Trajectory Optimization and Control for Ball Bouncing Quadrotors

Ryan Sandzimier, Jerry Ng and Filippos E. Sotiropoulos,

***Abstract*—The control of quadrotors to impact and juggle multiple balls in the air with the aim of transporting them to a specific target is tackled. Specifically this report addresses the formulation of nonlinear optimizations for the generation of trajectories necessary to perform the ball juggling task. By specifying a mode sequence, applying direct collocation constraints using cubic splines and further constraining the optimization feasible trajectories were generated. Thereafter, using time varying LQR we control the quadrotors to track the reference optimal trajectories. Successful juggling of multiple balls with multiple quadrotors was achieved.**

***Index Terms*—Trajectory optimization, nonlinear optimization, hybrid dynamics, quadrotor control, time varying LQR**

I. Introduction

In this report we investigate the trajectory optimization and control of multiple planar quadrotor vehicles to collaboratively keep balls in the air by bouncing them off of their top surface. Despite operating in an idealised 2D environment with perfect collisions the problem still presents some technical challenges due to the both nonlinear and hybrid dynamics which will need to be optimized over.

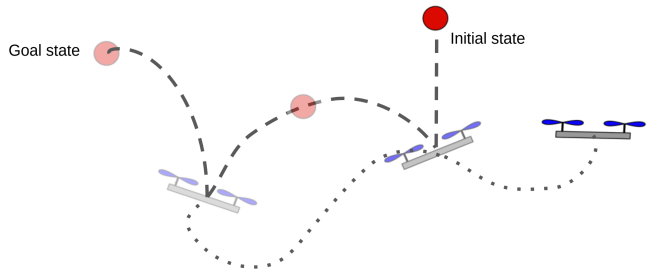

Fig. 1. Our objective is to transport balls by bouncing them multiple times with planar quadrotors

II. Literature Review

There has been a lot of work on control and trajectory optimization of quadrotors for performing various navigation and manipulation tasks [1]–[3].

Furthermore there has been work on the subject of quadrotors and juggling balls using a badminton racquet head attached to a quadrotor [4]. In this work, the group was able to generate trajectories for: 1) a single quadrotor to return a ball that is thrown to it, 2) two quadrotors passing a ball back and forth, 3) a single quadrotor to juggle a ball. The system operated based on generating trajectories for a single impact at a time.

In this project we will attempt to generate trajectories for multiple quadrotors and balls while also using multiple impacts to achieve a specific desired end state.

F. E. Sotiropoulos (fes@mit.edu), R. Sandzimier (rsandz@mit.edu) and J. Ng (jerryng@mit.edu) are with the Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, USA. Code used for this project can be found at:
https://github.com/rsandzimier/quadrotor_juggling

III. Method

A. Simulated environment

In our task we aim to optimize the juggling of n_{balls} balls using n_{quads}. In order to do this, we must first simulate the free dynamics of the quadrotors and balls, as well as the collision dynamics between quadrotors and balls.

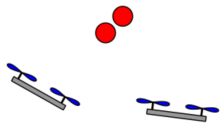

Fig. 2. Simulated quadrotor and ball environment.

1) Free Dynamics: Here we describe the system dynamics used for both simulation and optimization. The quadrotor dynamics in free motion are described by:

$$\begin{aligned} m_q\ddot{x} &= -(u_1+u_2)\sin\theta \\ m_q\ddot{y} &= (u_1+u_2)\cos\theta - m_q g \\ I_q\ddot{\theta} &= r(u_1-u_2) \end{aligned} \tag{1}$$

where x, y, and θ are the Cartesian coordinates of the quadrotor, and u_1 and u_2 are the thrusts of the propellers, which we treat as inputs to the system. m_q and I_q are the mass and moment of inertia of the quadrotor, respectively. r is the length of the moment arm between the propeller and center of mass of the quadrotor. The full state of the i-th quadrotor is then:

$$\mathbf{x}_i = [x_i \;\; y_i \;\; \theta_i \;\; \dot{x}_i \;\; \dot{y}_i \;\; \dot{\theta}_i]^T \tag{2}$$

and the input to the i-th quadrotor is then:

$$\mathbf{u}_i = [u_{1_i} \;\; u_{2_i}]^T \tag{3}$$

The ball dynamics in free motion are described by projectile motion with negligible air resistance:

$$\begin{aligned} \ddot{x} &= 0 \\ \ddot{y} &= -g \end{aligned} \tag{4}$$

where x and y are the Cartesian coordinates of the ball. The full state of the j-th ball is then:

$$\mathbf{x}_j = [x_j \ y_j \ \dot{x}_j \ \dot{y}_j]^T \tag{5}$$

2) Collision Dynamics: We model collisions using discrete updates to the quadrotor and ball states whenever either two balls collide with each other or a quadrotor collides with a ball. We did not implement collision dynamics for collisions between two quadrotors. We implement collision detection using witness functions. The witness functions calculate the distance between the outer surfaces of two objects. When a witness function evaluates to a positive value when the two objects are not colliding and to a negative value when the two objects are colliding. Therefore, when a witness function changes from a positive value to a non-positive value, a discrete update is applied to the states of the corresponding objects.

First, we describe these discrete collision dynamics for collisions between two balls. We treat these collisions as elastic collisions. Therefore, both momentum and energy are conserved.

$$\begin{aligned} m_b\mathbf{v}_{1_i} + m_b\mathbf{v}_{2_i} &= m\mathbf{v}_{1_f} + m_b\mathbf{v}_{2_f} \\ \frac{1}{2}m_b\|\mathbf{v}_{1_i}\|^2 + \frac{1}{2}m_b\|\mathbf{v}_{2_i}\|^2 &= \frac{1}{2}m_b\|\mathbf{v}_{1_f}\|^2 + \frac{1}{2}m_b\|\mathbf{v}_{2_f}\|^2 \end{aligned} \tag{6}$$

where $\mathbf{v}_{1_i}$ and $\mathbf{v}_{2_i}$ are vectors representing the pre-collision velocities of the two balls involved in the collision. $\mathbf{v}_{1_f}$ and $\mathbf{v}_{2_f}$ are the corresponding post-collision velocities of the two balls. Fig. 3 illustrates two balls in collision. The unit vector $\mathbf{i}$ points in the direction normal to the collision surfaces. Likewise, the unit vector $\mathbf{j}$ points in the direction perpendicular to $\mathbf{i}$. Solving 6 for $\mathbf{v}_{1_f}$ and $\mathbf{v}_{2_f}$ and using the fact that the component of the velocities in the direction of $\mathbf{j}$ remains unchanged:

$$\begin{aligned} \mathbf{v}_{1_f} &= (\mathbf{v}_{2_i} \cdot \mathbf{i})\mathbf{i} + (\mathbf{v}_{1_i} \cdot \mathbf{j})\mathbf{j} \\ \mathbf{v}_{2_f} &= (\mathbf{v}_{1_i} \cdot \mathbf{i})\mathbf{i} + (\mathbf{v}_{2_i} \cdot \mathbf{j})\mathbf{j} \end{aligned} \tag{7}$$

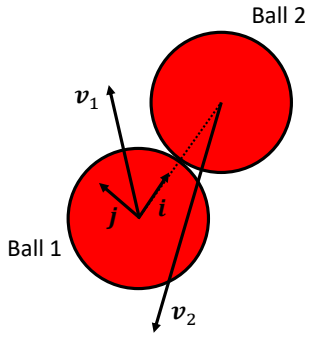

Fig. 3. Collision between two balls

Whenever the witness function triggers, 7 is used to update the state of the balls involved in the collision.

Next, we describe the discrete collision dynamics for collisions between quadrotors and balls. We treat these collisions as elastic collisions. Therefore, linear momentum, angular momentum, and energy are conserved.

$$\begin{aligned} m_q\mathbf{v}_{q_i} + m_b\mathbf{v}_{b_i} &= m_q\mathbf{v}_{q_f} + m_b\mathbf{v}_{b_f} \\ L_{q_i} + L_{b_i} &= L_{q_f} + L_{b_f} \\ L_{q_i} &= I_q\omega_{q_i} + m_q\mathbf{r}_q \times \mathbf{v}_{q_i} \\ L_{b_i} &= m_b\mathbf{r}_b \times \mathbf{v}_{b_i} \\ L_{q_f} &= I_q\omega_{q_f} + m_q\mathbf{r}_q \times \mathbf{v}_{q_f} \\ L_{b_f} &= m_b\mathbf{r}_b \times \mathbf{v}_{b_f} \\ E_{trans_i} + E_{rot_i} &= E_{trans_f} + E_{rot_f} \\ E_{trans_i} &= \frac{1}{2}m_q\|\mathbf{v}_{q_i}\|^2 + \frac{1}{2}m_b\|\mathbf{v}_{b_i}\|^2 \\ E_{rot_i} &= \frac{1}{2}I_q\|\omega_{q_i}\|^2 \\ E_{trans_f} &= \frac{1}{2}m_q\|\mathbf{v}_{q_f}\|^2 + \frac{1}{2}m_b\|\mathbf{v}_{b_f}\|^2 \\ E_{rot_f} &= \frac{1}{2}I_q\|\omega_{q_f}\|^2 \end{aligned} \tag{8}$$

where $\mathbf{v}_{q_i}$ and $\mathbf{v}_{b_i}$ are vectors representing the pre-collision velocities of the quadrotor and ball, respectively, involved in the collision. $\mathbf{v}_{q_f}$ and $\mathbf{v}_{b_f}$ are the corresponding post-collision velocities of the quadrotor and ball, respectively. ω_{q_i} and ω_{q_f} are vectors representing the pre-collision and post-collision angular velocities of the quadrotor, respectively. Fig. 4 illustrates a quadrotor and ball in collision. The unit vector $\mathbf{i}$ points in the direction normal to the collision surfaces. Likewise, the unit vector $\mathbf{j}$ points in the direction perpendicular to $\mathbf{i}$. Vectors $\mathbf{r}_q$ and $\mathbf{r}_b$ represent the moment arms between the center of mass of the quadrotor-ball system and the quadrotor and ball, respectively. Solving 8 for $\mathbf{v}_{q_f}$, $\mathbf{v}_{b_f}$, and ω_{q_f} and using the fact that the component of the velocities in the direction of $\mathbf{j}$ remains unchanged:

$$\begin{aligned} \mathbf{v}_{q_f} &= v_{q_{fpar}}\mathbf{i} + (\mathbf{v}_{q_i} \cdot \mathbf{j})\mathbf{j} \\ \mathbf{v}_{b_f} &= v_{b_{fpar}}\mathbf{i} + (\mathbf{v}_{b_i} \cdot \mathbf{j})\mathbf{j} \\ \omega_{q_f} &= \omega_{q_i} + \frac{m_q}{I_q}\mathbf{r}_q \times (\mathbf{v}_{q_i} - \mathbf{v}_{q_f}) + \frac{m_b}{I_q}\mathbf{r}_b \times (\mathbf{v}_{b_i} - \mathbf{v}_{b_f}) \\ v_{q_{fpar}} &= \frac{c_1(\mathbf{v}_{q_i}(m_q - m_b) + 2m_b\mathbf{v}_{b_i}) + c_2\mathbf{v}_{q_i}}{c_1(m_q + m_b) + c_2} \cdot \mathbf{i} \\ v_{b_{fpar}} &= \frac{c_1(\mathbf{v}_{b_i}(m_b - m_q) + 2m_q\mathbf{v}_{q_i}) + c_2\mathbf{v}_{b_i}}{c_1(m_q + m_b) + c_2} \cdot \mathbf{i} \\ c_1 &= m_q m_b I_q \\ c_2 &= m_q^2 m_b^2((\mathbf{r}_q - \mathbf{r}_b) \cdot \mathbf{j})^2 \end{aligned} \tag{9}$$

Whenever the witness function triggers, 9 is used to update the state of the quadrotor and ball involved in the collision.

B. Trajectory Optimization

In the optimization the trajectory is represented by a series of knot points with cubic splines used to interpolate between them. This allows one to constrain the optimization to our system dynamics using a Direct Collocation approach [5]. As in the standard direct collocation method we constrain the mathematical program such that the derivative of the spline at

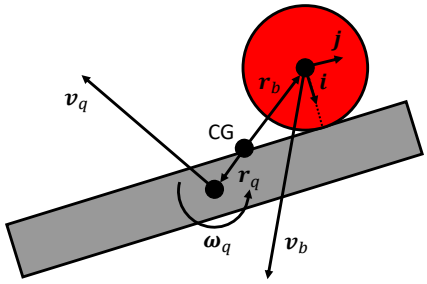

Fig. 4. Collision between a quadrotor and a ball

the collocation points (which fall at the midpoints between the breakpoint) is equal to the transition from the system dynamics.

$$\begin{aligned}
\text{find} \quad & \mathbf{x}[\cdot], \mathbf{h}[\cdot], \mathbf{u}[\cdot] \\
\text{s.t.} \quad & \dot{\mathbf{x}}(t_{c,n}) = f(\mathbf{x}(t_{c,n}), \mathbf{x}(t_{c,n})), \quad \forall n \in [0, N-1] \setminus \mathbf{n}_{skip} \\
& \mathbf{x}[0] = \mathbf{x}_0 \\
& \mathbf{u} \leq u_{max} \\
& \mathbf{x}_{\mathbf{i}}[t_f] = \mathbf{x}_{\mathbf{i}f} \\
& \mathbf{x}_{\mathbf{j}f} - \varepsilon \leq \mathbf{x}_{\mathbf{j}}[t_f] \geq \mathbf{x}_{\mathbf{j}f} + \varepsilon \\
& h_{min} \times N \leq \mathbf{h}[N] \leq h_{max} \times N
\end{aligned} \tag{10}$$

where $\mathbf{n}_{skip} = \{\mathbf{n}_{impact}, \mathbf{n}_{impact}+1, \mathbf{n}_{impact}-1\}$, and i corresponds to the i-th quadrotor and j corresponds to the j-th ball. Instead for those time steps backward Euler constraints are added to constrain the program to the system dynamics. This is to allow for the discrete update at the collision time.

Additional constraints are added to further constrain the problem. The input is constrained under a maximum value. Guards are used to implement discrete collision dynamics and decide on the time steps that correspond to impacts. Because there is only a single mode for the system, the guards reset the state of the system to the same mode. These guards essentially measured the distance from the outer radius of each ball to the contact box for each quadrotor, such that when any quadrotor and any ball made contact the guard equaled zero. The specific time steps for these guards to trigger is prescribed to the optimization problem. These guards are implemented as witness functions.

Other constraints include:

1) **Bounds** for variable time steps
2) **Time steps** for **collisions**.
3) **Pitch angles** for quadrotors.
4) **Final** and **initial** state constraints.
5) Collisions between balls and quadrotors are constrained in position to be at the center of each quadrotor

In addition, we perform re-optimizations over the course of the entire simulation to correct for impreciseness in the impacts compared to the previous optimization. As opposed to the continuous control of the quadrotors clearly it is not possible to change the course of a ball after it has been impacted. Hence, it is consequently necessary to re-optimize trajectories after collisions.

We determine the point in the simulation to perform the re-optimization based on the sequence of quadrotor-ball impacts. The re-optimization are scheduled to occur such that every ball has at most a single collision before re-optimization. Then the optimization is triggered to occur in between the last impact before re-optimizing and the impact directly following. To have any ball never have two consecutive impacts without a re-optimization in between we are assuming that all other balls will have collided before the ball impacts again. Furthermore, given the assumption that the new optimized trajectory will be very similar to the new result

Given the repeated optimizations with a fixed goal and a receding time horizon this is similar to a low frequency Model Predictive Controller.

C. Control

The numerical integration performed in simulation and that which is implied by the direct collocation constraints are not exactly equivalent and thus the open loop implementation of the optimal control inputs would lead to the optimal trajectory and simulated states to diverge. To mitigate this the simulated quadrotors are controlled using a time varying LQR controller with the objective of minimizing the error between the quadrotor state and the reference trajectory supplied by the optimization. The time varying linear control is implemented with the following steps. Firstly, the trajectory optimization outputs an optimal trajectory defined by the points on the spline $\{\mathbf{x}^*[\cdot], \mathbf{u}^*[\cdot]\}$ along the optimal time breaks $\mathbf{h}^*[\cdot]$. Then at each time-break t the linearized system based on a first order Taylor expansion is determined yielding the system matrices: A_t, B_t. Then a standard Linear Quadratic Regulator state feedback matrix, K_t is calculated for each time break. This creates a Zero-Order Hold trajectory of K gain matrices. Then when simulating the system the total control for each quadrotor:

$$\mathbf{u}_i(t) = \mathbf{u}_i^*(t) - K_t(\mathbf{x}_i(t) - \mathbf{x}_i^*(t)) \tag{11}$$

IV. Results

Using the methods outlined in section III we were able to achieve optimal trajectories for various scenarios where the quadrotors were able to successfully juggle balls from an initial position to a desired final state. In Fig. 5 the simulated trajectory for a single ball.

Furthermore, in Fig. 6 one can see the trajectory for one of the quadrotors during the 2 ball 2 quadrotor task which the ball trajectory in Fig. 5 was also illustrated for. As can be seen the controller keeps the vehicle on the optimized trajectory.

V. Discussion

A. Limitations, Challenges and Development steps

The most overarching limitation of the approach and method that we have implemented is that the sequence of collisions is fixed a priori. This has several consequences on the capability of the optimization to find solutions. Firstly, the fixed time index when collision occurs means that the

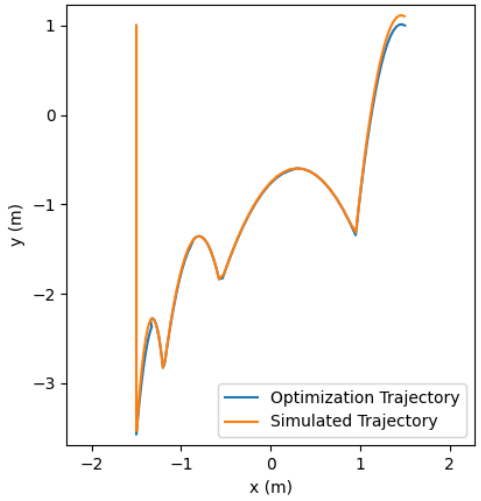

Fig. 5. The example optimized and simulated trajectory of one ball with initial position $(-1.5, 1.0)$ and goal position $(1.5, 1.0)$

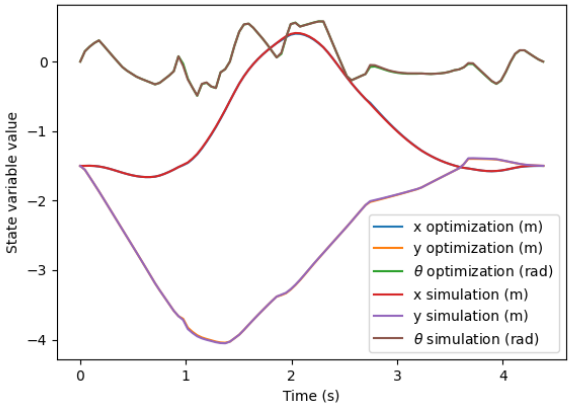

Fig. 6. Position and orientation variables for a single quadrotor from both optimization and simulation. The controller keeps the vehicle on the optimized trajectory.

timing of collisions is constrained by the minimum and maximum time increments h_{min} and h_{max}. Furthermore, the predetermined sequence of ball-quadrotor combinations limits the task and likely misses solutions with alternative impact combinations that would be more effective or could solve specific problems which failed to solve in our case. Given that simply evaluating all the potential combinations would be prohibitively computationally intensive ($\mathcal{O}((qb)^c)$ for q number of quadrotors, b number of balls and c number of collision) a higher level planner would need to be introduced to optimize or determine feasible combination sequences. Mixed-integer programming methods may need to be applied.

Several considerations were made regarding applying the final state to the optimization. One attempt was to not fix the final state of the optimization, and instead to assign a quadratic error cost only to the final time step. This however led to issues with the solver unable to solve the optimization problem during re-optimization towards the last set of collisions. In addition, it led to an increased time for optimization, which was somewhat expected. Another attempt at assigning cost was done to penalize the control inputs for all quadrotors and also for specific quadrotors. However, this attempt resulted in an inability to solve the mathematical program. We believe this may be due to the solver converging towards a local minimum solution that violates some of the prescribed constraints.

The final state constraints are not implemented as exact constraints due to an inability to solve the problem generally. Instead, the final state constraints were implemented in two different ways. The first was to constrain the final state to not care about the velocities of the balls and only constrain the positions. The second method used was to give a range for the final ball states; this relaxation allowed for the solver to solve the problem more readily for a wider variety of conditions.

In the implementation process, initially the collision dynamics were implemented in continuous time using a spring-damper model. However, this led to issues with the optimization. Firstly, without using a guard the optimizer has no gradient to use that can help it find a solution where the quadrotors and balls collide. Also, based on previous issues we had with no constraints for the control input, it seemed that this could have been due to high accelerations. As such, we converted to discrete collision dynamics. This limited us in that it required us to choose time steps for the collisions. Because of the variable time step, by choosing this time step we were essentially choosing a time window for the collision to happen. Because the controller for the quadrotor must be of a certain frequency for the optimization and the simulation to match, the variable time step is bound in to be under a threshold. With that in mind, the time window for collision is also fairly limited.

The current implementation does not have collision dynamics between the quadrotors, and in fact the quadrotors are allowed to overlap in the simulation. This allows for solutions from the solver that would be infeasible in real life. However, we faced issues in implementing guards related to the quadrotor-quadrotor interactions as these additional constraints led to the solver being unable to solve the mathematical program.

References

[1] S. Tang and V. Kumar, "Autonomous flight," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 29–52, 2018.

[2] M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2958–2964.

[3] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza, "Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload," in *Robotics: Science and Systems*, 2017.

[4] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *2011 IEEE/RSJ international conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 5113–5120.

[5] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.