# Whole-body Trajectory Optimization and Tracking for Agile Maneuvers for a Single-Spherical-Wheeled Balancing Mobile Manipulator

Juan Alvarez-Padilla
*Department of Electrical & Computer Engineering*
*Carnegie Mellon University*
jralvare@andrew.cmu.edu

Christian Berger
*The Robotics Institute*
*Carnegie Mellon University*
cberger2@andrew.cmu.edu

Sayan Mondal
*The Robotics Institute*
*Carnegie Mellon University*
sayanmon@andrew.cmu.edu

Haoru Xue
*The Robotics Institute*
*Carnegie Mellon University*
haorux@andrew.cmu.edu

Zhikai Zhang
*Department of Mechanical Engineering*
*Carnegie Mellon University*
zhikaiz@andrew.cmu.edu

***Abstract*—Several ballbots have been developed, yet only a handful have been equipped with arms to enhance their maneuverability and manipulability. The incorporation of 7-DOF arms to the CMU ballbot has presented challenges in balancing and navigation due to the constantly changing center of mass. This project aims to propose a control strategy that incorporates the arms dynamics. Our approach is to use a simplified whole-body dynamics model, with only the shoulder and elbow joints moving for each arm. This reduces the number of states and accelerates convergence. We focused on two specific tasks: navigation (straight and curved paths) and pushing against a wall. Trajectories were generated using direct collocation for the navigation task and hybrid contact trajectory optimization for pushing the wall. A time-variant linear quadratic regulator (TVLQR) was designed to track the trajectories. The resulting trajectories were tracked with a mean-average error of less than 4 cm, even for the more complex path. These experiments represent an initial step towards unlocking the full potential of ballbots in dynamic and interactive environments. Supplementary information, including code and animations, can be found at https://github.com/jrapudg/ocrl_ballbot_navigation_project.**

***Index Terms*—ballbot, contact, agile maneuvers, dircol, tvlqr, control, navigation**

## I. Introduction

Ballbots constitute a special class of mobile robots that can use their full dynamics to efficiently move through cluttered, interactive environments. They locomote using a single ball, on which they balance. This balancing behaviour gives them an inherent compliance and enables them to interact with people more effectively. A ballbot can be guided by a gentle nudge, but cannot be toppled by a strong shove. Several versions of these ballbots have been developed. They include the BallIP [4], Kugle [2], Rezero [5], and CMU ballbots [6]. Of these ballbots, only Rezero and the CMU ballbot have been given arms. Rezero is short and has a single 3-DOF arm [5]. The CMU ballbot is human-sized and has a pair of 7-DOF Barrett WAM arms [7]. While these arms add immense potential for manipulability and maneuverability, they are bulky and highly mobile, so that the center of mass of the robot is constantly changing. This greatly complicates the balancing task.

Project code: https://github.com/jrapudg/ocrl_ballbot_navigation_project

The manipulative potential of the CMU ballbot has only begun to be explored. Experiments have validated the ability of the ballbot to balance weights on its hands and move them around while stationary, as well as balance its body while slowly moving its arms [6]. However, these experiments have not shown the ballbot dynamically interacting with obstacles or people. Example goal tasks include opening a door, pushing a wheelchair, and cooperating with a human to carry a heavy object. A first step toward these tasks is generating dynamically feasible trajectories for agile maneuvers that can effectively interact with the environment.

This project aims to generate and track trajectories for the following tasks, as performed by the CMU ballbot:

1) Navigating with arms.
2) Making contact and pushing off a wall using the arms.

## II. Rigid Body Model

The CMU ballbot can be modeled as a set of rigid bodies and joints, as shown in Fig. 1 and Equation 1.

$$M(q)\dot{q} + c(q, \dot{q}, \omega_{ext}) = \tau \tag{1}$$

Where the matrix $M$ is the inertia matrix at a given configuration $q$, $c$ is the dynamic bias term, $\omega_{ext}$ is the external wrench and $\tau$ is the the motor inputs of the mechanism.

The full state space or configuration of the ballbot is given in Equation 2:

$$q = \begin{bmatrix} P_S & \phi & q_{a_L} & q_{a_R} \end{bmatrix}^T \in \mathbb{R}^{19} \tag{2}$$

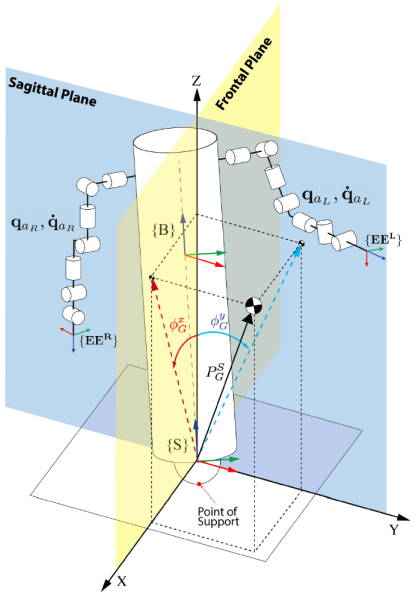


Fig. 1. Diagram of the CMU ballbot, as illustrated in [6], Fig. 4.17.

The vector $P_S = \begin{bmatrix} p_x & p_y \end{bmatrix}^T \in \mathbb{R}^2$ describes the position of the center of the ball on the horizontal ground plane. The vector $\phi = \begin{bmatrix} \phi_x & \phi_y & \phi_z \end{bmatrix}^T \in \mathbb{R}^3$ gives the Euler angles that together comprise the lean angle (ie. $\phi_x$, $\phi_y$) and the Yaw (i.e. $\phi_z$) of the ballbot. The vectors $q_{a_L}$, $q_{a_R} \in \mathbb{R}^7$ represent the joint angles of the left and right arms, respectively [6].

The full control inputs of ballbot are given in Equation 3:

$$\tau = \begin{bmatrix} f_s & \tau_{\phi_z} & \tau_{a_L} & \tau_{a_R} \end{bmatrix}^T \in \mathbb{R}^{17} \tag{3}$$

The vector $f_s = \begin{bmatrix} f_s^x & f_s^y \end{bmatrix}^T \in \mathbb{R}^2$ is the linear force applied by the ball at its point of contact with the ground. The value $\tau_{\phi_z}$ is the torque applied on the yaw axis. The vectors $\tau_{a_L}$, $\tau_{a_R} \in \mathbb{R}^7$ represent the joint torques of the left and right arms, respectively [6]. We use the RigidBodyDynamics library in Julia to simulate the CMU ballbot [3]. This library allows for quick and easy prototyping.

## III. Navigation

Ballbots are unique robots that use a single ball as the means of locomotion. These robots are known for their agile and omnidirectional movement capabilities, which make them suitable for a wide range of applications such as surveillance, inspection, and entertainment. The ballbot with arms is an interesting underactuated system to study for agile navigation like tasks. In case of the ballbot without arms, its behaviour is somewhat similar to that of a "Cartpole", but on a 2D plane instead of 1D. When the effective center of mass (COM) of the ballbot without arms is directly on top of the ball (ie. lean angle = 0), it is in an unstable equilibrium position. In order for the ballbot to navigate it has to first move the ball position in the opposite direction in order to generate a lean angle of the COM position with respect to the ball position which is responsible for the ballbot's motions. When arms are added it brings an additional level of complexity. The COM of a ballbot with arms can be changed with the movement of the arms which can also lead to the motion of the system. In this section, we focused on the navigation task of ballbots and investigated the use of trajectory optimization and tracking controllers.

Initially, we worked with a ballbot without arms, and later, we extended our work to a ballbot with arms. In both cases, we used direct collocation as the optimization method for generating the trajectories offline. Specifically, we specified the positions of the ball as the waypoints for the ballbot to follow, and used direct collocation to encode the dynamics of the robot and generate dynamically feasible optimal trajectories for a LQR cost function.

Once we had the trajectories, we designed a Time-Varying Linear Quadratic Regulator (TVLQR) as the tracking controller for the ballbot. TVLQR is a feedback control strategy that minimizes a quadratic cost function over a time horizon while taking into account the time-varying dynamics of the system. By using TVLQR, we were able to track the generated trajectories, even in the presence of disturbances and uncertainties.

In this report, we present our methodology and results for both the ballbot without arms and the ballbot with arms, and compare the performance of the two systems. We believe that our findings can contribute to the development of advanced navigation techniques for ballbots and other similar robots.

### A. Trajectory Optimization

To get the trajectory, we have defined a quadratic cost function and we have incorporated the discrete-time dynamics as one of the equality constraints. Other equality constraints involve the initial state, the goal state. We also enforce state-limits and torque limits as inequality constraints as shown in Equation 4. We have used IPOPT to solve this nonlinear optimization problem to get the state and the action trajectories. Hermite-Simpson Collocation is our selected medium-order direct collocation method.

$$\begin{aligned}
\min_{x_{1:N}, u_{1:N-1}} \quad & \sum_{i=1}^{N-1} \Big[ \frac{1}{2}(x_i - x_{ref,i})^T Q (x_i - x_{ref,i}) \\
& \quad + \frac{1}{2}(u_i - u_{ref,i})^T R u_i \Big] \\
& \quad + \frac{1}{2}(x_N - x_{ref,N})^T Q_f (x_N - x_{ref,N}) \\
\\
\text{st} \quad & x_1 = x_{\text{IC}} \\
& x_N = x_{\text{G}} \\
& x_{i+1} = f(x_i, u_i) \quad \text{for } i = 1, 2, \ldots, N-1 \\
& x_{min} \le x_i \le x_{max} \quad \text{for } i = 1, 2, \ldots, N \\
& u_{min} \le u_i \le u_{max} \quad \text{for } i = 1, 2, \ldots, N-1
\end{aligned} \tag{4}$$

Here, the state of the ballbot at each time step $i$ is defined by $x_i = \begin{bmatrix} q_i & \dot{q}_i \end{bmatrix}^T$. The full configuration space of the ballbot without arms, $q_{no\_arms} \in \mathbb{R}^5$ and with arms $q \in \mathbb{R}^{19}$. However, it is computationally very expensive to optimize over such high dimensional state space.

$$q_{no\_arms} = \begin{bmatrix} P_S & \phi \end{bmatrix}^T \in \mathbb{R}^5 \tag{5}$$

In order to compute an optimal solution in a reasonable time we lowered the state space of the system by fixing some of the revolute joints in the 7-DOF arms. We only allowed the shoulder joint and the elbow joint in each arm of the ballbot to be actuated. This helped us to have a reduced state dynamical system which was sufficient for the intended tasks to accomplish. Thus, the configuration space of this reduced system $q_{arms} \in \mathbb{R}^9$ as shown in Equation 6.

$$q_{arms} = \begin{bmatrix} P_S & \phi & q_{a_{L_{reduced}}} & q_{a_{R_{reduced}}} \end{bmatrix}^T \in \mathbb{R}^9 \tag{6}$$

$$q_{a_{L_{reduced}}} = \begin{bmatrix} q_{a_{L_{shoulder}}} & q_{a_{L_{elbow}}} \end{bmatrix}^T \in \mathbb{R}^2 \tag{7}$$

$$q_{a_{R_{reduced}}} = \begin{bmatrix} q_{a_{R_{shoulder}}} & q_{a_{R_{elbow}}} \end{bmatrix}^T \in \mathbb{R}^2 \tag{8}$$

Working with the low dimensional system also resulted in reducing the dimension of the control inputs as shown in Equation 9.

$$\tau_{arms} = \begin{bmatrix} \tau_{no\_arms}^T & \tau_{a_{L_{reduced}}} & \tau_{a_{R_{reduced}}} \end{bmatrix}^T \in \mathbb{R}^7 \tag{9}$$

where,

$$\tau_{no\_arms} = \begin{bmatrix} f_s & \tau_{\phi_z} \end{bmatrix}^T \in \mathbb{R}^3 \tag{10}$$

$$Q_{no\_arms} = \begin{pmatrix} Q_{P_S} & 0 \\ 0 & Q_\phi \end{pmatrix} \tag{11}$$

$$Q_{arms} = \begin{pmatrix} Q_{P_S} & 0 & 0 & 0 \\ 0 & Q_\phi & 0 & 0 \\ 0 & 0 & Q_{q_{a_L}} & 0 \\ 0 & 0 & 0 & Q_{q_{a_R}} \end{pmatrix} \tag{12}$$

An important thing to note here is that the value of $Q_{P_S}$ associated to the ball position $P_S$, is chosen to be significantly larger (100 times) as compared to the other values of $Q$, which are associated to the rest of the state (Equations 11 and 12). This encourages the ballbot to follow the desired path.

### B. Time-Variant Linear Quadratic Regulator

Once we obtained the trajectories using Direct Collocation, we implemented TVLQR as a tracking controller.

$$P_N = Q_N \tag{13}$$

$$K_i = (R + B_i^T P_{i+1} B_i)^{-1} B_i^T P_{i+1} A_i \tag{14}$$

$$P_i = Q + A_i^T P_{i+1} (A_i - B_i K_i) \tag{15}$$

where

$$A_i = \left. \frac{\partial f}{\partial x} \right|_{x_{ref,i}, u_{ref,i}} \tag{16}$$

$$B_i = \left. \frac{\partial f}{\partial u} \right|_{x_{ref,i}, u_{ref,i}} \tag{17}$$

Hence, we get a feedback policy $K_i$ to compensate for disturbances around tracking points. We obtained the discrete-time dynamics $f$ (Equation 1) using Julia's RigidBodyDynamics package [3] and use Runge-Kutta 4 order integrator for simulation.

### C. Results

We show the performance of navigation for the ballbot without arms in Figs. 2, 3 and 4. Clearly, the ballbot without arms is able to navigate accurately as desired.

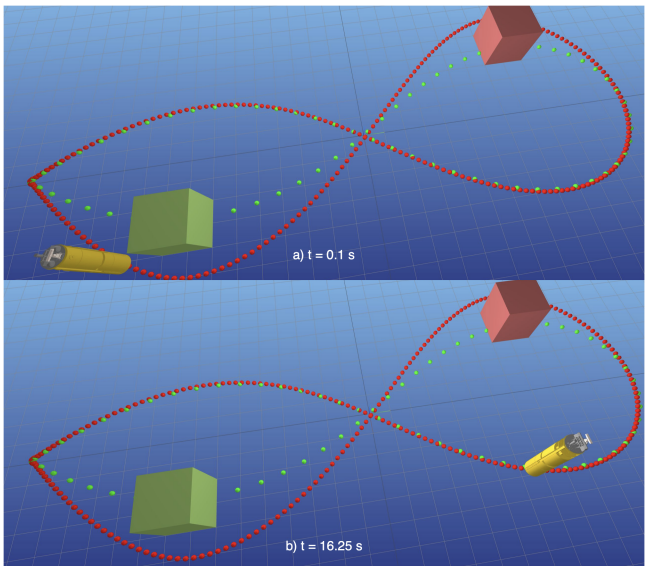


Fig. 2. Ballbot, with no arms, is navigating the obstacle-free curved trajectory.

It is critical to note that the planning is done offline. The two trajectories generated by the planner are shown in Fig. 2. The one in green is planned for no obstacles and the one in red is planned for static obstacles. The static obstacles are avoided by incorporating an additional $l_2$ norm in the cost due to obstacles.

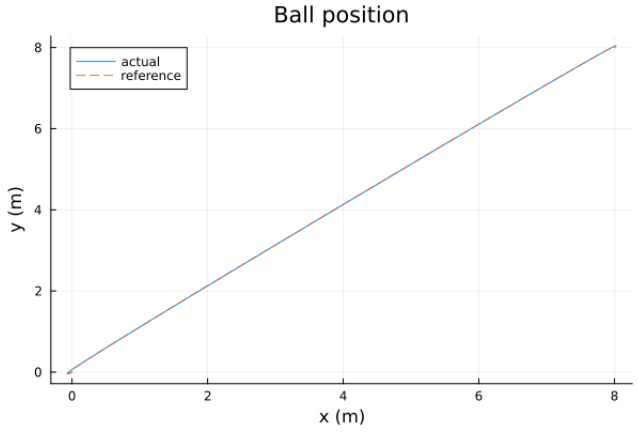


Fig. 3. Performance of ballbot without arms on the straight line reference path.

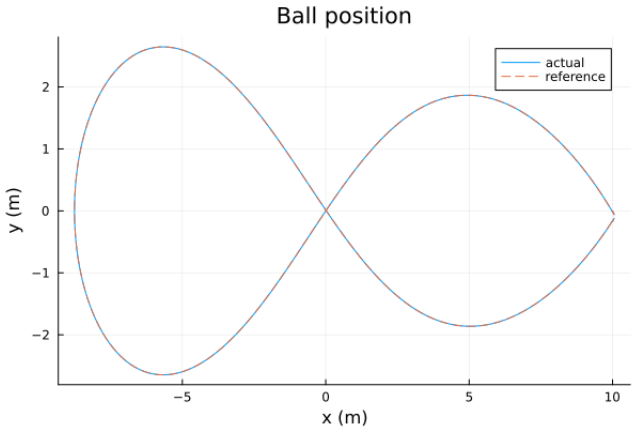


Fig. 4. Performance of ballbot without arms on the curve reference path and obstacles.

The performance of the ballbot with arms during navigation is presented in Figs. 5, 6, 7, and 8. To assess the robustness of the controller, we added noise ($\sim \mathcal{N}(0, 0.01)$) to the ball position and introduced model mismatches (e.g., a 10% increase in body mass). As shown in Fig. 8, the tracking error is higher when the curve is sharper in the presence of model mismatches. However, the controller is able to follow the desired path and reduces the error to less than 1 cm by the end of the trajectory.

An intriguing observation is that the optimal solution involves a combination of leaning and forward arm movement to gain momentum at the start of the navigation task, moving arms forwards or backwards during sharper turns, and leaning and moving the arm to come to a halt, as demonstrated by our simulated results.

The optimization problems were solved on a computer with Intel i7-10700 CPU and the implementation details are found on Table I.

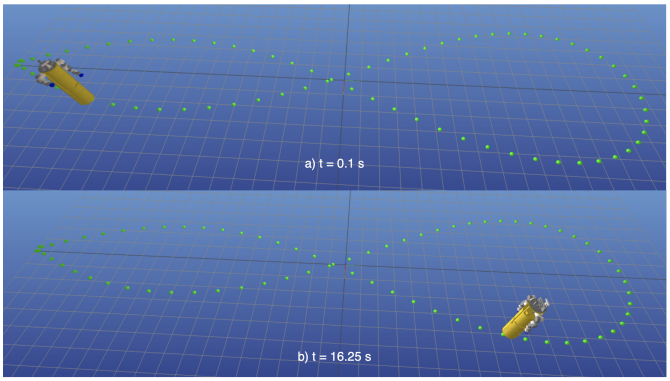


Fig. 5. Ballbot, with arms, is navigating the curved trajectory.

To have a quantitative comparison of the two systems, we also present the Table (I).

## IV. Pushing the Wall

In this task, we design an optimization problem for the robot to go towards a wall, and push the wall to generate

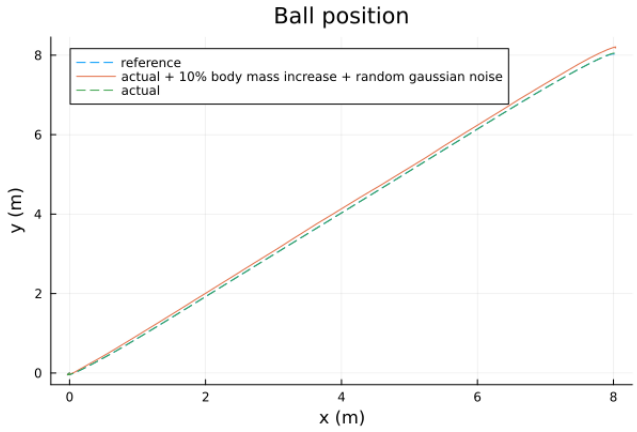


Fig. 6. Performance of ballbot with arms on the straight line reference path with gaussian noise in the ball pose and an increase of 10% in body mass.

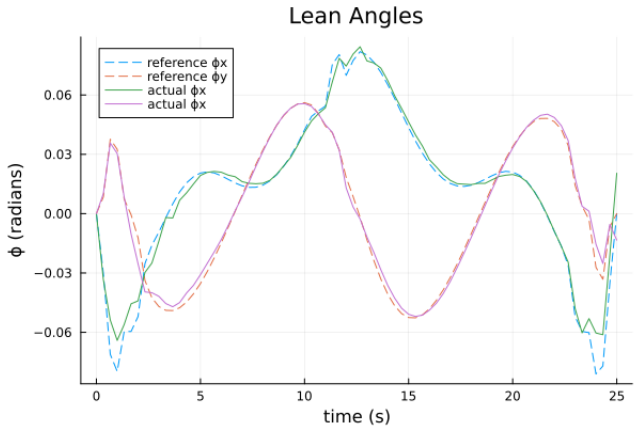


Fig. 7. Performance of the controller tracking the desired lean angles for the ballbot with arms.

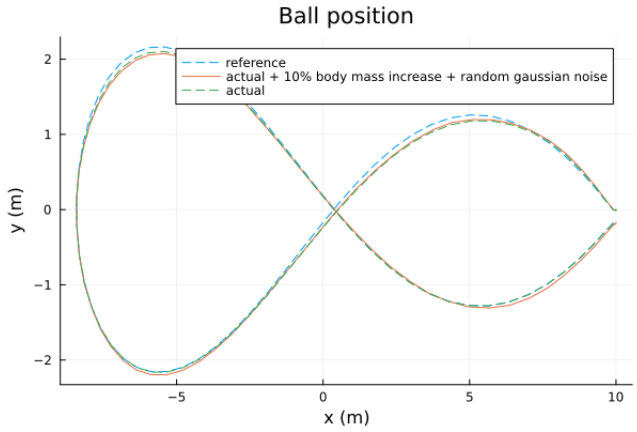


Fig. 8. Performance of ballbot with arms on the curved reference path.

TABLE I
PERFORMANCE COMPARISON OF THE TWO SYSTEMS.

| | **Ballbot without arms** | | **Ballbot with arms** | |
|---|---|---|---|---|
| Trajectory shape | Straight | Curved | Straight | Curved |
| Knot points | 81 | 251 | 41 | 126 |
| TrajOpt computation time (s) | 60 | 176 | 714 | 1801 |
| TrajOpt time step (s) | 0.1 | 0.1 | 0.2 | 0.2 |
| Tracking mean error, actual (cm) | 1.2 | 0.013 | 1.4 | 3.3 |
| Tracking mean error, actual+noise (cm) | | | 1.5 | 3.5 |

momentum and return back to its original position. Compared with the navigation task, the wall-pushing task involves contact dynamics and more focus on the movement of the arms to push off the wall and balance.

We fixed some of the arm joints to simply the problem. The ballbot is equipped with two Barrett WAM arms, each with seven joints, most of which are non-essential and inflate the dimension of the problem. Therefore, we mimicked the human behavior when pushing off the wall, and fixed all the joints except the shoulder and elbow joints. Combined with the x-y translation and 3D rotation on the bottom of the ballbot, there are only 9 elements in the robot state $q_{arms}$ as in Equation 6, making it a more feasible problem to solve offline in minutes.

The dynamics of the robot is given by the Julia RigidBodyDynamics package, given the states and control efforts for each joint.

### A. Hybrid Trajectory Optimization

We can form the following optimization problem for the wall-pushing task:

$$
\begin{aligned}
\min_{x_{1:N}, u_{1:N-1}} \quad & \sum_{i=1}^{N-1} \Big[ \frac{1}{2}(x_i - x_{ref,i})^T Q (x_i - x_{ref,i}) \\
& \quad + \frac{1}{2}(u_i - u_{ref,i})^T R u_i \Big] \\
& + \frac{1}{2}(x_N - x_{ref,N})^T Q_f (x_N - x_{ref,N}) \\
\text{st} \quad & x_1 = x_{\text{IC}} \\
& x_{N/2} = x_{\text{GP}} \\
& x_N = x_{\text{G}} \\
& x_{i+1} = g(f(x_i, u_i)) \quad \text{for } i = 1, 2, \ldots, N-1 \\
& x_{min} \le x_i \le x_{max} \quad \text{for } i = 1, 2, \ldots, N \\
& u_{min} \le u_i \le u_{max} \quad \text{for } i = 1, 2, \ldots, N-1
\end{aligned}
\tag{18}
$$

Compared with Equation 4 for navigation, there are two major differences. First, we constrain the state at the midpoint $x_{N/2}$ to be a specific wall-pushing pose, $x_{\text{GP}}$, shown in Fig. 9 picture (c). Second, we incorporate a contact map, $g(X)$, in the dynamics constraint, which is inactive except at the midpoint to simulate an elastic contact.

We warm start the optimization with two linear interpolations. The first half of the trajectory is interpolated from $x_{\text{IC}}$ to $x_{\text{GP}}$, and the second half is interpolated from $x_{\text{GP}}$ to $x_{\text{IC}}$.

### B. Results

The optimization problem is solved within 4 minutes 11 seconds on a computer with Intel i9-12900H CPU.

Fig. 9 visualizes the optimization result with a 5 m/s constraint on the bottom ball velocity. The ballbot runs towards the wall, swings up the arms, makes contact to push away from the wall, and navigates back to the original position.

Fig. 10 shows a different optimization result where the bottom ball velocity is constrained to 1 m/s. Since the ballbot can no longer rely solely on the ball movement to reach the wall in time, it swings up its arm more aggressively to gain momentum, and push off the wall from the same position as in the first optimization.

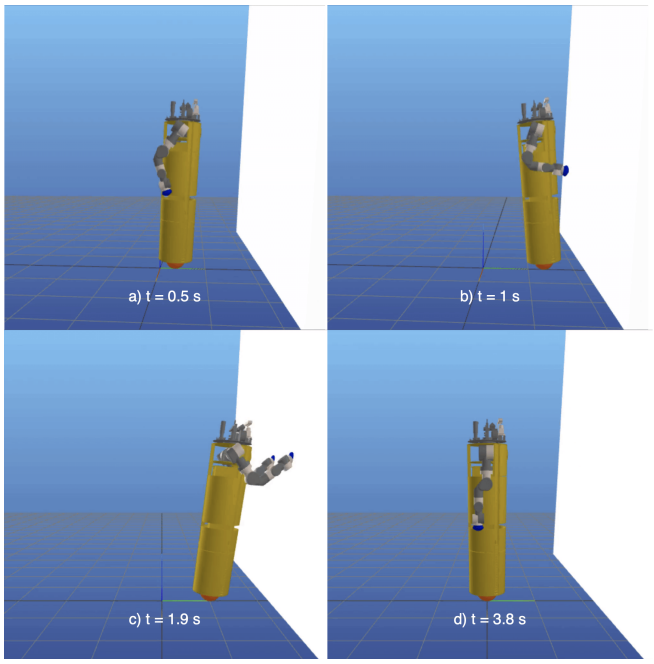


Fig. 9. Hybrid trajectory optimization result.

## V. CONCLUSIONS AND NEXT STEPS

The movement of the arms can influence the stability and balance of the ballbot. By actively controlling the movements of the arms in coordination with the ball's motion, the robot can adjust its balance and stability in response to various environmental conditions or tasks. This capability allows the ballbot to adapt to different situations and improve its overall performance. Our current formulation can generate trajectories offline and track them online. However, CMU ballbot with 7-DOF arms cannot interact with dynamically changing environments yet. The proposed next steps involve transferring the TVLQR navigation control approach to hardware using Iterative Learning Control to bridge the sim-to-real-gap. Initially, a simple trajectory will be employed to test and evaluate the effectiveness of the control method. For the wall pushing task, a reduced order model based on centroidal momentum dynamics [1] [8] and the whole-body kinematics of the robot will be utilized. This modeling approach accounts for the

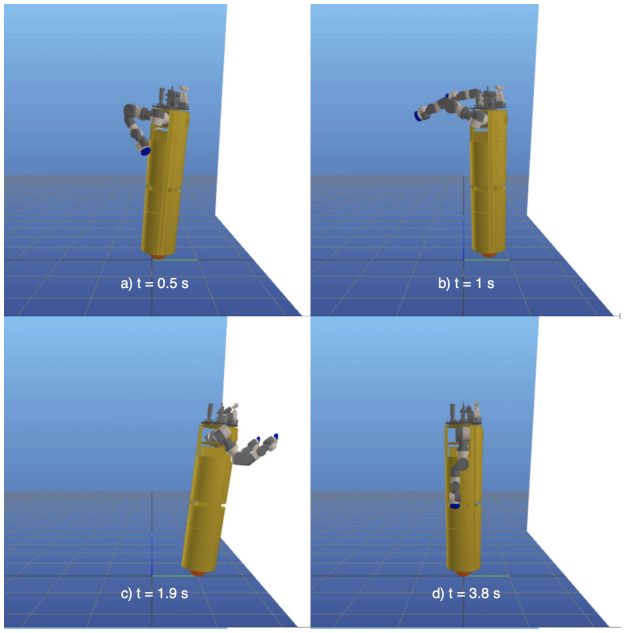


Fig. 10. Hybrid trajectory optimization result with ball speed constrained to 1 m/s.

balancing dynamics of the robot and enables a faster and more accurate control of its motion. Furthermore, friction cone constraints and contact force planning will be introduced to enhance the stability of the robot during agile maneuvers. The idea is to use a simplified dynamics model that can be used to generate trajectories online and dynamically interact with the environment.

### A. *Current progress: Centroidal Momemntum Dynamics*

Currently, we are testing an NLP based on the formulation in Equation 19 using centroidal momentum dynamics. The robot is able to navigate to the wall and touch the wall with its hands. However, the arms seem unable to generate output forces to push the robot back.

$$
\begin{aligned}
\min_{x_{1:N},u_{1:N-1}} \quad & \sum_{i=1}^{N-1} \Big[ \frac{1}{2}(P_{S,i} - P_{S\_ref,i})^T Q_{ps} (P_{S,i} - P_{S\_ref,i}) \\
& + \frac{1}{2}(v_{S,i} - v_{S\_ref,i})^T Q_{vs} \\
& (v_{S,i} - v_{S\_ref,i}) \Big] \\
& + \frac{1}{2}(P_{S,N} - P_{S\_ref,N})^T Q_{ps} (P_{S,N} - P_{S\_ref,N}) \\
& + \frac{1}{2}(v_{S,N} - v_{S\_ref,N})^T Q_{fvs} (v_{S,N} - v_{S\_ref,N}) \\
& + \frac{1}{2}(h_N)^T Q_{fh} (h_N) + \frac{1}{2}(\dot{h}_N)^T Q_{fh} (\dot{h}_N) \\
\text{st} \quad & x_1 = x_{\text{IC}}
\end{aligned}
\tag{19}
$$

**stage constraints:**

$$
\begin{aligned}
& h = x_{\text{GP}} \\
& r = com(q) \\
& x_N = x_{\text{G}} \\
& h_i = A(q_i) * \dot{q} \quad \text{for } i = 1, 2, \ldots, N-1 \\
& \dot{h}_i = \sum_j (c_{j,i} - r_i) \times F_{j,i} + \tau_{j,i} \\
& x_{min} \le x_i \le x_{max} \quad \text{for } i = 1, 2, \ldots, N \\
& x = [q, \dot{q}, \ddot{q}, h, \dot{h}, r, \dot{r}, \ddot{r}, F_{b,L,R}, \tau_{b,L,R}]^T \\
& P_S = q[1:2] \\
& v_S = \dot{q}[1:2] \\
& e^T * d_i <= \mu * {F^n}_{L,R,i} \\
& \quad d_i >= 0 \\
& \quad {f^t}_{L,R,i} = [I, -I] * d_i \\
& \qquad \text{for} \quad i = N_{contact}, \ldots, N_{push} \\
& \qquad \textbf{where } e = [1,1,1,1]^T
\end{aligned}
\tag{20}
$$

**collocation constraints:**

$$
\begin{aligned}
& q_i - q_{i-1} = \dot{q}_i * dt_i \\
& \dot{q}_i - \dot{q}_{i-1} = \ddot{q}_i * dt_i \\
& h_i - h_{i-1} = \dot{h}_i * dt_i \\
& \textbf{COM constraints:} \\
& r_i - r_{i-1} = \frac{(\dot{r})_i + (\dot{r})_{i-1}}{2} * dt_i \\
& \dot{r}_i - \dot{r}_{i-1} = \ddot{r}_i * dt_i
\end{aligned}
\tag{21}
$$

In the above formulation in Equations 19 and 20, $x$ is the state space of this task. $A$ is the centroidal momentum matrix, $h$ and its derivatives are momentum of the robot at a given time step, $r$ and its derivatives are the centre of mass of the robot, $F_{b,L,R}$ and $\tau_{b,L,R}$ are the force and torque at contact points respectively, (i.e. at the ball, left end-effector, and right end-effector of the arm). We used direct collocation with the forward Euler method to integrate the $r$, $q$, and $h$ variables as in Equation 21 for numerical stability, where $dt_k$ is the time step size. ${F^n}_{L,R}$ are the normal forces acting on the end-effector, and ${F^t}_{L,R}$ are the tangential components. The $com()$ functions calculate the centre of mass based on the robot's state $q$, which is using the reduced form as explained in Section IV. Similar to Section IV, we are tracking a reference containing only the the ball's position and velocity.


## ACKNOWLEDGMENT

We would like to express our sincere gratitude to our project supervisor Dr. Zac Manchester and TA Kevin Tracy, for their invaluable guidance, expertise, and continuous support throughout the duration of this project. We would also like to thank CMU Ballbot leading team Dr. Ralph Hollis, Dr. Nancy Pollard, Dr. Oliver Kroemer and ballbot expert Dr.Roberto Shu for their insights and constructive feedback.

## REFERENCES


[1] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302, 2014. doi: 10.1109/HUMANOIDS.2014.7041375.

[2] Thomas Kølbæk Jespersen. Kugle-modelling and control of a ball-balancing robot. *Master Thesis, Aalborg University, Aalborg, Denmark*, 2019.

[3] Twan Koolen and contributors. Rigidbodydynamics.jl, 2016. URL https://github.com/JuliaRobotics/RigidBodyDynamics.jl.

[4] Masaaki Kumagai and Takaya Ochiai. Development of a robot balancing on a ball. In *2008 International Conference on Control, Automation and Systems*, pages 433–438. IEEE, 2008.

[5] Maria Vittoria Minniti, Farbod Farshidian, Ruben Grandia, and Marco Hutter. Whole-body mpc for a dynamically stable mobile manipulator. *IEEE Robotics and Automation Letters*, 4(4):3687–3694, 2019.

[6] Roberto Shu. An agile and dexterous balancing mobile manipulator robot. *PhD thesis*, 2022.

[7] Roberto Shu and Ralph Hollis. Development of a humanoid dual arm system for a single spherical wheeled balancing mobile robot. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 499–504. IEEE, 2019.

[8] Roberto Shu and Ralph Hollis. Momentum based whole-body optimal planning for a single-spherical-wheeled balancing mobile manipulator. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3221–3226, 2021. doi: 10.1109/IROS51168.2021.9636752.