sim_bidirectional_rrt

October 18, 2019

1 Bidirectional Sampling-Based Motion Planning

In [38]:
```
# 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
%matplotlib inline
%autoreload 2

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

plt.rcParams['figure.figsize'] = [7, 7] # Change default figure size
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

1.0.1 Set up workspace

In [39]:
```
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

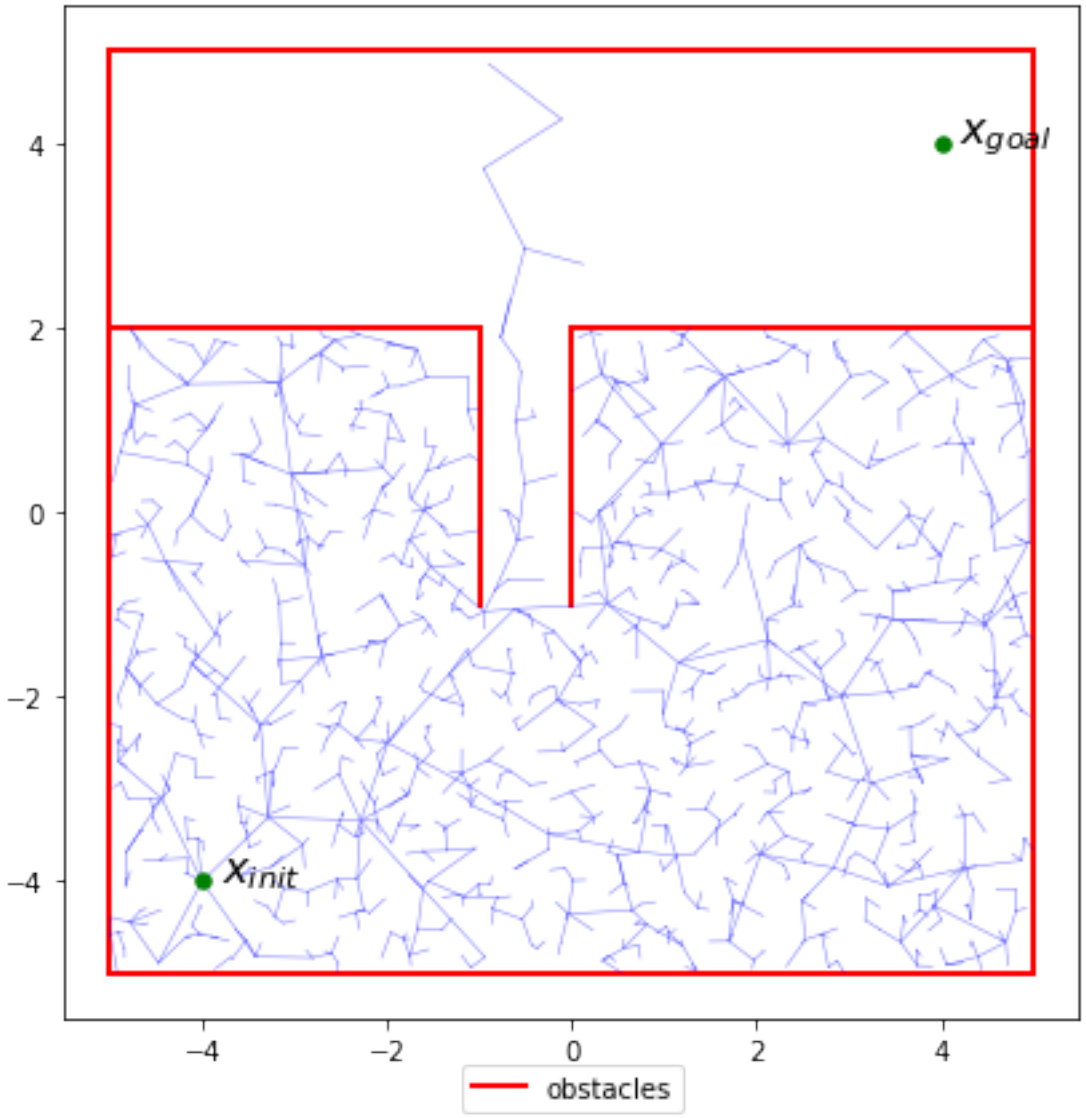
On this "bugtrap" problem, normal RRT often will fail to find a find a path.

1.1.1 Geometric planning

```
In [40]: grrt = GeometricRRT([-5,-5], [5,5], [-4,-4], [4,4], MAZE)
         grrt.solve(1.0, 2000)
```

```
Solution not found!
```

```
Out[40]: False
```

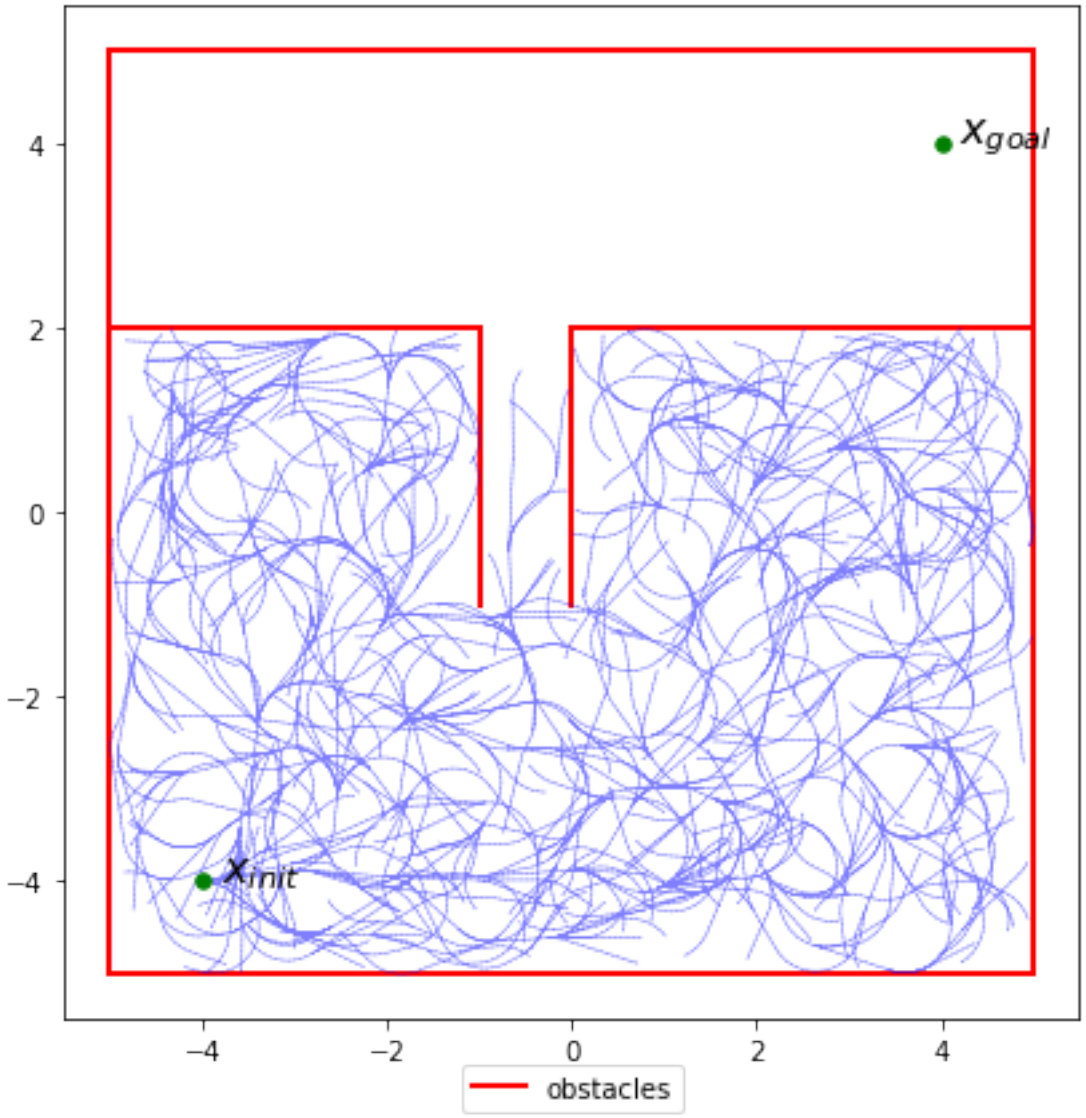


1.1.2 Dubins car planning

```
In [41]: drrt = DubinsRRT([-5,-5,0], [5,5,2*np.pi], [-4,-4,0], [4,4,np.pi/2], MAZE, .5)
         drrt.solve(1.0, 1500)
```

```
Solution not found!
```

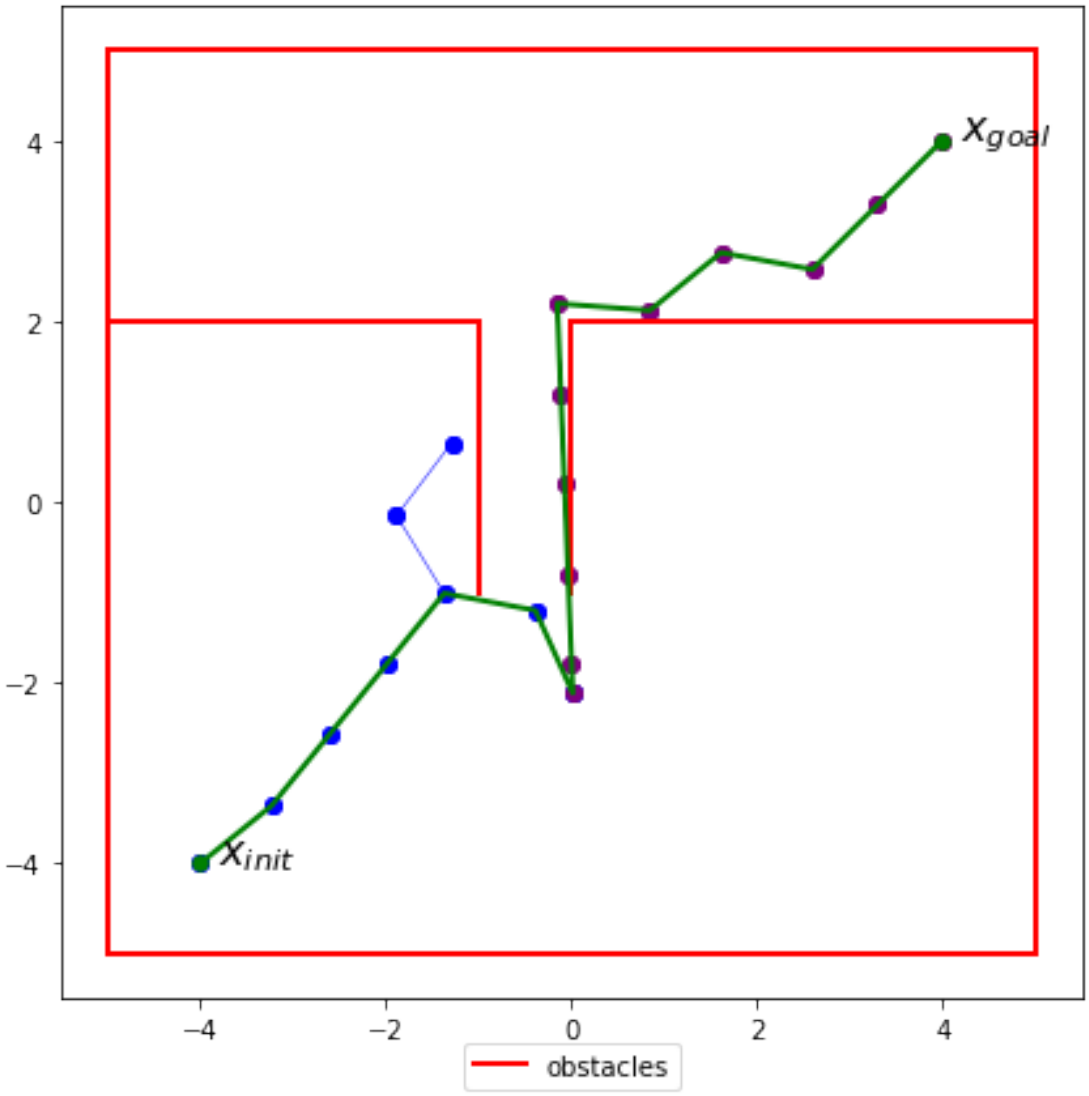
Out[41]: False



1.2 RRTConnect

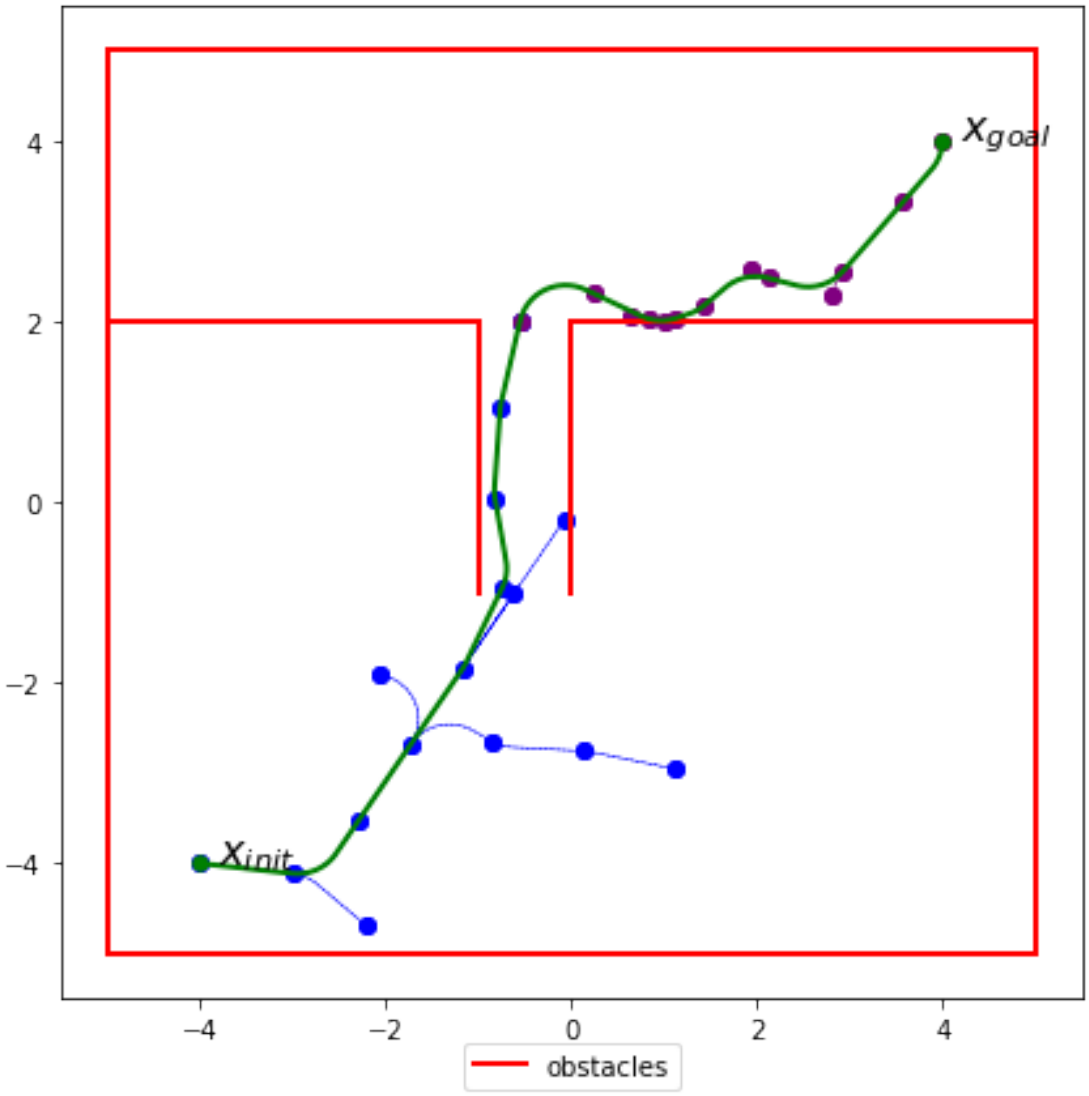
1.2.1 Geometric planning

In [42]:
```
grrt = GeometricRRTConnect([-5,-5], [5,5], [-4,-4], [4,4], MAZE)
grrt.solve(1.0, 2000)
```



1.2.2 Dubins car planning

```
In [59]: drrt = DubinsRRTConnect([-5,-5,0], [5,5,2*np.pi], [-4,-4,0], [4,4,np.pi/2], MAZE, .5)
         drrt.solve(1.0, 4000)
```


x_{goal}
x_{init}
4
2
0
−2
−4
−4
−2
0
2
4
obstacles