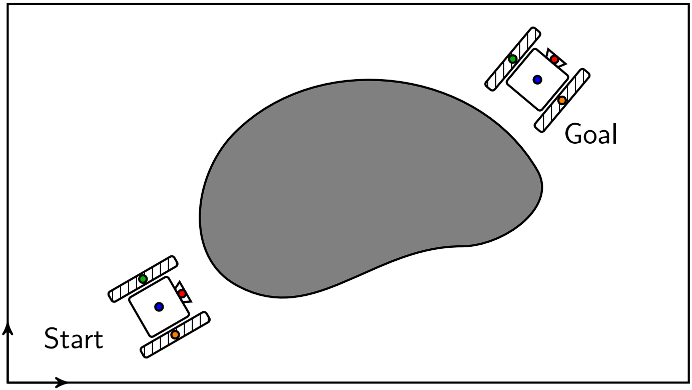
Motion Planning II

Spring 2020

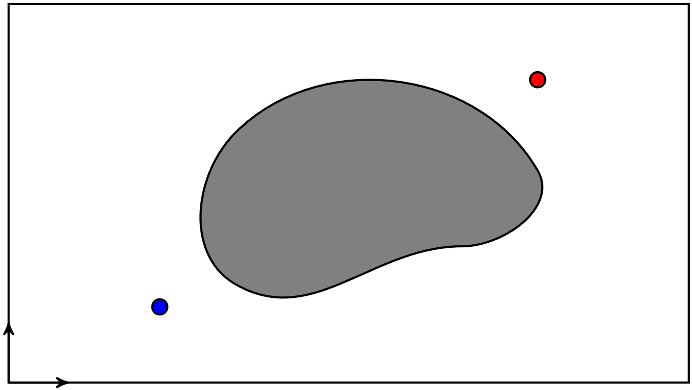
Hasan Poonawala

Potential Function Methods



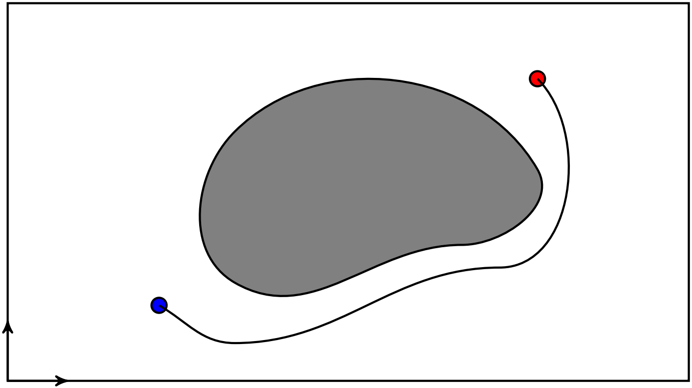
Motion Planning Problem

Potential Function Methods

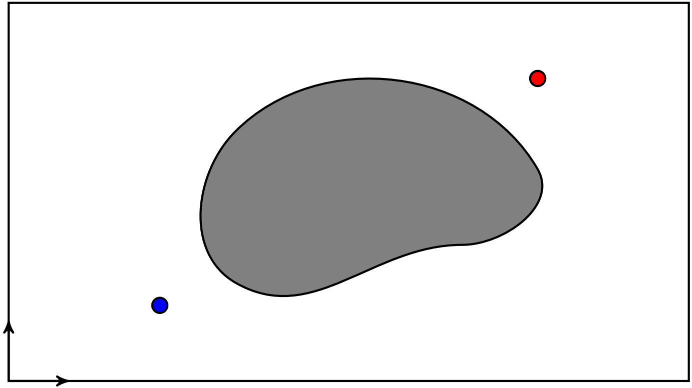


Simplified Problem

Potential Function Methods

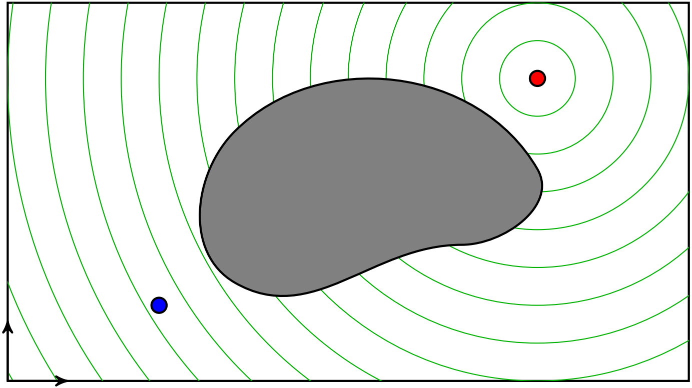


Potential Function Methods



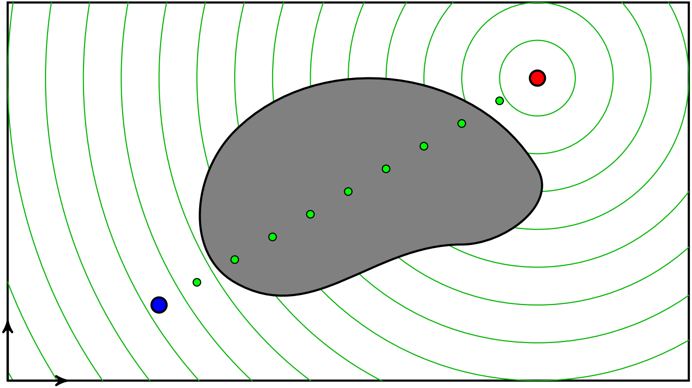
Choose potential function $U(q) = \|q - q_g\|^2$

Potential Function Methods



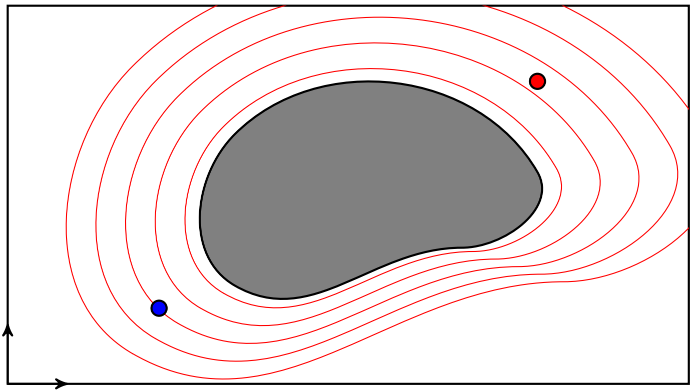
Level sets of $U(q)$ are circles, gradients are perpendicular to level sets

Potential Function Methods



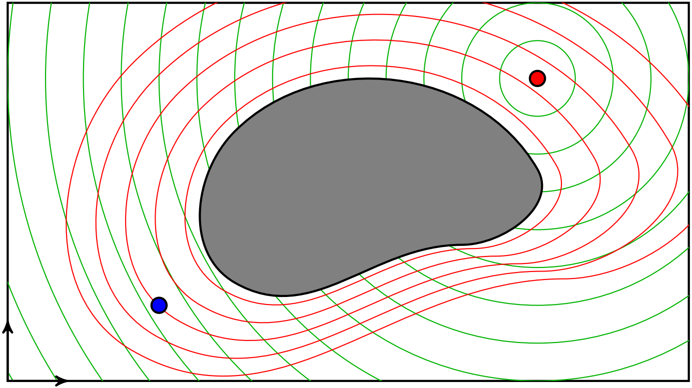
Gradient steps generate a sequence of points

Potential Function Methods



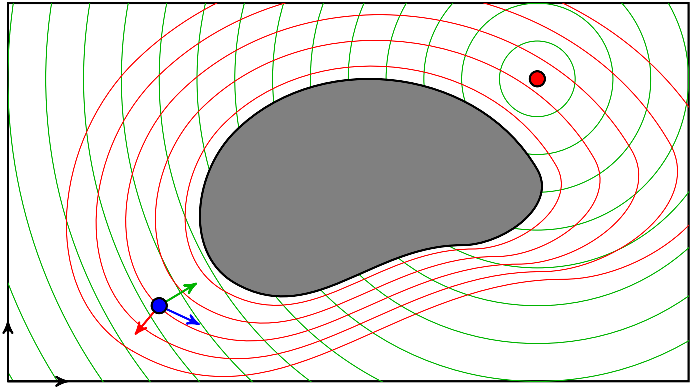
Clearly this solution is invalid. Need to add a term to handle obstacles

Potential Function Methods



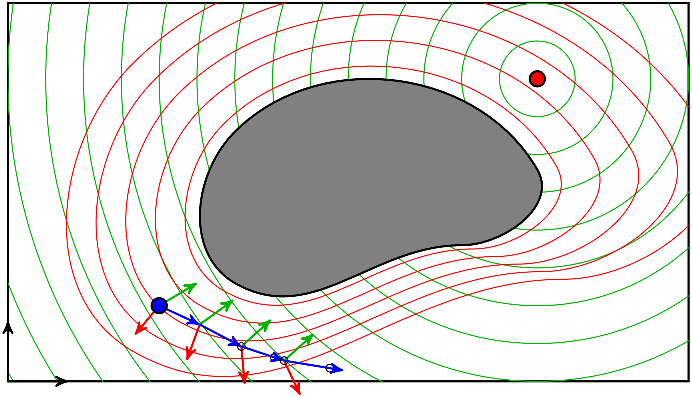
Our potential is the sum of the potential $U_{attr}(q)$ due to the goal and $U_{rep}(q)$ due to the obstacle

Potential Function Methods



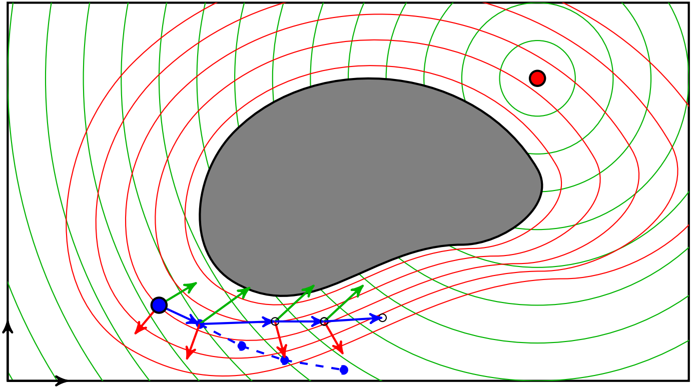
The negative gradient $-\nabla U_{attr}(q)$ pulls us to goal, $-\nabla U_{rep}(q)$ pushes us away from obstacle, their sum is the blue arrow.

Potential Function Methods



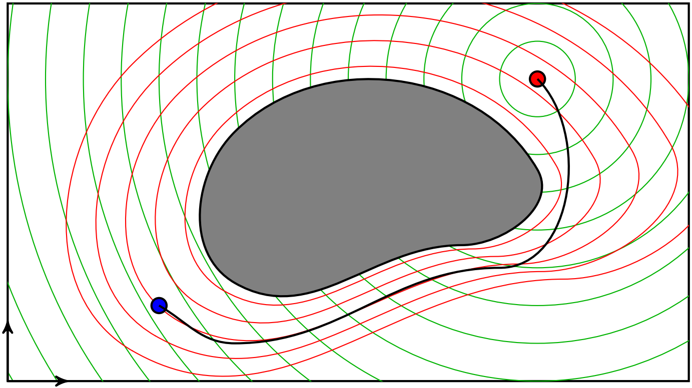
Repeating this process after every step along the blue arrows generates a sequence.

Potential Function Methods



Scaling $U_{attr}(q)$, which scales its gradient, pulls path closer to the obstacle.

Potential Function Methods



Next step: convert sequence of nodes/configurations into a trajectory.

Full Trajectory

- PRM, RRT, Potential-functions etc generate a sequence of configurations/nodes; a path in a graph.

Full Trajectory

- PRM, RRT, Potential-functions etc generate a sequence of configurations/nodes; a path in a graph.
- To get a full trajectory, we need to use a local planner to convert an edge in the graph to a trajectory.

Full Trajectory

- PRM, RRT, Potential-functions etc generate a sequence of configurations/nodes; a path in a graph.
- To get a full trajectory, we need to use a local planner to convert an edge in the graph to a trajectory.
- Essentially, we will fit parametrized functions of time to pairs of points.

Polynomial Blends

Let two nodes in the sequence by q_1 and q_2, where we want the trajectory to pass through them at t_1 and t_2 $(> t_1)$ respectively

Polynomial Blends

Let two nodes in the sequence by q_1 and q_2, where we want the trajectory to pass through them at t_1 and t_2 ($> t_1$) respectively

Linear interpolation in time provides a simply trajectory

$$q(t) = q_1 + \frac{t - t_1}{t_2 - t_1}(q_2 - q_1) = a_1 t + a_0$$

where

$$a_0 = \frac{q_1 t_2 - q_2 t_1}{t_2 - t_1}, \quad a_1 = \frac{q_2 - q_1}{t_2 - t_1}.$$

This interpolation satisfies $q(t_1) = q_1$, $q(t_2) = q_2$

Polynomial Blends

Let two nodes in the sequence by q_1 and q_2, where we want the trajectory to pass through them at t_1 and t_2 $(> t_1)$ respectively

Linear interpolation in time provides a simply trajectory

$$q(t) = q_1 + \frac{t - t_1}{t_2 - t_1}(q_2 - q_1) = a_1 t + a_0$$

where

$$a_0 = \frac{q_1 t_2 - q_2 t_1}{t_2 - t_1}, \quad a_1 = \frac{q_2 - q_1}{t_2 - t_1}.$$

This interpolation satisfies $q(t_1) = q_1$, $q(t_2) = q_2$

For $t \in [t_1, t_2]$,

$$\dot{q}(t) = a_1 = \frac{q_2 - q_1}{t_2 - t_1}.$$

The velocity is constant during this time interval.

Polynomial Blends

Take three configurations q_1, q_2, and q_3. Let times be t_1, t_2, t_3.

Polynomial Blends

Take three configurations q_1, q_2, and q_3. Let times be t_1, t_2, t_3.

Use linear interpolation for q_1 and q_2 to get a function $q^a(t)$, and also for q_2 and q_3 to get $q^b(t)$.

We know that we will achieve $q^a(t_1) = q_1$, $q^a(t_2) = q_2 = q^b(t_2)$, and $q^b(t_3) = q_3$

Polynomial Blends

Take three configurations q_1, q_2, and q_3. Let times be t_1, t_2, t_3.

Use linear interpolation for q_1 and q_2 to get a function $q^a(t)$, and also for q_2 and q_3 to get $q^b(t)$.

We know that we will achieve $q^a(t_1) = q_1$, $q^a(t_2) = q_2 = q^b(t_2)$, and $q^b(t_3) = q_3$

However, for $t \in [t_1, t_2]$,

$$\dot{q}^a(t) = \frac{q_2 - q_1}{t_2 - t_1},$$

and for $t \in [t_2, t_3]$,

$$\dot{q}^b(t) = \frac{q_3 - q_2}{t_3 - t_2}.$$

Polynomial Blends

Take three configurations q_1, q_2, and q_3. Let times be t_1, t_2, t_3.

Use linear interpolation for q_1 and q_2 to get a function $q^a(t)$, and also for q_2 and q_3 to get $q^b(t)$.

We know that we will achieve $q^a(t_1) = q_1$, $q^a(t_2) = q_2 = q^b(t_2)$, and $q^b(t_3) = q_3$

However, for $t \in [t_1, t_2]$,

$$\dot{q}^a(t) = \frac{q_2 - q_1}{t_2 - t_1},$$

and for $t \in [t_2, t_3]$,

$$\dot{q}^b(t) = \frac{q_3 - q_2}{t_3 - t_2}.$$

Generally, $\dot{q}^a(t_2) \neq \dot{q}^b(t_2)$. [Why is this bad?]

Polynomial Blends

To make the velocity continuous at t_2, maybe we should use quadratic functions for $q^a(t)$ and $q^b(t)$, and make sure that $\dot{q}^a(t_2) = \dot{q}^b(t_2)$.

Polynomial Blends

To make the velocity continuous at t_2, maybe we should use quadratic functions for $q^a(t)$ and $q^b(t)$, and make sure that $\dot{q}^a(t_2) = \dot{q}^b(t_2)$.

We can use higher order polynomials to make acceleration, jerk, snap, and so on continuous at each node.

Polynomial Blends

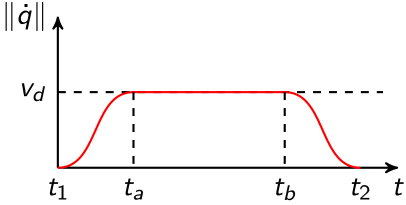
To make the velocity continuous at t_2, maybe we should use quadratic functions for $q^a(t)$ and $q^b(t)$, and make sure that $\dot{q}^a(t_2) = \dot{q}^b(t_2)$.

We can use higher order polynomials to make acceleration, jerk, snap, and so on continuous at each node.

We use all these polynomials to define $q(t)$ over the time interval $[t_0, t_N]$ where there are $N + 1$ nodes.

Other Approaches

- Parabolic blends: assume we have to be stopped at node (sensor task/way-station task).
  Divide time interval into three intervals: middle has a given velocity v_d, first and third represent smooth transition from 0 to v_d and v_d back to 0.



- Minimum-time parabolic blends: make the transition times $t_a - t_1$ and $t_2 - t_b$ as short as possible, and v_d as high as possible.
- B-Splines, Bezier Curves etc.

Summary

- Use graph search to find a path - CS approach

Summary

- Use graph search to find a path - CS approach
- Challenge: converting continuous state space to graph (PRM, RRT, etc)

Summary

- Use graph search to find a path - CS approach
- Challenge: converting continuous state space to graph (PRM, RRT, etc)
- Challenge: converting graph to continuous trajectory (Polynomials, parabolic, splines, etc)

Summary

- Use graph search to find a path - CS approach
- Challenge: converting continuous state space to graph (PRM, RRT, etc)
- Challenge: converting graph to continuous trajectory (Polynomials, parabolic, splines, etc)
- Challenge: implementing continuous trajectory (feedback control & state estimation)