Lecture 19, Oct 17, 2025

Obstacle Avoidance via Potential Field (Continued)

- Given the initial joint variables $q^0 = q^s \in \mathbb{R}^n$ and final joint variables $q^f \in \mathbb{R}^n$, with the gradients, we can plan a path iteratively as $q^{k+1} = q^k - \alpha_k \nabla_q \mathcal{U}(q^k)$ where $\alpha_k > 0$ is the learning rate
- $\mathcal{U}(q)$ is the total potential function, $\mathcal{U}(q) = \sum_{i=1}^{n} \left(\mathcal{U}_{i,att}(O_i^0(q)) + \mathcal{U}_{i,rep}(O_i^0(q))\right)$
  - A simple sum might not always work; sometimes we cannot find the global minimum of the potential this way
- $\nabla_q \mathcal{U}(q) = \left(\dfrac{\partial \mathcal{U}(q)}{\partial q}\right)^T$

$$= \left(\sum_{i=1}^{n} \left(\frac{\partial \mathcal{U}_{i,att}}{\partial O_i^0} + \frac{\partial \mathcal{U}_{i,rep}}{\partial O_i^0}\right) \frac{\partial O_i^0(q)}{\partial q}\right)^T$$

$$= \sum_{i=1}^{n} \left(\frac{\partial O_i^0(q)}{\partial q}\right)^T \left(\nabla \mathcal{U}_{i,att}(O_i^0(q)) + \nabla \mathcal{U}_{i,rep}(O_i^0(q))\right)$$

  - We have $\nabla \mathcal{U}_{i,att}(O_i^0(q)) + \nabla \mathcal{U}_{i,rep}(O_i^0(q))$ from the previous lecture
  - Now we need $\dfrac{\partial O_i^0(q)}{\partial q}$, which for $i = n$ is the linear velocity Jacobian $J_v(q)$ that we already have
  - Note, for this algorithm we need the Jacobian for every base point, not just the end-effector
  - In practice, when computing $\dfrac{\partial O_i^0(q)}{\partial q}$ for $i < n$, we can do it more efficiently by starting from $J_v(q)$ and zeroing out some columns
- Let $J_{v,O_i}(q) = \dfrac{\partial O_i^0(q)}{\partial q}$, then $J_{v,O_i} = \begin{bmatrix} J_{v,O_i,1} & \cdots & J_{v,O_i,i} & 0_{3\times(n-i)} \end{bmatrix}$
  - $J_{v,O_i,j} = \begin{cases} z_{j-1}^0 & \text{joint } j \text{ is prismatic} \\ z_{j-1}^0 \times (O_i^0 - O_{j-1}^0) & \text{joint } j \text{ is revolute} \end{cases}$
  - This is similar to $J_v(q)$, but notice instead of O_n^0 we have O_i^0, because we are essentially cutting off the manipulator after link i
- The overall algorithm:
  1. Initialize: $q^0 = q^s$
  2. Iterate: $q^{k+1} = q^k + \alpha_k \sum_{i=1}^{n} J_{v,O_i}^T(q^k) \left(F_{i,att}(O_i^0(q^k)) + F_{i,rep}(O_i^0(q^k))\right)$
  3. Termination condition: $\|q^{k+1} - q^f\| < \varepsilon$ where $\varepsilon > 0$ is a termination threshold
     - In practice, this won't always converge, so often we put a cap on the max iterations and give up if we hit this number
  4. Output: $q^0, q^1, \ldots, q^N$, a set of waypoints in q-space
     - However, these waypoints are often not smooth enough and results in jerky motion
     - Therefore we usually do a spline fit over these waypoints, to get a continuous a second or third-order derivative