Carnegie Mellon University

Table Tennis Playing Robots?!

Mohit Vikas Javale, Atharva Sunder Ramdas

Problem Statement

Motivation

Table tennis is a fast-paced, high-dexterity task that is easy to learn but difficult to master—even for humans. A major challenge arises from the ball's lightweight and smooth design, which makes it highly susceptible to aerodynamic forces. These effects, primarily accentuated by heavy spin, significantly influence the ball's motion and make accurate placement of the ball difficult.

Also, both of us are table tennis player and enjoy it very much: we think it would be cool to figure out what the optimal stroke looks like in many cases to improve our own gameplay :)

Goals

- To figure out dynamics of interactions taking place in a table-tennis scenario and implement then to create a custom simulator.
- Devise an optimal control strategy for a robotic 6DOF manipulator to successfully return a ball.
- Get 2 such manipulator robots to successfully hold long rallies with one another.
- Hopefully, also get this setup to where it could be deployed in real-time over actual hardware

Approach:

- **Focused Player**: In this approach, given some initial condition of the ball, the control algorithm tries to find a solution that return the ball to a specified desired location at a specified desired time while satisfying table tennis rules.
- **Defensive Player**: In this approach, given some initial condition, the control algorithm itself figures out an optimal location and time to return the ball, such that table tennis rules are followed over the trajectory.

Dynamics Equations

Ball Flight Dynamics

Accounts for gravity, air drag due to ball velocity and Magnus effect due to spin.

$$\ddot{\mathbf{b}} = \mathbf{g} - C_D \nu \dot{\mathbf{b}} + C_L \boldsymbol{\omega} \times \dot{\mathbf{b}}$$

Table Contact Reset Map

Accounts the interchange between linear and angular velocity due to friction and restitution.

$$\boldsymbol{v}_b' = \boldsymbol{A}_v \boldsymbol{v}_b + \boldsymbol{B}_v \boldsymbol{\omega}_b$$
$$\boldsymbol{\omega}_b' = \boldsymbol{A}_\omega \boldsymbol{v}_b + \boldsymbol{B}_\omega \boldsymbol{\omega}_b$$

$$\boldsymbol{A}_v := \begin{bmatrix} 1-\alpha & 0 & 0 \\ 0 & 1-\alpha & 0 \\ 0 & 0 & -e_t \end{bmatrix}, \boldsymbol{B}_v := \begin{bmatrix} 0 & \alpha r & 0 \\ -\alpha r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{A}_\omega := \begin{bmatrix} 0 & -\frac{3\alpha}{2r} & 0 \\ \frac{3\alpha}{2r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \boldsymbol{B}_\omega := \begin{bmatrix} 1-\frac{3\alpha}{2} & 0 & 0 \\ 0 & 1-\frac{3\alpha}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{v}_{bT} := [v_{bx} \ v_{by} \ 0]^{\mathrm{T}} + \boldsymbol{\omega} \times \boldsymbol{r} = \begin{bmatrix} v_{bx} - r\omega_{by} \\ v_{by} + r\omega_{bx} \\ 0 \end{bmatrix}$$

$$\alpha := \mu(1+e_t)\frac{|v_{bz}|}{\|\boldsymbol{v}_{bT}\|}$$

Racket Contact Reset Map

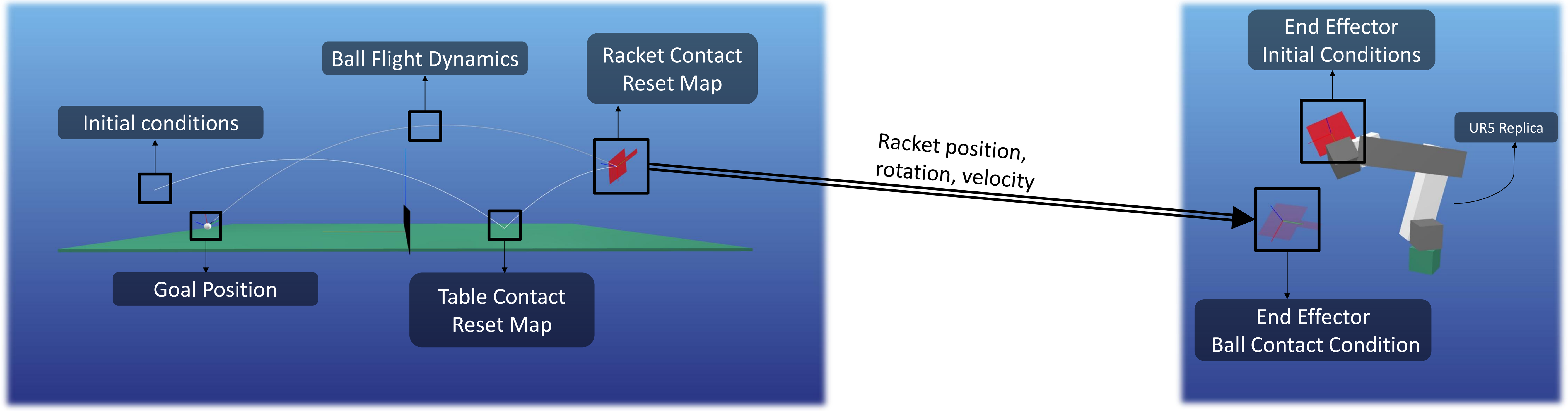
Very similar to table contact reset map, just expressed in the racket frame.

$$\dot{\mathbf{b}}_{\text{out}} = (\mathbf{I} - \bar{\mathbf{A}}_r)\mathbf{v} + \bar{\mathbf{A}}_r \dot{\mathbf{b}}_{\text{in}} + \mathbf{R}_{\text{rot}} \mathbf{B}_r \boldsymbol{\omega}$$

$$\bar{\mathbf{A}}_r := \mathbf{R}_{\text{rot}} \mathbf{A}_r \mathbf{R}_{\text{rot}}^{\mathrm{T}}$$

$$\mathbf{A}_r = \begin{bmatrix} 1-\kappa & 0 & 0 \\ 0 & 1-\kappa & 0 \\ 0 & 0 & -\epsilon_r \end{bmatrix}, \mathbf{B}_r = \begin{bmatrix} 0 & \kappa r_R & 0 \\ -\kappa r_R & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Implementation Details



Ball Trajectory Optimization

Free-time Formulation

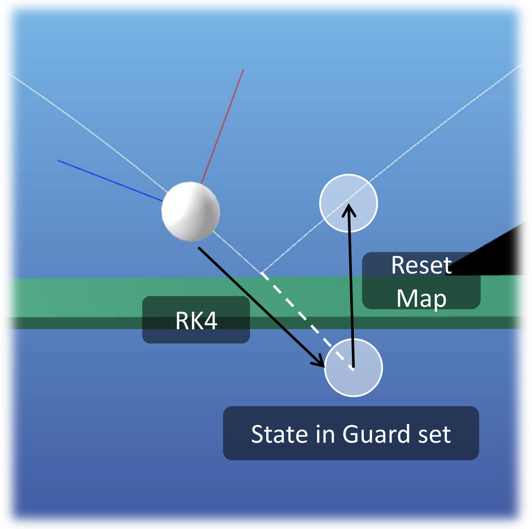
The problem is modelled as a free-time problem, to give the algorithm freedom to choose a striking time that makes it possible for the ball to successfully reach the goal position.
For hybrid transitions, we enforce first bounce at n_1 step, racket contact at $n_1 + n_2$ step, and second bounce at $n_1 + n_2 + n_3$ step.

States

- $[ball\ state \quad step\ length]\ \forall\, t \in (1, N)$
- $[racket\ state]\ at\ t_{strike}$

Objective Function Components

- Step length regularization (to prevent large step lengths)
- Racket state regularization (to restrict angles to $(-\pi, \pi)$, reduce striking velocity)
- Reach goal state at final timestep



Constraints

- Initial condition constraint
- Ball flight dynamics for aerial phases
- Table contact reset map at n_1 and $n_1 + n_2 + n_3$ timesteps.
- Racket contact reset map at $n_1 + n_2$ timestep.
- Sum of step lengths between $n_1 + n_2$ and $n_1 + n_2 + n_3$ equals desired return time
- Step length positivity (and upper bound for safety)

Manipulator Trajectory Optimization

Kinematic Formulation

Trajectory optimization is done over kinematics and not dynamics due to compute limitations. (6DOF manipulator dynamics are highly non-linear)

States

- $[joint\ pos \quad joint\ vel \quad joint\ accel]\ \forall\, t \in (1, N)$

Objective Function Components

- Acceleration regularization (to enforce smooth trajectories)
- Reach racket state at strike time
- Reach zero config state at final timestep

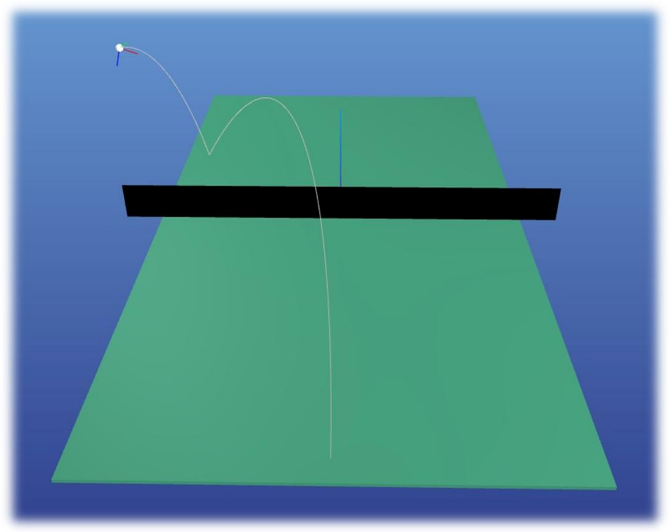
Constraints

- Initial condition constraints
- Kinematic derivative constraints

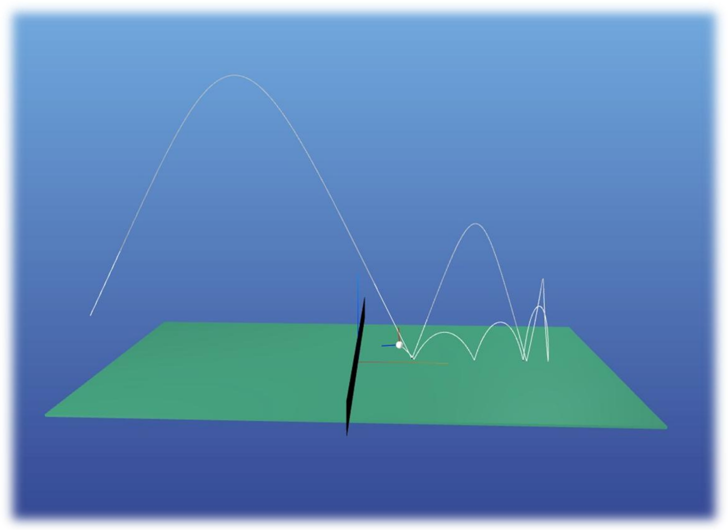
Future Work

- Make manipulator capable to hit backhand shots.
- Implement defensive player formulation
- Add second manipulator
- Perform live optimization for both manipulators
- Optimize solver for near real-time deployment

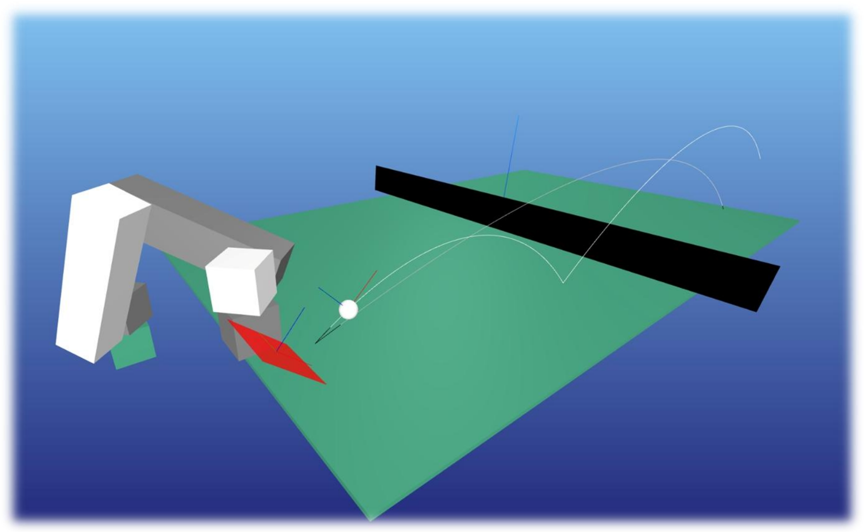
Preliminary Results



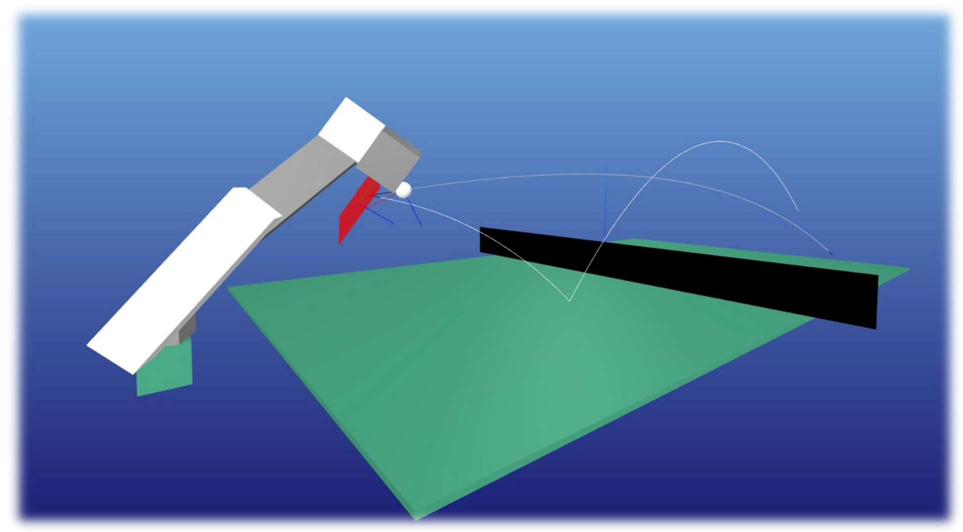
Aerial swing due to sidespin, (caused by aerodynamic effects)



Ball returning due to backspin (caused by frictional effects)



Slice shot to counter incoming backspin



Closed racket angle to counter incoming topspin