Lecture 15, Oct 6, 2025

Basic Motion Planning With Velocity Jacobian

- We can use velocity Jacobians to create a very basic motion planning algorithm for the manipulator through gradient descent
  - Given a desired $O_d^0 \in \mathbb{R}^3$, we want to find $(q_1(t), \dots, q_n(t))$, the joint variables as a function of time, that takes us to the desired position
- Recall that the gradient for $f : \mathbb{R}^n \mapsto \mathbb{R}$ is $\nabla_x f(x) = \left(\frac{\partial f(x)}{\partial x}\right)^T$, a column vector
- Also, $\frac{\partial \|x\|}{\partial x} = \frac{1}{\|x\|} x^T$
  - This can be shown by checking componentwise, with $\|x\| = \sqrt{x^T x}$
- The algorithm: Define a quadratic cost function $\mathcal{U}(q) = \frac{1}{2}\|O_d^0 - O_n^0(q)\|^2$ (penalizing the difference between desired and current end-effector position), we move in the direction of the negative gradient
  - $$\begin{aligned} \frac{\partial \mathcal{U}}{\partial q} &= \|O_d^0 - O_n^0(q)\| \left.\frac{\partial \|O_d^0 - O_n^0(q)\|}{\partial (O_d^0 - O_n^0(q))}\right|_{O_n^0(q)} \left(-\frac{\partial O_n^0(q)}{\partial q}\right) \\ &= \|O_d^0 - O_n^0(q)\| \frac{1}{\|O_d^0 - O_n^0(q)\|} (O_d^0 - O_n^0(q))^T \left(-\frac{\partial O_n^0(q)}{\partial q}\right) \\ &= -(O_d^0 - O_n^0(q))^T \frac{\partial O_n^0(q)}{\partial q} \end{aligned}$$
  - $\nabla_q \mathcal{U}(q) = \left(\frac{\partial \mathcal{U}}{\partial q}\right)^T = -\left(\frac{\partial O_n^0(q)}{\partial q}\right)^T (O_d^0 - O_n^0(q))$
  - Notice that the second term is simply the direction we want to move in, in the workspace of the robot (Euclidean space), and then we transformed it using the Jacobian to get the direction in q space
- In continuous time, $\dot{q} = -\gamma \nabla_q \mathcal{U}(q) = \gamma \left(\frac{\partial O_n^0(q)}{\partial q}\right)^T (O_d^0 - O_n^0(q))$ where $\gamma > 0$ is the learning rate
  - We can forward simulate this with any numerical ODE solver to get a reference signal to track
- In discrete time, $q(t_{k+1}) = q(t_k) + \gamma \left(\frac{\partial O_n^0}{\partial q}(q_{t_k})\right)^T (O_d^0 - O_n^0(q_{t_k}))$
- Note this is an asymptotic algorithm, i.e. it will never reach the goal exactly, instead closing in on the goal as the error decays exponentially
- This can also be seen as a method to do inverse kinematics without relying on kinematic decoupling or any geometric constraints on the manipulator
- In the future we will add collision avoidance and other features to this algorithm

Velocity Kinematics – Examples

- Example: 3-link articulated manipulator, RRR, find $J_v(q), J_w(q)$
  - $\rho = 1$ for all joints
  - $J_w(q) = \begin{bmatrix} z_0^0 & z_1^0 & z_2^0 \end{bmatrix}$
  - $J_v(q) = \begin{bmatrix} J_{v,1} & J_{v,2} & J_{v,3} \end{bmatrix}$
    - $J_{v,1} = z_0^0 \times (O_3^0 - O_0^0)$
    - $J_{v,2} = z_1^0 \times (O_3^0 - O_1^0)$
    - $J_{v,3} = z_2^0 \times (O_3^0 - O_2^0)$
  - Now find z_0^0, z_1^0, z_2^0 and O_1^0, O_2^0, O_3^0
    - $z_0^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
    - z_1^0 can be found by extracting the last column of R_1^0, which is contained in H_1^0
    - Find O_1^0 in H_1^0

        * Repeat for all the other variables
- Example: A helicopter flies horizontally along a circle of radius 30 at a height of 100, speed of 10, counterclockwise about z_0; frame 1 is attached to the helicopter, with x_1 forward and z_1 up; find the linear and angular velocities of the helicopter with respect to frame 0
    - We want $\dot{O}_1^0$ and w_1^0
    - Can think of there being a revolute joint between the two frames, so q_1 is the angle between x_0, x_1 about z_0
    - Because we can easily write a closed form for O_1^0, the simplest method to get the linear velocity is to simply differentiate it
        * $O_1^0 = \begin{bmatrix} 30\cos(\omega t) + c_1 \\ 30\sin(\omega t) + c_2 \\ 100 \end{bmatrix}$
        * c_1, c_2 account for the initial conditions
        * $\dot{O}_1^0 = \begin{bmatrix} -30\omega\sin(\omega t) \\ 30\omega\cos(\omega t) \\ 0 \end{bmatrix}$
        * Now $\|\dot{O}_1^0\| = \sqrt{(-30\omega\sin(\omega t))^2 + (30\omega\cos(\omega t))^2} = 30\omega = 10 \implies \omega = \frac{1}{3}$
        * $\dot{O}_1^0 = \begin{bmatrix} -10\sin(t/3) \\ 10\cos(t/3) \\ 0 \end{bmatrix}$
    - For the angular velocity, we make the assumption that the helicopter always flying forward, i.e. the direction of x_1 is the same as the direction of the linear velocity $\dot{O}_1^0$
    - Since the angular velocity Jacobian is trivial, we can simply compute $\begin{aligned} w_1^0 &= J_w(q)\dot{q} \\ &= J_w(q_1)\dot{q}_1 \\ &= \begin{bmatrix}\rho_1 z_0^0\end{bmatrix}\dot{q}_1 \\ &= \begin{bmatrix} 0 \\ 0 \\ 1/3 \end{bmatrix} \end{aligned}$
    - The brute-force approach is to find R_1^0, compute its derivative $\dot{R}_1^0$, and extract w_1^0 from $S(w_1^0) = \dot{R}_1^0(R_1^0)^T$
        * Find x_1^0 by normalizing $\dot{O}_1^0$, so $x_1^0 = \begin{bmatrix} -\sin(t/3) \\ \cos(t/3) \\ 0 \end{bmatrix}$
        * We also have z_1 parallel to z_0 so $z_1^0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$
        * Finally to get y_1^0 we need to take a cross product, so its normal to the other 2 and the right hand rule holds
        * $R_1^0 = \begin{bmatrix} x_1^0 & y_1^0 & z_1^0 \end{bmatrix} = \begin{bmatrix} -\sin(t/3) & -\cos(t/3) & 0 \\ \cos(t/3) & -\sin(t/3) & 0 \\ 0 & 0 & 1 \end{bmatrix}$
        * Now we can take the derivative of each term with respect to t, multiply by $(R_1^0)^T$ and simplify
        * In the end we're left with $S(w_1^0) = \begin{bmatrix} 0 & -1/3 & 0 \\ 1/3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \implies w_1^0 = \begin{bmatrix} 0 \\ 0 \\ 1/3 \end{bmatrix}$