Motion Planning for Recycling Robotics

Nicholas Link, Samuel Tormey, Ricky LeVan

December 10, 2014

- MRF sorting video

Introduction

We are implementing a robotic arm algorithm to efficiently grab items on a conveyor belt.

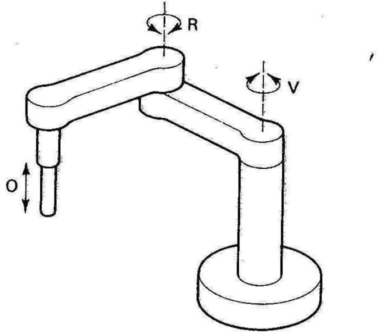


FIGURE 11.8 SCARA body-and-arm assembly (VRO).

Motivation

- Inspiration from improved recycling efficiency.
- WM Houston's Material Recycling Facility (MRF) hires full time employees
- We aim to control a robotic arm to efficiently sort the most recyclable materials

Overview

Our project considers

- Minimal-time arm motion
    - Robotic Dynamics
    - Optimization Formation
    - Implementation
    - Results
- Optimal grab strategies
- Robot hardware

Lagrangian Physics

We can calculate the kinetic energy of a simpler revolute system as

$$E = \frac{1}{2}\dot{\theta}_1^2(I_4 + 2I_5\cos\theta_2) + \frac{1}{2}\dot{\theta}_2^2 I_6 + \dot{\theta}_1\dot{\theta}_2(I_3 + I_5\cos\theta_2) \tag{1}$$

We ignore gravity and thus potential energy. By the Euler-Lagrange equation, the system dynamics are constrained by

$$\frac{\partial E}{\partial \theta_j} - \frac{d}{dt}\frac{\partial E}{\partial \dot{\theta}_j} = 0 \tag{2}$$

Equations of Motion

We desire to solve for the acceleration terms so we can predict future state:

$$\ddot{\theta} = H^{-1}(h - u) \tag{3}$$

where H is the inertial matrix, h represents the centripetal and Coriolis forces, and u is the control torque. We take derivatives with respect to position, velocity, and time to obtain

$$H = \begin{bmatrix} I_4 + 2I_5 \cos\theta_2 & I_3 + I_5 \cos\theta_2 \\ I_3 + I_5 \cos\theta_2 & I_6 \end{bmatrix} \tag{4}$$

$$h = \begin{bmatrix} -2I_5\dot{\theta}_2\dot{\theta}_1 \sin\theta_2 - I_5\dot{\theta}_2^2 \sin\theta_2 \\ I_5\dot{\theta}_1^2 \sin\theta_2 \end{bmatrix} \tag{5}$$

Optimization Problem

$$
\begin{aligned}
\min \quad & T \\
\text{s.t.} \quad & \frac{dx(t)}{dt} = f(x(t), u(t)) && t \in [0, T] \\
& x(0) = x_0 \\
& x(T) = x_T
\end{aligned}
$$

where

- T = total time taken by path
- t = time
- $x(t)$ = state at time t
- $u(t)$ = control at time t
- x_0 = initial state
- x_T = end state

Transformation

$$\tau = \frac{t}{T} \qquad \tau \in [0,1]$$

$$\tilde{u}(\tau) = u(t)$$

$$\tilde{x}(\tau) = x(t)$$

$$\tilde{x}(0) = x_0$$

$$\tilde{x}(1) = x_\mathrm{T}$$

$$\frac{d\tilde{x}(\tau)}{d\tau} = Tf\left(\tilde{x}(\tau), \tilde{u}(\tau)\right) \quad \left(\text{because } \tfrac{d\tilde{x}(\tau)}{d\tau} = \tfrac{dx(t)}{d\tau} = \tfrac{dx(T\tau)}{d\tau} = Tf(\tilde{x}(\tau), \tilde{u}(\tau))\right)$$

Transformed Optimization Problem

$$
\begin{aligned}
&\min \;\; T \\
&\text{s.t. } \frac{d\tilde{x}(\tau)}{d\tau} = Tf(\tilde{x}(\tau), \tilde{u}(\tau)) && \tau \in [0, 1] \\
&\qquad \tilde{x}(0) = x_0 \\
&\qquad \tilde{x}(1) = x_{\mathrm{T}}
\end{aligned}
$$

where

- T = total time taken by path
- τ = time/(total time)
- $\tilde{x}(\tau)$ = state at time $\tau = \frac{t}{T}$
- $\tilde{u}(\tau)$ = control at τ
- x_0 = initial state
- x_{T} = end state

Algorithm

Forward Euler:

$$\frac{\tilde{x}(\tau_{\mathrm{k+1}}) - \tilde{x}(\tau_{\mathrm{k}})}{d\tau} \approx \frac{d\tilde{x}(\tau_{\mathrm{k}})}{d\tau}$$

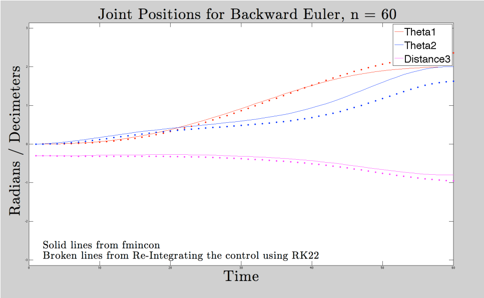
$$\frac{\tilde{x}(\tau_{\mathrm{k+1}}) - \tilde{x}(\tau_{\mathrm{k}})}{d\tau} - Tf(\tilde{x}(\tau_{\mathrm{k}}), \tilde{u}(\tau_{\mathrm{k}})) \approx 0$$

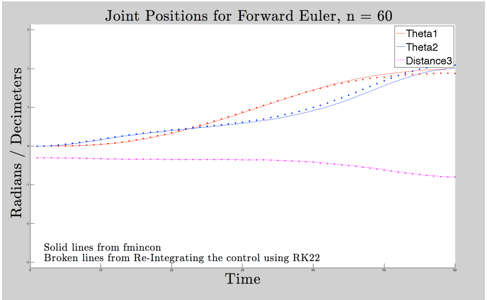
Backward Euler:

$$\frac{\tilde{x}(\tau_{\mathrm{k+1}}) - \tilde{x}(\tau_{\mathrm{k}})}{d\tau} - Tf(\tilde{x}(\tau_{\mathrm{k+1}}), \tilde{u}(\tau_{\mathrm{k+1}})) \approx 0$$

for $\tau_{\mathrm{k}} = \mathrm{k}d\tau,\ \ \mathrm{k} = 1..N,\ \ d\tau = \dfrac{1}{N+1}$

Results


Joint Positions for Backward Euler, n = 60
Theta1
Theta2
Distance3
Radians / Decimeters
Solid lines from fmincon
Broken lines from Re-Integrating the control using RK22
Time


Joint Positions for Forward Euler, n = 60
Theta1
Theta2
Distance3
Radians / Decimeters
Solid lines from fmincon
Broken lines from Re-Integrating the control using RK22
Time

Optimal Grabbing Strategy

Some example grab strategies:

- First-in First-out

$$p(k) = t_j(k), \quad j = min(i : t_i(k) \in P(k))$$

- Shortest Processing Time (SPT)

$$\min_{1 \leq j \leq n} t_j$$

- Entry-biased SPT

$$\min_{1 \leq j \leq n} \lambda d_j + t_j$$

Hardware Demonstration

We hope to demonstrate our solution with real hardware, which would consist of:

- A moving conveyor belt, possibly a treadmill
- Objects with QR codes
- Robotic arm for gripping
    - Adept Cobra SCARA
- Extra Controllers?

Future Work

- Finalize minimum path finding algorithm
  - Hook into optimal control
  - Test options
- Improve Optimal Motion Planning
  - More algorithms
  - Increase efficiency, ex: Jacobian
- Hardware Demonstration
- Complexify with Temporal Goals