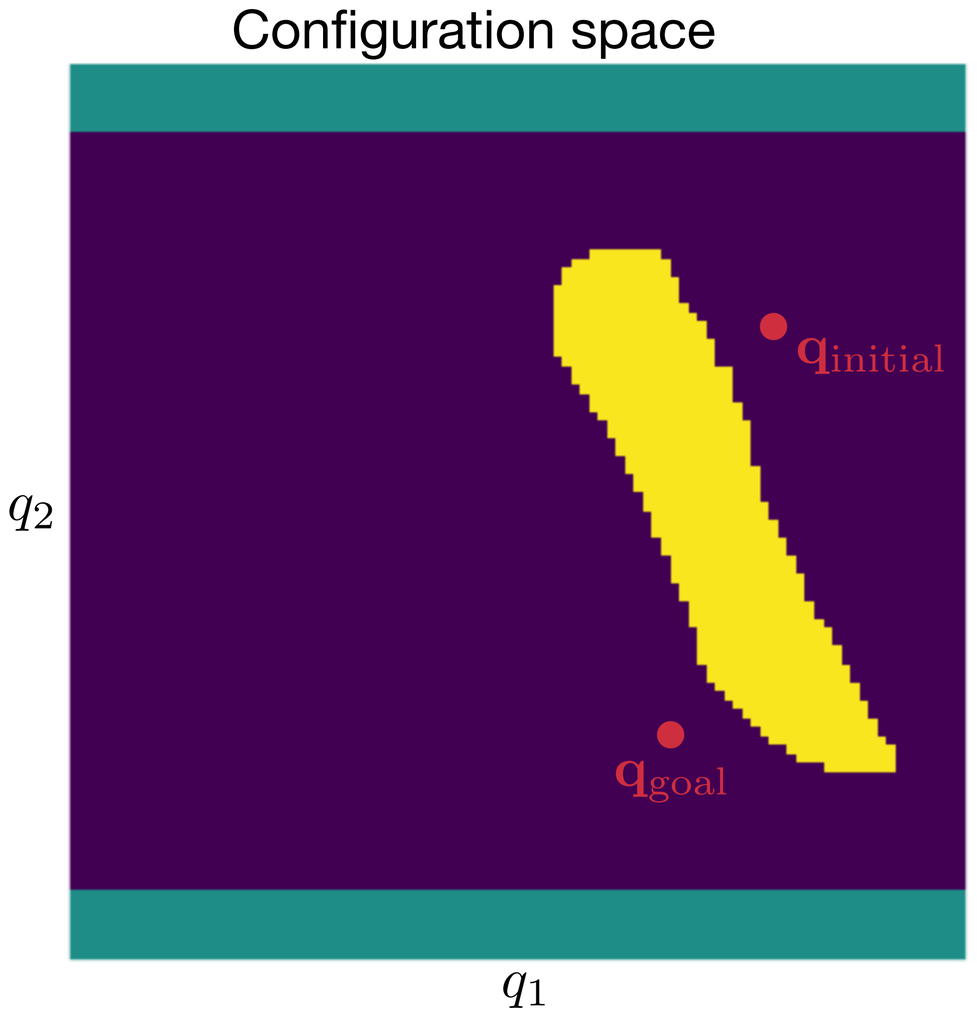


(Let’s assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



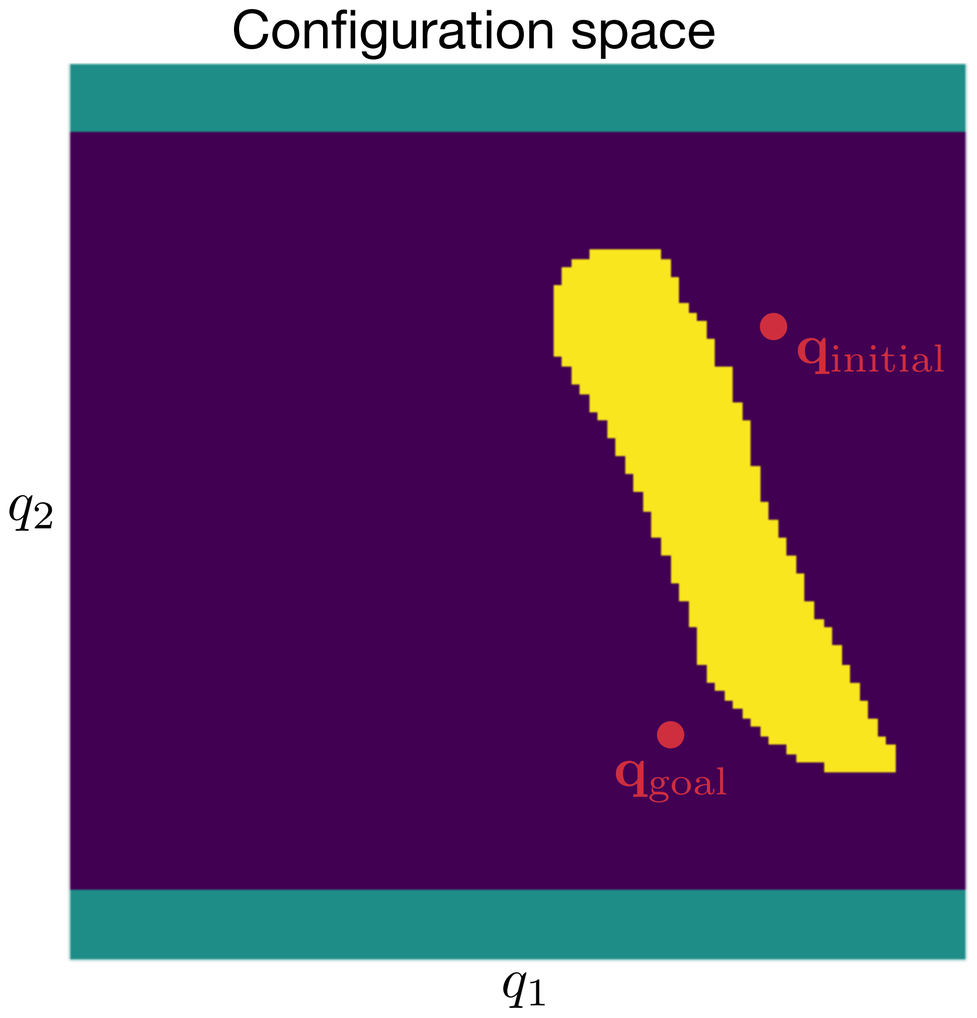
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



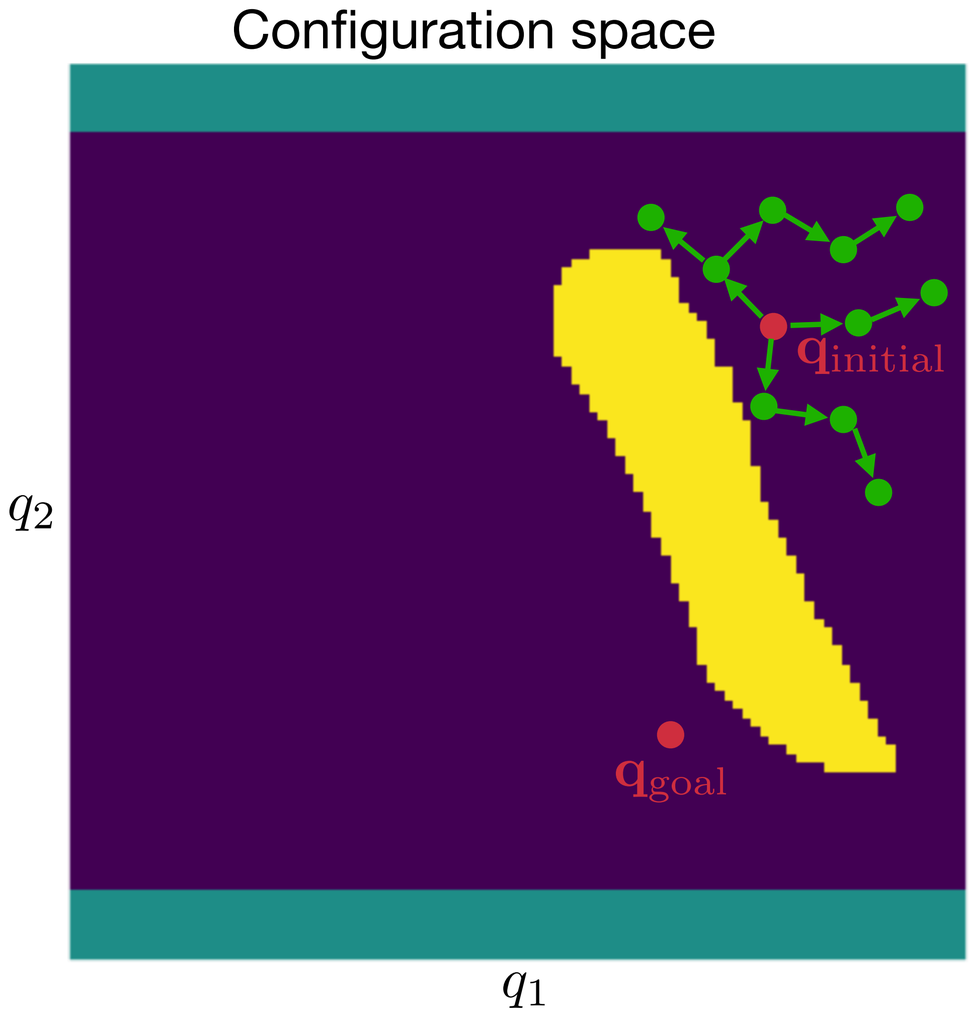
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



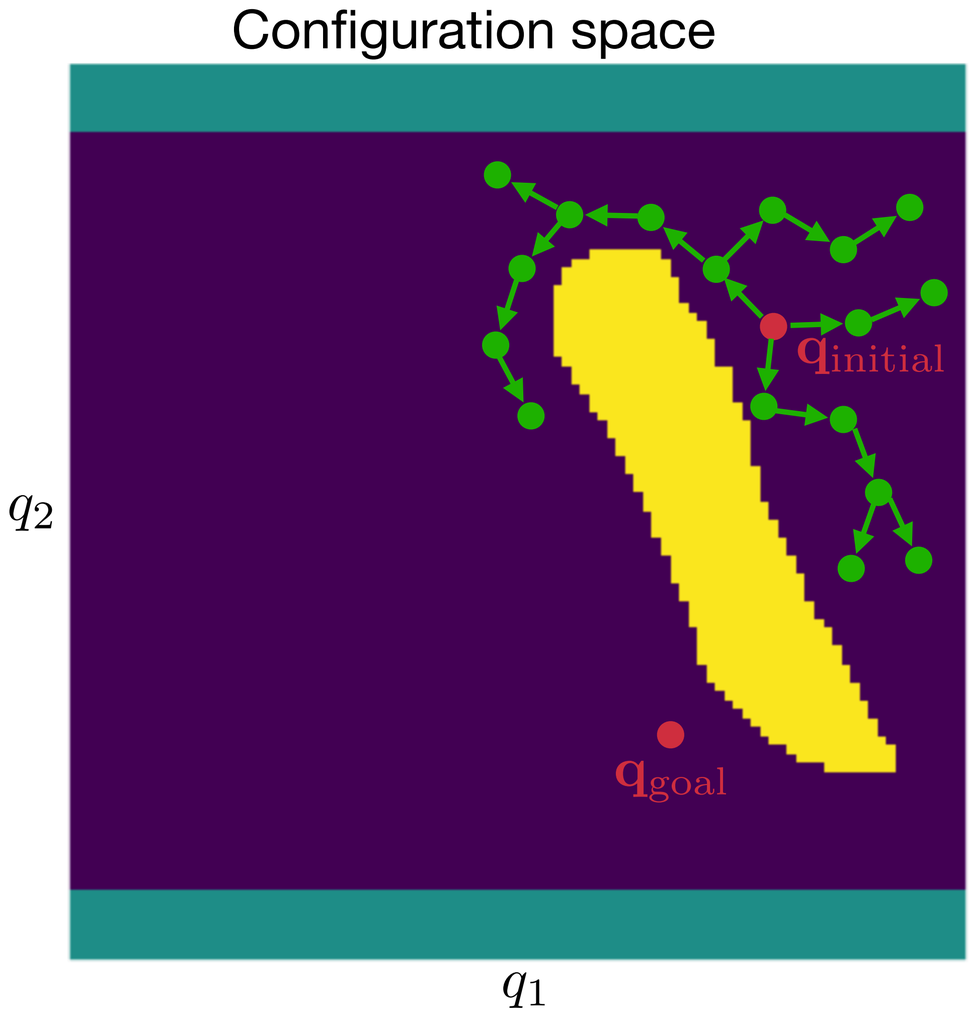
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



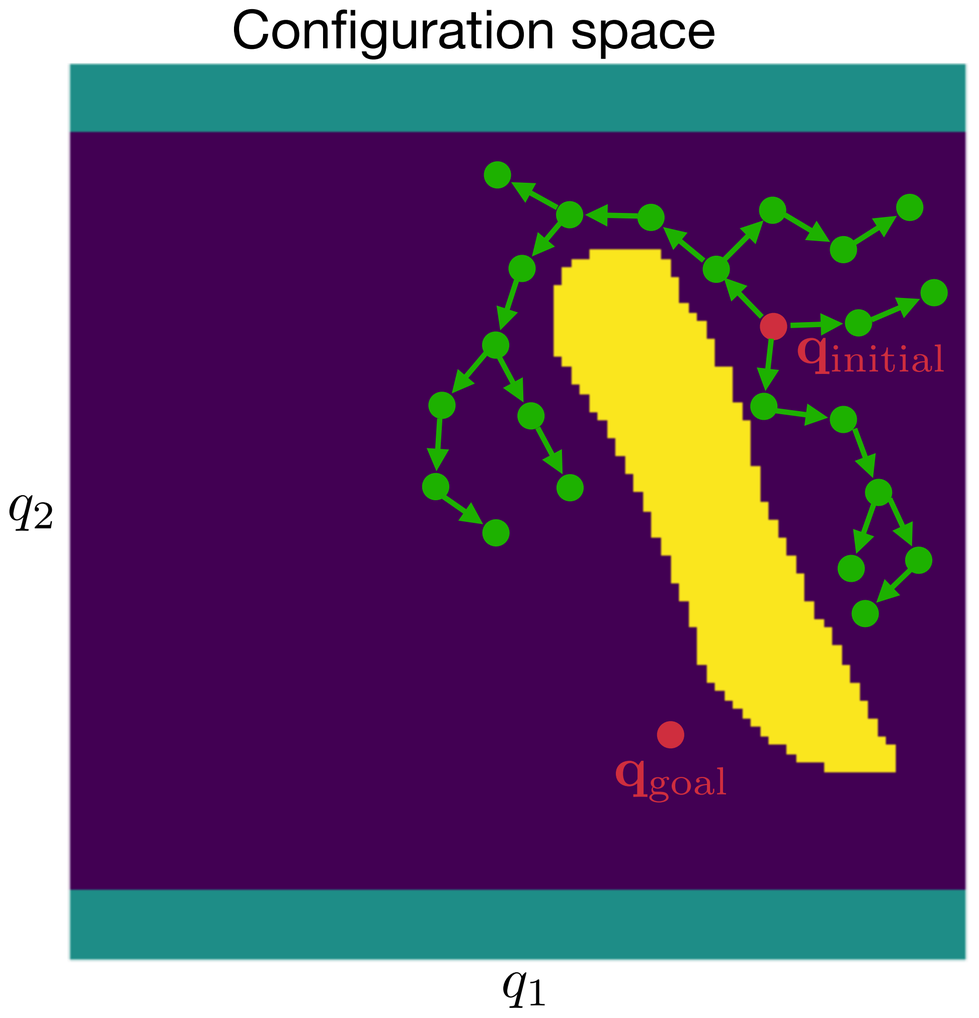
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



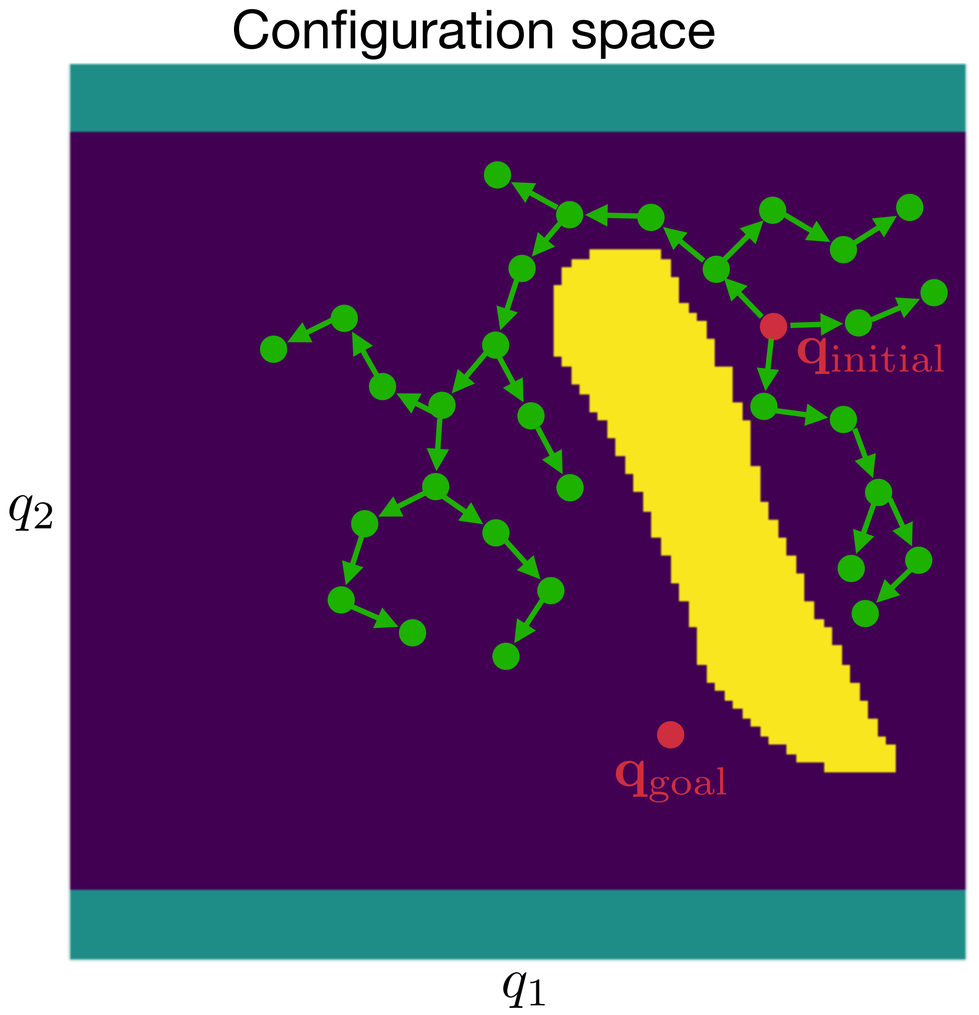
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



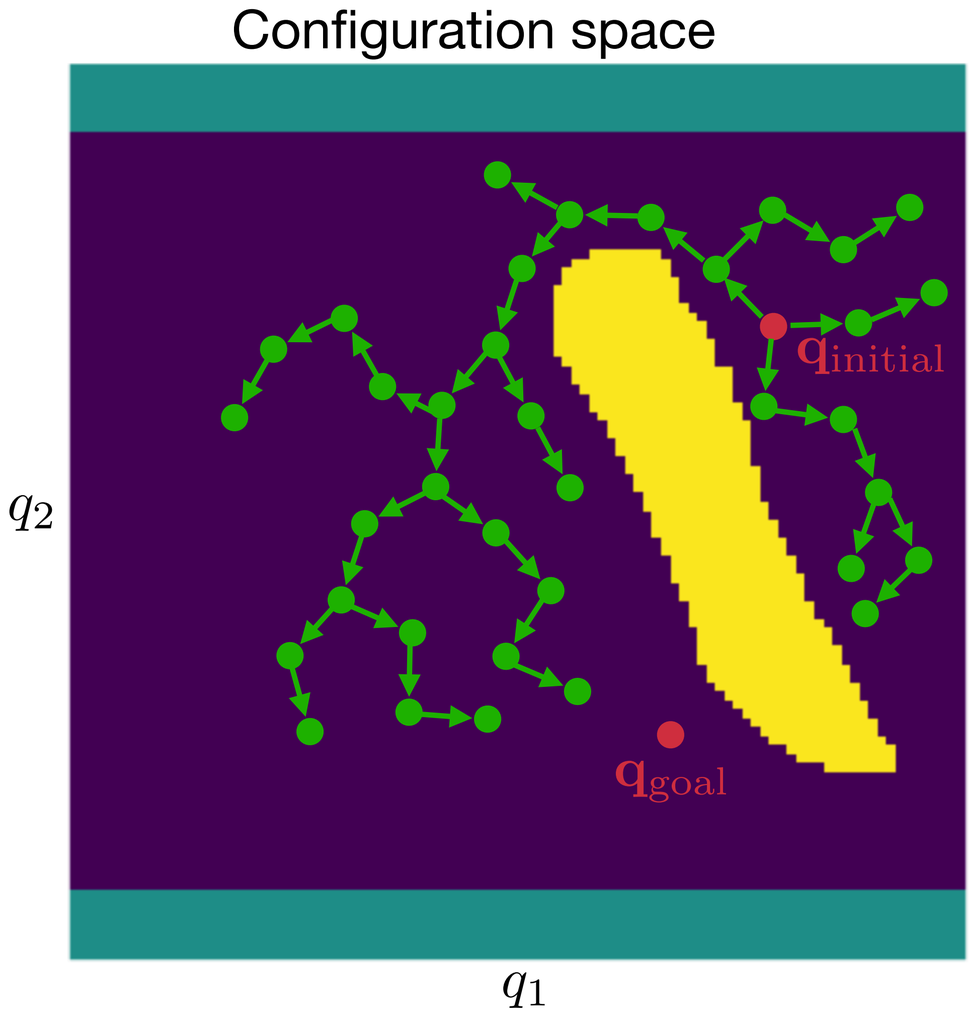
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



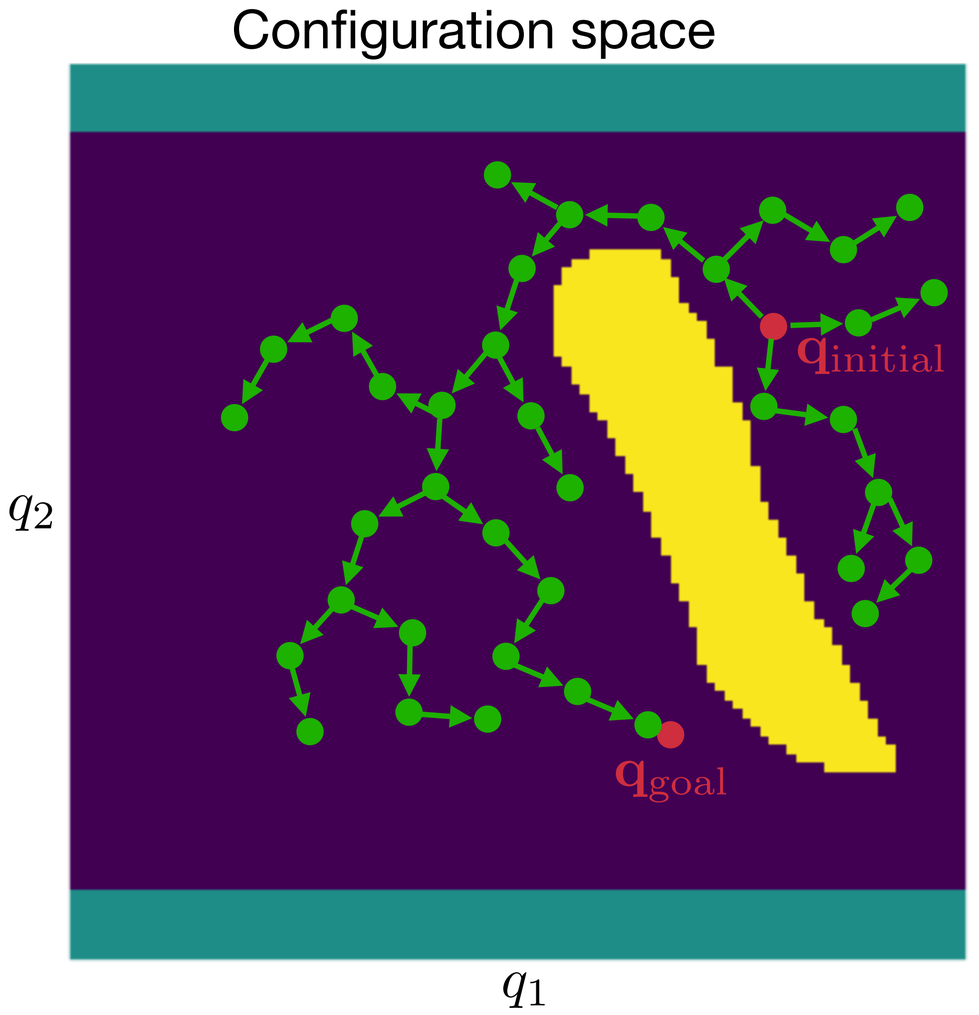
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



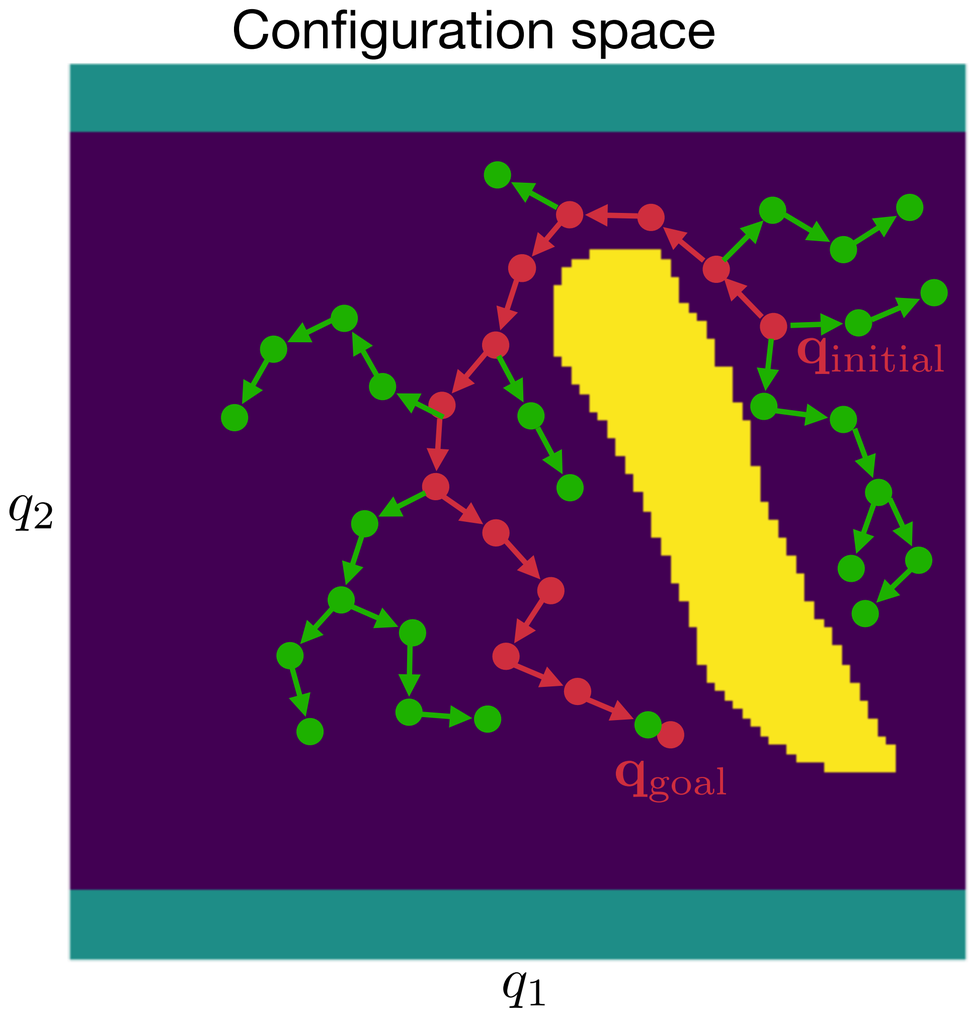
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



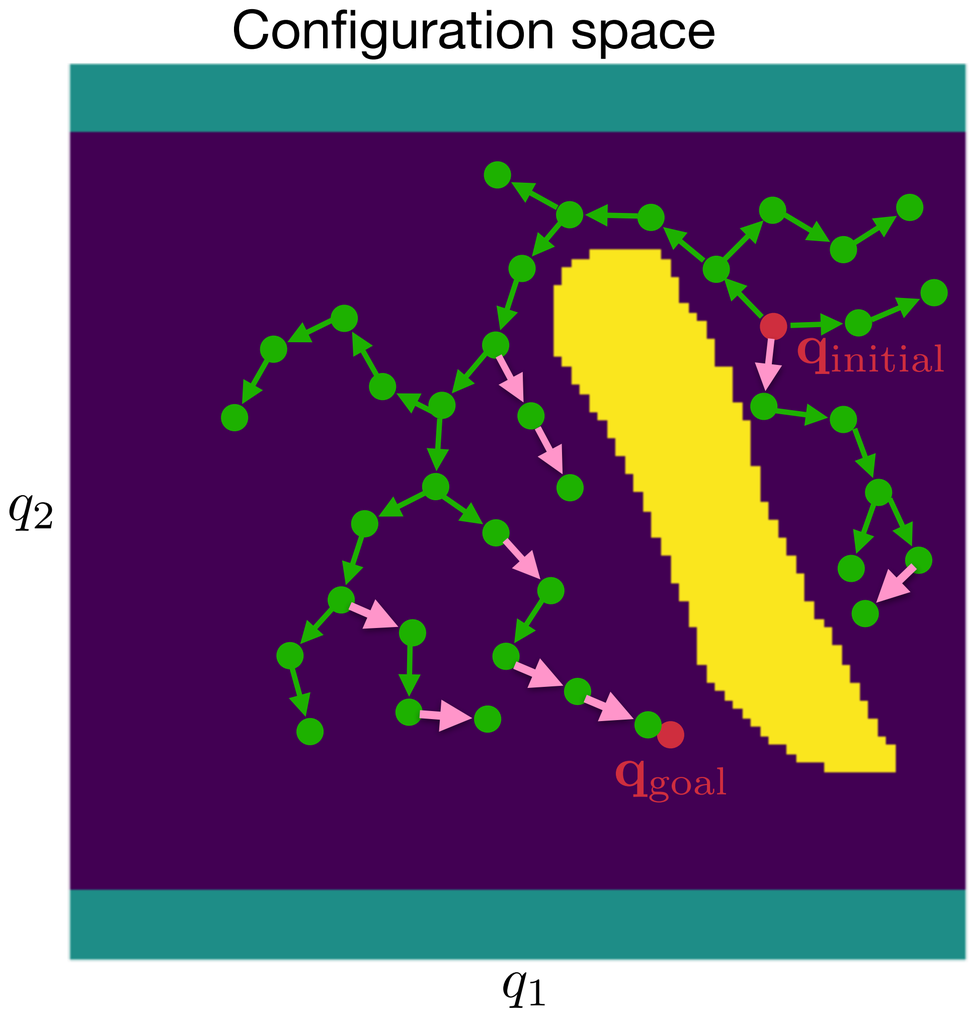
(Let's assume, we know the goal configuration)

1.) Goal Bias

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

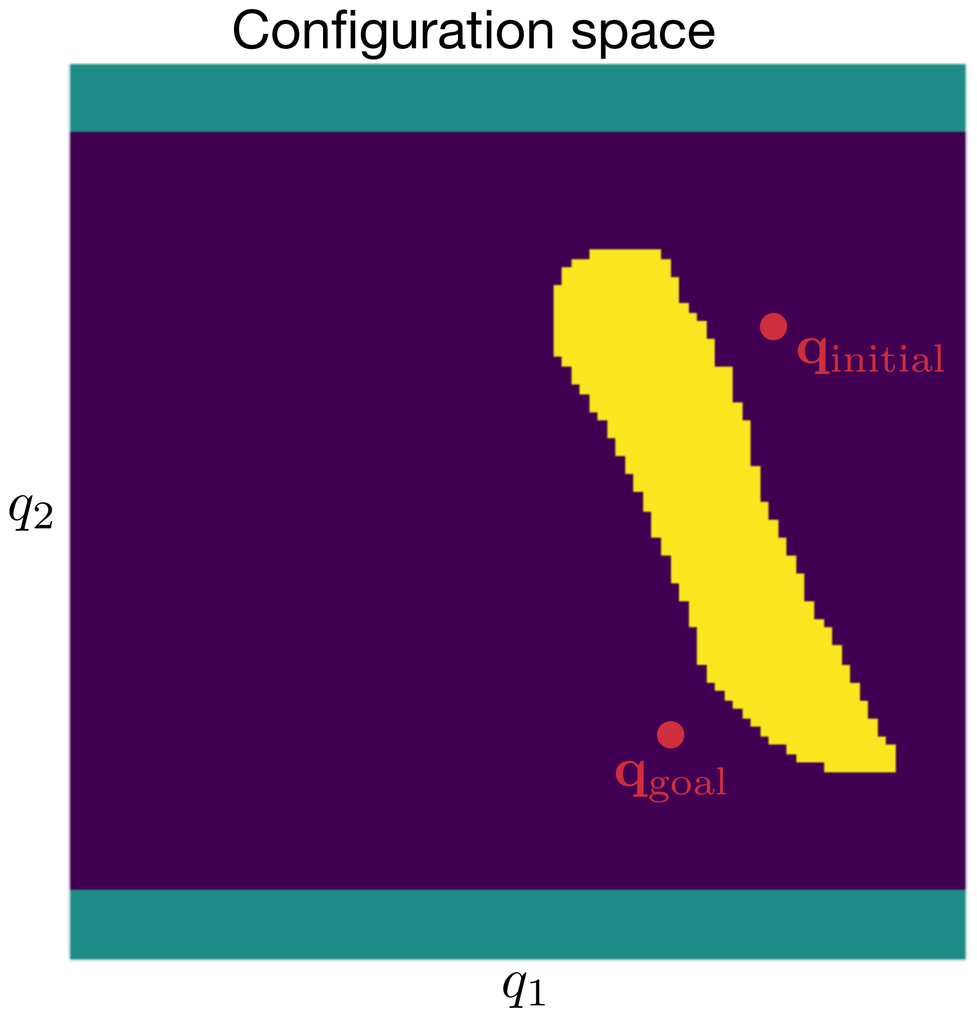
Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ
- If that new configuration is valid (i.e., no collisions), add new node and edge to tree
- If distance between new node and $\mathbf{q}_{\text{goal}}$ is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect



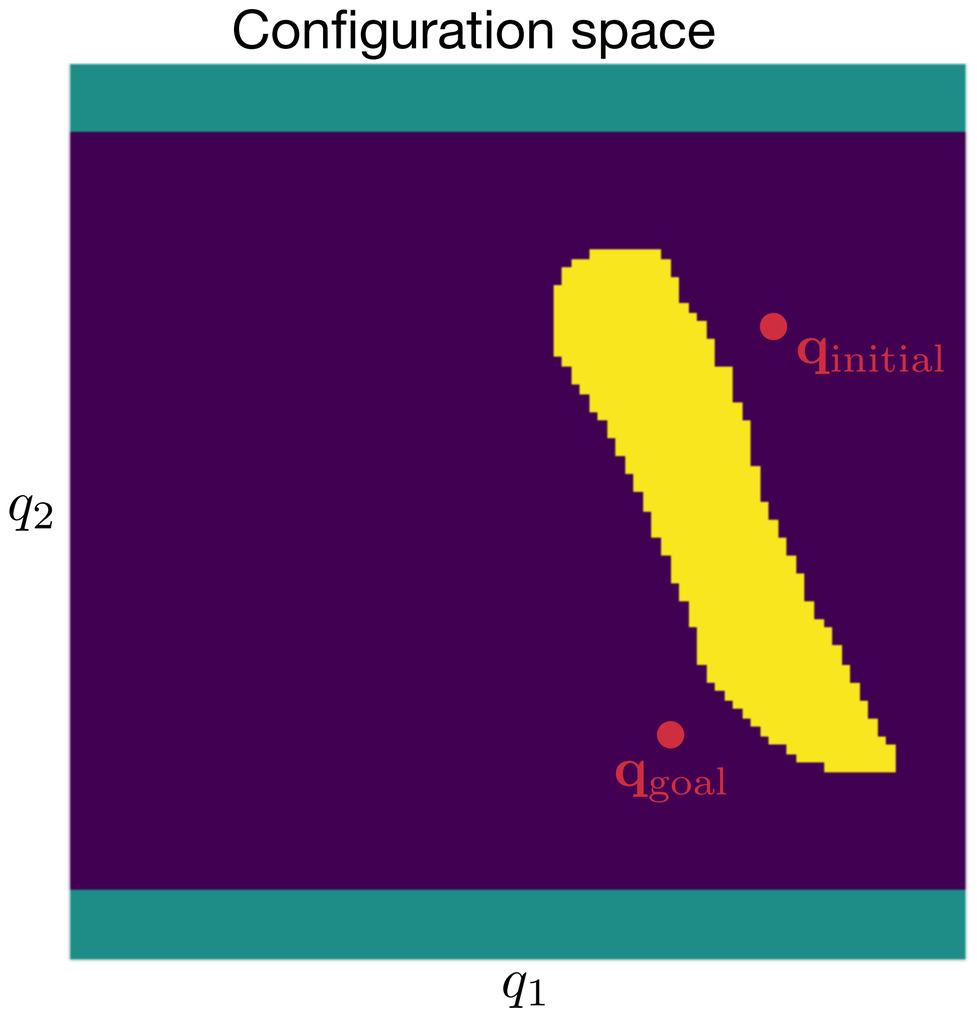
(Let’s assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$
- Find closest node $\mathbf{v}_{\text{closest}}$ in tree
- Move <u>repeatedly</u> from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration
- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

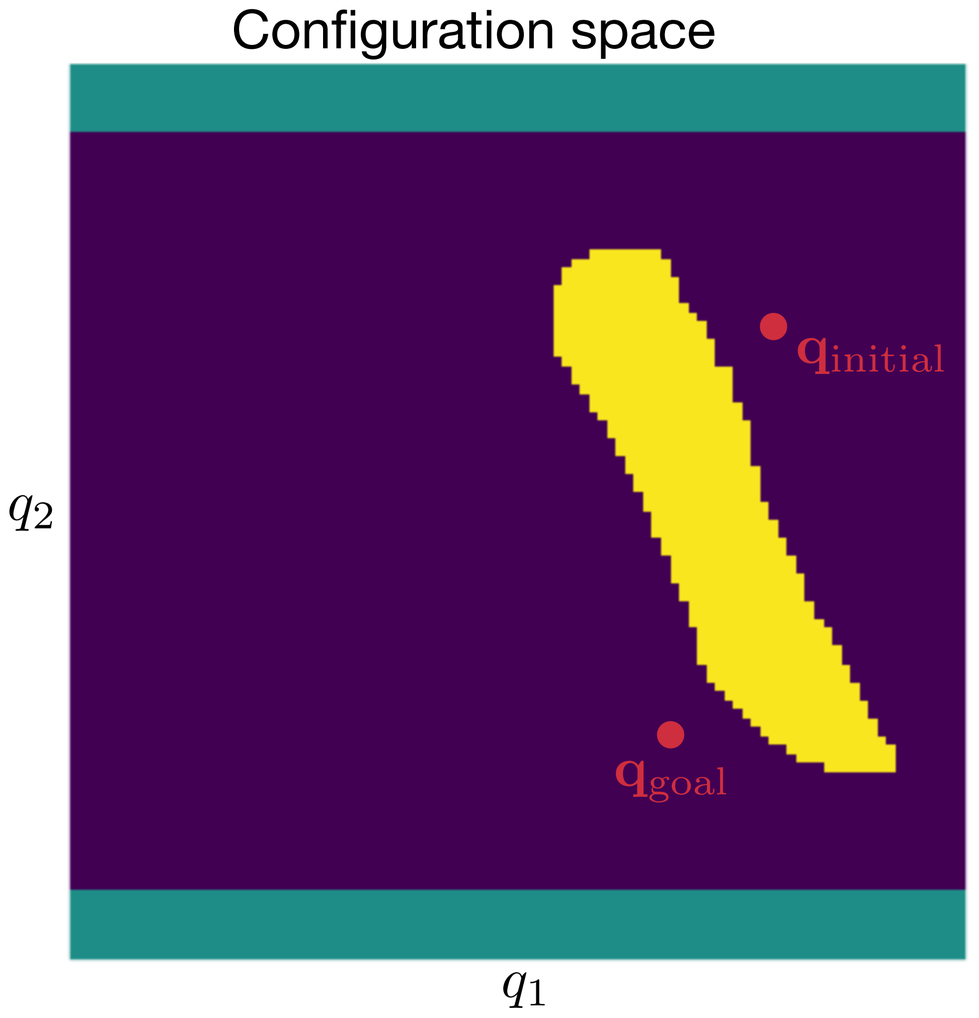
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

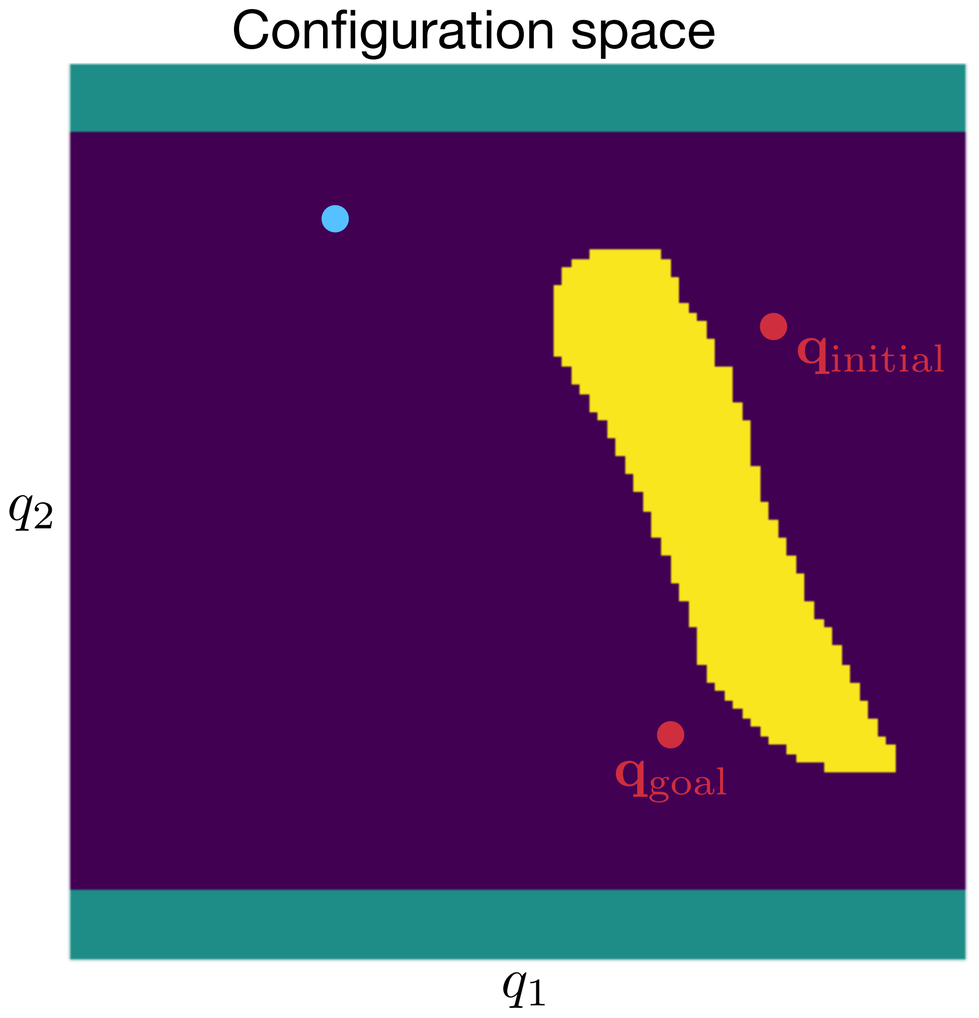
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

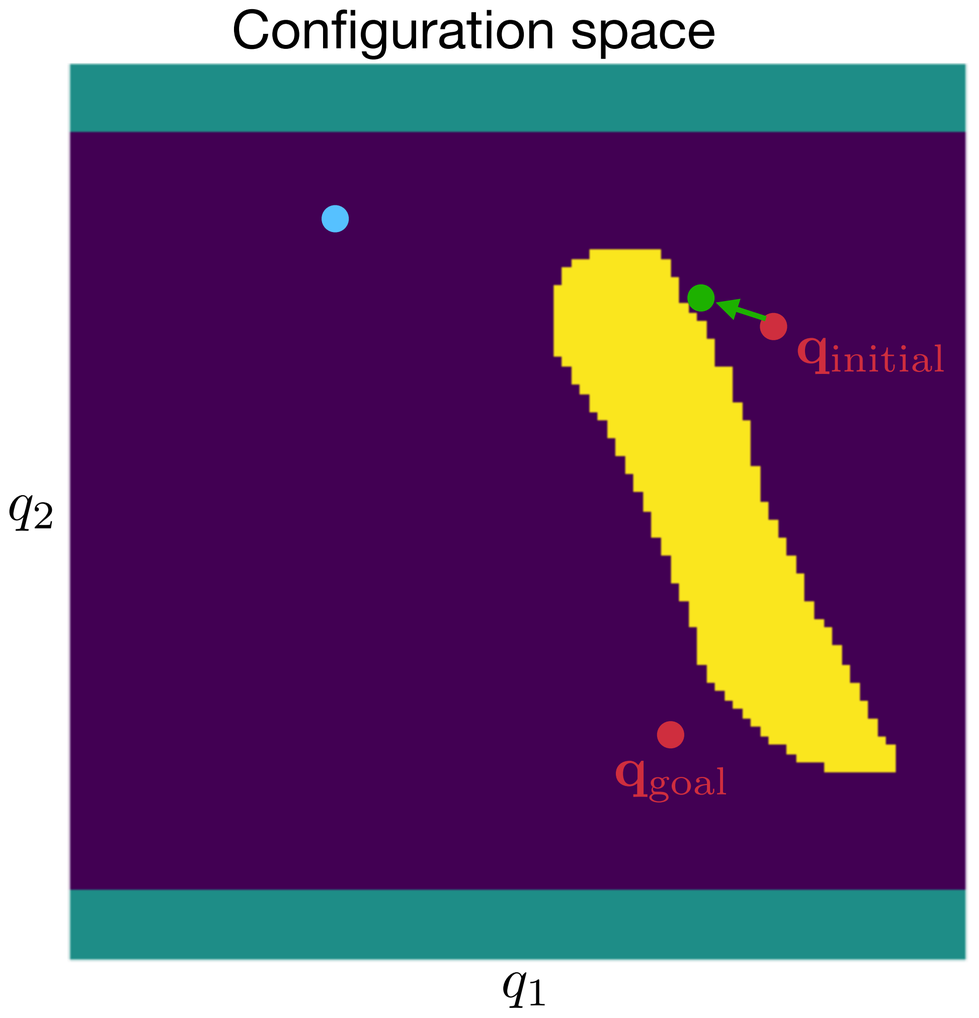
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

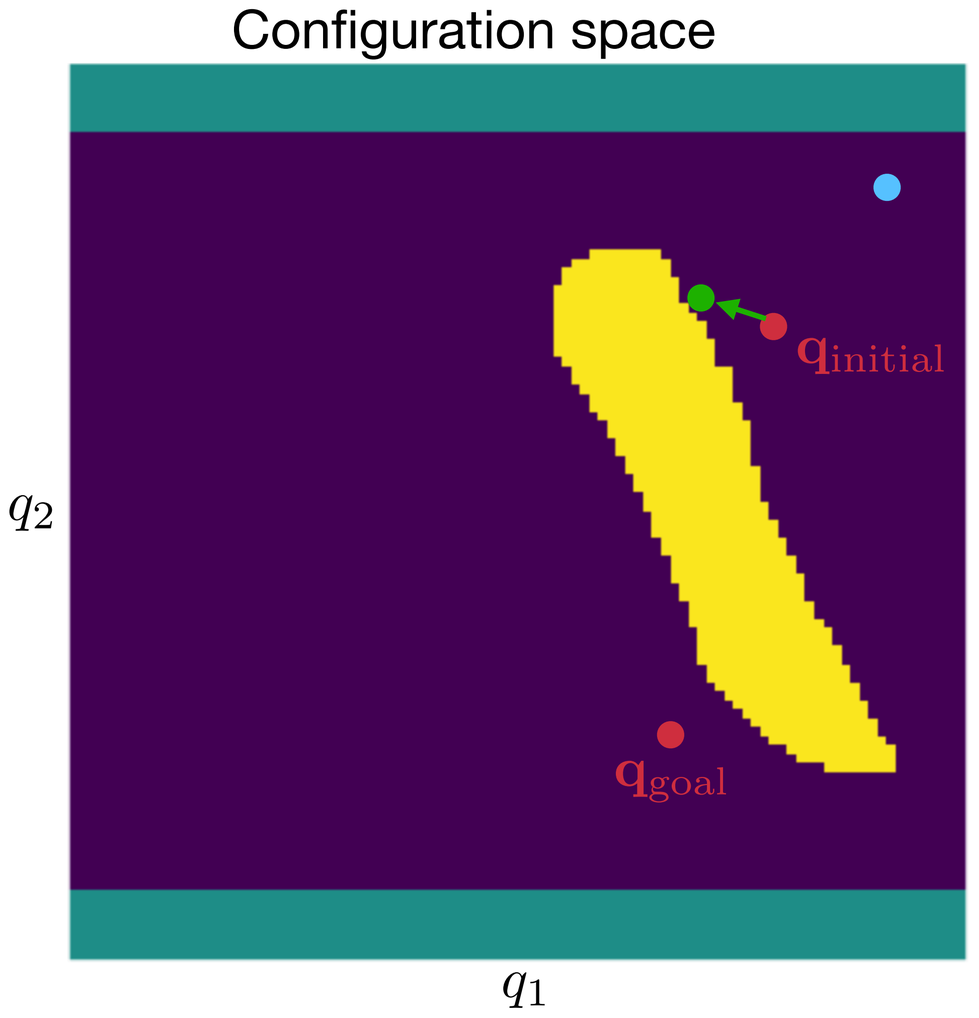
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

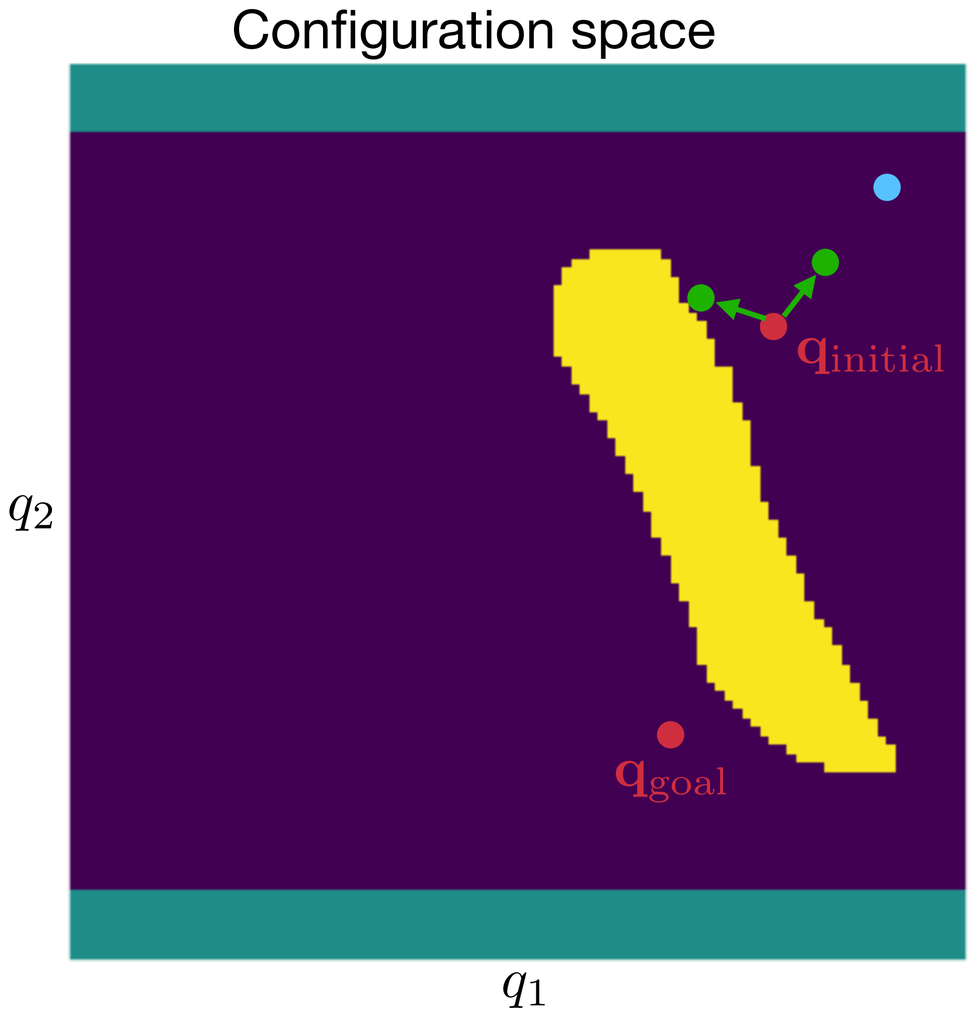
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

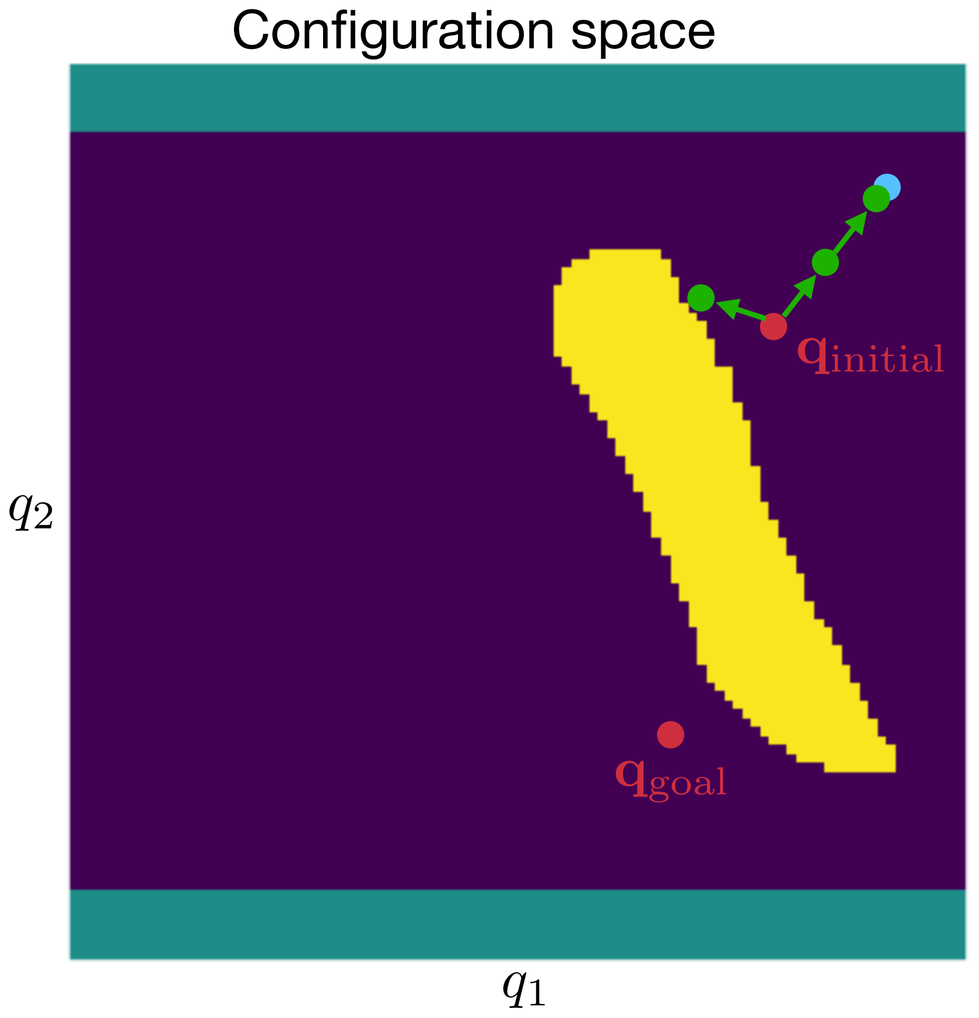
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

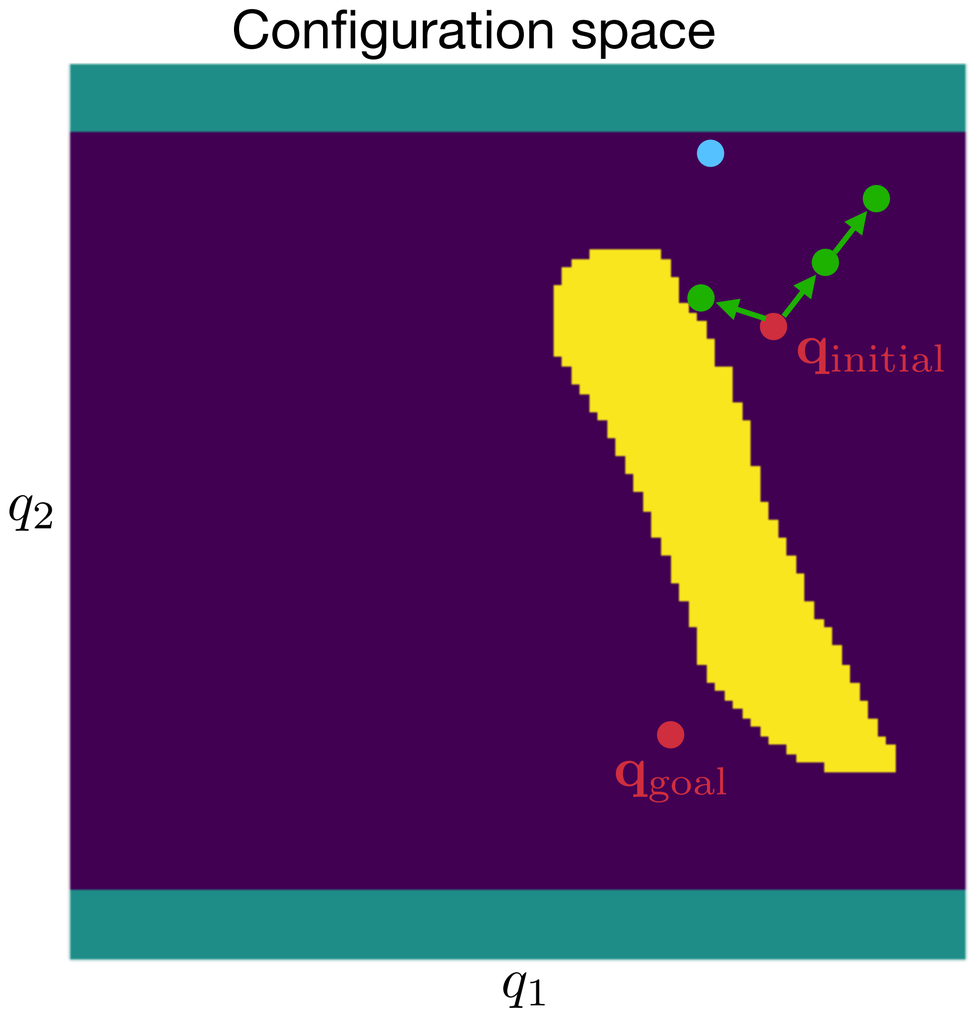
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

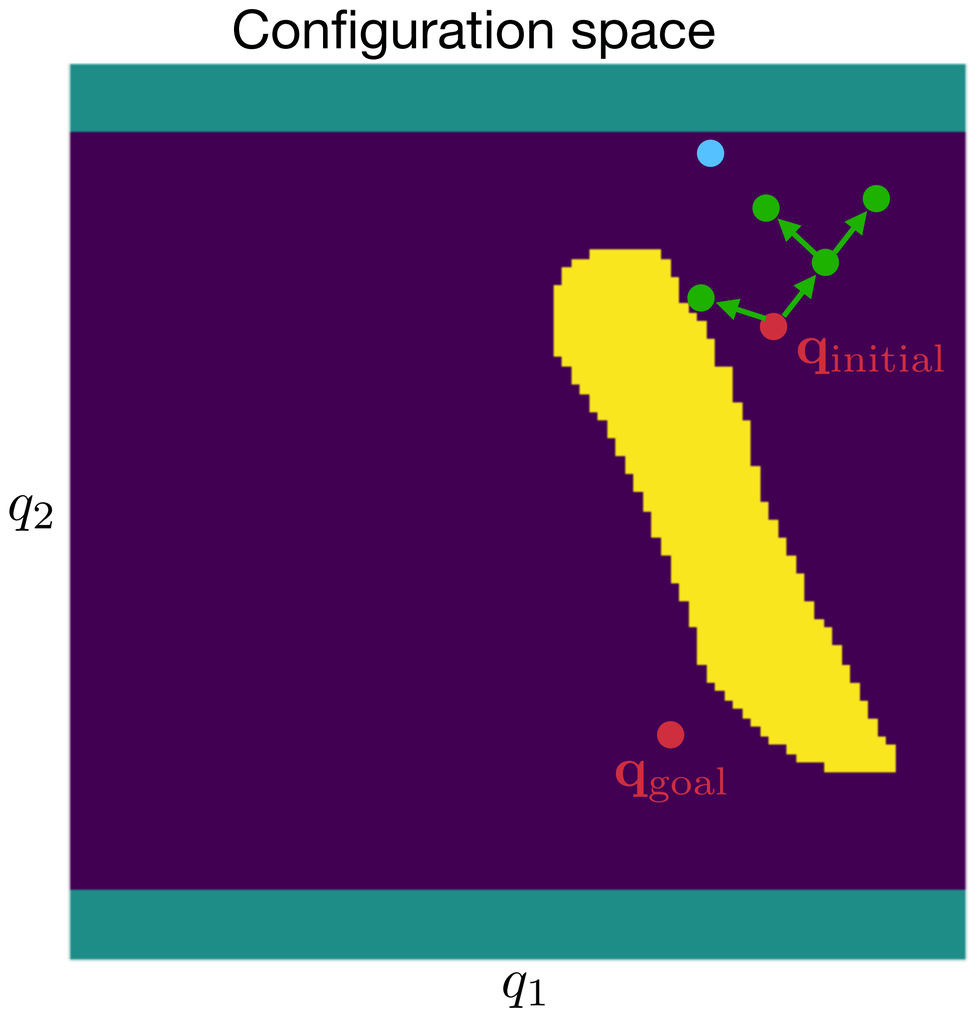
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

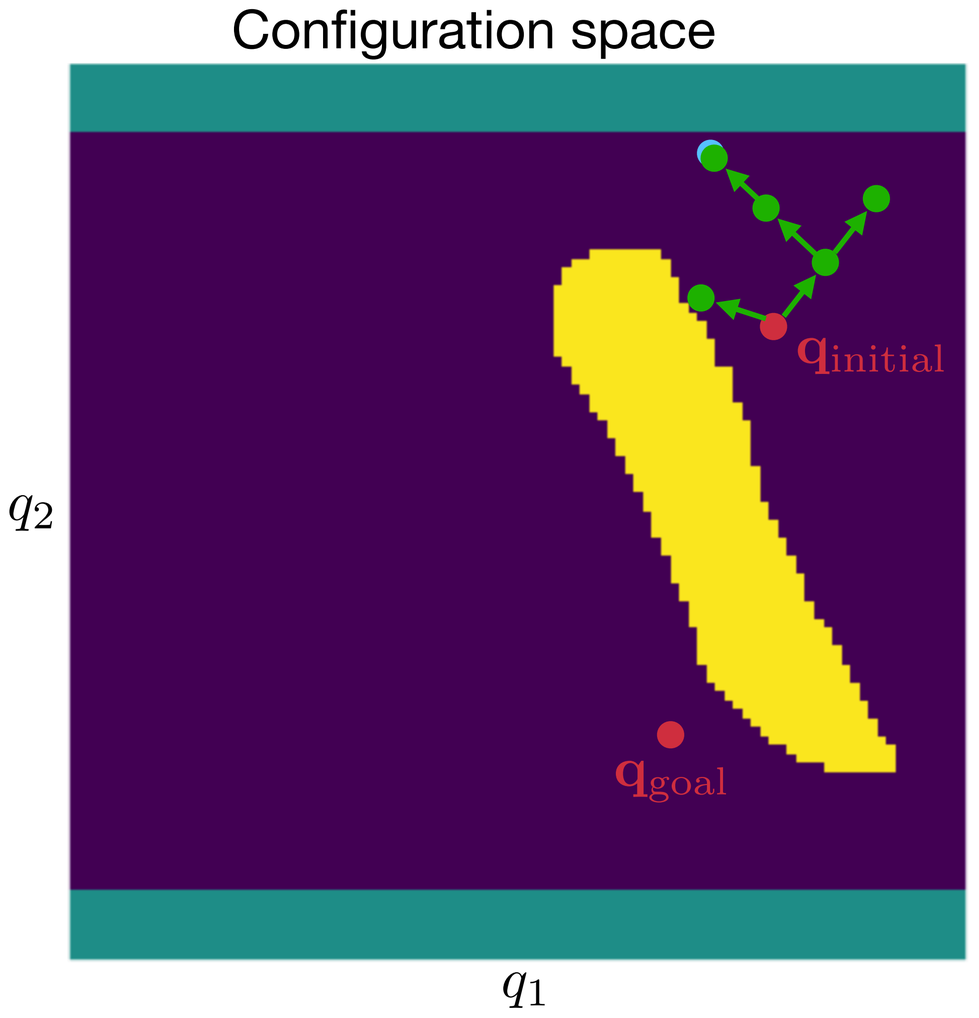
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

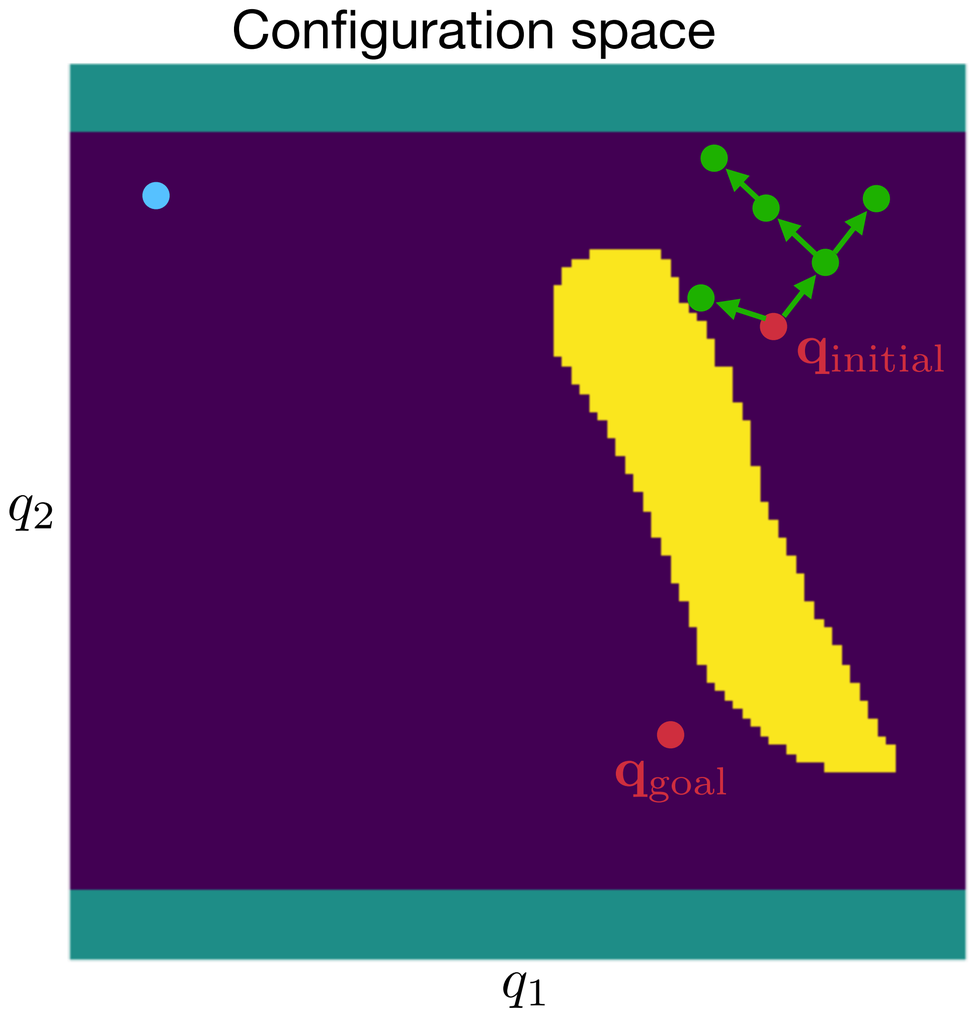
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

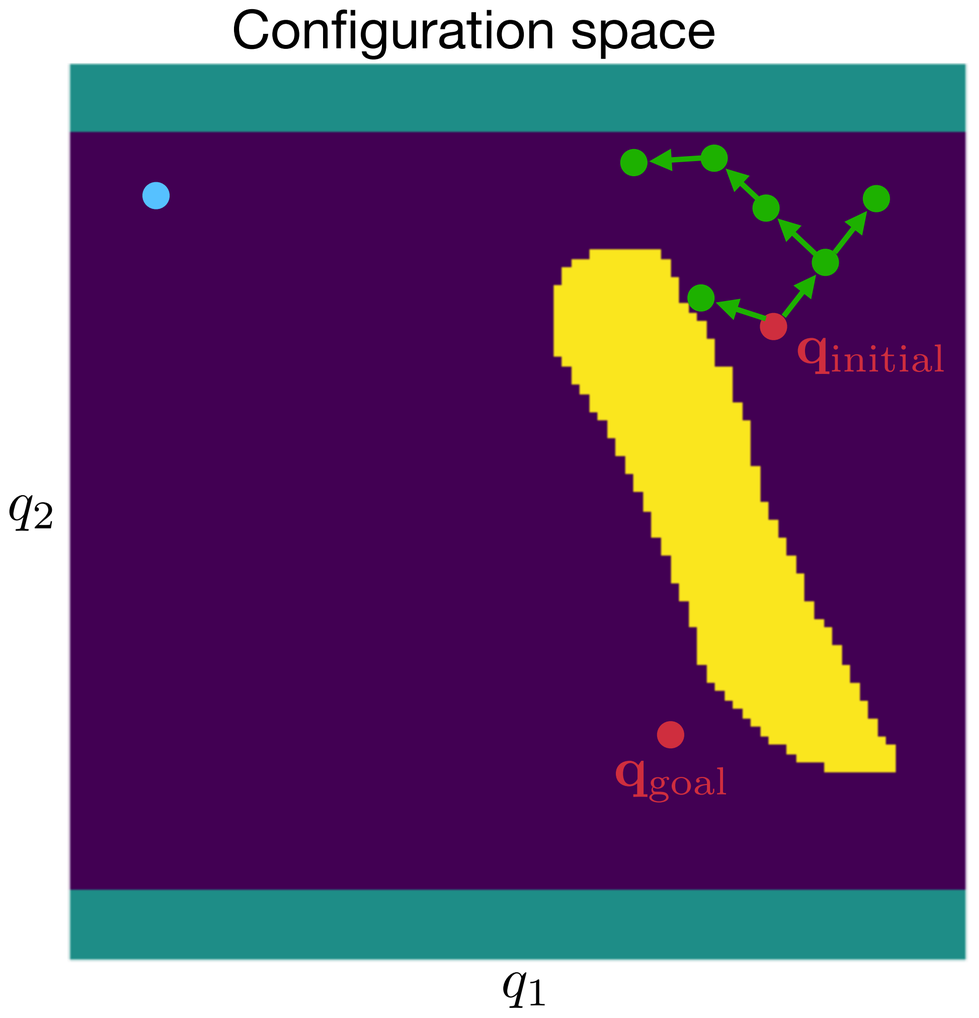
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

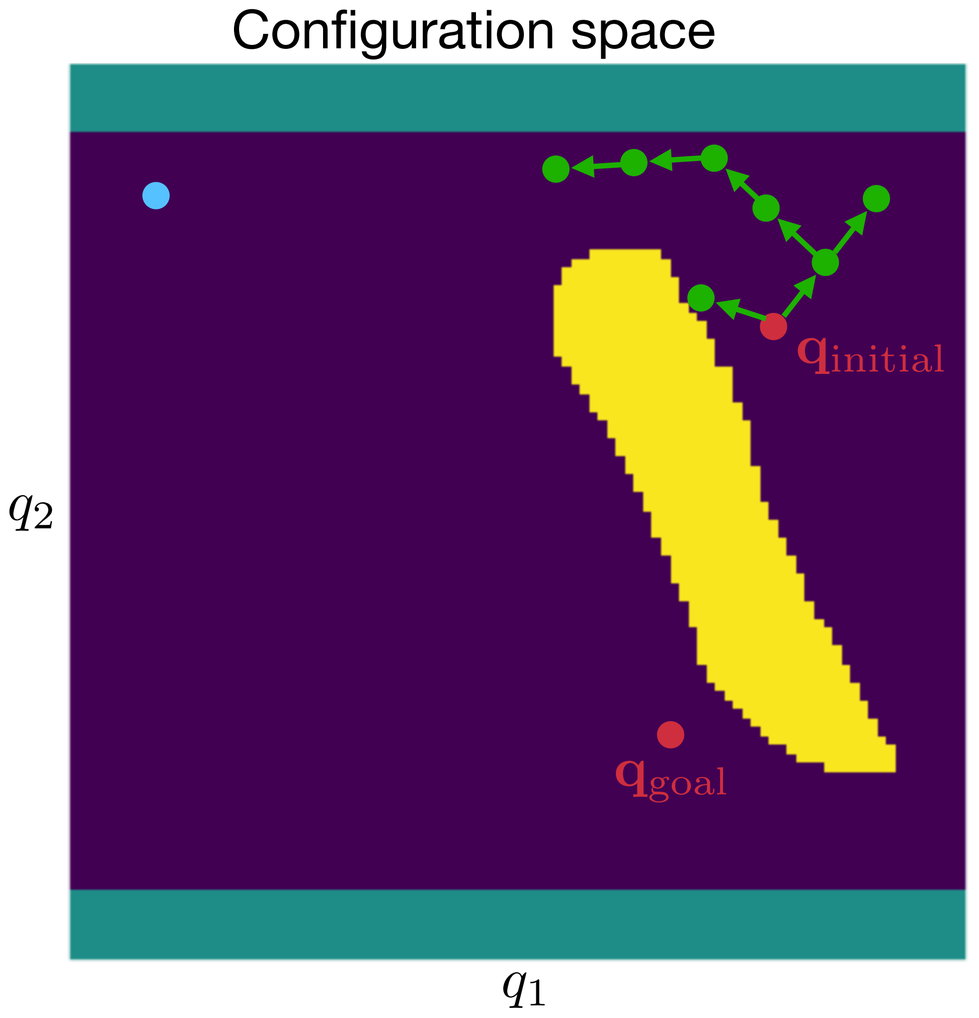
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

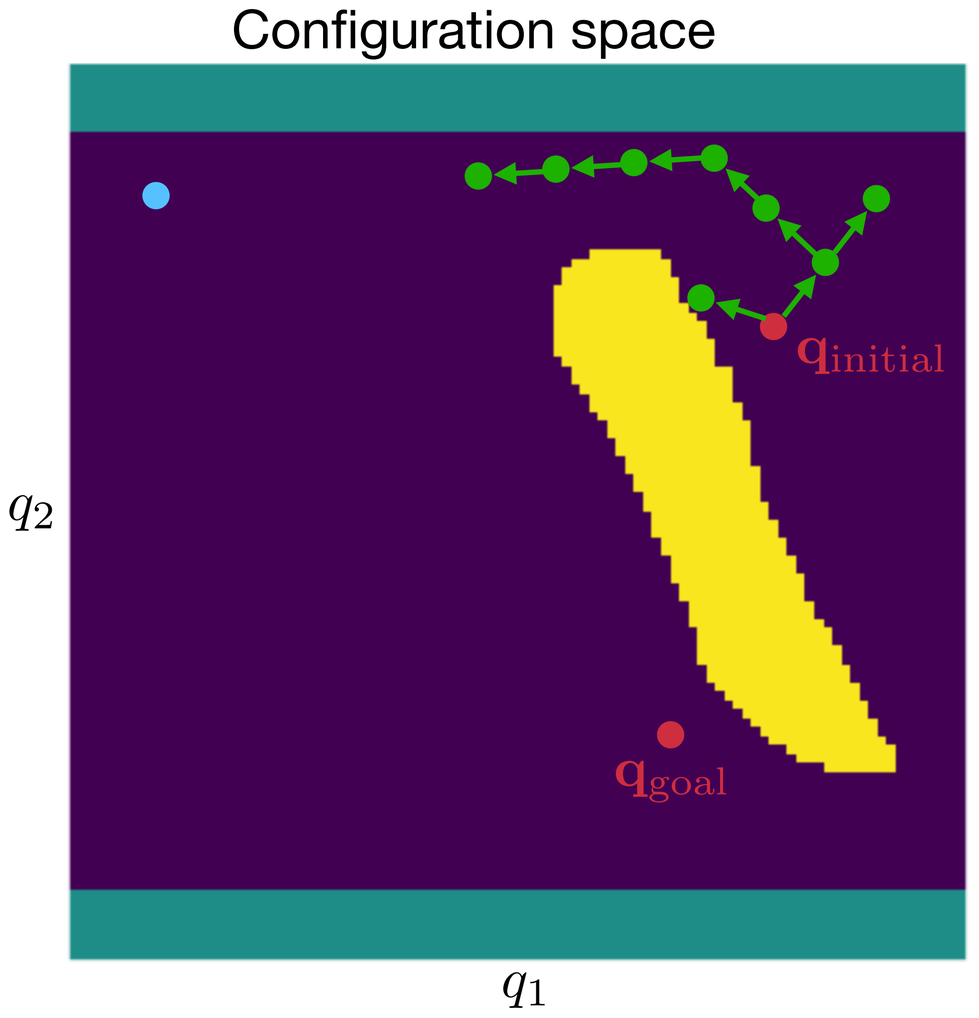
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

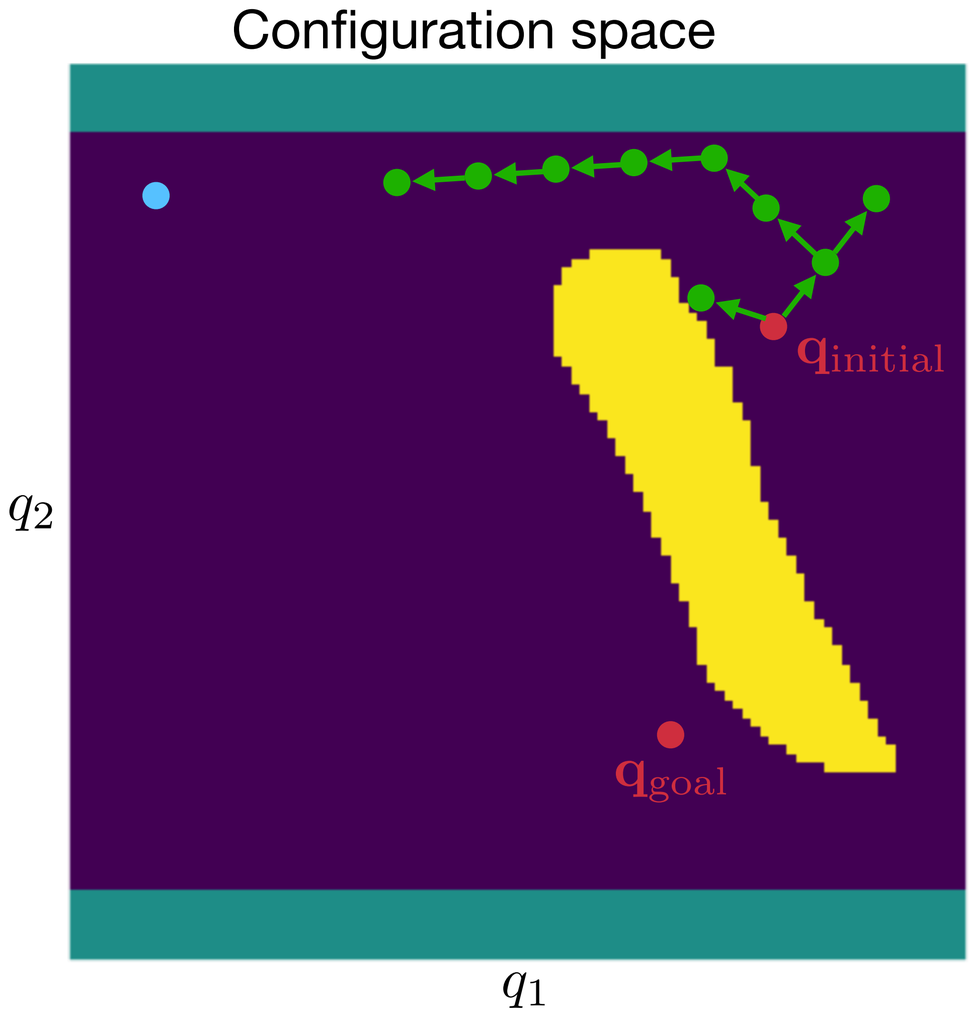
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

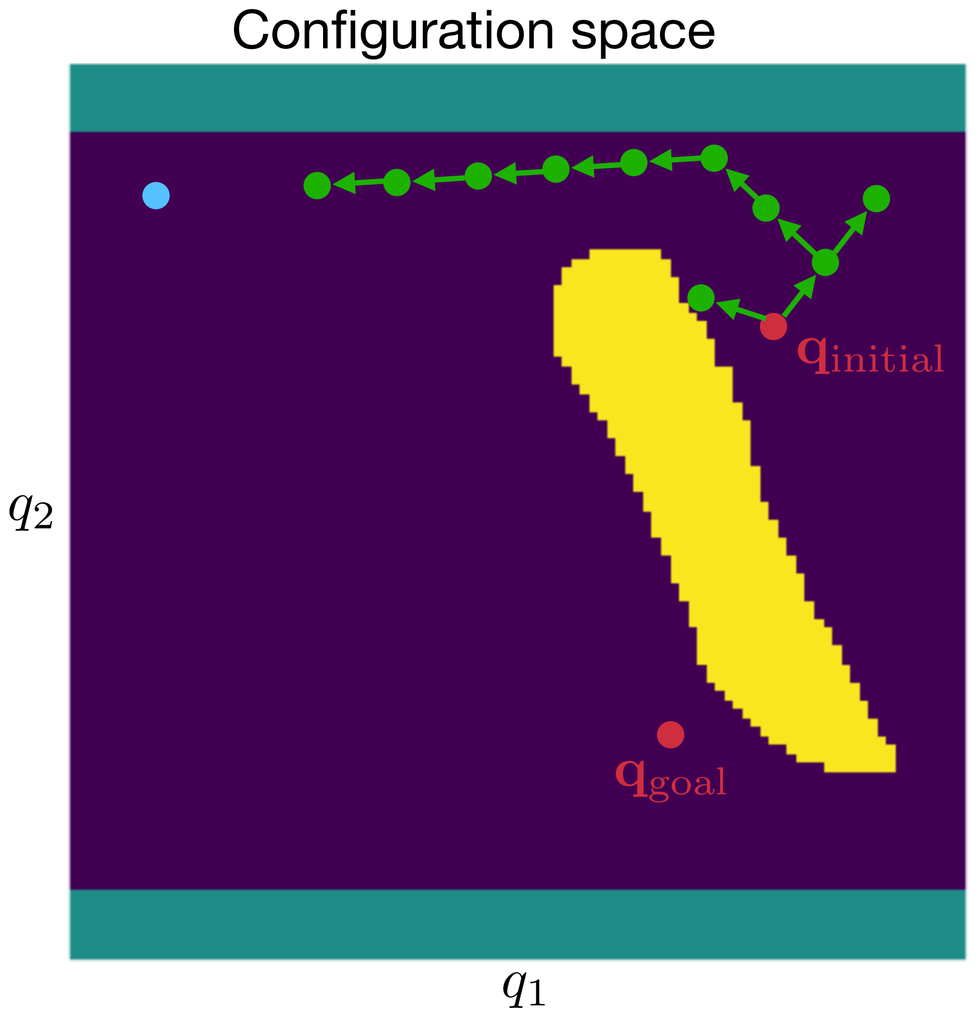
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

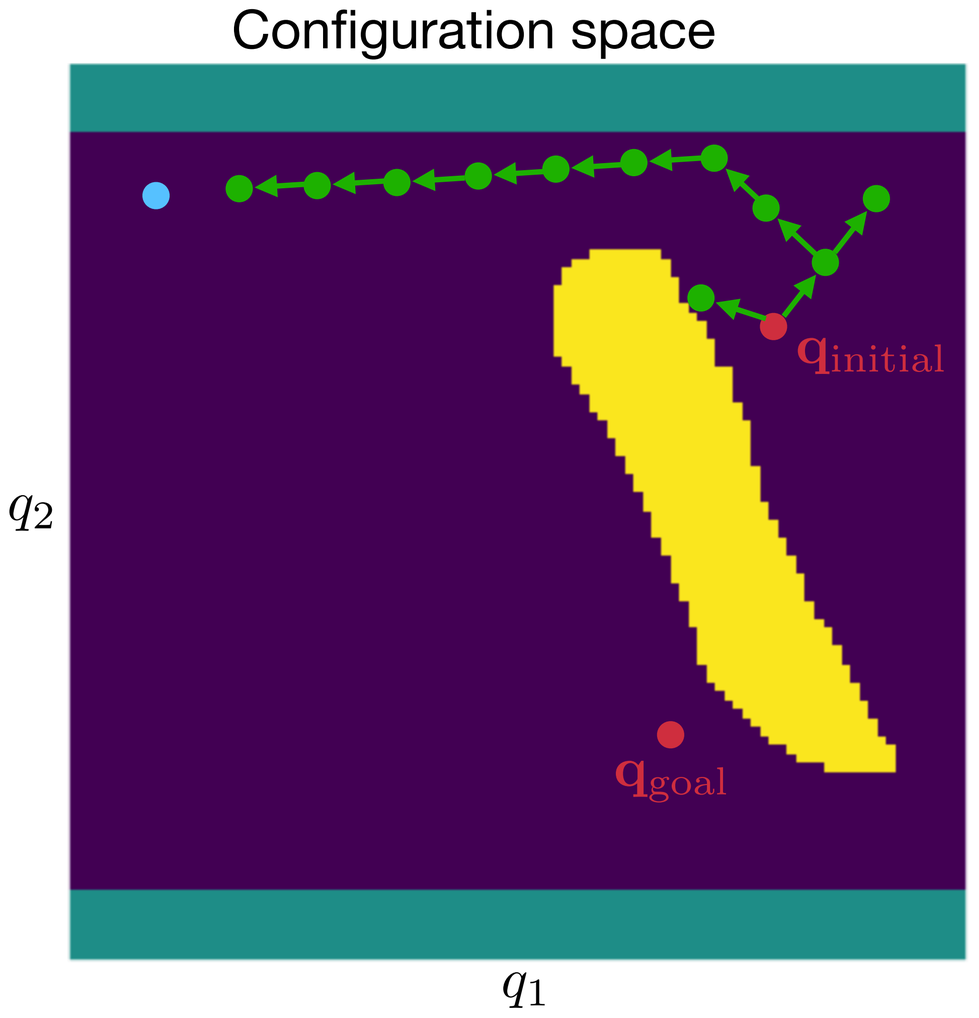
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

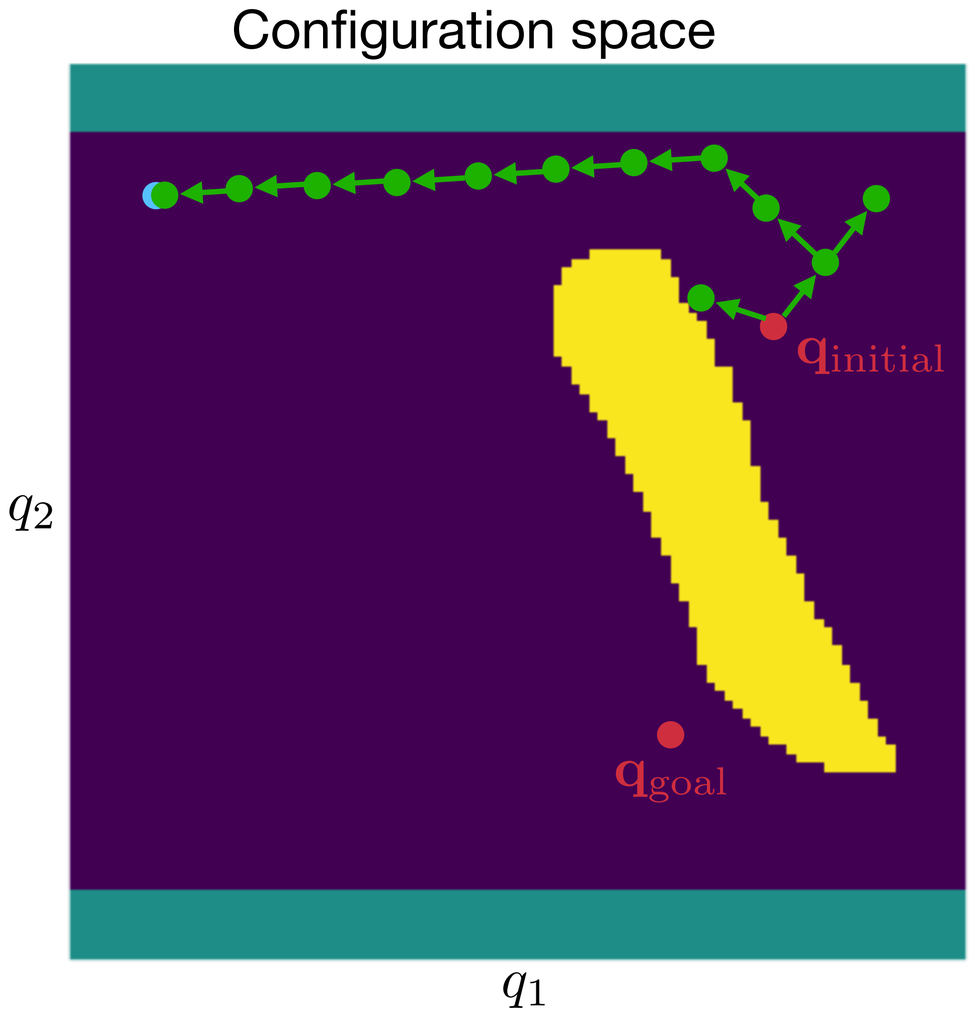
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

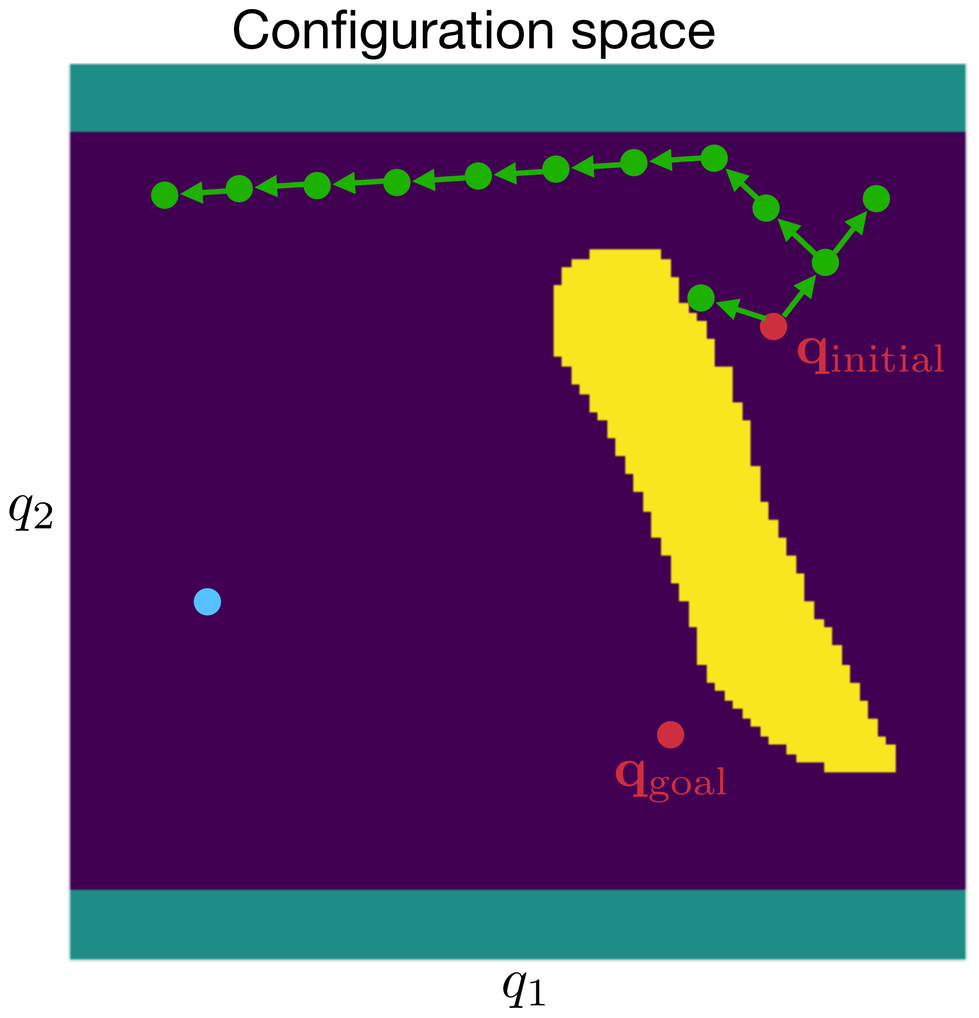
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

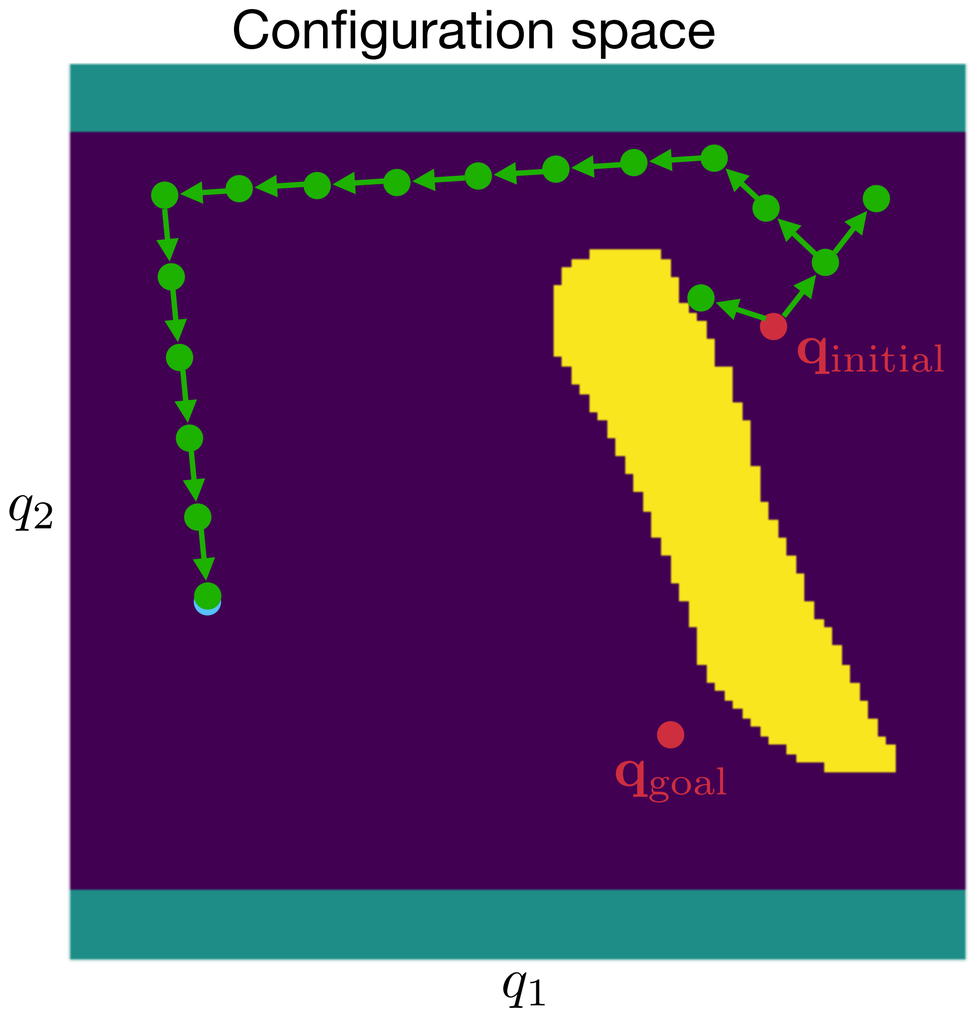
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

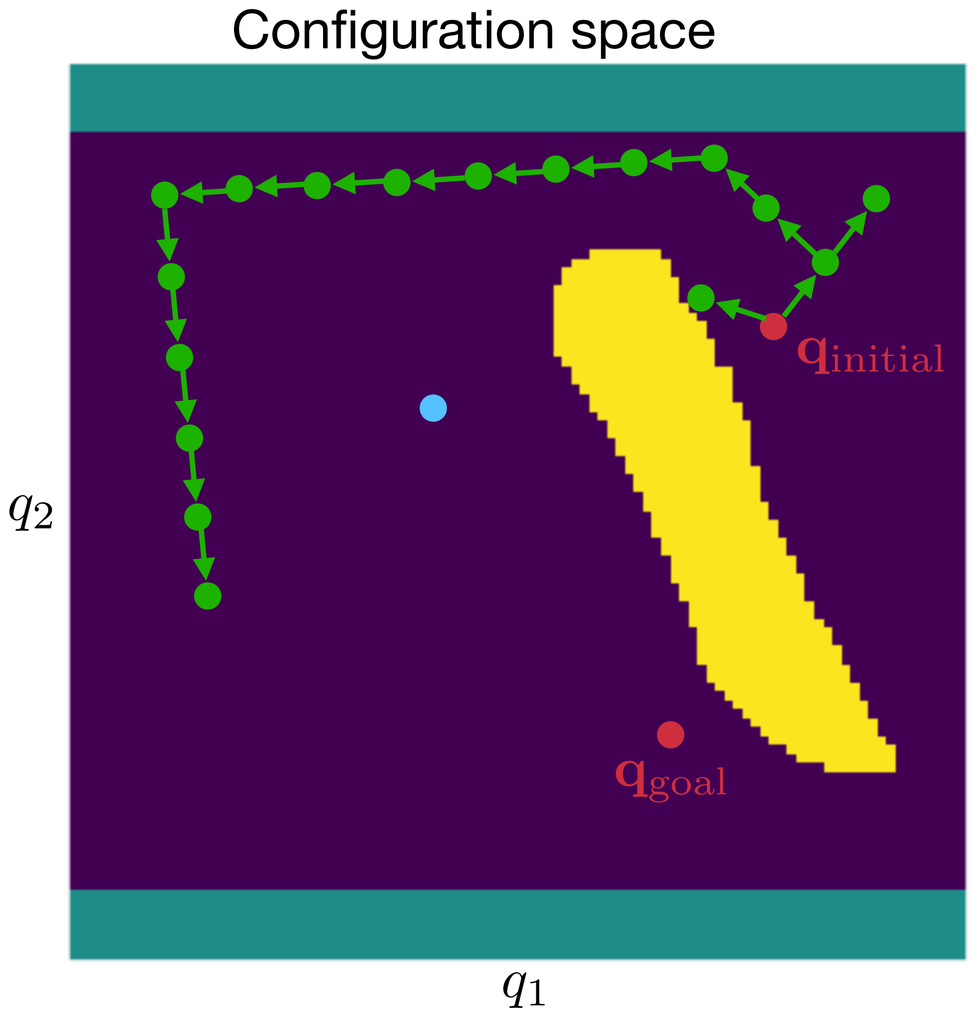
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

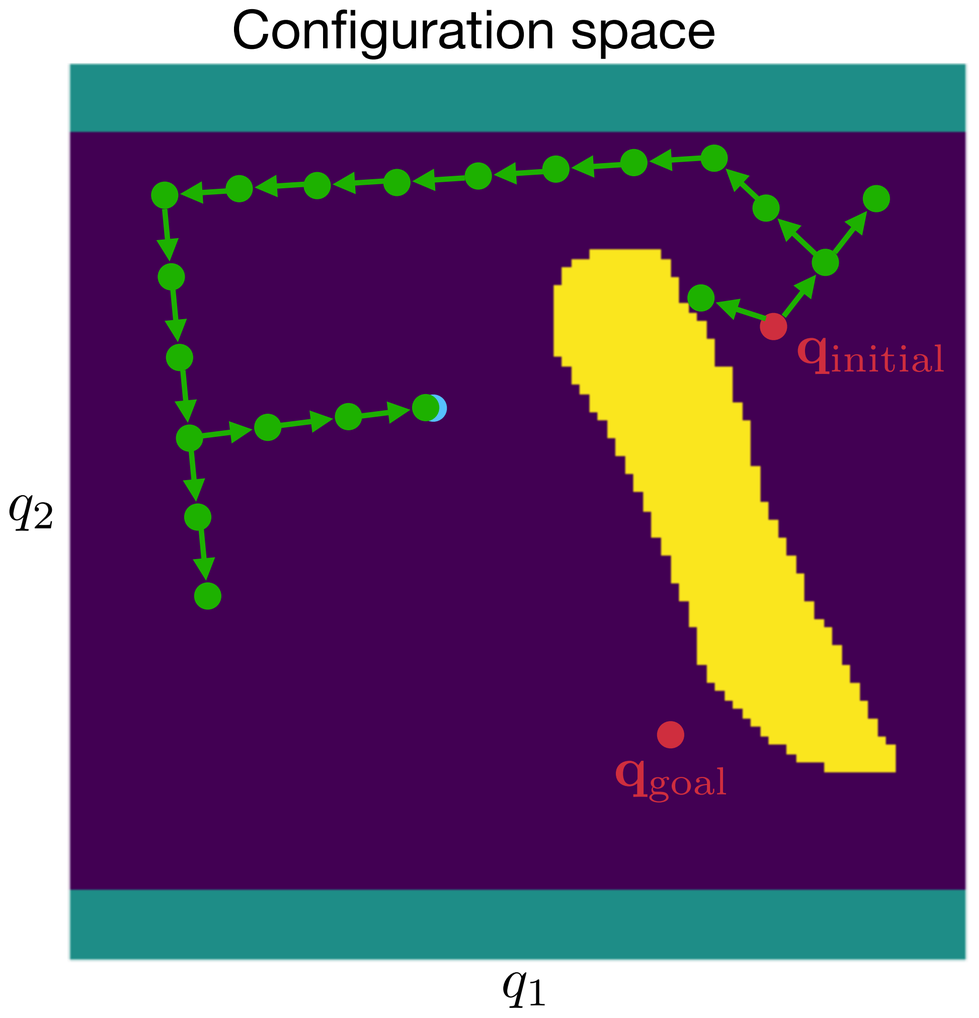
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

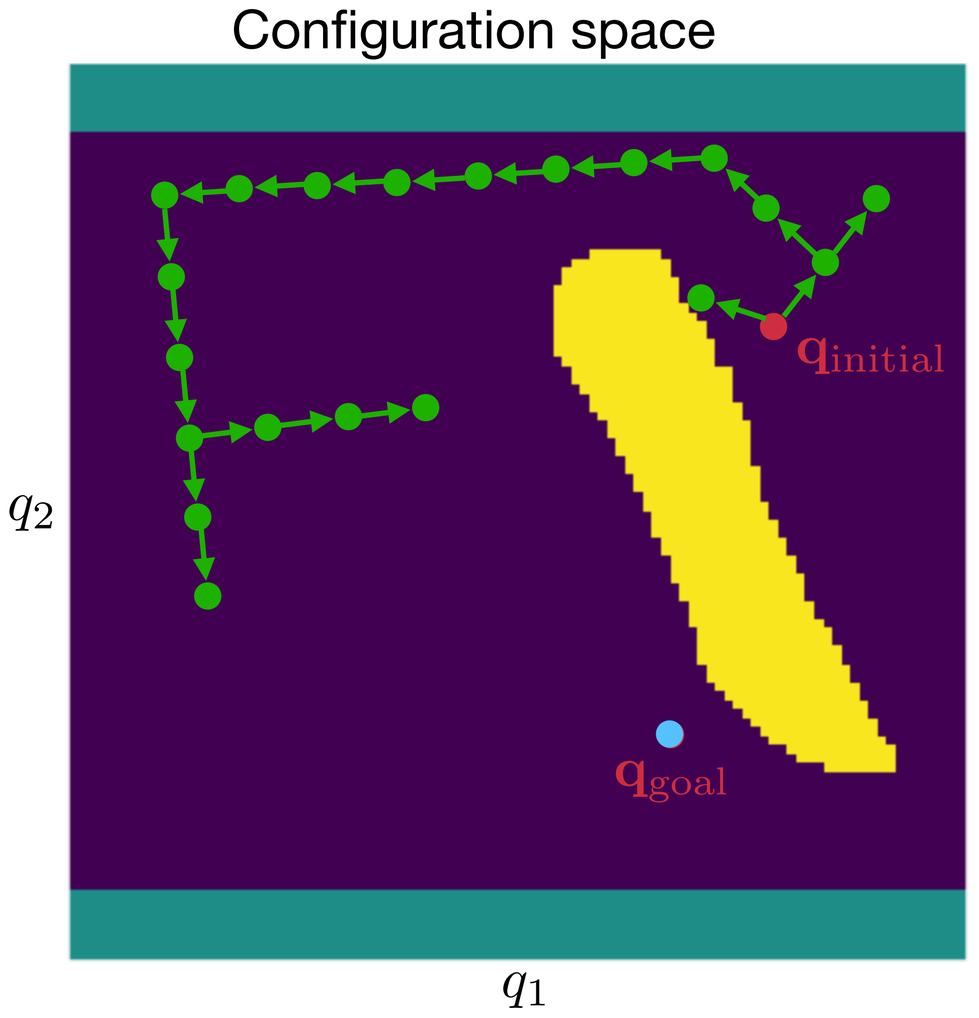
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

2.) RRT-Connect

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Goal Bias

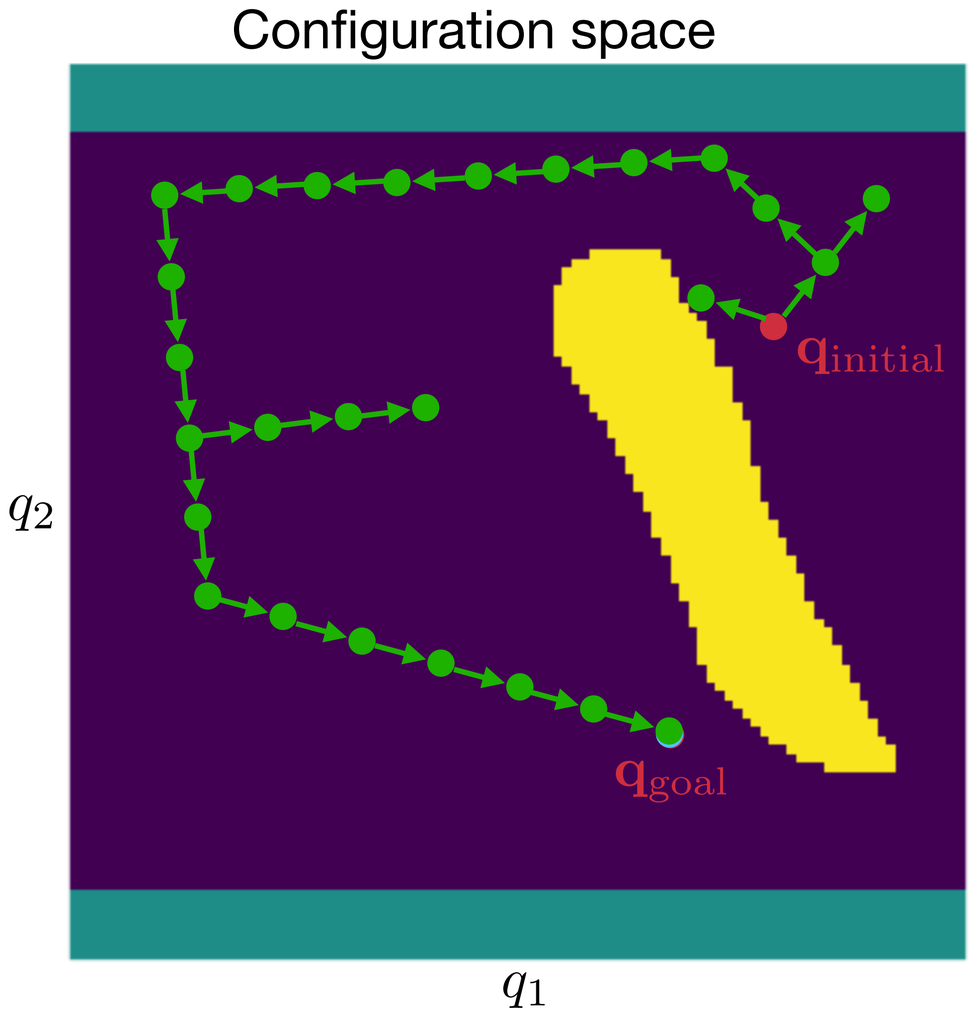
- With probability p_{goal}, sample $\mathbf{q}_{\text{goal}}$ and with probability $(1 - p_{\text{goal}})$, sample random configuration $\mathbf{q}_{\text{rnd}}$

- Find closest node $\mathbf{v}_{\text{closest}}$ in tree

Connect

- Move repeatedly from $\mathbf{v}_{\text{closest}}$ towards sampled node with a fixed step size γ and add new nodes along the way until hitting an invalid configuration

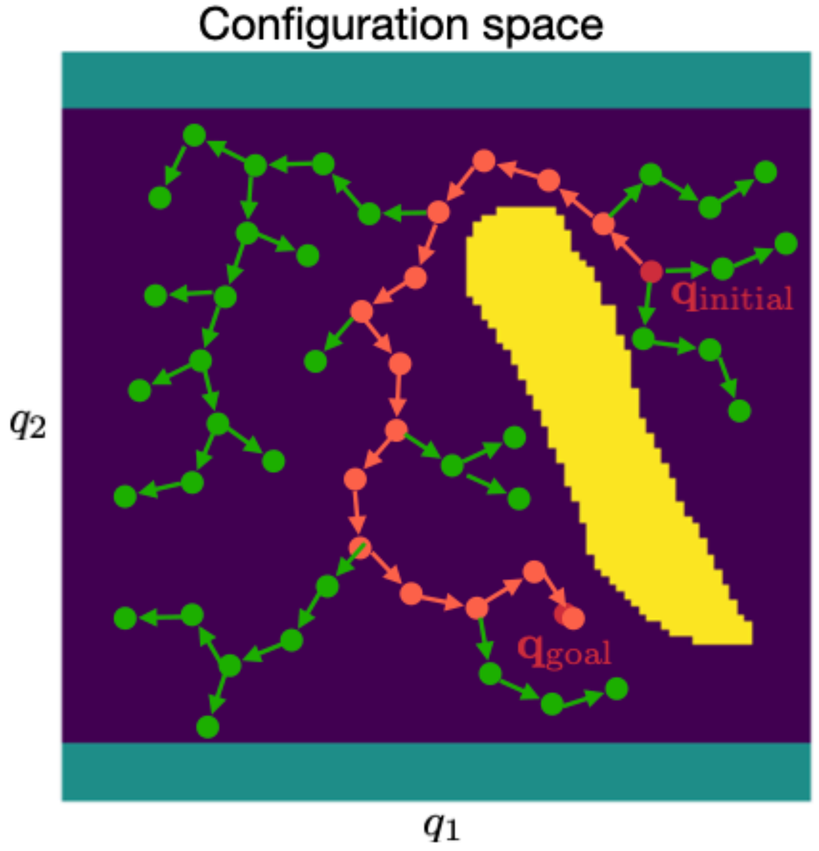
- If distance between new node and is below threshold: $\mathbf{q}_{\text{goal}}$ is found



(Let's assume, we know the goal configuration)

There are actually many more possible improvements…

Are the Problems with path planning in configuration Space solved by RRT?



Are the Problems with path planning in configuration Space solved by RRT?

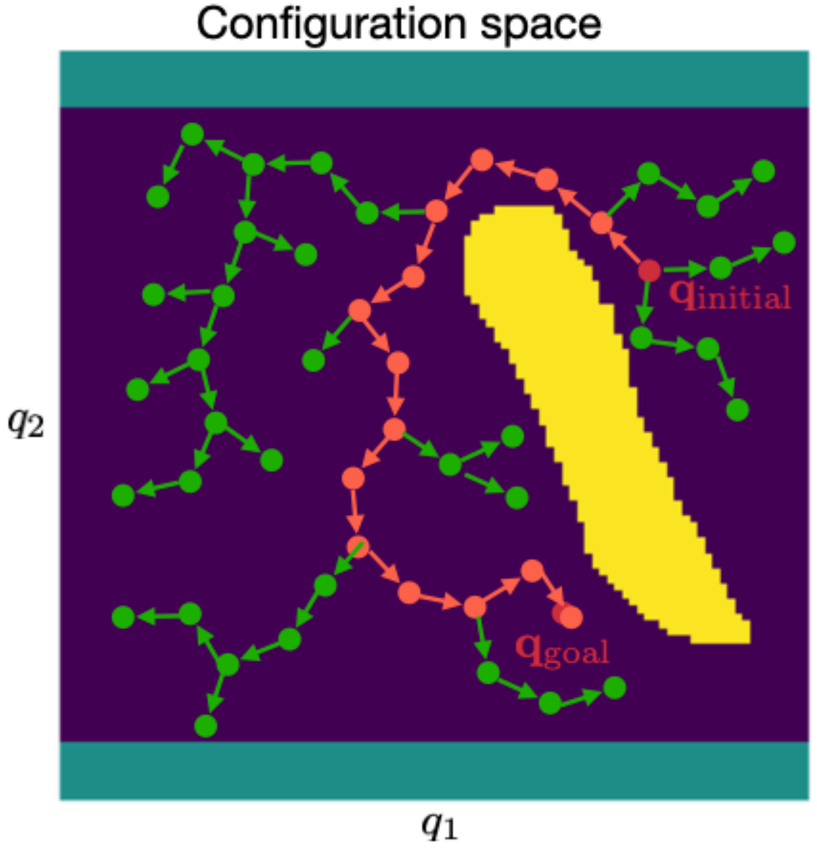
1. Projecting the Workspace into the C-Space is computationally expensive!

   -> We cannot project the whole Workspace!

RRT:
We only project for the sampled configurations between the Workspace and the Configuration space!

2. The goal configuration $\mathbf{q}_{\text{goal}}$ is usually <u>not</u> known!

   -> How to compute a path?

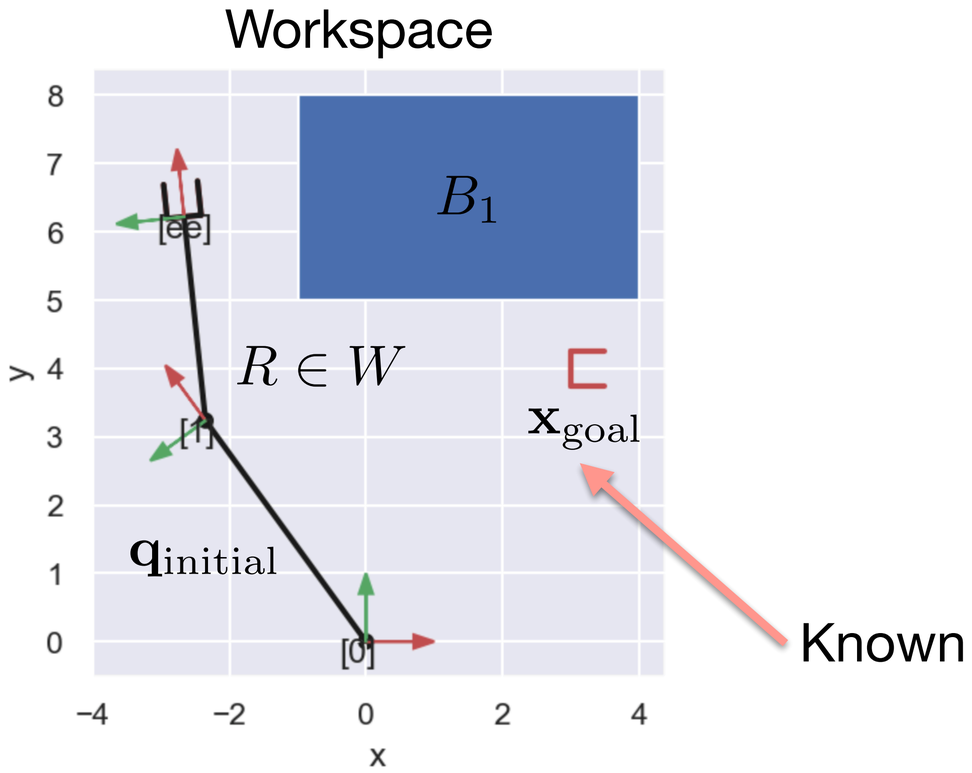


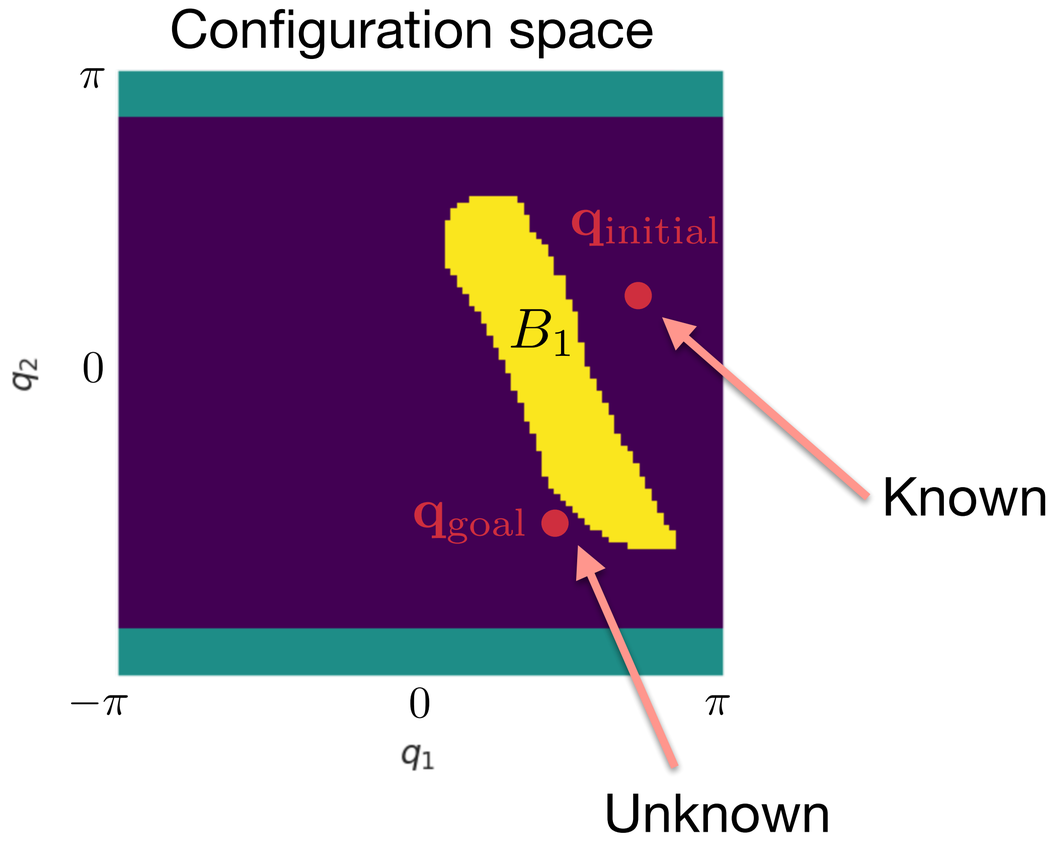
How to compute a path to a goal pose?

In end effector space

How to compute a path to a goal pose?

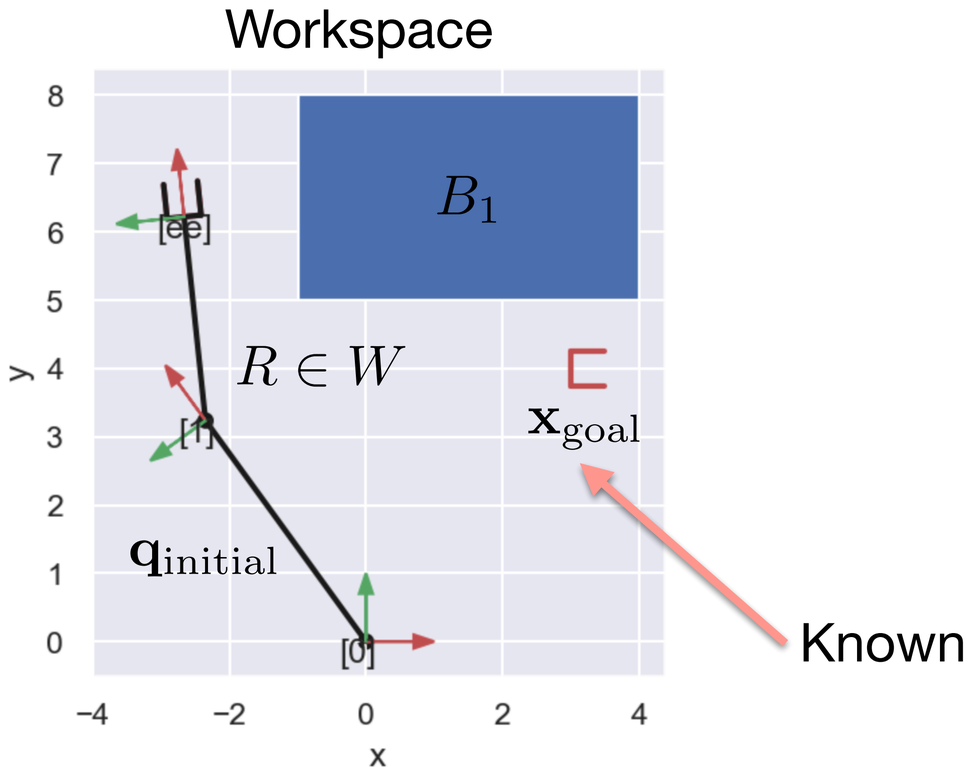
In end effector space

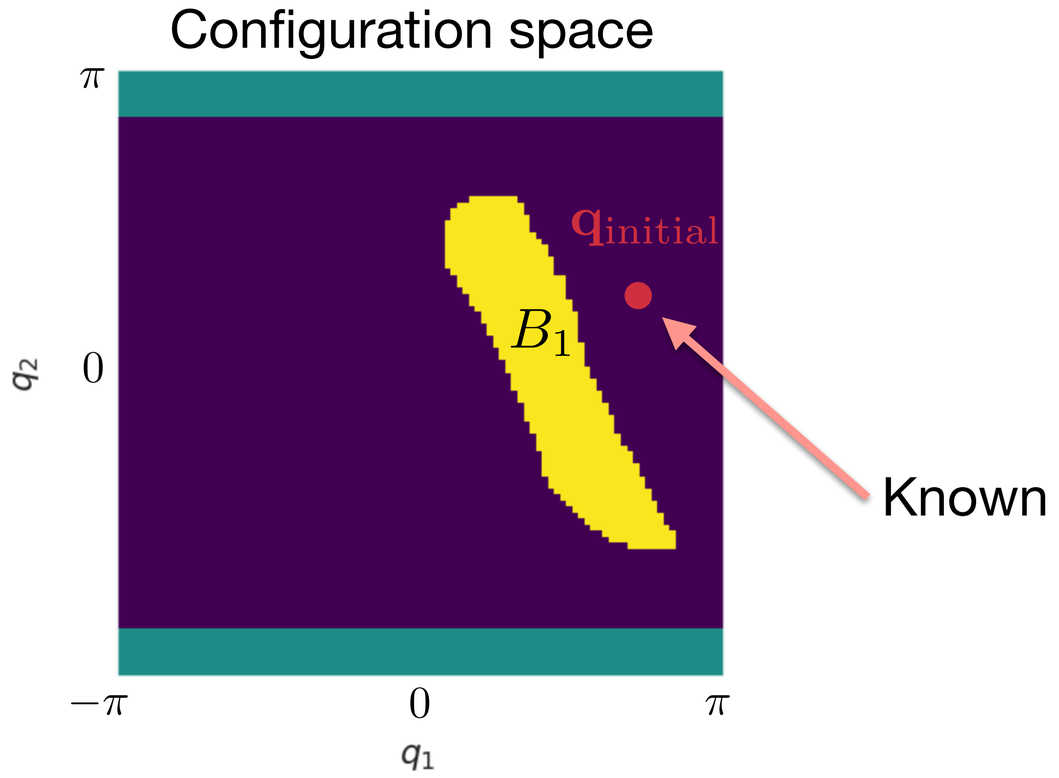




How to compute a path to a goal pose?

In end effector space

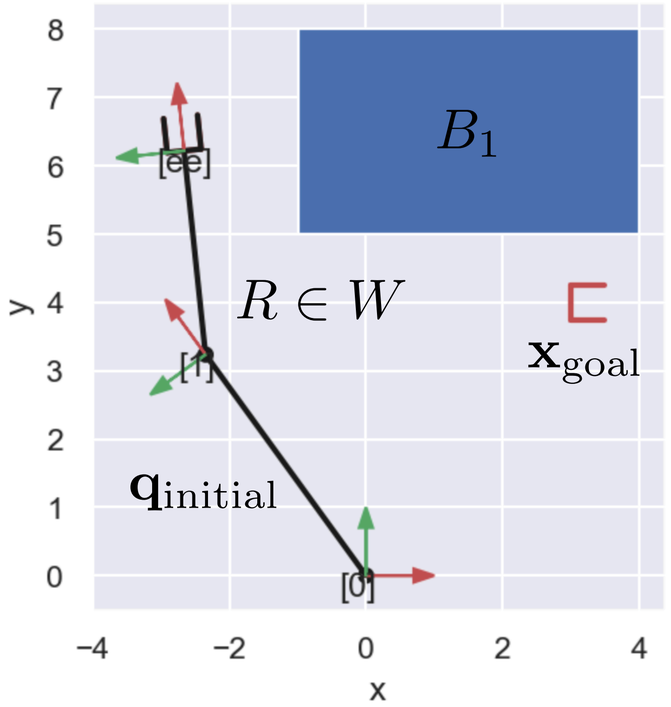




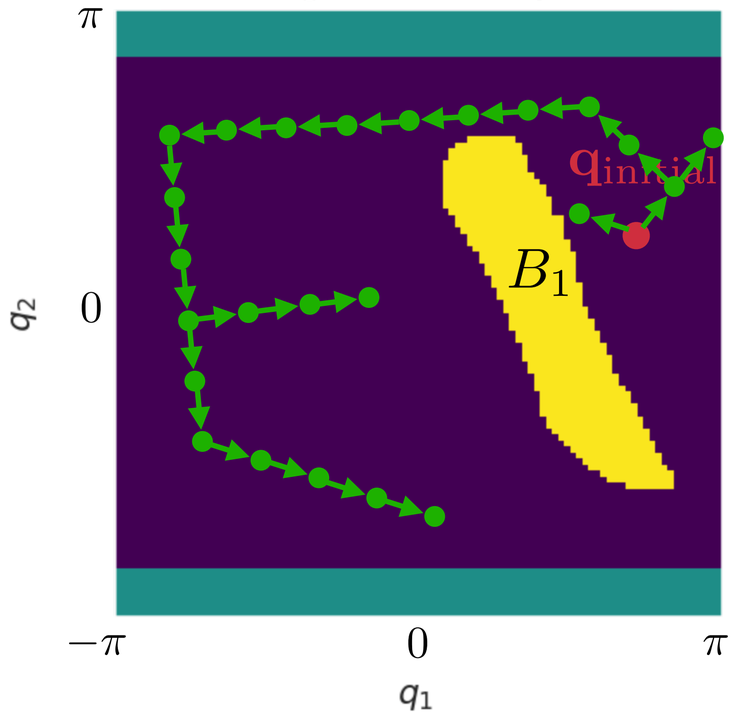
How to compute a path to a goal pose?

In end effector space

Workspace

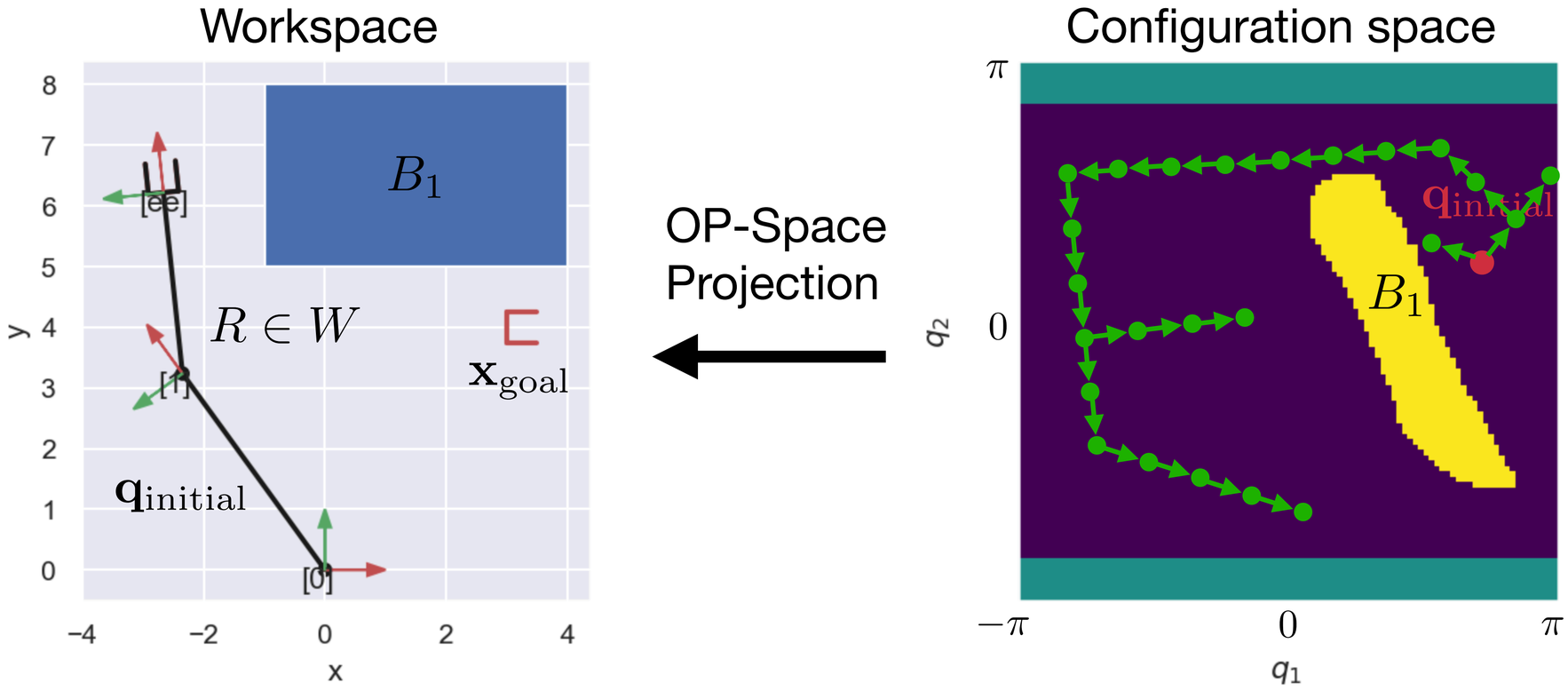


Configuration space



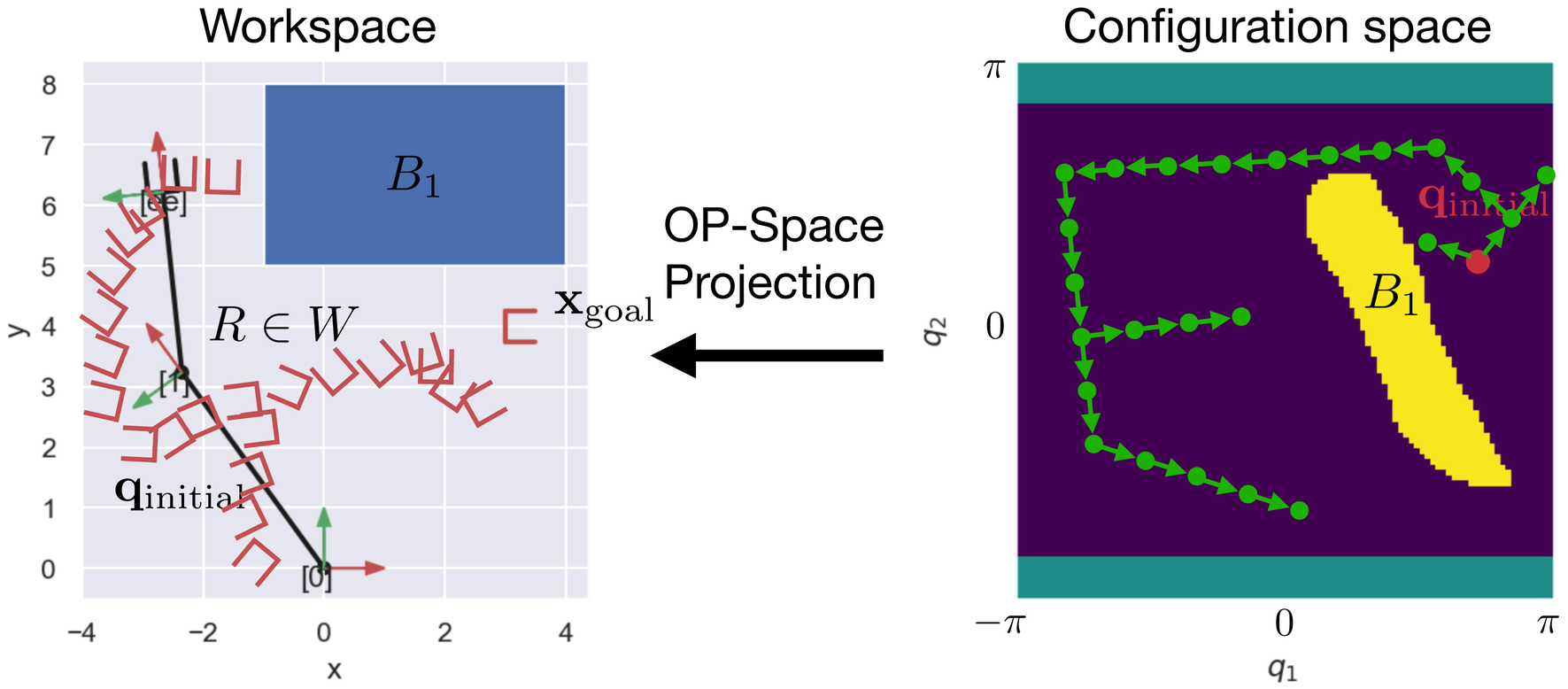
How to compute a path to a goal pose?

In end effector space



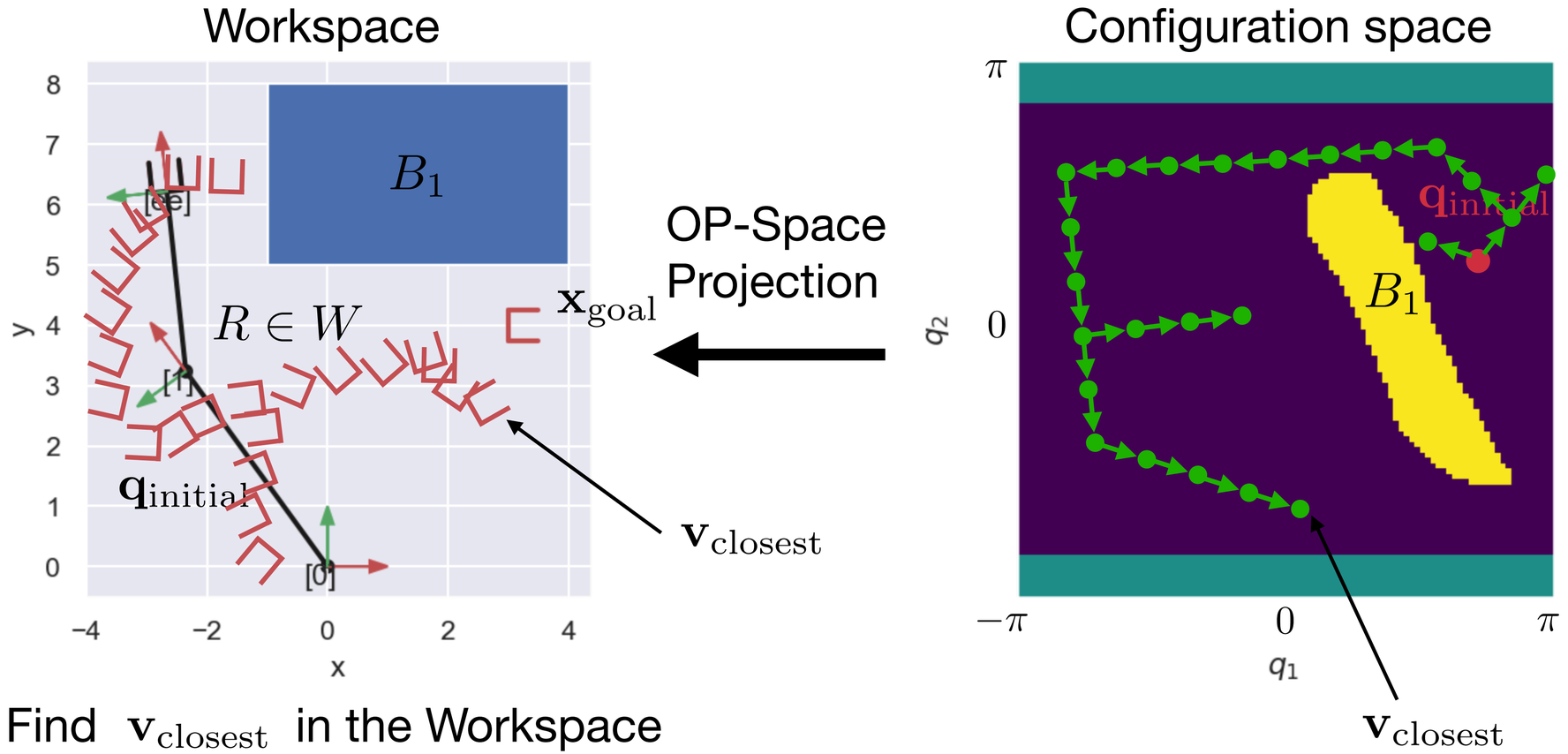
How to compute a path to a goal pose?

In end effector space



How to compute a path to a goal pose?

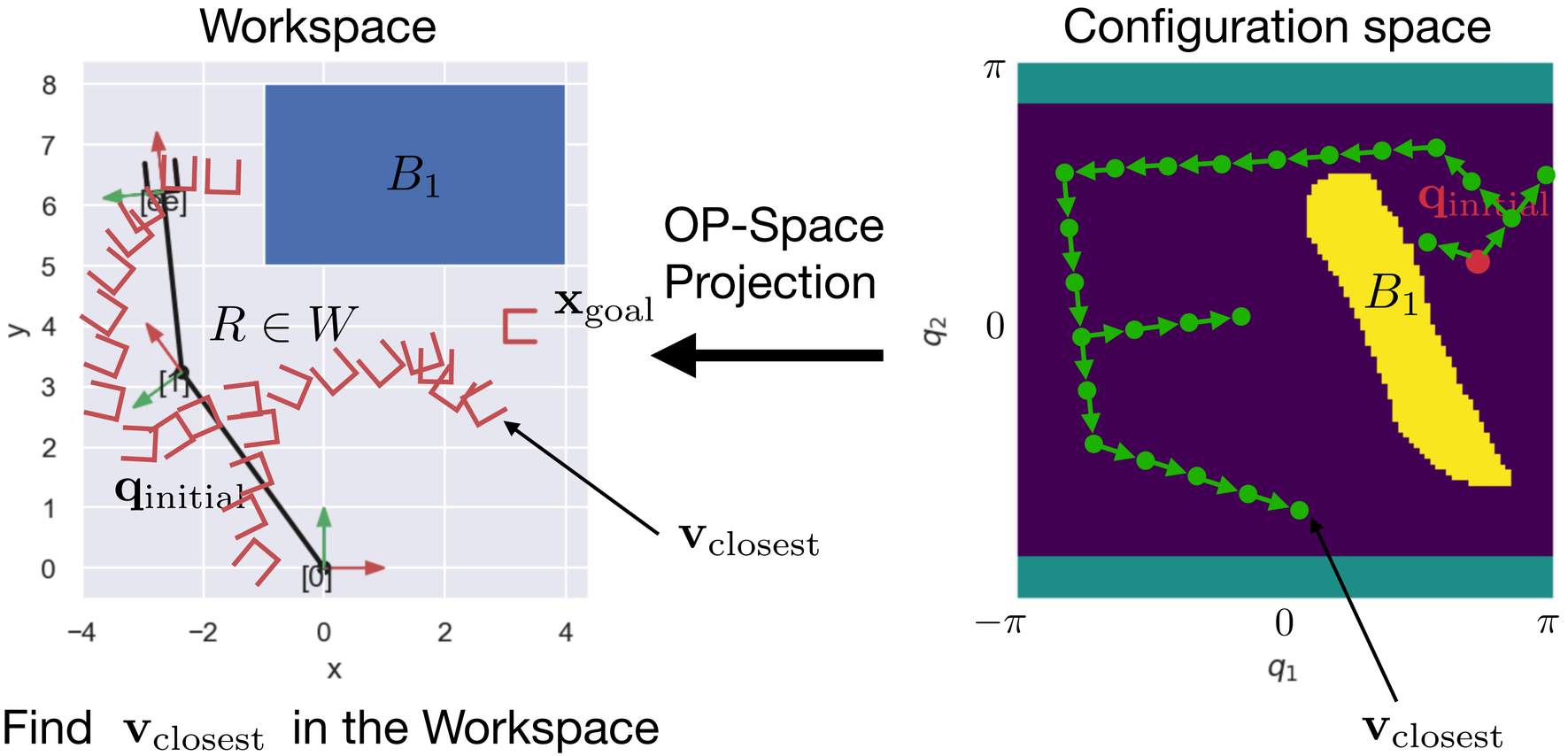
In end effector space



Find $\mathbf{v}_{\text{closest}}$ in the Workspace

How to compute a path to a goal pose?

In end effector space

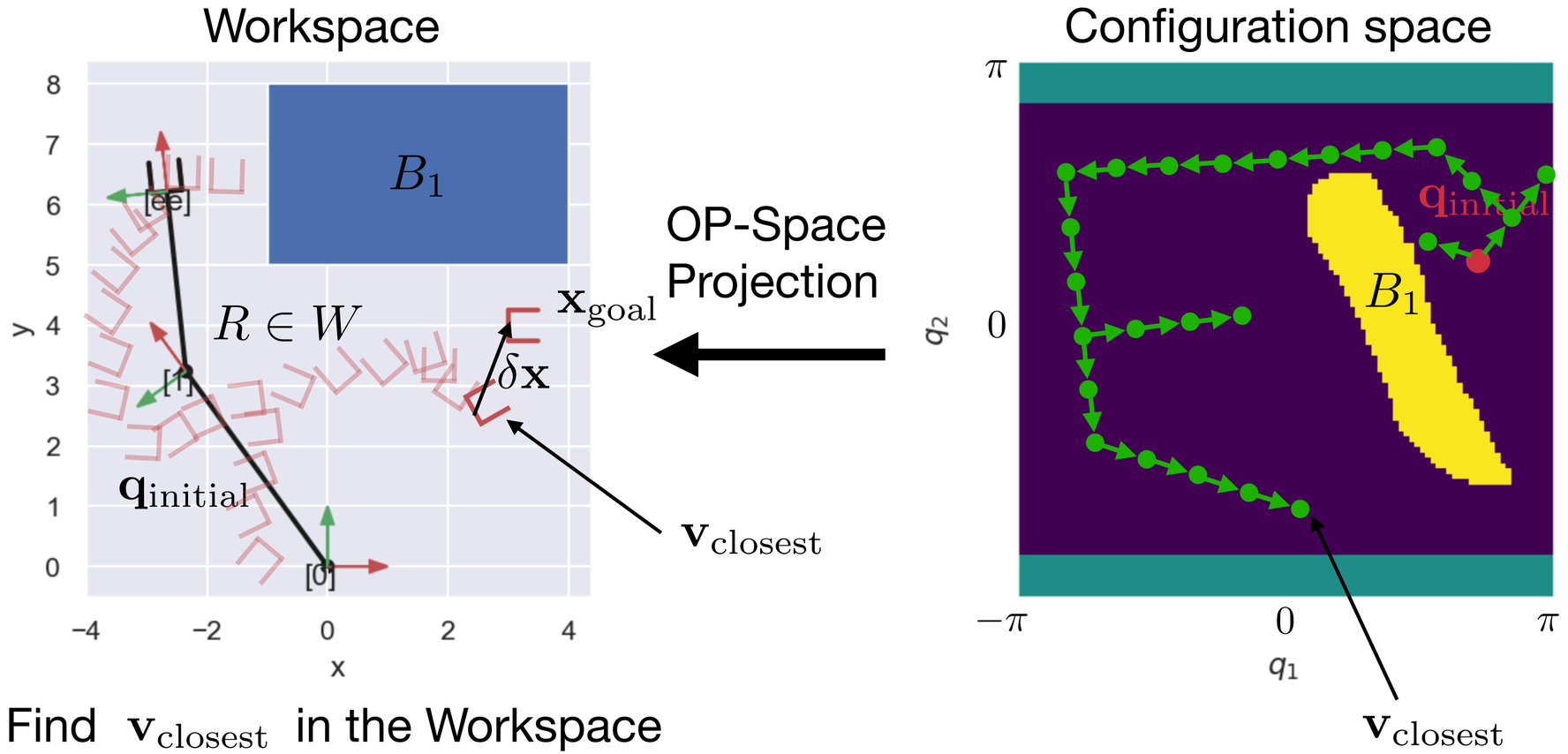


Find $\mathbf{v}_{\text{closest}}$ in the Workspace

Compute step in Workspace $\delta\mathbf{x} = \gamma(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{closest}})$

How to compute a path to a goal pose?

In end effector space

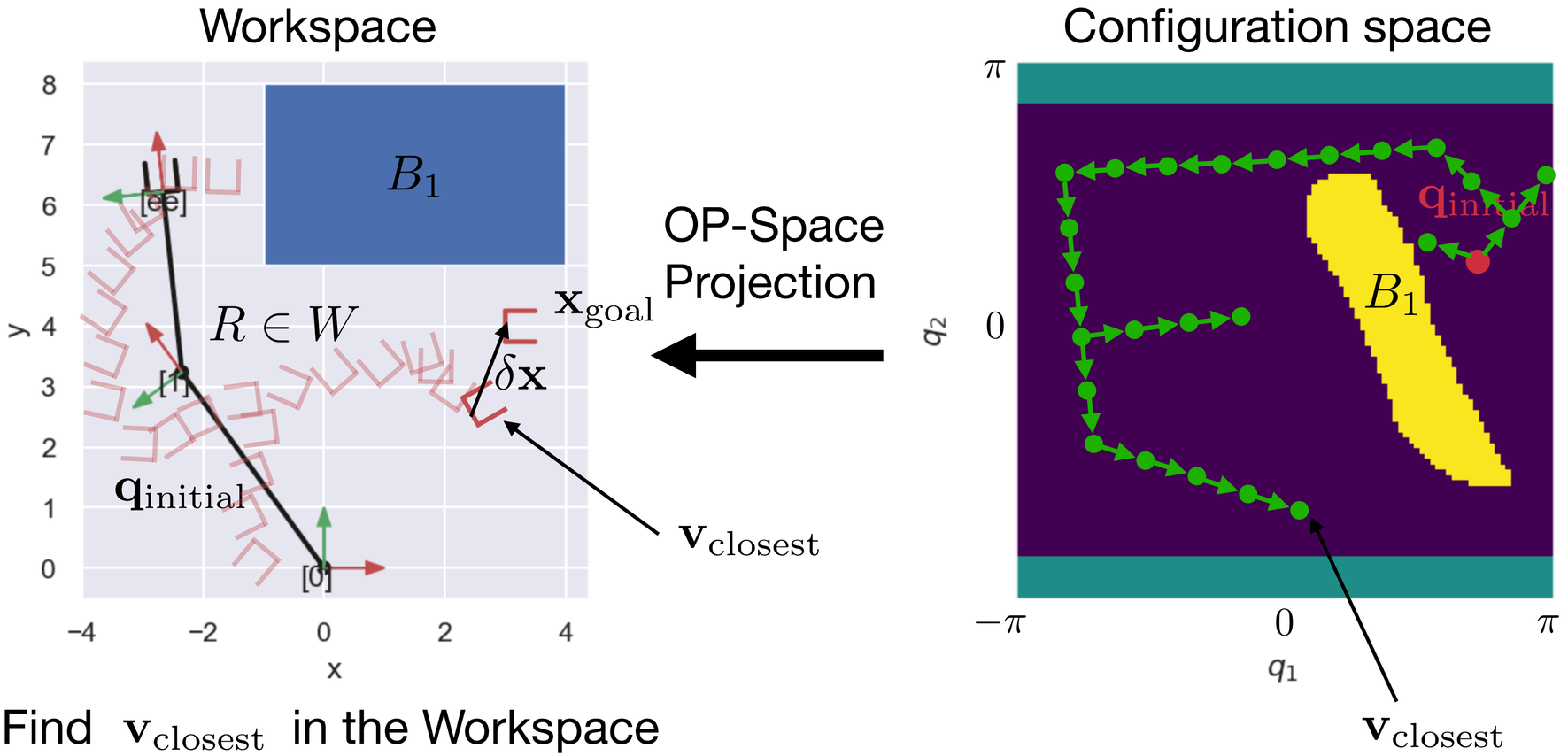


Find $\mathbf{v}_{\text{closest}}$ in the Workspace

Compute step in Workspace $\delta\mathbf{x} = \gamma(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{closest}})$

How to compute a path to a goal pose?

In end effector space



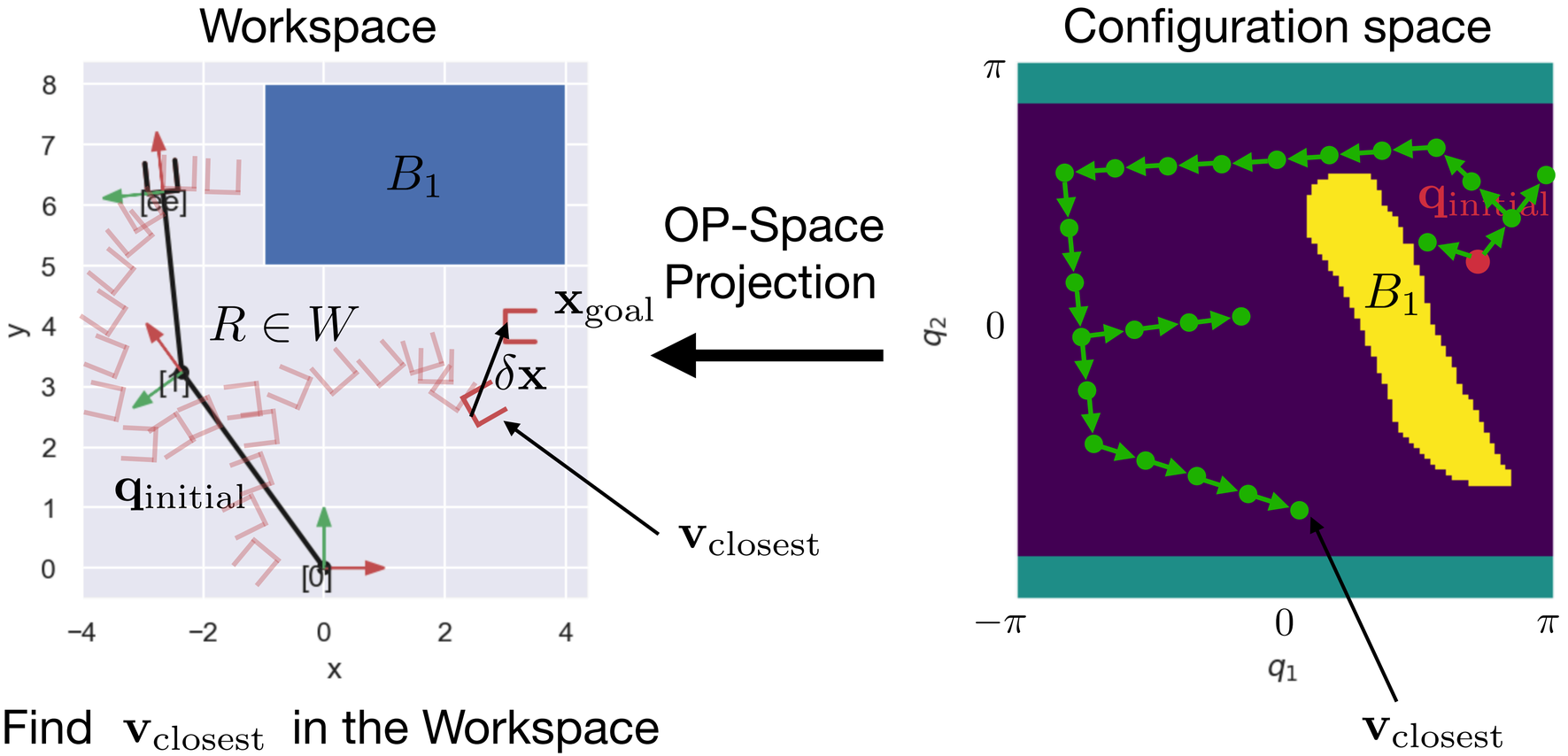
Find $\mathbf{v}_{\text{closest}}$ in the Workspace

Compute step in Workspace $\delta\mathbf{x} = \gamma(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{closest}})$

Compute step in Configuration Space: $\delta\mathbf{q} =$

How to compute a path to a goal pose?

In end effector space



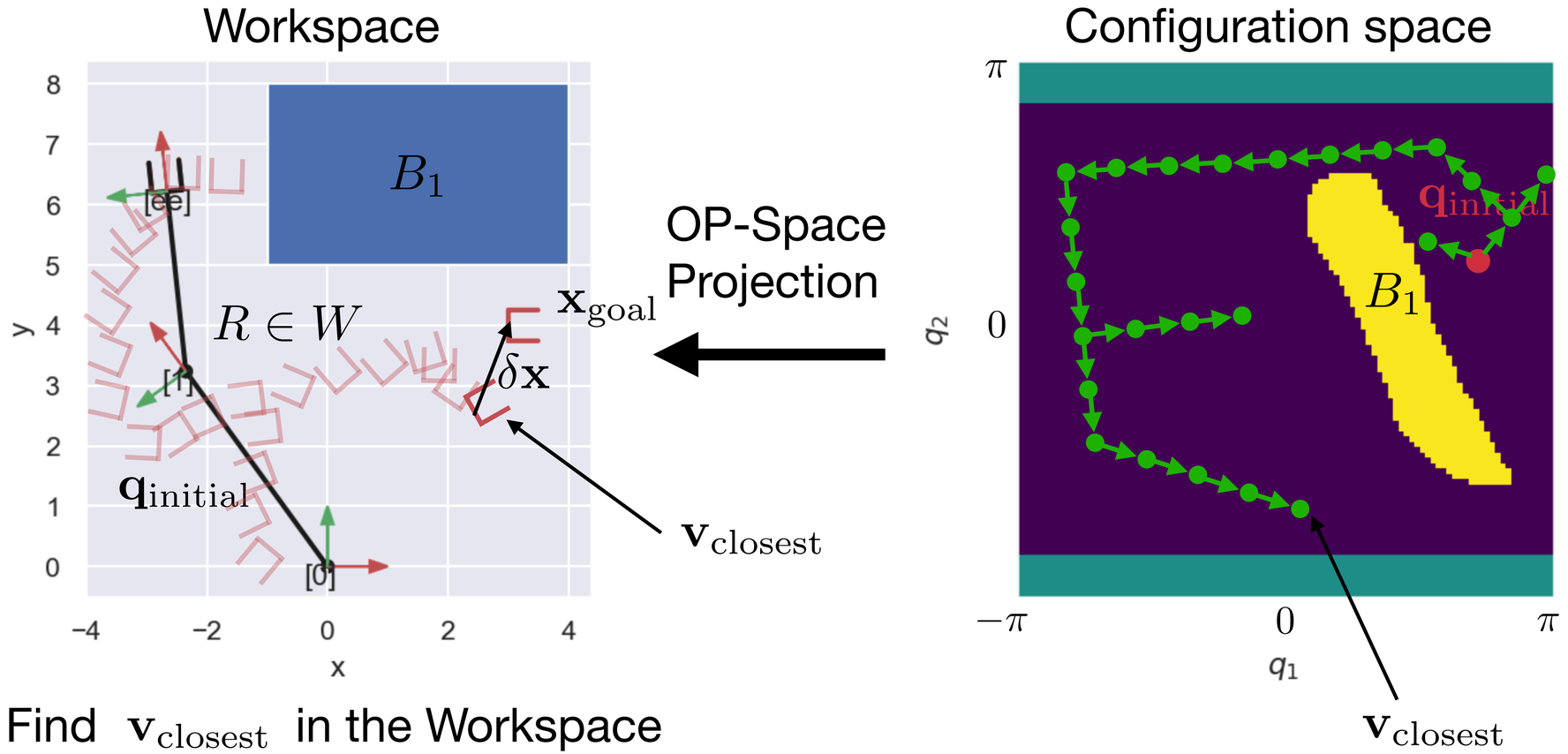
Find $\mathbf{v}_{\text{closest}}$ in the Workspace

Compute step in Workspace $\delta\mathbf{x} = \gamma(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{closest}})$

Compute step in Configuration Space: $\delta\mathbf{q} = J^{-1}\delta\mathbf{x}$

How to compute a path to a goal pose?

In end effector space



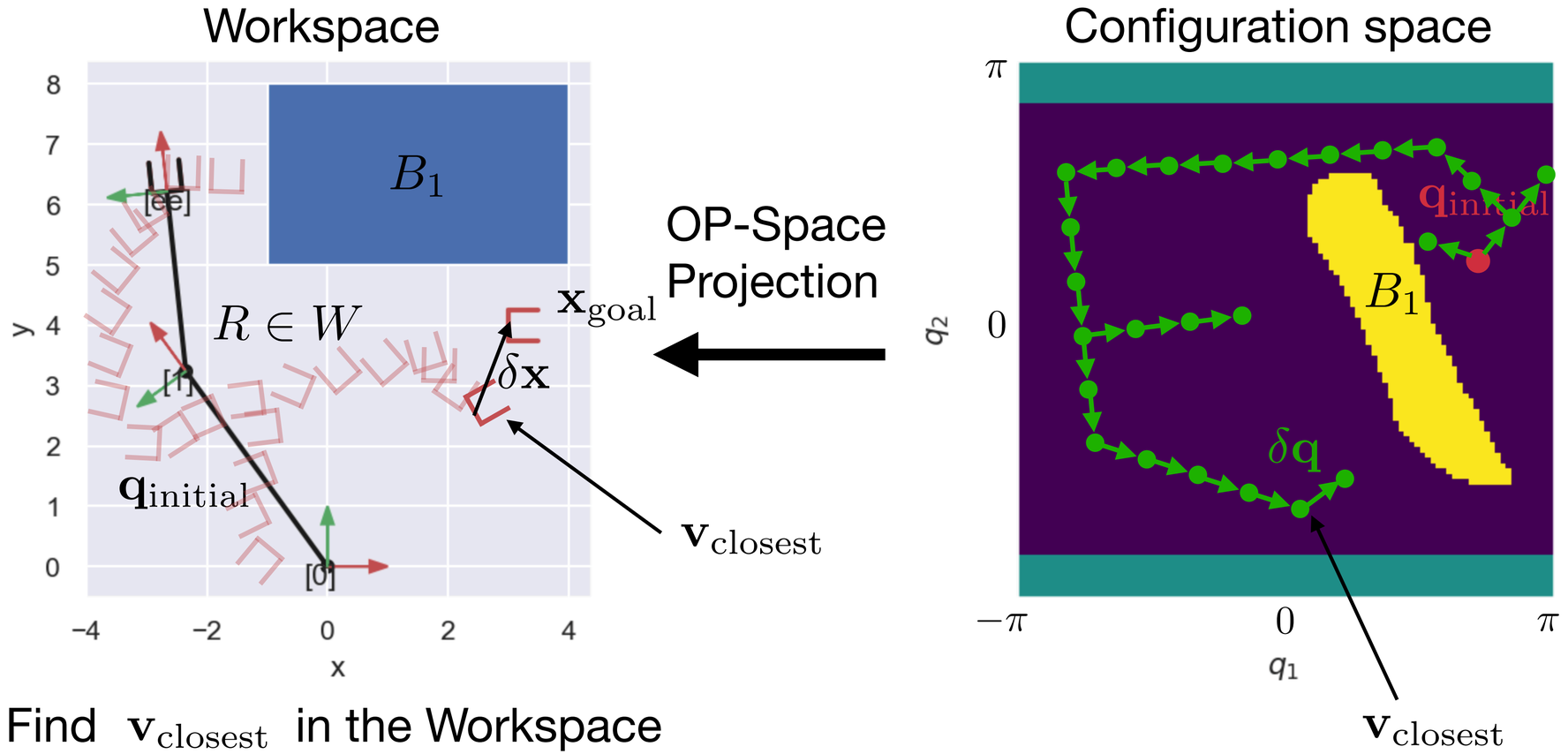
Find $\mathbf{v}_{\text{closest}}$ in the Workspace

Repeat until $\mathbf{x}_{\text{goal}}$ is found

- Compute step in Workspace $\delta\mathbf{x} = \gamma(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{closest}})$
- Compute step in Configuration Space: $\delta\mathbf{q} = J^{-1}\delta\mathbf{x}$

How to compute a path to a goal pose?

In end effector space



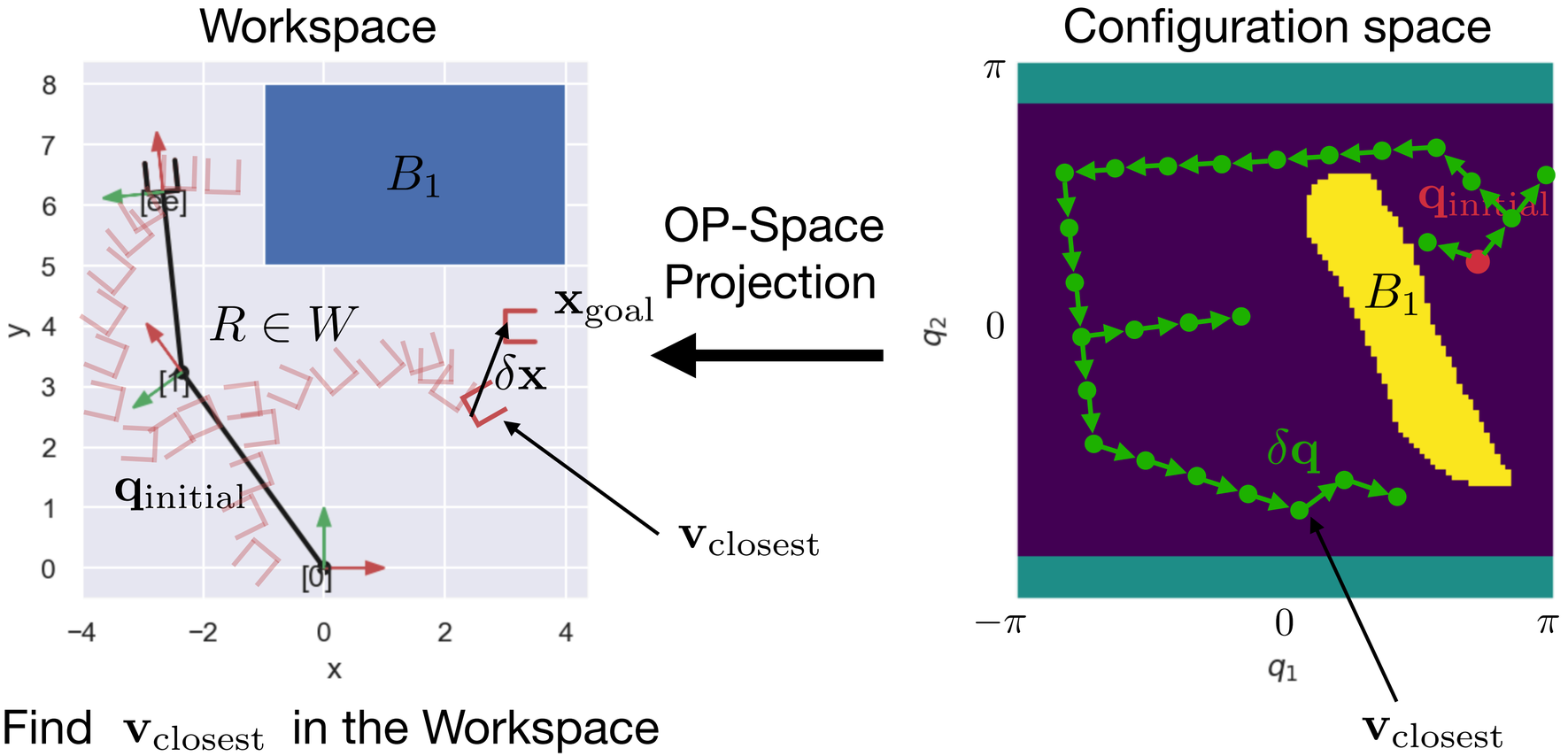
Find $\mathbf{v}_{\text{closest}}$ in the Workspace

Repeat until $\mathbf{x}_{\text{goal}}$ is found

- Compute step in Workspace $\delta\mathbf{x} = \gamma(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{closest}})$
- Compute step in Configuration Space: $\delta\mathbf{q} = J^{-1}\delta\mathbf{x}$

How to compute a path to a goal pose?

In end effector space



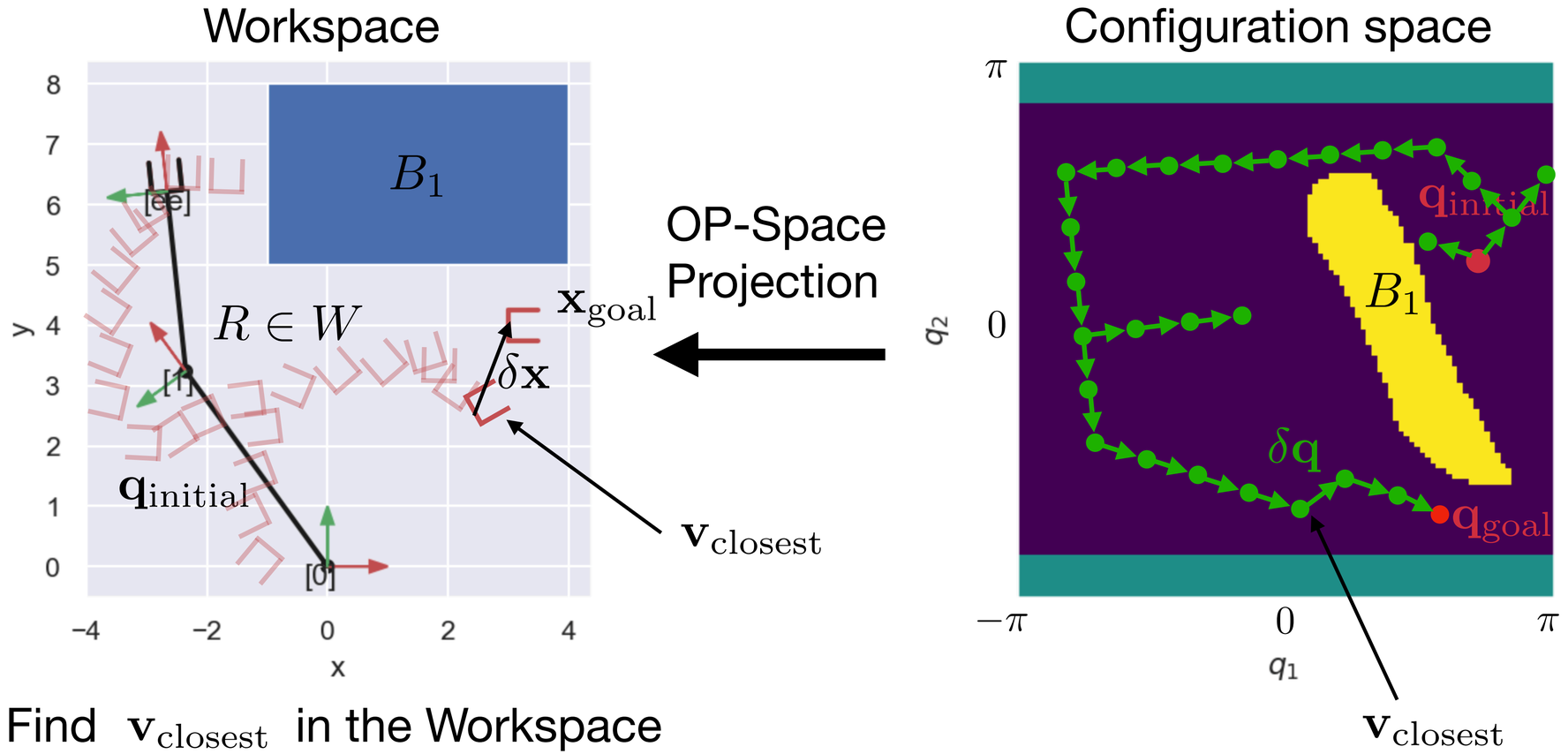
Find $\mathbf{v}_{\text{closest}}$ in the Workspace

Repeat until $\mathbf{x}_{\text{goal}}$ is found

- Compute step in Workspace $\delta\mathbf{x} = \gamma(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{closest}})$
- Compute step in Configuration Space: $\delta\mathbf{q} = J^{-1}\delta\mathbf{x}$

How to compute a path to a goal pose?

In end effector space

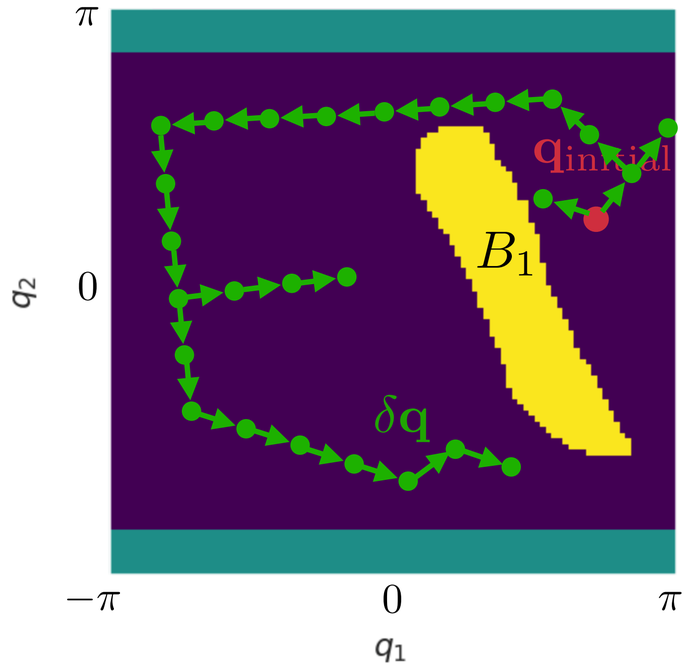


Find $\mathbf{v}_{\text{closest}}$ in the Workspace

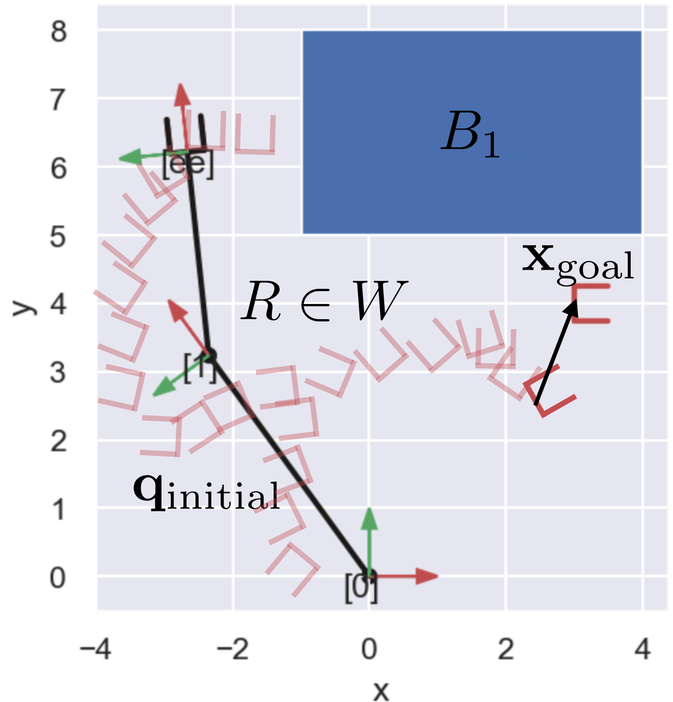
Repeat until $\mathbf{x}_{\text{goal}}$ is found

- Compute step in Workspace $\delta\mathbf{x} = \gamma(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{closest}})$
- Compute step in Configuration Space: $\delta\mathbf{q} = J^{-1}\delta\mathbf{x}$

RRT-Connect with Jacobian



Configuration space



Workspace

RRT-Connect with Jacobian

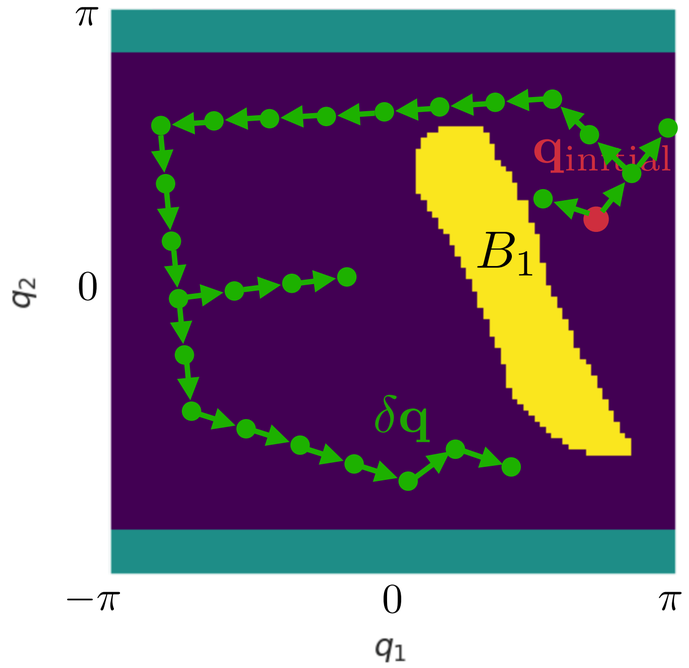
$$\mathbf{v}_i = (\mathbf{q}_i, \mathbf{x}_i, \mathbf{v}_i^{\text{parent}})$$

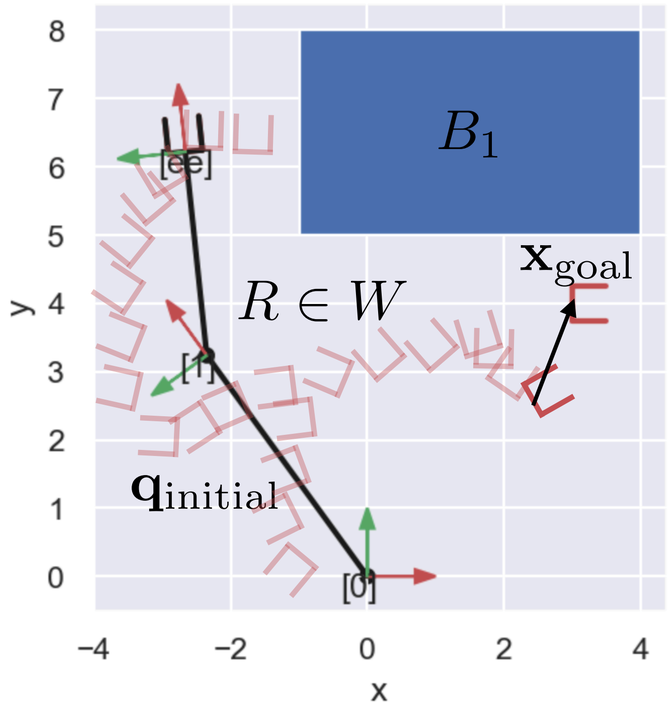
Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Exploration

- With probability $(1 - p_{\text{goal}})$ do:
  - Sample random node $\mathbf{v}_{\text{smpl}} = \mathbf{v}_{\text{rnd}}$
  - Find closest node $\mathbf{v}_{\text{closest}}$ in configuration space
    - Repeatedly move in direction $(\mathbf{q}_{\text{smpl}} - \mathbf{q}_{\text{closest}})$ with step size γ and add nodes along the way

- With probability p_{goal} do:
  - Choose goal node $\mathbf{v}_{\text{smpl}} = \mathbf{v}_{\text{goal}}$
  - Find closest node $\mathbf{v}_{\text{closest}}$ in ee-space
  - Repeatedly move in direction $J^{-1}(\mathbf{x}_{\text{smpl}} - \mathbf{x}_{\text{closest}})$ with step size γ and add nodes along the way





RRT-Connect with Jacobian

$$\mathbf{v}_i = (\mathbf{q}_i, \mathbf{x}_i, \mathbf{v}_i^{\text{parent}})$$

Repeat until $\mathbf{q}_{\text{goal}}$ is found:

Exploration

- With probability $(1 - p_{\text{goal}})$ do:
  - Sample random node $\mathbf{v}_{\text{smpl}} = \mathbf{v}_{\text{rnd}}$
  - Find closest node $\mathbf{v}_{\text{closest}}$ in configuration space
    - Repeatedly move in direction $(\mathbf{q}_{\text{smpl}} - \mathbf{q}_{\text{closest}})$ with step size γ and add nodes along the way

Exploitation

- With probability p_{goal} do:
  - Choose goal node $\mathbf{v}_{\text{smpl}} = \mathbf{v}_{\text{goal}}$
  - Find closest node $\mathbf{v}_{\text{closest}}$ in ee-space
  - Repeatedly move in direction $J^{-1}(\mathbf{x}_{\text{smpl}} - \mathbf{x}_{\text{closest}})$ with step size γ and add nodes along the way