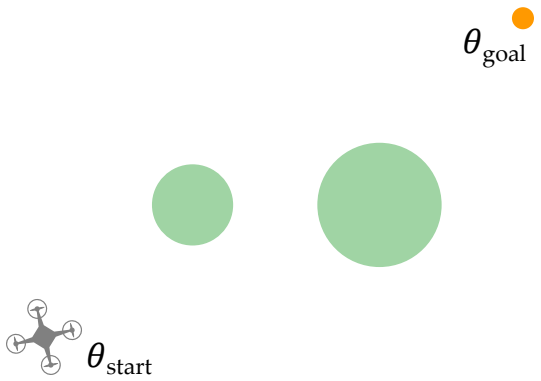
Problem Set 8

Robotics & Automation
Dylan Losey, Virginia Tech

Instructions. Please write legibly and do not attempt to fit your work into the smallest space possible. It is important to show all work, but basic arithmetic can be omitted. You are encouraged to use Matlab when possible to avoid hand calculations, but print and submit your commented code for non-trivial calculations. You can attach a pdf of your code to the homework, use live scripts or the publish feature in Matlab, or include a snapshot of your code. Do not submit .m files — we will not open or grade these files.

1 Potential Fields



In this problem you will use potential fields to get a motion plan for the 2-DoF environment shown above. Here the drone's position is $\theta = [x, y]^T$.

1.1 (15 points)

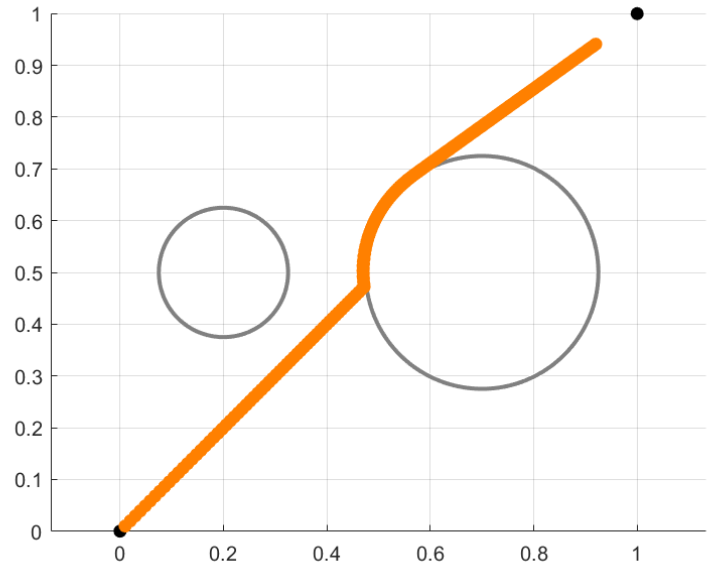
Implement the potential fields approach:

- Set $\theta_{start} = [0, 0]^T$ and $\theta_{goal} = [1, 1]^T$
- The first obstacle has center $c_1 = [0.2, 0.5]^T$ and radius $r_1 = 0.125$
- The second obstacle has center $c_2 = [0.7, 0.5]^T$ and radius $r_2 = 0.225$
- **Hint:** Start with a low learning rate α in your gradient descent algorithm. The result shown below was obtained with $\alpha = 0.01$.

The motion plan must reach a final position within 0.1 units of the goal. Turn in your code and a plot of the result using **Publish** in Matlab. Visualize the obstacles in your plot: your solution should look like the example below.

1.2 (10 points)

Modify the position of the obstacles so that a valid plan from θ_{start} to θ_{goal} exists but the potential fields planner fails (i.e., gets stuck). Turn in a **plot** that shows the obstacles and the failed motion plan. **Explain** why potential fields fail in your environment.



2 Trajectory Optimization

In this problem you will use trajectory optimization to perform motion planning in 2-DoF environments. As before, the mobile robot's position is $\theta = [x, y]^T$.

2.1 (15 points)

Implement the trajectory optimization algorithm. Your code should be able to work with an arbitrary number of waypoints and circular obstacles. Set the initial trajectory ξ^0 as:

```
xi_0 = [linspace(theta_start(1), theta_goal(1), k);
        linspace(theta_start(2), theta_goal(2), k)];
```

2.2 (15 points)

Use your code to find a desired trajectory for the following environments. In each environment $\theta_{start} = [0, 0]^T$ and $\theta_{goal} = [1, 1]^T$.

- **Environment 1.** One obstacle with center $c_1 = [0.55, 0.5]^T$ and radius $r_1 = 0.3$. Your trajectory should have $k = 10$ waypoints.
- **Environment 2.** One obstacle with center $c_1 = [0.5, 0.3]^T$ and radius $r_1 = 0.3$. A second obstacle with center $c_2 = [0.5, 0.7]^T$ and radius $r_2 = 0.2$. Set $k = 15$.
- **Environment 3.** One obstacle with center $c_1 = [0.2, 0.35]^T$ and radius $r_1 = 0.2$. A second obstacle with center $c_2 = [0.5, 0.3]^T$ and radius $r_2 = 0.2$. A third obstacle with center $c_3 = [0.7, 0.5]^T$ and radius $r_3 = 0.2$. Set $k = 20$.

Turn in a **published** version of your code and plots. Each plot should visualize the obstacles, show the initial trajectory ξ^0, and show the optimal trajectory ξ. For instance, your solution for the first environment should look like the plot below.

2.3 (10 points)

Consider an environment with two obstacles:

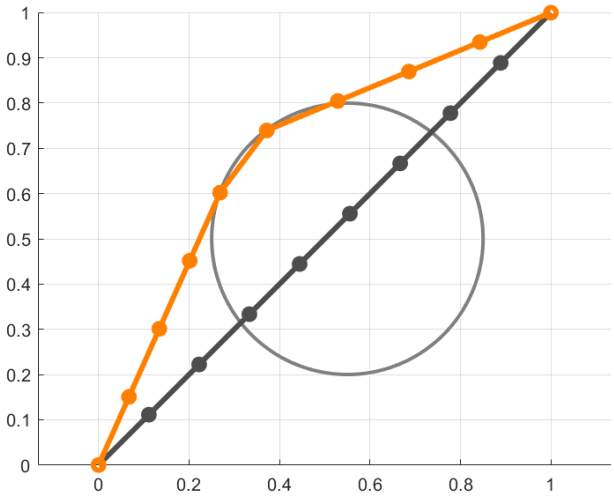


Figure 1: Trajectory optimization solution for **Environment 1**

- $\theta_{start} = [0, 0]^T$ and $\theta_{goal} = [1, 1]^T$
- First obstacle with center $c_1 = [0.4, 0.6]^T$ and radius $r_1 = 0.2$
- Second obstacle with center $c_2 = [0.6, 0.4]^T$ and radius $r_2 = 0.2$
- The trajectory ξ should have $k = 20$ waypoints

Modify the initial trajectory ξ^0 so that the optimal trajectory goes around both obstacles. Submit a **plot** of your result and **list** the initial trajectory that you used.

3 RRT Algorithm

In this problem you will use the RRT algorithm to perform motion planning in 2-DoF environments. As before, the mobile robot's position is $\theta = [x, y]^T$.

3.1 (10 points)

Implement the RRT algorithm. Your code should be able to work with an arbitrary number of circular obstacles.

- The bounds of the workspace are $x \in [0, 1]$, $y \in [0, 1]$
- The motion plan must end within $\epsilon = 0.1$ units of the goal.

3.2 (15 points)

Use your code to find a desired trajectory for the following environments. In each environment $\theta_{start} = [0, 0]^T$ and $\theta_{goal} = [1, 1]^T$.

- **Environment 1.** One obstacle with center $c_1 = [0.55, 0.5]^T$ and radius $r_1 = 0.3$.
- **Environment 2.** One obstacle with center $c_1 = [0.5, 0.3]^T$ and radius $r_1 = 0.3$. A second obstacle with center $c_2 = [0.5, 0.7]^T$ and radius $r_2 = 0.2$.
- **Environment 3.** One obstacle with center $c_1 = [0.2, 0.35]^T$ and radius $r_1 = 0.2$. A second obstacle with center $c_2 = [0.5, 0.3]^T$ and radius $r_2 = 0.2$. A third obstacle with center $c_3 = [0.7, 0.5]^T$ and radius $r_3 = 0.2$.

Turn in a **published** version of your code and plots. Each plot should visualize the obstacles, show the tree G, and show the final motion plan. For instance, your solution for the first environment should look like the plot below.

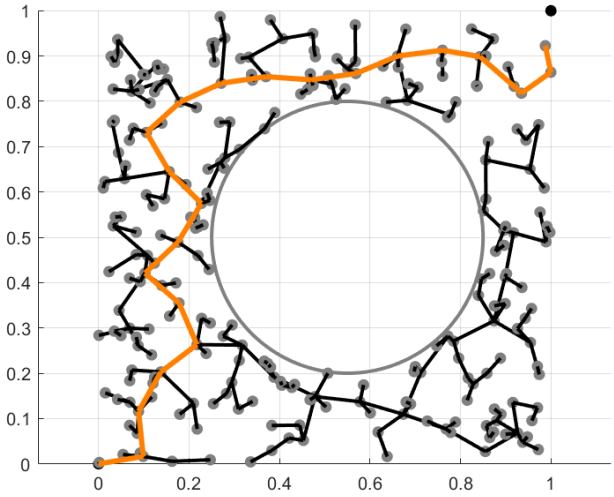


Figure 2: RRT solution for **Environment 1**

3.3 (10 points)

Using **Environment 3** from the previous part, compare two versions of RRT. The first version is the standard RRT algorithm you have implemented (let's refer to this as the **baseline**). The second version will sample the goal more frequently (let's refer to this as **goal bias**). For **goal bias**, with probability 0.2 set θ_{rand} as θ_{goal}. Otherwise sample randomly as normal.

Run your code 10 times for **baseline** and 10 times for **goal bias**. Write down how many samples it takes on average to find a motion plan. Which approach is more sample-efficient: **baseline** or **goal bias**? Write a few sentences to explain and support your answer.