sim_bidirectional_rrt

October 8, 2020

1 Bidirectional Sampling-Based Motion Planning

[1]:
```
# The autoreload extension will automatically load in new code as you edit␣
 ↪files,
# so you don't need to restart the kernel every time
%load_ext autoreload
%autoreload 2

import numpy as np
import matplotlib.pyplot as plt
from P2_rrt import *
from P4_bidirectional_rrt import *

plt.rcParams['figure.figsize'] = [20, 20] # Change default figure size
```

1.0.1 Set up workspace

[2]:
```
MAZE = np.array([
    (( 5, 5), (-5, 5)),
    ((-5, 5), (-5,-5)),
    ((-5,-5), ( 5,-5)),
    (( 5,-5), ( 5, 5)),
    ((-5, 2), (-1, 2)),
    ((-1, 2), (-1,-1)),
    (( 0, 2), ( 0,-1)),
    (( 0, 2), ( 5, 2))
])
```

1.1 Normal RRT

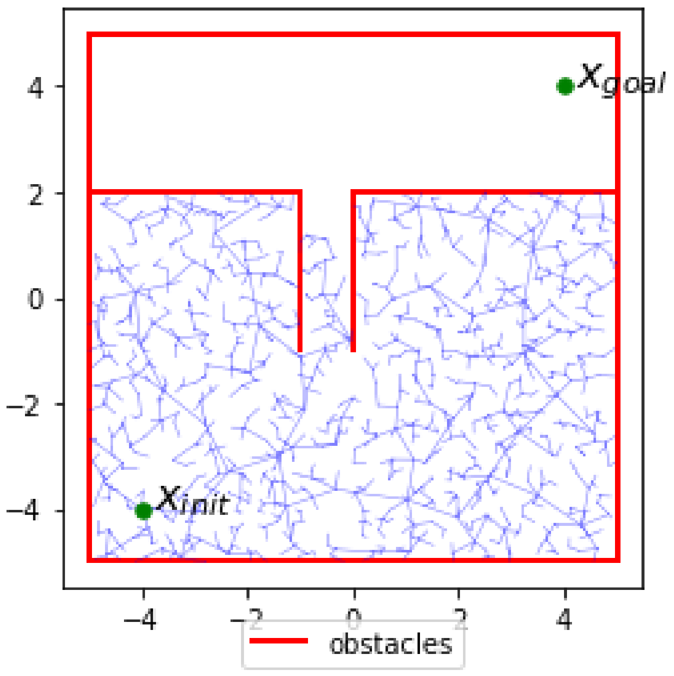
On this “bugtrap” problem, normal RRT often will fail to find a find a path.

1.1.1 Geometric planning

[3]:
```
grrt = GeometricRRT([-5,-5], [5,5], [-4,-4], [4,4], MAZE)
grrt.solve(1.0, 2000)
```

```
Solution not found!
```

[3]:
```
False
```

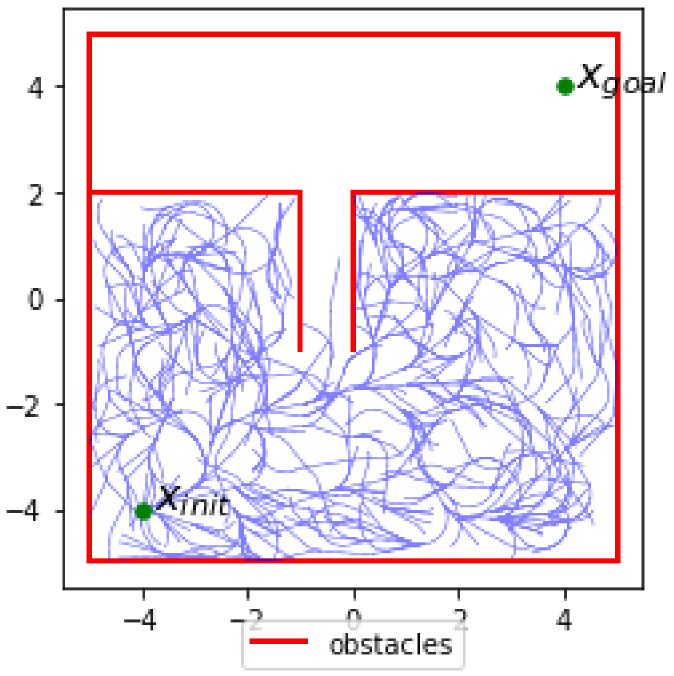


1.1.2 Dubins car planning

[4]:
```
drrt = DubinsRRT([-5,-5,0], [5,5,2*np.pi], [-4,-4,0], [4,4,np.pi/2], MAZE, .5)
drrt.solve(1.0, 1000)
```

```
Solution not found!
```

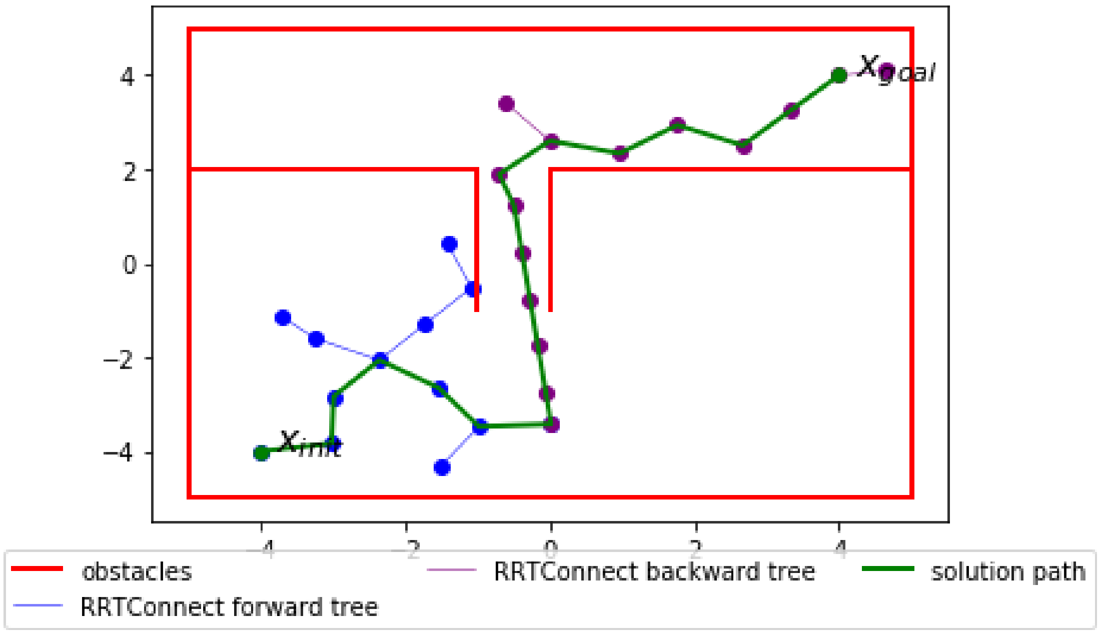
[4]:
```
False
```



1.2 RRTConnect

1.2.1 Geometric planning

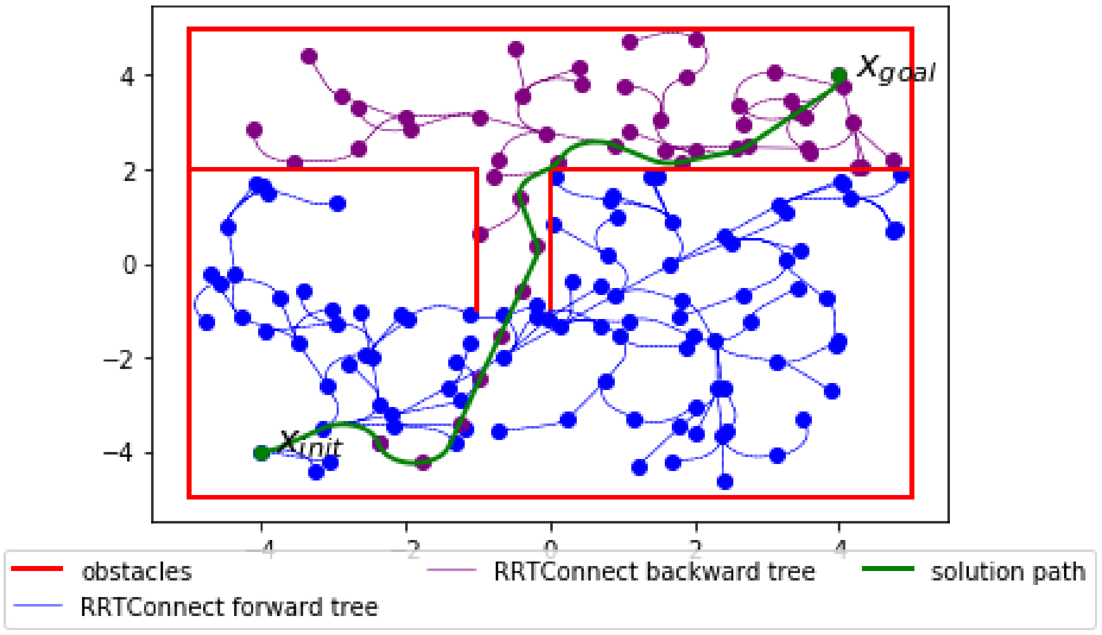
```
[5]: grrt = GeometricRRTConnect([-5,-5], [5,5], [-4,-4], [4,4], MAZE)
     grrt.solve(1.0, 2000)
```



```
[5]: True
```

1.2.2 Dubins car planning

```
[6]: drrt = DubinsRRTConnect([-5,-5,0], [5,5,2*np.pi], [-4,-4,0], [4,4,np.pi/2],␣
      ↪MAZE, .5)
     drrt.solve(1.0, 1000)
```


X_{goal}
X_{init}
4
2
0
−2
−4
−4
−2
0
2
4
obstacles
RRTConnect backward tree
solution path
RRTConnect forward tree

```
[6]: True
```

```
[ ]:
```

```
[ ]:
```