Momentum based Whole-Body Optimal Planning for a Single-Spherical-Wheeled Balancing Mobile Manipulator

Roberto Shu* and Ralph Hollis*

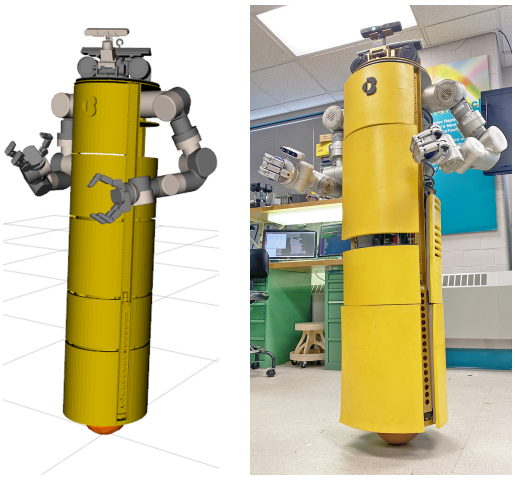
***Abstract*— In this paper, we present a planning and control framework for dynamic, whole-body motions for dynamically stable shape-accelerating mobile manipulators. This class of robots are inherently unstable and require careful coordination between the upper and lower body to maintain balance while performing arm motion tasks. Solutions to this problem either use a complex, full-body nonlinear dynamic model of the robot or a highly simplified model of the robot. Here we explore the use of centroidal dynamics which has recently become a popular approach for designing balancing controllers for humanoid robots. We describe a framework where we first solve a trajectory optimization problem offline. We define balancing for a ballbot in terms of the centroidal momentum instead of other approaches like ZMP or angular velocity that are more commonly used. The generated motion is tracked using a PD-PID cascading balancing controller for the body and torque controller for the arms. We demonstrate that this framework is capable of generating dynamic motion plans and control inputs with examples on the CMU ballbot, a single-spherical-wheeled balancing mobile manipulator.**

I. Introduction

In recent years dynamically-stable (DS) mobile manipulators have become very attractive alternatives to their statically-stable (SS) counterparts that usually consist of large, heavy, bases that may or may not be omni-directional. Typical DS mobile manipulator robots are in the form factor of a bipedal humanoid robot or of a manipulator mounted on top of a two-wheel base (e.g., Segway) that balances along a single axis. Wheeled based manipulators have become a compelling alternative to bipedal robots due to the potential for increased efficiency on flat ground. Though balancing strategies have been explored for wheeled DS robots, only a few consider manipulation. This paper aims to provide a solution to this issue with a whole-body optimization based planning and control framework.

The CMU ballbot mobile manipulator, shown in Fig. 1, is a unique type of DS robot that balances and locomotes on a single spherical wheel. It has a pair of 7-DOF arm mounted on to the body. Ballbot type robots belong to the family of *shape-accelerated underactuated balancing systems* [1]. Shape-accelerated balancing systems are a special class of underactuated systems wherein their shape configurations, *i.e.*, body and arms' configuration, can be mapped to the accelerations of the position variables, *i.e.*, ball position. Changes in the shape configuration will result in an acceleration in the position variables.

Dynamically stable robots, like the ballbot, are interesting systems to investigate from a control perspective due to their instability when balancing on a very small area of support. The control problem becomes more challenging if such a robot is equipped with a multi-DOF manipulator that can move objects and interact with the environment. For example, when trying to perform a dexterous manipulation task where the end-effector position relative to an inertial frame is important. As the manipulator moves toward the target position, the body will accelerate, and the position of the base will move too. Thus, to realize complex whole-body tasks while maintaining balance careful coordination between the upper and lower body is required.



(a) (b)

Fig. 1: The human-size CMU ballbot in (a) dynamic simulation environment and (b) dynamically balancing in reality.

Dynamic based solutions to this problem mostly use simplified models such as the Linear Inverted Pendulum (LIP) model that fail to exploit the whole body dynamic capabilities of the platform in use [2]–[4]. This model assumes that angular momentum is constant, which is not valid for motions requiring fast arm swinging. On the other hand, dynamic motion planning can be done using full body nonlinear dynamics of the system [5], [6]. These methods can produce smooth trajectories but due to the complexity of the full-body nonlinear dynamics, these optimizations can take an excessively long time to run. In this work we represent the dynamics constraint of the ballbot by its *centroidal dynamics*, since postural balance can be defined in terms of centroidal momentum [7]. This method is a balance between the two extremes and has become a popular approach to plan and control legged robots [8]–[11]. However, it has seldom been explored for dynamically stable wheeled robots (e.g., Segway

* The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {rshu,rhollis}@cs.cmu.edu

or ballbot bases) that present different challenges compared to legged robots.

II. Related Work

While many ballbots have been built in recent years [12]–[14] and different balancing control strategies have been studied, only a few consider manipulation capabilities. In [15], a ballbot with a single robot arm is controlled by regulating the position of the center of mass (COM) to always be on top of the point of support. The robot arm is used to regulate the COM position and not to perform a manipulation task. To mitigate this they propose adding a stabilizer mass to compensate for the effects of the manipulator. In [16] a MPC approach is presented that considers the full non-linear dynamics of the system and treats balancing and manipulation in a unified planner. In each MPC iteration, they linearize the dynamics of the system which limits the operating range.

Our research group has previously demonstrated several dynamic physical human-robot interaction capabilities with our person-size ballbot equipped with a pair of 2-DOF arms [3], [17]–[19]. Recently, the 2-DOF arms in the CMU ballbot were upgraded with a pair of far more capable 7-DOF arms and multi-DOF hands [20]. Our proposed strategies in the past considered the balancing and manipulation tasks separately. The dynamics effects of the arm motion were compensated for by a robust balancing controller with COM compensation [18]. We modeled the ballbot with arms as a decoupled, planar, wheeled inverted pendulum, with a massless arm with weights at its end. This was a valid simplification with the previous 2-DOF arms. However, for the 7-DOF arms this approach limits the range of possible tasks.

A. Momentum based controllers

Until recently, most balance control methods have attempted to maintain balance by controlling only the linear motion of the robot. An interesting departure from this are *momentum-based-balance controllers* [21]. These approaches control both the linear and angular components of the spatial momentum to perform whole-body motions.

In [21] a whole-body momentum based controller for humanoid robots on non-level ground is presented. The controller regulates the linear and angular momentum by solving an optimization problem to find whole body motion. It gives higher priority to linear momentum over angular momentum. We also give a higher priority to linear momentum when both cannot be simultaneously attained.

In [22] a WBC for a torque controlled humanoid robot balancing on top of a two-wheel balancing platform is presented. The controller is formulated as a quadratic optimization problem (QP) to generate joint torques that satisfy the whole-body dynamics constraint. The quadratic cost function minimizes the error between the desired and actual linear and angular momentum for the system. However, the issue of how to set the desired angular momentum for more complex motions such as performing locomotion and manipulation simultaneously was not fully explored.

This issue can be tackled using offline trajectory optimization to generate complex whole-body motions. This paper takes inspiration from [8], [23] which combine the simple centroidal dynamics with a full kinematic model to generate robots' whole-body motions through offline non-linear trajectory optimization. However, we differentiate in that we simplify the optimization problem by not including the complementary constraint for contact implicit optimization. For the ballbot we assume a single rolling point of contact between the ball and the ground, whereas in actuality there is a small patch of the urethane ball outer layer.

III. Dynamic Model

A. System Description

The CMU ballbot is a human-size robot that balances on a ball. It has a pair of 7-DOF torque-controllable arms mounted onto the body. The ball is actuated using a four-motor inverse mouse-ball drive mechanism (IMBD). A pair of actuated opposing rollers drive the ball in each of the two orthogonal motion directions on the floor. Omnidirectional motion is achieved by moving the ball in any direction using this mechanism. The IMBD mechanism is attached to the body using a large thin-section bearing, which allows yaw rotation of the body (*i.e.*, rotation about its vertical axis). Another DC servomotor actuates this yaw degree of freedom. Unlimited yaw rotation of the body is enabled by a slip ring assembly. The model makes the following assumptions: (i) there is no slip between the ball and the floor; and (ii) the ball height relative to the floor remains constant, *i.e.*, the ball is always in contact with the floor.

B. Modelling

The configuration of the robot is described by $n_b + \sum^i n_{a_i}$ DOF, where n_b is the number of DOF of the mobile base and n_{a_i} is the number of DOF of the i^{th} manipulator attached. The motion of the ballbot base is described by $\mathbf{q}_b = [\mathbf{p}_S, \boldsymbol{\phi}]^T \in \mathbb{R}^{n_b}$, $\mathbf{p}_S = [p_x, p_y]^T \in \mathbb{R}^2$ is the 2D position of the ball in the horizontal plane with respect to the inertial frame $\{I\}$, $\boldsymbol{\phi} = [\phi_x, \phi_y, \phi_z]^T \in \mathbb{R}^3$ is a vector of Euler angles representing the body orientation. The 7-DOF of the left and right arms are represented by $\alpha_L \in \mathbb{R}^{n_a}$ and $\alpha_R \in \mathbb{R}^{n_a}$. These notations are shown in Fig. 2. For the CMU ballbot we define the set of generalized coordinates

$$\mathbf{q} = \begin{bmatrix} \mathbf{p}_S \\ \boldsymbol{\phi} \\ \alpha_L \\ \alpha_R \end{bmatrix} \text{ and } \dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{p}}_S \\ \dot{\boldsymbol{\phi}} \\ \dot{\alpha}_L \\ \dot{\alpha}_R \end{bmatrix}. \tag{1}$$

The system has in total $n = 19$ DOFs ($n_b = 5, n_a = 7$). Grasp-planning is outside the scope of this work and does not consider the DOFs of the hand fingers. Instead, we only consider the left and right arm end-effector pose $\mathbf{p}_{EE,L} \in SE(3)$ and $\mathbf{p}_{EE,R} \in SE(3)$, respectively. We also define $\mathbf{r} \in \mathbb{R}^3$ the robot's COM position with respect to $\{I\}$.

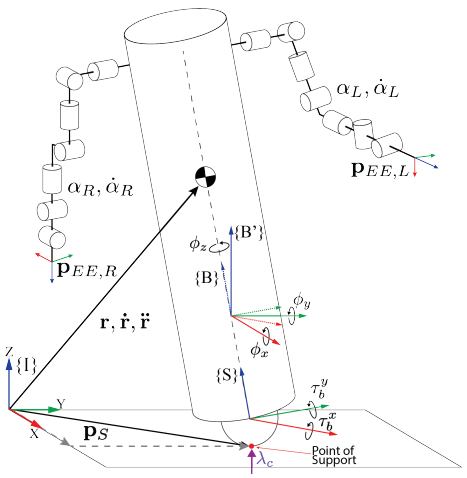


Fig. 2: Schematic of the CMU ballbot generalized coordinate representation. $\{I\}, \{S\}, \{B\}$ are the inertial frame, the ball (sphere) frame and the body frame. Frame $\{B'\}$ is coincident with $\{B\}$ but aligned with $\{I\}$. $\mathbf{p}_{EE,R}$ and $\mathbf{p}_{EE,L}$ are right and left end-effector pose with respect to $\{I\}$.

Considering the generalized coordinates defined in (1) the Euler-Lagrange Equation of Motion are as follow:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{S}\boldsymbol{\tau} + \boldsymbol{J}_c^T \boldsymbol{\lambda}_c \tag{2}$$

where, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ is the matrix composed of the sum of Coriolis and centrifugal forces, $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^n$ is the gravity force vector, $\boldsymbol{\lambda}_c$ is the ground reaction force, $\boldsymbol{J}_c$ is the corresponding Jacobian. $\boldsymbol{\tau} = [\mathbf{f}_s, \tau_{\phi_z}, \boldsymbol{\tau}_{a,L}^T, \boldsymbol{\tau}_{a,R}^T]^T \in \mathbb{R}^{n_\tau}$ is the vector of generalized forces and torque inputs. The torque exerted by the IMBD at the center of the ball is related to a linear force at the point of contact between the ball and the ground by

$$\mathbf{f}_s = \begin{bmatrix} f_s^x \\ f_s^y \end{bmatrix} = \begin{bmatrix} r_b \cdot \tau_b^x \\ r_b \cdot \tau_b^y \end{bmatrix} \tag{3}$$

where r_b is the radius of the ball and the τ_b^x, τ_b^y are the torques exerted by the IMBD in the X and Y axis respectively. τ_{ϕ_z} is the torque input to control the body yaw. The joint torque of the manipulators are described by the vectors $\boldsymbol{\tau}_{a,L} \in \mathbb{R}^{n_a}$ and $\boldsymbol{\tau}_{a,R} \in \mathbb{R}^{n_a}$. The system has a total of $n_\tau = 17$ control inputs. The actuation selection $\mathbf{S} \in \mathbb{R}^{n\times(n_\tau)}$ separates the $n_c = n - n_f$ controlled joints from the $n_f = 2$ unactuated body lean angle joints ϕ_x and ϕ_y. Since the number of degrees of freedom of the robot is larger than the number of independent control inputs, the system is underactuated.

C. Centroidal Dynamics

In this section, we provide a brief recap of centroidal dynamics, the dynamics of a robot system at its COM [7]. The centroidal momentum vector $\mathbf{h} \in \mathbb{R}^{6\times1}$ composed of the linear momentum $\mathbf{l} \in \mathbb{R}^{3\times1}$ and angular momentum $\mathbf{k} \in \mathbb{R}^{3\times1}$ is linearly related to the generalized joint velocities vector $\dot{\mathbf{q}}$ by

$$\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} \mathbf{k} \\ \mathbf{l} \end{bmatrix} = \mathbf{A}(\mathbf{q})\dot{\mathbf{q}} \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{6\times n}$ is the centroidal momentum matrix (CMM). Taking the time derivative of (4) results in the second order *centroidal dynamics*

$$\dot{\mathbf{h}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{A}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}. \tag{5}$$

The rate of centroidal linear and angular momentum $\dot{\mathbf{h}}(\mathbf{q}, \dot{\mathbf{q}}) = [\dot{\mathbf{k}}, \dot{\mathbf{l}}]^T$, computed from the robot's joint angles and velocities, equals the total wrench generated by the external contacts and the gravitational forces:

$$\dot{\mathbf{k}} = \mathbf{m}\ddot{\mathbf{r}} = \sum_j \mathbf{F}_j + \mathbf{mg} \tag{6}$$

$$\dot{\mathbf{l}}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_j (\mathbf{c}_j - \mathbf{r}) \times \mathbf{F}_j + \tau_j \tag{7}$$

where $\mathbf{m}$ is the total mass of the robot, $g \in \mathbb{R}^3$ is the gravitational acceleration, $\mathbf{c}_j \in \mathbb{R}^3$ is the position of the j^{th} contact point, $\mathbf{F}_j \in \mathbb{R}^3$ and $\tau_j \in \mathbb{R}^3$ are the force and torque at the j^{th} contact point, respectively. Since, CMM has been shown to be useful when generating dynamic motions of multiple limbs to maintain balance [7], we define balancing in terms of the linear and angular momentum. For balancing maintenance it is desired that $\mathbf{k}$ and $\mathbf{l}$ be zero and for $\mathbf{r}$ to be above the point of support. This will form the basis of the optimization problem presented in section IV.

IV. Planning and Control

A. Trajectory Optimization

To compute the feasible motion plan that includes the centroidal momentum trajectory and joint trajectories, we formulate a nonlinear optimization problem (NLP) that uses the centroidal dynamics and a full kinematic model [8]. We implement a direct collocation method [24] to transcribe the continuous-time dynamics in (4) and (5) to their discrete form. We sample all time-varying quantities at N knot points. The nonlinear constraint optimization problem minimizes the cost function

$$\begin{aligned} \min_{\substack{\mathbf{q}[k],\dot{\mathbf{q}}[k],\ddot{\mathbf{q}}[k],\\ \mathbf{r}[k],\dot{\mathbf{r}}[k],\ddot{\mathbf{r}}[k],\,\mathbf{F_j}[k],\tau_\mathbf{j}[k]\\ \mathbf{h}[k],\dot{\mathbf{h}}[k]}} \sum_{k=0}^{N} \Big(& \left|\mathbf{p}_{EE,i}[k] - \mathbf{p}_{EE,i}^d[k]\right|^2_{\mathbf{Q}_{EE,i}} \\ & + \left|\mathbf{e}_{oEE,i}[k]\right|^2_{\mathbf{Q}_{oEE,i}} \\ & + \left|\mathbf{p}_S[k] - \mathbf{p}_S^d[k]\right|^2_{\mathbf{Q}_S} \\ & + \left|\mathbf{p}_S[k] - \mathbf{p}_{com}[k]\right|^2_{\mathbf{Q}_{com}} + \left|\dot{\mathbf{h}}[k]\right|^2_{\mathbf{Q}_{\dot{\mathbf{h}}}} + \left|\ddot{\mathbf{q}}[k]\right|^2_{\mathbf{Q}_{\ddot{\mathbf{q}}}} \Big) \end{aligned} \tag{8}$$

where $|x|^2_{\mathbf{Q}}$ is the abbreviation for the quadratic cost $x^T Q x$. The square bracket $[k]$ means the sampled value at the k^{th} knot point. The cost function (8) tries to minimize the sum of different tasks that the robot is required to perform. The semidefinite positive matrices $Q_{EE,i}$ and $Q_{oEE,i}$ weight the task of tracking a desired end-effector position and orientation for $i = L, R$ corresponding to the left and right end-effectors. The orientation error is computed using quaternion difference. The task of tracking a desired base position is weighted by the matrix Q_S. The balancing task is

defined by the error tracking term between the COM position to be on top of the point of support and the term penalizing the centroidal momentum rate. The momentum penalizing term is weighted higher by $Q_{\dot{\mathbf{h}}}$ than the COM tracking term by Q_{com}. The regulation term penalizing joint acceleration in the cost function is included to provide numerical stability and is weighted by $Q_{\ddot{\mathbf{q}}}$.

The optimization constraints include the centroidal dynamics defined in (4) and (5) discretized at each knot point

$$m\ddot{\mathbf{r}}[k] = \sum_i \mathbf{F}_i[k] + m\mathbf{g} \tag{9}$$

$$\dot{\mathbf{h}}[k] = \sum_i (\mathbf{c}_i[k] - \mathbf{r}[k]) \times \mathbf{F}_i[k] + \tau_i[k] \tag{10}$$

$$\mathbf{h}[k] = \mathbf{A}(\mathbf{q}[k])\dot{\mathbf{q}}[k]. \tag{11}$$

Note that in our case there is only one contact point $\mathbf{c}_i$ between the ball and the ground. To enforce continuity between the discrete system state knot points $\mathbf{q}[k], \dot{\mathbf{q}}[k], \ddot{\mathbf{q}}[k], \mathbf{h}[k]$, and $\dot{\mathbf{h}}[k]$ we formulate equality constraints such that the change in state between two knot points is equal to the integral of the system dynamics. We approximate the integral using Euler integration. For numerical stability, this is implemented using backward-Euler integration. The collocation constraints are

$$\mathbf{q}[k] - \mathbf{q}[k-1] = \dot{\mathbf{q}}[k]dt \tag{12}$$

$$\dot{\mathbf{q}}[k] - \dot{\mathbf{q}}[k-1] = \ddot{\mathbf{q}}[k]dt \tag{13}$$

$$\mathbf{h}[k] - \mathbf{h}[k-1] = \dot{\mathbf{h}}[k]dt \tag{14}$$

where dt is the sample time between knot points. We approximate the COM position using a piecewise quadratic polynomial. Its time integration constraints are

$$\mathbf{r}[k] - \mathbf{r}[k-1] = \frac{\dot{\mathbf{r}}[k] + \dot{\mathbf{r}}[k-1]}{2}dt \tag{15}$$

$$\dot{\mathbf{r}}[k] - \dot{\mathbf{r}}[k-1] = \ddot{\mathbf{r}}[k]dt \tag{16}$$

To ensure the full kinematics of the robot are obeyed we implement the kinematic constraint that relates the robot joint configuration and COM position

$$\mathbf{r}[k] = \text{com}\left(\mathbf{q}[k]\right) \tag{17}$$

where $\text{com}(\mathbf{q})$ computes the corresponding COM position to a given robot joint configuration $\mathbf{q}$. Joint position, velocity, and acceleration limits are also enforced through the inequality constraints

$$\mathbf{q}_{lb} \leq \mathbf{q}[k] \leq \mathbf{q}_{ub} \tag{18}$$

$$\dot{\mathbf{q}}_{lb} \leq \dot{\mathbf{q}}[k] \leq \dot{\mathbf{q}}_{ub} \tag{19}$$

$$\ddot{\mathbf{q}}_{lb} \leq \ddot{\mathbf{q}}[k] \leq \ddot{\mathbf{q}}_{ub}. \tag{20}$$

Equality boundary constraints are also included to enforce initial and final robot states. We set the initial constraints for the joint states

$$\mathbf{q}[0] = \mathbf{q}_0, \quad \dot{\mathbf{q}}[0] = \dot{\mathbf{q}}_0, \quad \text{and} \quad \ddot{\mathbf{q}}[0] = \ddot{\mathbf{q}}_0. \tag{21}$$

The final value of the centroidal momentum and centroidal momentum rate are constrained to be zero, so to ensure the robot is balancing at the end of the motion by

$$\mathbf{h}[N] = 0 \quad \text{and} \quad \dot{\mathbf{h}}[N] = 0. \tag{22}$$

B. Control

The robot's body motion plan, *i.e.,* $\hat{q}_b(t), \hat{\dot{q}}_b(t)$, generated by the trajectory optimization is tracked by a PD-PID cascading balancing controller. The inner PID loop maintains the ballbot balancing upright. It does so by tracking a desired lean angle by actuating the ball. The outer loop tracks the ball position on the floor and feeds lean-angle setpoints to the inner loop controller. By tracking the body lean angle we can indirectly track the ball position in the ground. The arms motion plan, *i.e.,* $\hat{\alpha}_i(t), \hat{\dot{\alpha}}_i(t)$, are tracked by a decentralized torque-impedance based feedback controller with feedforward gravity and torque sensing compensation, as shown in Eq. 23.

$$\tau_{des} = K_{P_\alpha} e_\alpha + K_{D_\alpha} \dot{e_\alpha} + g(\alpha, \dot{\alpha}) - \tau, \tag{23}$$

where $K_{P_\alpha}, K_{D_\alpha}$ are positive definite diagonal gain matrices, $g(\alpha, \dot{\alpha})$ is the gravity compensation term based on the dynamic model of the arm, $e_\alpha = \alpha_{des} - \alpha$ and $\dot{e}_\alpha = \dot{\alpha}_{des} - \dot{\alpha}$ are the joint position and velocity errors.

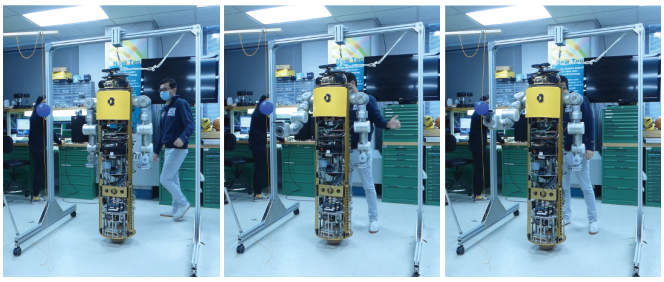
V. Results

The proposed whole body planning and control framework was tested to perform different tasks in a dynamic simulation environment of the CMU ballbot and in the real hardware. Experiment results are shown in the supplemental video[1]. The optimization problem described is implemented using *CasADi* [25] and Ipopt [26]. The dynamics equations were efficiently obtained using Pinocchio [27]. For all the tasks performed we first solve the trajectory optimization problem offline with a long time horizon $N \geq 40$ and $dt = 0.1$ s to obtain a reference trajectory to follow. The controller to track the reference plan runs at 500 Hz.

A. Hardware: Single End-Effector Position Tracking

In the hardware we tested reaching a desired position for the right end-effector that is outside the arm's reach without moving the base, as shown in Fig. 3. For this task we penalize the end-effector position error $\mathbf{Q}_{EE,r} = \text{diag}([100, 100, 100])$. All other weights matrices $\mathbf{Q}_i$ are set to the identity. The motion plan was generated offline in 22 seconds. Without explicitly defining a task for the base motion, the framework synthesizes a combined motion for the arm and base such that while maintaining balance the robot reaches its end-effector target position marked by the blue ball. The ball is only used for visual reference, it is not tracked online.

[1] https://youtu.be/HC_Qk1x228Y

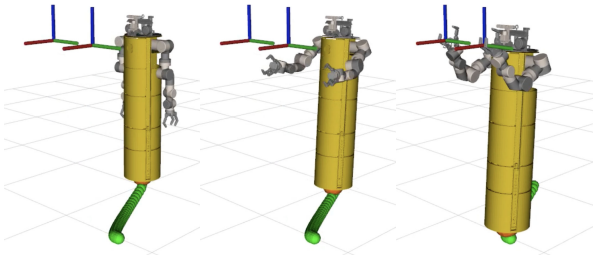


(a) t = 0 sec (b) t = 2 sec (c) t = 5 sec

Fig. 3: Screenshots of tracking a desired end-effector pose (blue ball for reference only) with high weight cost on left arm joint acceleration.

B. Simulation: Double End-Effector Position Tracking

In this experiment we control the desired position and orientation for both end-effectors as shown in Fig. 4. We set high weights on the position $Q_{EE,L} = Q_{EE,R} = \text{diag}\big([100, 100, 1000]\big)$ and orientation $Q_{oEE,L} = Q_{oEE,R} = \text{diag}\big([50, 50, 50]\big)$ error tracking terms. In Fig. 5 the end-effector position tracking error are shown. The linear and angular momentum trajectories are non-zero initially to realize the desired motion but quickly return to zero to stabilize the robot, as shown in Fig. 6. We set $Q_b = 0$ to give the optimization the freedom to find a motion to coordinate the base and arm motion to track the desired end-effector pose. As desired the controller can successfully track the planned motion that uses the body lean to compensate for the COM movement due to the arm motion. The offline plan was generated in 26 seconds. This behavior is very similar to that of humans when lifting heavy objects.



(a) t=0 sec (b) t=3 sec (c) t=6 sec

Fig. 4: Snapshots of tracking desired position and orientation for both end-effectors. Note the use of the arms instead of the body lean to move towards the target location.

C. Simulation: Base Position Tracking

There may be scenarios where we are not interested in the end-effector pose and are only interested in the base tracking a desired position. This can be accomplished by setting $Q_{EE,i} = 0$ and setting a large value for $Q_b = \text{diag}\big([100, 100]\big)$. We test this by commanding the robot to a position $\mathbf{P}_B^d = [1,\ 1]^T$ m, as shown in Fig. 7. Without setting a high weight to penalize the arm joint accelerations the trajectory optimization found an optimum motion in 21

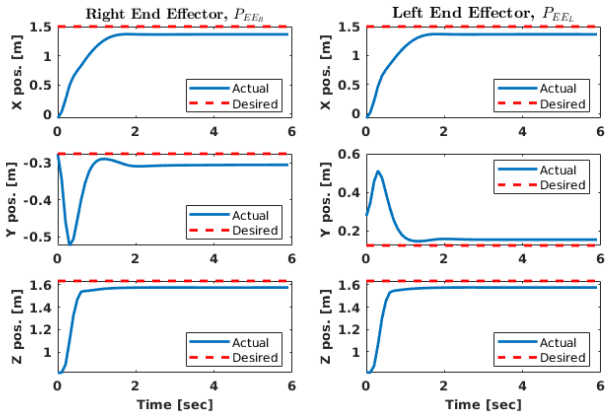


Fig. 5: Right and left end-effector cartesian position with respect to inertial frame.

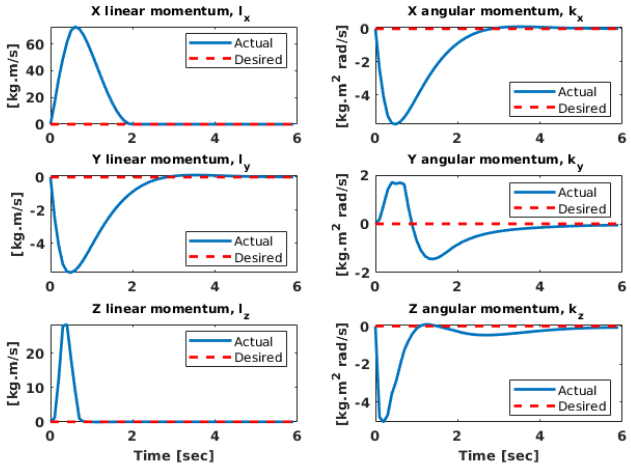


Fig. 6: Linear and angular momentum evolution while tracking a desired position for both end-effectors

seconds that primarily uses lean angle motion as shown in Fig. 8. This is desirable and expected since inducing a small lean angle alone produces enough momentum to realize the motion. Swinging the arms will produce undesirable large momentum changes.

VI. Conclusion and Future Work

A. Conclusion

In this paper, we present a whole-body optimal planning and control framework for dynamically stable mobile robots with multiple arms. In this framework, we first solve a trajectory optimization problem offline. We define balancing for a ballbot in terms of the centroidal momentum instead of other approaches like ZMP or angular velocity that are more commonly used. The employed combination of the centroidal dynamics and a full kinematics model represented the system's dynamics sufficiently well to generate stable motion plans for the ballbot. We tracked the offline generated motion plan using a combination of arm and body controllers for the ballbot. We demonstrate the effectiveness of this algorithm performing different locomotion and arms motion tasks that require simultaneous control of the ball, body, and arms both in simulation and hardware. This framework has

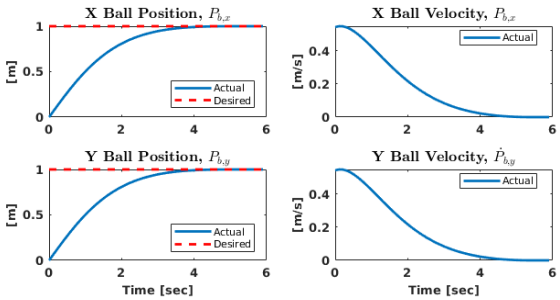
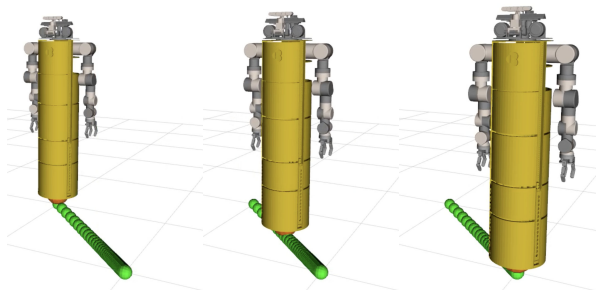


Fig. 7: Ball position and velocity trajectories for the task of tracking a desired ball position



(a) t=0 sec (b) t=3 sec (c) t=6 sec

Fig. 8: Snapshots of tracking desired base position. Despite arm joint acceleration not being penalized they are not used to generate forward momentum.

been shown in a ballbot manipulator, but we believe it can be easily extended to other similar systems such as humanoids with wheeled feet and two-wheeled balancing manipulators.

B. Future Work

The framework has no ability to replan online in the case of large external disturbances or environment changes. Also, fast motions required a significant amount of feedback compensation to track the planned motions in the CMU ballbot hardware. We will look to reformulate the optimization problem into a QP so that it can be solved online. Speeding up the optimization will allow to generate motion plans and control inputs online in an MPC fashion to be able to react fast and handle model and environment uncertainty.

Acknowledgment

This work was supported in part by NSF grant CNS-1629757. The authors thank Cornelia Bauer and Saumya Saxena for helping conducting experiments.

References

[1] U. Nagarajan, "Dynamic constraint-based optimal shape trajectory planner for shape-accelerated underactuated balancing systems," in *Robotics: Science and Systems*, 2010.

[2] M. Stilman, J. Olson, and W. Gloss, "Golem Krang: Dynamically stable humanoid robot for mobile manipulation," in *2010 IEEE International Conference on Robotics and Automation*, pp. 3304–3309, IEEE, 2010.

[3] U. Nagarajan and R. Hollis, "Shape space planner for shape-accelerated balancing mobile robots," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1323–1341, 2013.

[4] S. Xin and S. Vijayakumar, "Online dynamic motion planning and control for wheeled biped robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3892–3899, 2020.

[5] J. Wang, E. C. Whitman, and M. Stilman, "Whole-body trajectory optimization for humanoid falling," in *2012 American Control Conference (ACC)*, pp. 4837–4842, IEEE, 2012.

[6] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[7] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[8] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295–302, IEEE, 2014.

[9] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for whole-body control," *International Journal of Humanoid Robotics*, 2016.

[10] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[11] C. Li, Y. Ding, and H.-W. Park, "Centroidal-momentum-based trajectory generation for legged locomotion," *Mechatronics*, vol. 68, p. 102364, 2020.

[12] P. Fankhauser and C. Gwerder, "Modeling and control of a ballbot," B.S. thesis, Eidgenössische Technische Hochschule Zürich, 2010.

[13] T. K. Jespersen, M. al Ahdab, F. M. Juan de Dios, M. R. Damgaard, K. D. Hansen, R. Pedersen, and T. Bak, "Path-following model predictive control of ballbots," in *2020 IEEE International Conference on Robotics and Automation*, pp. 1498–1504, IEEE, 2020.

[14] T. Hoshino, S. Yokota, and T. Chino, "Omniride: A personal vehicle with 3 dof mobility," in *2013 International Conference on Control, Automation, Robotics and Embedded Systems*, pp. 1–6, IEEE, 2013.

[15] P. Asgari, P. Zarafshan, and S. A. A. Moosavian, "Manipulation control of an armed ballbot with stabilizer," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 229, no. 5, pp. 429–439, 2015.

[16] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body mpc for a dynamically stable mobile manipulator," *2019 IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3687–3694, 2019.

[17] M. Shomin, J. Forlizzi, and R. Hollis, "Sit-to-stand assistance with a balancing mobile robot," in *IEEE Int'l. Conf. on Robotics and Automation*, (Seattle, WA), May 26-30 2015.

[18] F. Sonnleitner, R. Shu, and R. L. Hollis, "The mechanics and control of leaning to lift heavy objects with a dynamically stable mobile robot," *Int'l. Conf. on Robotics and Automation*, May 20-24 2019.

[19] Z. Li and R. Hollis, "Toward a ballbot for physically leading people: A human-centered approach," in *2019 IEEE International Conference on Intelligent Robots and Systems*, pp. 4827–4833, IEEE, 2019.

[20] R. Shu and R. 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)*, pp. 499–504, IEEE, 2019.

[21] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground," *Autonomous Robots*, vol. 33, no. 4, pp. 399–414, 2012.

[22] S. Xin, Y. You, C. Zhou, C. Fang, and N. Tsagarakis, "A torque-controlled humanoid robot riding on a two-wheeled mobile platform," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1435–1442, IEEE, 2017.

[23] S. Dafarra, S. Bertrand, R. J. Griffin, G. Metta, D. Pucci, and J. Pratt, "Non-linear trajectory optimization for large step-ups: Application to the humanoid robot atlas," *arXiv preprint arXiv:2004.12083*, 2020.

[24] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[25] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[26] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[27] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *2019 IEEE International Symposium on System Integrations (SII)*, 2019.