International Journal of Robotics Research (IJRR)
Volume 32, Issue 11, pp. 1323 – 1341, September 2013

Shape Space Planner for Shape-Accelerated Balancing Mobile Robots

Umashankar Nagarajan and Ralph Hollis

***Abstract*—This paper introduces *shape-accelerated balancing systems* as a special class of underactuated systems wherein their shape configurations can be mapped to the accelerations in the position space. These systems are destabilized by gravitational forces and have non-integrable constraints on their dynamics. Balancing mobile robots, like the ballbot, are examples of such systems. The ballbot is a human-sized dynamically stable mobile robot that balances on a single ball. This paper presents a shape trajectory planner that uses dynamic constraint equations to plan trajectories in the shape space, which when tracked will result in approximate tracking of desired position trajectories. The planner can handle systems with more shape variables than position variables, and can also handle cases where a subset of the shape variables is artificially constrained. Experimental results are shown on the ballbot with arms where different desired position space motions are achieved by tracking shape space motions of either body lean angles, or arm angles or combinations of the two; and also by tracking only the body lean motions while the arm angles are artificially constrained.**

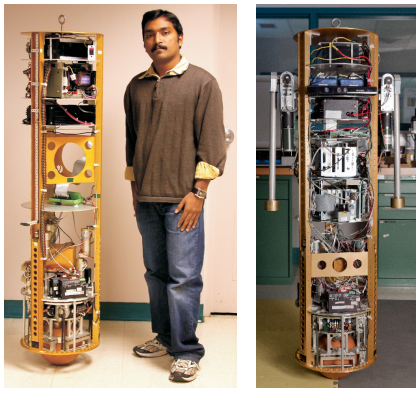
***Index Terms*—Balancing mobile robots, underactuated systems, dynamics and control, shape trajectory planning.**

I. Introduction

Mobile robots will soon be operating and interacting with us in human environments. They will be offering a variety of assistive technologies that will augment our capabilities and enhance the quality of our lives. It is becoming increasingly apparent that, to be most effective, many of these robots will be dynamically stable machines that actively balance just like humans do. Such a balancing robot rapidly and continuously adjusts the relationship between its center of gravity and base of support to enable motion (Hollis 2006). Balancing mobile robots have underlying dynamic properties that can be exploited in order to carry out fast, graceful and efficient motions. Balancing mobile robots can be tall enough to interact with people at eye-level, narrow enough to easily negotiate cluttered environments, and they can move with speed and grace comparable to that of humans. They are also capable of safe, gentle physical interaction.

Two-wheeled balancing mobile robots (Deegan et al. 2006; Stilman et al. 2010; Takahashi et al. 2000) became popular after the introduction of the Segway Robotic Mobility Platform (Nguyen et al. 2004). Grupen and his group introduced *uBot* (Deegan et al. 2006), a two-wheeled balancing mobile manipulation platform, and demonstrated that balancing mobile robots can be effective mobile manipulators with the ability to maintain postural stability, generate forces on external objects and withstand greater impact forces (Deegan et al. 2007). Stilman and his group introduced *Golem Krang* (Stilman et al. 2010), a two-wheeled balancing mobile manipulation platform that has the capability to autonomously stand and sit. These two-wheeled balancing mobile robots balance only in a single vertical plane, and their kinematic constraints do not allow lateral motion.

Our group introduced the *ballbot* (Hollis 2006; Lauwers et al. 2005, 2006), the first successful dynamically stable mobile robot that balances on a single ball, shown in Fig. 1. The ballbot balances in both vertical planes,



(*a*) (*b*)

Fig. 1. (*a*) The ballbot balancing; and (*b*) a more recent version of the ballbot having a pair of 2-DOF arms.

U. Nagarajan is with Disney Research, Pittsburgh, PA 15213, USA. This work was done when he was with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. email: umashankar@disneyresearch.com.

R. Hollis is with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. email: rhollis@cs.cmu.edu.

and hence is omnidirectional. A detailed description of the ballbot's hardware, its control architecture, its dynamic motion and physical interaction capabilities can be found in our previous work (Nagarajan et al. 2009a,b,c). Since the introduction of the ballbot, several other groups have developed single-wheeled balancing mobile robots (Havasi 2005; Kumagai and Ochiai 2008; Rezero 2010). Kumagai developed the *BallIP* (Kumagai and Ochiai 2008), and demonstrated several cooperative transportation tasks with ball balancing robots (Kumagai and Ochiai 2009). A group of mechanical engineering students at ETH Zurich developed the *Rezero* (Rezero 2010), and re-emphasized the dynamic capabilities of ball balancing mobile robots.

Balancing mobile robots like the ballbot are underactuated systems with unstable zero dynamics (Isidori 1989). They have second-order, non-integrable constraints on their dynamics that restrict the family of configuration trajectories that they can follow. The configuration space of any dynamic system can be divided into the *position space* and the *shape space*. The position variables represent the position of the system in the world, whereas the shape variables are those that affect the inertia matrix of the system and dominate the system dynamics. Navigation tasks for mobile robots are generally posed as desired motions in the position space, and do not deal with shape space motions. However, for balancing mobile robots, the strong coupling between the position dynamics and the shape dynamics makes it impossible to ignore shape space motions while tracking desired motions in the position space. Therefore, it is necessary to plan appropriate shape space motions in order to achieve desired position space motions. Moreover, the dynamic constraints allow accurate tracking of arbitrary trajectories in the shape space but restricts the trajectories that can be followed in the position space, which implies that arbitrary position space motions can only be approximately achieved.

A. Approach

This paper presents a trajectory planner that plans shape trajectories, which when tracked will result in approximate tracking of position trajectories. The shape trajectory planner presented in this paper is restricted to a special class of underactuated systems called *shape-accelerated balancing systems* to which balancing mobile robots like the ballbot belong. In shape-accelerated balancing systems, one can map their shape configurations to the accelerations in the position space. Moreover, in the neighborhood of the origin, one can find a linear map from the accelerations in the position space to the shape configurations. The shape trajectory planner presented in this paper finds such a time-invariant linear map (constant gain matrix) that transforms the desired acceleration trajectory in the position space to a planned shape trajectory such that the sum of squared error between the desired acceleration trajectory and the acceleration trajectory resulting from tracking the planned shape trajectory is minimized. Therefore, the shape trajectory planning for shape-accelerated balancing systems is reduced to an optimization problem of finding a time-invariant linear map (constant gain matrix) from desired accelerations in the position space to shape configurations, and standard nonlinear leasts-squares optimization tools are used to solve this problem.

B. Contributions

This paper introduces shape-accelerated balancing systems as a special class of underactuated systems wherein one can map their shape configurations to accelerations in the position space (Sec. III-B). It presents a trajectory planner that plans shape trajectories, which when tracked result in approximate tracking of desired acceleration trajectories in the position space. The shape trajectory planner can handle systems with more shape variables than position variables, and can also account for additional shape constraints (Sec. IV). This paper also presents a control architecture that achieves closed-loop tracking of desired position trajectories. Several experimental results on the ballbot with arms, which validate the shape trajectory planner and the control architecture are presented (Sec. V). This paper is a significantly improved and extended version of the work presented in (Nagarajan 2010; Nagarajan et al. 2012).

II. Related Work

Several nonlinear control procedures based on partial feedback linearization (Isidori 1989; Isidori and Byrnes 1990; Spong 1994) are available in the nonlinear control literature for regulation of underactuated mechanical systems. Underactuated balancing systems have unstable zero dynamics, and are called nonminimum-phase systems. A variety of nonlinear inversion based approaches (Devasia and Paden 1994; Devasia et al. 1996; Getz 1994; Getz and Hedrick 1995; Getz 1996) have been used in the nonlinear control literature to achieve approximate tracking of desired trajectories for such systems. One such dynamic inversion method was developed by Getz (Getz 1994). He developed a nonlinear controller based on internal equilibrium manifold (Getz and Hedrick 1995) for nonlinear nonminimum-phase systems that provided a larger region of attraction over

linear regulators, and enabled better output tracking while maintaining balance (Getz 1996). These control procedures were demonstrated on bicycle models (Getz 1994). These dynamic inversion based approaches are computationally expensive, and cannot be run real-time on robots. Moreover, these approaches are sensitive to modeling uncertainties, especially to uncertainties in the actuator dynamics. This paper presents a trajectory planning algorithm that is fast enough to run real-time on robots. Moreover, since the trajectory planner presented in this paper uses only the dynamic constraint equations, a subset of the equations of motion, it is more robust to modeling uncertainties in actuator mechanisms and nonlinear friction effects.

Geometric mechanics tools have been used to study the effect of internal shape changes on net changes in position and orientation in mechanical systems with nonholonomic constraints and symmetries (Ostrowski and Burdick 1996; Ostrowski 1999). Ostrowski presented the mechanical connection and the reconstruction equation that relate shape changes to momentum and position (Ostrowski and Burdick 1995). He presented various gaits for snakeboards (Lewis et al. 1994) and Hirose snakes (Hirose 1993), and addressed their controllability issues (Ostrowski and Burdick 1995). However, planning procedures that plan for motions in the shape space to achieve desired motions in the position space were not presented. Shammas et al. presented a variety of gait design tools for generating kinematic and dynamic gaits for principally kinematic, purely mechanical systems (Shammas et al. 2006, 2007a), and dynamic systems with nonholonomic velocity constraints (Shammas et al. 2005, 2007b). However, these gait design tools are not applicable to dynamic systems with nonholonomic acceleration constraints. Hatton and Choset (Hatton and Choset 2008) used the connection, which relates the body velocity to internal shape changes, to create a set of vector fields on the shape space called *connection vector fields*. Each connection vector field corresponds to one component of the body velocity, and informs how a given shape change will move the system through its position space. The main advantage of this approach is that it is not restricted to gaits, and can be used for any general shape change. However, this procedure was restricted to principally kinematic and purely mechanical systems (Shammas et al. 2006, 2007a).

Direct collocation methods (Hargraves and Paris 1987; von Stryk 1993; von Stryk and Bulirsch 1992) have emerged as popular numerical techniques for solving optimal control and feasible trajectory generation problems for nonlinear systems. These approaches use piecewise polynomial approximations for the state and control trajectories and transform the problem into an optimization problem subject to nonlinear constraints given by the equations of motion. The trajectory planner presented in this paper is significantly faster than direct collocation methods in finding feasible trajectories that approximate desired position space motions.

III. Background

This section presents the general equations of motion of underactuated systems, and introduces position and shape variables. It also introduces shape-accelerated balancing systems and their dynamic constraint equations. A brief description of the ballbot with arms, its 3D dynamic model, and its control architecture are also presented.

A. Underactuated Mechanical Systems

The forced Euler-Lagrange equations of motion for an underactuated mechanical system are given by:

$$\frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}} - \frac{\partial \mathscr{L}}{\partial q} = \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}, \tag{1}$$

where $q \in \mathbb{R}^n$ is the configuration vector, $\mathscr{L}(q, \dot{q}) = K(q, \dot{q}) - V(q)$ is the Lagrangian with kinetic energy $K(q, \dot{q})$ and potential energy $V(q)$, and $\tau \in \mathbb{R}^m$ is the vector of generalized forces. The mechanical system satisfying Eq. 1 is called an *underactuated system* (Spong 1994) because there are fewer independent control inputs than configuration variables, $i.e.$, $m < n$. Equation 1 can be written in matrix form as follows:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}, \tag{2}$$

where $M(q) \in \mathbb{R}^{n \times n}$ is the mass/inertia matrix, $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$ is the Coriolis and centrifugal matrix, and $G(q) \in \mathbb{R}^n$ is the vector of gravitational forces.

The configuration variables $q \in \mathbb{R}^n$ of any dynamic system can be split into *position variables* $q_x \in \mathbb{R}^{n_x}$, and *shape variables* $q_s \in \mathbb{R}^{n_s}$, $i.e.$, $q = [q_x, q_s]^T$ and $n_x + n_s = n$. The shape variables q_s are those that appear in the mass/inertia matrix $M(q)$, whereas the position/external variables q_x are those that do not appear in the mass/inertia matrix $M(q)$ (Olfati-Saber 2001). This implies that $M(q)$ is a function of only the shape variables q_s. Position variables represent the position of the robot in the world frame, and the dynamics of mobile robots are independent of transformations of their position variables. However, shape variables affect the mass/inertia matrix of the system and hence dominate the system dynamics.

B. Shape-Accelerated Balancing Systems

The work presented in this paper focuses on a special class of underactuated systems called *shape-accelerated balancing systems* to which balancing mobile robots like the ballbot belong. Other examples of shape-accelerated balancing systems include planar and 3D cart-pole systems with unactuated lean angles, planar balancing wheeled robots like the Segway (Nguyen et al. 2004) moving in a plane, and marble-maze robots. Shape-accelerated balancing systems have several special properties, some which are exploited by the shape trajectory planner presented in Sec. IV. A detailed presentation of the properties of shape-accelerated balancing systems can be found in (Nagarajan 2012).

The degree of underactuation of a shape-accelerated balancing system matches the number of its position variables $q_x \in \mathbb{R}^{n_x}$, $i.e, n_x = n - m$. Moreover, the number of shape variables $q_s \in \mathbb{R}^{n_s}$ is an integral multiple of the number of position variables $q_x \in \mathbb{R}^{n_x}$, $i.e.$, $n_s = kn_x$, for some $k \in \mathbb{Z}^+$, where $\mathbb{Z}^+$ represents the set of positive integers. This work defines a *shape set* to be a set of n_x shape variables that can independently affect the dynamics of all position variables. Let's consider shape-accelerated balancing systems whose position variables q_x are actuated, while their shape variables q_s contain both actuated $q_{s_a} \in \mathbb{R}^{n_{s_a}}$ and unactuated variables $q_{s_u} \in \mathbb{R}^{n_{s_u}}$. Since the number of unactuated shape variables matches the number of position variables, such a system has one unactuated shape set and $k-1$ actuated shape sets.

Another important property of a shape-accelerated balancing system is that its equations of motion as shown in Eq. 2 are independent of both position and velocity of its position variables, $i.e.$, q_x and $\dot{q}_x$. Therefore, its mass/inertia matrix is of the form:

$$M(q_s)=\begin{bmatrix} M_{xx}(q_s) & M_{xs_a}(q_s) & M_{xsu}(q_s) \\ M_{s_ax}(q_s) & M_{s_as_a}(q_s) & M_{s_as_u}(q_s) \\ M_{s_ux}(q_s) & M_{s_us_a}(q_s) & M_{s_us_u}(q_s) \end{bmatrix}, \quad (3)$$

the vector of Coriolis and centrifugal forces is of the form:

$$C(q_s,\dot{q}_s)\dot{q}=\begin{bmatrix} C_{xs_a}(q_s,\dot{q}_s)\dot{q}_{s_a}+C_{xs_u}(q_s,\dot{q}_s)\dot{q}_{s_u} \\ C_{s_as_a}(q_s,\dot{q}_s)\dot{q}_{s_a}+C_{s_as_u}(q_s,\dot{q}_s)\dot{q}_{s_u} \\ C_{s_us_a}(q_s,\dot{q}_s)\dot{q}_{s_a}+C_{s_us_u}(q_s,\dot{q}_s)\dot{q}_{s_u} \end{bmatrix}, \quad (4)$$

and the vector of gravitational forces is of the form:

$$G(q_s) = \begin{bmatrix} \mathbf{0} \\ G_{s_a}(q_s) \\ G_{s_u}(q_s) \end{bmatrix}. \quad (5)$$

The last $n-m$ equations of motion that correspond to the unactuated degrees of freedom are given by

$$M_{s_ux}(q_s)\ddot{q}_x + M_{s_us_a}(q_s)\ddot{q}_{s_a} + M_{s_us_u}(q_s)\ddot{q}_{s_u} + C_{s_us_a}(q_s,\dot{q}_s)\dot{q}_{s_a} + C_{s_us_a}(q_s,\dot{q}_s)\dot{q}_{s_u} + G_{s_u}(q_s) = \mathbf{0} \quad (6)$$

can be written as:

$$\Phi(q_s,\dot{q}_s,\ddot{q}_s,\ddot{q}_x) = \mathbf{0}. \quad (7)$$

Equations 6 and 7 are called *second-order nonholonomic constraints*, or *dynamic constraints* because they are non-integrable (Oriolo and Nakamura 1991). They are not even partially integrable. The dynamic constraint equations in Eq. 7 are independent of the position and velocity of position variables, $i.e.$, q_x and $\dot{q}_x$, but relate the acceleration of position variables $\ddot{q}_x$ to the position, velocity and acceleration of shape variables, $i.e.$, $\left(q_s,\dot{q}_s,\ddot{q}_s\right)$.

The shape-accelerated balancing systems are named so because one can map their shape configurations to the acceleration of their position variables. Such a map can be derived from the dynamic constraint equations as shown in Eq. 9, and the proof of its existence is presented in Theorem 1.

Theorem 1. *For a shape-accelerated balancing system, there exists a nonlinear map $\Gamma(q_s)$ in the neighborhood of the origin that maps the shape configurations to accelerations in the position space such that its dynamic constraint equations in Eq. 7 are satisfied.*

Proof: The Jacobian of $\Phi(q_s,\dot{q}_s,\ddot{q}_s,\ddot{q}_x)$ in Eq. 7 w.r.t. $\ddot{q}_x$ at the origin is given by

$$\left.\frac{\partial\Phi}{\partial\ddot{q}_x}\right|_{(q_s,\dot{q}_s,\ddot{q}_s,\ddot{q}_x)=(\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0})} = \left. M_{s_ux}(q_s)\right|_{q_s=\mathbf{0}}. \quad (8)$$

By the implicit function theorem (Marsden and Hoffman 1993), if the Jacobian in Eq. 8 exists and is invertible, then there exists a map $\Gamma : (q_s,\dot{q}_s,\ddot{q}_s) \rightarrow \ddot{q}_x$ in the neighborhood of the origin such that the dynamic constraints in Eq. 7 are satisfied. For shape-accelerated balancing systems, $M_{s_ux}(q_s)$ exists and is also invertible, and hence, the map Γ exists as shown below:

$$\Gamma(q_s,\dot{q}_s,\ddot{q}_s) = -M_{s_ux}(q_s)^{-1}\Big(M_{s_us_a}(q_s)\ddot{q}_{s_a} + M_{s_us_u}(q_s)\ddot{q}_{s_u} + C_{s_us_a}(q_s,\dot{q}_s)\dot{q}_{s_a} + C_{s_us_a}(q_s,\dot{q}_s)\dot{q}_{s_u} + G_{s_u}(q_s)\Big) \quad (9)$$

such that $\Phi(q_s,\dot{q}_s,\ddot{q}_s,\Gamma(q_s,\dot{q}_s,\ddot{q}_s)) = \mathbf{0}$ in the neighborhood of the origin. ■

International Journal of Robotics Research (IJRR)
Volume 32, Issue 11, pp. 1323 – 1341, September 2013

C. The Ballbot

The ballbot, shown in Fig. 1, is a human-sized mobile robot that balances on a single ball. It is an underactuated system wherein the ball is directly actuated, while the body is not. The ball is actuated using a four-motor inverse mouse-ball drive mechanism shown in Fig. 2(a). A pair of actuated opposing rollers drive the ball in each of the two orthogonal motion directions on the floor. The encoders on the ball motors provide odometry information of the ball, while the body lean angles are measured using an inertial measurement unit (IMU). A more detailed description of the ballbot's hardware can be found in (Nagarajan et al. 2009c). Recently, a pair of 2-DOF arms driven by series-elastic actuators were added to the robot as shown in Fig. 1(b). The arm angles are measured using the encoders on the series-elastic actuators, and a detailed hardware description can be found in (Nagarajan et al. 2012). The system parameters of the ballbot with arms are presented in Table I.

1) 3D Ballbot Model with Arms: The ballbot with arms, shown in Fig. 1(b), is modeled as a rigid cylinder on top of a rigid sphere with a pair of massless arms having weights at their ends. The model assumes that: (i) there is no slip between the ball and the floor; (ii) the floor is flat and level; and (iii) the ball, the body and the arms have two degrees of freedom each with no yaw motion, $i.e.$, rotation about the vertical axis. The arms are placed symmetrically about the body's sagittal plane, and the hardware components within the body are placed such that its center of mass is on its central axis.

The 3D ballbot model with arms has eight configuration variables given by $q = [\theta, \alpha^l, \alpha^r, \phi] \in \mathbb{R}^8$, where, $\theta = [\theta_x, \theta_y]^T \in \mathbb{R}^2$ are configurations of the ball, $\alpha^l = [\alpha^l_x, \alpha^l_y]^T \in \mathbb{R}^2$ are configurations of the left arm, $\alpha^r = [\alpha^r_x, \alpha^r_y]^T \in \mathbb{R}^2$ are configurations of the right arm, and $\phi = [\phi_x, \phi_y]^T \in \mathbb{R}^2$ are configurations of the body. The 3D ballbot model with arms with all of its configurations are shown in Fig. 3.

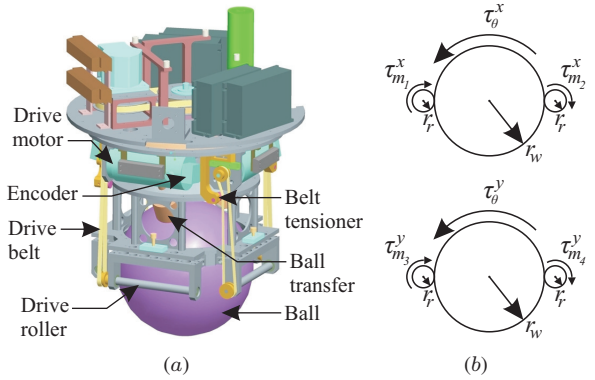


Fig. 2. (a) CAD model of the inverse mouse-ball drive; and (b) Ball actuation in the two orthogonal motion directions.

TABLE I
SYSTEM PARAMETERS FOR THE BALLBOT WITH ARMS

Parameter	Symbol	Value
Ball radius	r_w	0.1058 m
Roller radius	r_r	0.006335 m
Ball mass	m_w	2.44 kg
Ball moment of inertia	I_w	0.0174 kgm^2
Body mass	m_b	70.3 kg
Body CoM distance along z-axis from ball center	ℓ_b	0.87 m
Body roll moment of inertia about its CoM	I^b_{xx}	12.59 kgm^2
Body pitch moment of inertia about its CoM	I^b_{yy}	12.48 kgm^2
Body yaw moment of inertia about its CoM	I^b_{zz}	0.66 kgm^2
Arm mass	m_a	1 kg
Arm length	ℓ_a	0.55 m
Arm joint distance along y-axis from ball center	d^y_a	0.18415 m
Arm joint distance along z-axis from ball center	d^z_a	1.3 m
Arm roll moment of inertia about its CoM	I^a_{xx}	0.0016 kgm^2
Arm pitch moment of inertia about its CoM	I^a_{yy}	0.0016 kgm^2
Arm yaw moment of inertia about its CoM	I^a_{zz}	0.0010 kgm^2

The origin of the world frame is fixed to the initial position of the center of the ball. Since we have assumed a flat and level floor, the position (x_w, y_w) of the center

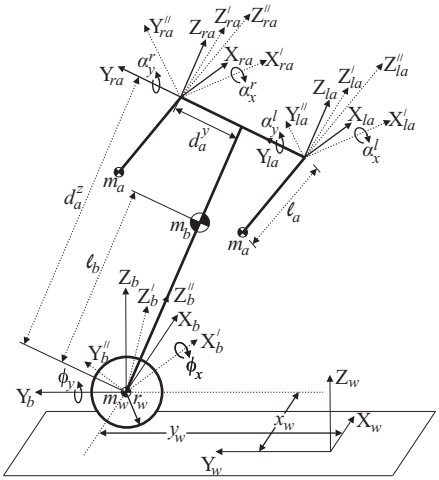


Fig. 3. The 3D ballbot model with a pair of 2-DOF arms: (ϕ_x, ϕ_y) are the body configurations, (α^l_x, α^l_y) are the left arm configurations, and (α^r_x, α^r_y) are the right arm configurations. The ball configurations (θ_x, θ_y) are chosen such that the ball position (x_w, y_w) is given by $x_w = r_w(\theta_x + \phi_y)$ and $y_w = r_w(\theta_y - \phi_x)$, where r_w is the radius of the ball.

of the ball matches the position of the ball's contact point on the floor. In this model, we are interested only in the position of the ball and not in its orientation. The ball configurations (θ_x, θ_y) are chosen such that $x_w = r_w(\theta_x + \phi_y)$ and $y_w = r_w(\theta_y - \phi_x)$, where r_w is the radius of the ball. The ball configurations (θ_x, θ_y) are angular configurations that represent the ball position, and they do not represent the orientation of the ball. Therefore, $\theta_x, \theta_y \in (-\infty, \infty)$. There are two advantages in choosing these coordinates: one is that the ball configurations (θ_x, θ_y) directly correspond to the encoder readings on the ball motors, and the other is that this coordinate choice removes input coupling between the ball and the body from the equations of motion.

The forced Euler-Lagrange equations of motion of the ballbot with arms can be written in matrix form as shown in Eq. 2. The ball configurations form the actuated position variables, $i.e., q_x = \theta \in \mathbb{R}^2$, the arm angles form the actuated shape variables, $i.e., q_{s_a} = [\alpha^l, \alpha^r]^T \in \mathbb{R}^4$, and the body angles form the unactuated shape variables, $i.e., q_{s_u} = \phi \in \mathbb{R}^2$. The ballbot with arms has one unactuated shape set and two actuated shape sets, whereas, the ballbot without arms has one unactuated shape set and no actuated shape sets. For the ballbot with arms, the system matrices in Eq. 2 are of the form given below:

$$M(q) = \begin{bmatrix} M_{\theta\theta} & M_{\theta\alpha^l}(q_s) & M_{\theta\alpha^r}(q_s) & M_{\theta\phi}(q_s) \\ M_{\alpha^l\theta}(q_s) & M_{\alpha^l\alpha^l}(q_s) & M_{\alpha^l\alpha^r}(q_s) & M_{\alpha^l\phi}(q_s) \\ M_{\alpha^r\theta}(q_s) & M_{\alpha^r\alpha^l}(q_s) & M_{\alpha^r\alpha^r}(q_s) & M_{\alpha^r\phi}(q_s) \\ M_{\phi\theta}(q_s) & M_{\phi\alpha^l}(q_s) & M_{\phi\alpha^r}(q_s) & M_{\phi\phi}(q_s) \end{bmatrix}, \tag{10}$$

$$C(q, \dot{q}) = \begin{bmatrix} \mathbf{0} & C_{\theta\alpha^l}(q_s, \dot{q}_s) & C_{\theta\alpha^r}(q_s, \dot{q}_s) & C_{\theta\phi}(q_s, \dot{q}_s) \\ \mathbf{0} & C_{\alpha^l\alpha^l}(q_s, \dot{q}_s) & \mathbf{0} & C_{\alpha^l\phi}(q_s, \dot{q}_s) \\ \mathbf{0} & \mathbf{0} & C_{\alpha^r\alpha^r}(q_s, \dot{q}_s) & C_{\alpha^r\phi}(q_s, \dot{q}_s) \\ \mathbf{0} & C_{\phi\alpha^l}(q_s, \dot{q}_s) & C_{\phi\alpha^r}(q_s, \dot{q}_s) & C_{\phi\phi}(q_s, \dot{q}_s) \end{bmatrix}, \tag{11}$$

$$G(q) = \begin{bmatrix} \mathbf{0} \\ G_{\alpha^l}(q_s) \\ G_{\alpha^r}(q_s) \\ G_{\phi}(q_s) \end{bmatrix}, \tag{12}$$

where, each $M_{ij} \in \mathbb{R}^{2\times2}$, each $C_{ij} \in \mathbb{R}^{2\times2}$ and each $G_i \in \mathbb{R}^{2\times1}$. These submatrices are functions of the position and velocity of the shape variables, and the system parameters. They have long symbolic expressions, and hence are not presented in this paper.

The vector of generalized forces is given by $\tau = [\tau_\theta, \tau_{\alpha^l}, \tau_{\alpha^r}]^T \in \mathbb{R}^{6\times1}$, where $\tau_\theta \in \mathbb{R}^{2\times1}$ is the torque vector on the ball, $\tau_{\alpha^l} \in \mathbb{R}^{2\times1}$ is the torque vector on the left arm, $\tau_{\alpha^r} \in \mathbb{R}^{2\times1}$ is the torque vector on the right arm. Although there are four motors that drive the ball, the ball torque vector $\tau_\theta = [\tau_\theta^x, \tau_\theta^y]^T \in \mathbb{R}^{2\times1}$. As shown in Fig. 2($a$), a pair of active opposing rollers drive the ball along each orthogonal motion direction on the floor. The ball torque vector τ_θ is given by:

$$\begin{bmatrix} \tau_\theta^x \\ \tau_\theta^y \end{bmatrix} = \frac{r_w}{r_r} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \tau_{m1}^x \\ \tau_{m2}^x \\ \tau_{m3}^y \\ \tau_{m4}^y \end{bmatrix}, \tag{13}$$

where r_w is the radius of the ball, r_r is the radius of the roller, and $\tau_{m1}^x, \tau_{m2}^x, \tau_{m3}^y$ and τ_{m4}^y are the torques on the four ball motors. The motor pairs (m_1, m_2) and (m_3, m_4) drive the ball in the orthogonal X and Y directions respectively as shown in Fig. 2(b). In the current setup, the amplifiers that drive the opposing motors are hardwired to command the same torque, and hence, $\tau_{m1}^x = \tau_{m2}^x = \frac{r_r}{2r_w}\tau_\theta^x$ and $\tau_{m3}^y = \tau_{m4}^y = \frac{r_r}{2r_w}\tau_\theta^y$.

2) Ballbot Control Architecture: The ballbot uses a balancing controller to achieve desired body angles. Since the body angles are unactuated, the balancing controller cannot directly track desired body angles. The balancing controller indirectly achieves this objective by actuating the ball such that the projection of the body's center of mass on the floor tracks the projection of the desired center of mass obtained from desired body angles. The balancing controller is a Proportional-Integral-Derivative (PID) controller and more details of it can be found in (Nagarajan et al. 2009c). The trajectory tracking controllers on the arms use the computed torque (Murray et al. 1994) method for feedforward terms and a PID controller for feedback terms (Nagarajan et al. 2012).

The ballbot is also capable of achieving unlimited yaw rotation of its body, $i.e.$ rotation about its vertical axis. The ballbot's body attaches to the ball drive mechanism via a bearing and a slip ring assembly, which make the yaw motion possible. However, the yaw drive mechanism can only yaw the body relative to the ball and cannot directly control the yaw of the ball. The details of the yaw controller that achieves the desired yaw motion of the body can be found in (Nagarajan et al. 2009c). The relative yaw of the ball drive unit and the body is measured by an absolute encoder. The work presented in this paper assumes that the ballbot's body cannot yaw, and hence for the experimental results presented in Sec. V, the ballbot uses its yaw controller to maintain its heading. In order to account for the yaw rotation of the ball drive, the ball drive commands issued by the balancing controller are transformed into the frame of the ball drive unit using the angular offsets measured by

the absolute yaw encoder.

IV. Dynamic Constraint-based Shape Trajectory Planner

The dynamic constraint equations of an underactuated system map its shape configurations to its accelerations in the position space, and vice versa. For a shape-accelerated balancing system, Theorem 1 proved the existence of a map from shape configurations to the accelerations in the position space. In the neighborhood of the origin, one can also find a linear map from accelerations in the position space to the shape configurations. This section presents a shape trajectory planner that uses the dynamic constraint equations to find a time-invariant linear map (constant gain matrix) that transforms the desired acceleration trajectory in the position space to a planned shape trajectory such that the sum of squared error between the desired acceleration trajectory in the position space and the acceleration trajectory resulting from tracking the planned shape trajectory is minimized.

Such a time-invariant linear map can be analytically derived for a special case, wherein the shape-accelerated balancing system sticks to a constant shape configuration, $e.g.$, the ballbot leaning at a constant body angle as described in Sec. IV-A. For the general case, Sec. IV-B formulates the shape trajectory planning as an optimization problem of finding the time-invariant linear map that minimizes the sum of squared error between the desired acceleration trajectory and the acceleration trajectory resulting from tracking the planned shape trajectory. The optimization algorithm uses the analytically derived time-invariant linear map from the special case as its initial guess. The shape trajectory planner is also extended to handle additional shape constraints as described in Sec. IV-C, and the control architecture that enables closed-loop tracking of desired position trajectories is presented in Sec. IV-D.

A. Special Case: Constant Shape Configuration

A constant, non-zero shape configuration q_s with $\dot{q}_s = \mathbf{0}$ and $\ddot{q}_s = \mathbf{0}$ reduces the dynamic constraint equations $\Phi(q_s, \dot{q}_s, \ddot{q}_s, \ddot{q}_x)$ in Eq. 7 to $\Phi'(q_s, \ddot{q}_x)$ given by

$$\begin{aligned}\Phi'(q_s, \ddot{q}_x) &= \Phi(q_s, \mathbf{0}, \mathbf{0}, \ddot{q}_x) \\ &= M_{s_u x}(q_s)\ddot{q}_x + G_{s_u}(q_s). \end{aligned} \quad (14)$$

It follows from Eq. 7 that $\Phi'(q_s, \ddot{q}_x) = \mathbf{0}$. Theorem 1 holds in this case too, and hence the map $\Gamma(q_s, \dot{q}_s, \ddot{q}_s)$ in Eq. 9 reduces to $\Gamma'(q_s) : q_s \rightarrow \ddot{q}_x$ given by:

$$\Gamma'(q_s) = -M_{s_u x}(q_s)^{-1} G_{s_u}(q_s) \quad (15)$$

The Jacobian linearization of $\Gamma'(q_s)$ in Eq. 15 w.r.t. q_s at $q_s = \mathbf{0}$ gives a linear map $K^0_{q_s}$:

$$K^0_{q_s} = -\frac{\partial\big(M_{s_u x}(q_s)^{-1} G_{s_u}(q_s)\big)}{\partial q_s}\bigg|_{q_s=\mathbf{0}} \in \mathbb{R}^{n_x \times n_s}, \quad (16)$$

which is a function of only system parameters, and hence it is a constant gain matrix. Therefore, in the neighborhood of the origin, tracking a constant shape configuration results in a constant acceleration in the position space given by

$$\ddot{q}_x = K^0_{q_s} q_s. \quad (17)$$

In order to find a linear map that maps desired accelerations in position space to shape configurations, $K^0_{q_s}$ must be invertible. Theorem 2 presents the conditions on the invertibility of $\Gamma'(q_s)$ and $K^0_{q_s}$. It is important to note that the definitions of the neighborhood of origin in which $\Gamma'(q_s)$ and $K^0_{q_s}$ are valid are different from each other since $\Gamma'(q_s)$ is an exact map, whereas $K^0_{q_s}$ is an approximate map.

Theorem 2. *For a shape-accelerated balancing system in a constant shape configuration, the nonlinear map $\Gamma'(q_s)$ in Eq. 15 and the linear map $K^0_{q_s}$ in Eq. 16 are invertible in the neighborhood of the origin only when the shape space and the position space are of equal dimensions.*

Proof: The Jacobian of $\Phi'(q_s, \ddot{q}_x)$ in Eq. 14 w.r.t. q_s at $(q_s, \ddot{q}_x) = (\mathbf{0}, \mathbf{0})$ is given by

$$\frac{\partial \Phi'}{\partial q_s}\bigg|_{(q_s, \ddot{q}_x)=(\mathbf{0},\mathbf{0})} = \frac{\partial G_{s_u}(q_s)}{\partial q_s}\bigg|_{q_s=\mathbf{0}}. \quad (18)$$

By the implicit function theorem (Marsden and Hoffman 1993), if the Jacobian in Eq. 18 exists and is invertible then, the map Γ' in Eq. 15 is invertible in the neighborhood of the origin, $i.e.$, given an acceleration in the position space, the constant shape configuration that causes it will be given by Γ'^{-1}. For shape-accelerated balancing systems, $\dfrac{\partial G_{s_u}(q_s)}{\partial q_s} \neq \mathbf{0} \in \mathbb{R}^{n_{s_u} \times n_s}$ at $q_s = \mathbf{0}$ exists but is invertible only when the shape space and the position space are of equal dimensions, and all shape variables are unactuated. This implies that the map Γ' in Eq. 15 is invertible only when the shape space and the position space are of equal dimensions, and all shape variables are unactuated. Similarly, the linear map $K^0_{q_s}$ in Eq. 16 is also invertible only when the shape space and the position space are of equal dimensions, and all shape variables are unactuated. ■

Theorem 2 shows that for systems with equal number of shape and position variables like the ballbot without

International Journal of Robotics Research (IJRR)
Volume 32, Issue 11, pp. 1323 – 1341, September 2013

arms, both the maps $\Gamma'(q_s)$ and $K^0_{q_s}$ exist and are invertible in the neighborhood of the origin, whereas for systems with more shape variables than position variables like the ballbot with arms, both the maps $\Gamma'(q_s)$ and $K^0_{q_s}$ exist but are not invertible. A discussion on both these cases is presented below.

1) Shape space and position space of equal dimensions: Consider shape-accelerated balancing systems with equal number of shape and position variables, $e.g.$, the ballbot without arms. Here, all shape variables are unactuated, $i.e.$, $n_{s_a} = 0$. For such systems, Theorem 2 shows that the map $\Gamma'(q_s)$ in Eq. 15 and the linear map $K^0_{q_s}$ in Eq. 16 are both invertible. This implies that there exists a linear map $K^0_{q_x}$ given by

$$K^0_{q_x} = (K^0_{q_s})^{-1} \in \mathbb{R}^{n_s \times n_x} \tag{19}$$

such that

$$q_s = K^0_{q_x}\, \ddot{q}_x. \tag{20}$$

Equations 17 and 20 show that $\ddot{q}_x$ is a constant if q_s is a constant, and vice versa. Therefore, for shape-accelerated balancing systems with equal number of shape and position variables, a constant desired acceleration $\ddot{q}_x$ in the position space is achieved by tracking a constant shape configuration q_s given by Eq. 20.

For example, the nonlinear map $\Gamma'(q_s)$ for the ballbot without arms is given by:

$$\Gamma'(q_s) = \frac{\eta_3}{\eta_1 + \eta_2 C_{\phi_x} C_{\phi_y}} \begin{bmatrix} C_{\phi_x} S_{\phi_y} \\ \dfrac{-S_{\phi_x}(\eta_1 C_{\phi_y} + \eta_2 C_{\phi_x})}{\eta_1 + \eta_2 C_{\phi_x}} \end{bmatrix} \in \mathbb{R}^{2\times 1}, \tag{21}$$

and the linear maps $K^0_{q_s}$ and $K^0_{q_x}$ are given by:

$$K^0_{q_s} = \frac{\eta_3}{\eta_1 + \eta_2} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{22}$$

$$K^0_{q_x} = \frac{\eta_1 + \eta_2}{\eta_3} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{23}$$

where $C_i = \cos(i)$, $S_i = \sin(i)$, and $\eta_1 = I_w + (m_b + m_w) r_w^2$, $\eta_2 = m_b \ell_b r_w$, $\eta_3 = m_b g \ell_b$ are non-zero, positive functions of the system parameters listed in Table I.

2) Shape space with more dimensions than position space: Now, let's consider shape-accelerated balancing systems with more shape variables than position variables, $e.g.$, the ballbot with arms. For such systems, Theorem 2 shows that the map $\Gamma'(q_s)$ in Eq. 15 and the linear map $K^0_{q_s}$ in Eq. 16 are both not invertible. This is obvious because the matrix $K^0_{q_s} \in \mathbb{R}^{n_x \times n_s}$ is singular with more columns than rows. This implies that there are infinite possible shape configurations that can produce the same acceleration in the position space.

A minimum-norm pseudo-inverse of $K^0_{q_s}$ given by $(K^0_{q_s})^{\#}$ minimizes $\|q_s\|_2$, where $q_s = (K^0_{q_s})^{\#} \ddot{q}_x$ for any acceleration $\ddot{q}_x$ in the position space. However, in order to have the flexibility of choosing and relatively weighing the contributions of the different shape sets to achieve a desired motion in the position space, we use a weighted pseudo-inverse (Nakamura 1991), which minimizes $\|W^{-1} q_s\|_2$, where $W \in \mathbb{R}^{n_s \times n_s}$ is a symmetric, positive definite weight matrix on the shape variables. Therefore, for a shape-accelerated balancing system with more shape variables than position variables, the time-invariant linear map $K^0_{q_x} : \ddot{q}_x \to q_s$ is chosen as the weighted pseudo-inverse of $K^0_{q_s}$ given by

$$K^0_{q_x} = W \big(K^0_{q_s} W\big)^{\#} \in \mathbb{R}^{n_s \times n_x}, \tag{24}$$

where, $(\cdot)^{\#}$ represents the minimum-norm pseudo-inverse. The weight matrix W is chosen as

$$W = \begin{bmatrix} W_{q_{s_a}} & \mathbf{0} \\ \mathbf{0} & W_{q_{s_u}} \end{bmatrix} \in \mathbb{R}^{n_s \times n_s}, \tag{25}$$

where $W_{q_{s_a}} \in \mathbb{R}^{n_{s_a} \times n_{s_a}}$ and $W_{q_{s_u}} \in \mathbb{R}^{n_{s_u} \times n_{s_u}}$ are symmetric, positive definite weight matrices on the actuated and unactuated shape sets respectively.

For the ballbot with arms, its nonlinear map $\Gamma'(q_s)$ has long symbolic expressions and hence is not presented in this paper. However, the Jacobian linearization of $\Gamma'(q_s)$ w.r.t. its shape variables is given by:

$$K^0_{q_s} = \frac{1}{\chi_3} \begin{bmatrix} 0 & -\chi_1 & 0 & -\chi_1 & 0 & \chi_2 \\ \chi_1 & 0 & \chi_1 & 0 & -\chi_2 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 6}, \tag{26}$$

where $\chi_1 = m_a g \ell_a$, $\chi_2 = m_b g \ell_b + 2 m_a g (d^z_a - \ell_a)$, and $\chi_3 = I_w + (2m_a + m_b + m_w) r_w^2 + m_b \ell_b r_w + 2 m_a (d^z_a - \ell_a) r_w$ are all non-zero, positive functions of the system parameters shown in Table I.

The weight matrix W can be chosen such that either pure body motions or pure arm motions or any combination of the two are used to achieve desired accelerations in the position space. The weight matrix W for the ballbot with arms is of the form:

$$W = \begin{bmatrix} W_{\alpha^l} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & W_{\alpha^r} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & W_{\phi} \end{bmatrix} \in \mathbb{R}^{6\times 6}, \tag{27}$$

where,

$$W_{\alpha^l} = \frac{\chi_3}{\chi_1} \sqrt{c_{\alpha^l}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{28}$$

$$W_{\alpha^r} = \frac{\chi_3}{\chi_1} \sqrt{c_{\alpha^r}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{29}$$

$$W_{\phi} = \frac{\chi_3}{\chi_2} \sqrt{c_{\phi}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{30}$$

where c_ϕ, c_{α^l} and c_{α^r} are user-picked contribution ratios for the body angle, left arm and right arm angles respectively such that $c_\phi + c_{\alpha^l} + c_{\alpha^r} = 1$, and χ_1, χ_2 and χ_3 are the same as in Eq. 26. The planned shape motions can be restricted to just body angles by picking $c_\phi = 1$ and $c_{\alpha^l} = c_{\alpha^r} = 0$. Similarly, equal contributions from the body, left arm and right arm angles can be achieved by picking $c_\phi = c_{\alpha^l} = c_{\alpha^r} = 1/3$.

B. General Case: Shape Trajectory Planner

Section IV-A showed that constant shape configurations needed to achieve constant desired accelerations in the position space can be obtained from the linear maps shown in Eq. 19 and Eq. 24. However, in order to accurately track arbitrary desired acceleration trajectories in position space, the map $\Gamma(q_s, \dot{q}_s, \ddot{q}_s)$ in Eq. 9 must be invertible, which is not the case. But any arbitrary acceleration trajectory in position space can be approximately tracked, and this section presents a shape trajectory planner that finds a time-invariant linear map (constant gain matrix), which transforms desired accelerations in the position space to planned shape configurations such that the sum of squared error between the desired acceleration trajectory in the position space and the acceleration trajectory resulting from tracking the planned shape trajectory is minimized.

Given a desired acceleration trajectory in the position space $\ddot{q}_x^d(t)$, the proposed shape trajectory planner finds a time-invariant linear map $K_{q_x} : \ddot{q}_x \rightarrow q_s$, similar to Eq. 19 and 24, such that the planned shape trajectory $q_s^p(t)$ given by

$$q_s^p(t) = K_{q_x} \ddot{q}_x^d(t), \tag{31}$$

when tracked, will result in an acceleration trajectory $\ddot{q}_x^p(t)$ given by

$$\ddot{q}_x^p(t) = \Gamma\Big(K_{q_x} \ddot{q}_x^d(t), K_{q_x} \dddot{q}_x^d(t), K_{q_x} \ddddot{q}_x^d(t)\Big), \tag{32}$$

which minimizes the sum of squared error J given by

$$J = \sum_{t=0}^{t_f} \Big\| \ddot{q}_x^p(t) - \ddot{q}_x^d(t) \Big\|_2^2, \tag{33}$$

where t_f is the time duration of the desired motion.

The optimization can be solved using nonlinear least-squares solvers like Nelder-Mead simplex (Nelder and Mead 1964) and Levenberg-Marquardt (Levenberg 1944) algorithms. As described in Sec. IV-A, a constant desired acceleration trajectory can be achieved using $K_{q_x} = K_{q_x}^0$ given in Eq. 19 and Eq. 24, whereas for any arbitrary $\ddot{q}_x^d(t)$, $K_{q_x}^0$ is used as the initial guess for the optimization process. It is important to note that the gain matrix K_{q_x} is time-invariant, *i.e.*, it is constant over the entire trajectory.

The shape trajectory planner presented above deals only with the tracking of a desired acceleration trajectory $\ddot{q}_x^d(t)$ and not of a desired position trajectory $q_x^d(t)$. The acceleration trajectory $\ddot{q}_x^p(t)$ can be numerically integrated to obtain the resulting position trajectory $q_x^p(t)$ using initial conditions of the desired acceleration trajectory $q_x^d(t)$. Therefore, with matching initial conditions, the position trajectory $q_x^p(t)$ approximately tracks the desired position trajectory $q_x^d(t)$ if its acceleration trajectory $\ddot{q}_x^p(t)$ approximately tracks the desired acceleration trajectory $\ddot{q}_x^d(t)$. The planned shape trajectory $q_s^p(t)$ and the position trajectory $q_x^p(t)$ form the feasible configuration trajectories that approximate the desired motion in the position space. When the initial conditions are not met, feedback tracking of desired position trajectories is necessary and the control architecture that achieves it is presented in Sec. IV-D.

C. Planning with Additional Shape Constraints

A system with more shape variables than position variables may need to use a subset of its shape configurations to achieve tasks other than navigation. For example, the ballbot with arms can use its arms for manipulation, which will constrain the arm angles to some specific manipulation trajectories. This section presents a variant of the shape trajectory planner that can handle these additional shape constraint trajectories, and still achieve desired motions in the position space using the other available shape configurations. The shape planner assumes that there is at least one shape set available without additional constraints so as to achieve desired motions in the position space.

With no loss of generality, let's assume that there is just one actuated shape set, and it is constrained to some reference trajectory, while the unactuated shape set has no additional constraints. The objective here is to plan trajectories for the unactuated shape configurations such that they achieve the desired motion in the position space, while counteracting the effect of additional constraints on the other shape set.

Tracking the additional constraint (ac) trajectories for actuated shape variables $q_{s_a}^{ac}(t)$ results in acceleration trajectories $\ddot{q}_x^{ac}(t)$ in the position space given by

$$\ddot{q}_x^{ac}(t) = \Gamma\Big(q_s^{ac}(t), \dot{q}_s^{ac}(t), \ddot{q}_s^{ac}(t)\Big), \tag{34}$$

where $q_s^{ac}(t) = \big[q_{s_a}^{ac}(t), \mathbf{0}\big]^T$, *i.e.*, zero angle trajectories for the unactuated shape configurations q_{s_u}, and Γ is obtained from Eq. 9. To achieve the desired acceleration

International Journal of Robotics Research (IJRR)
Volume 32, Issue 11, pp. 1323 – 1341, September 2013

Algorithm 1: Shape Trajectory Planner

input : Desired Position Trajectory $q_x^d(t)$
Constraint Shape Trajectory $q_s^{ac}(t)$
Weight Matrix W
output: Planned Shape Trajectory $q_s^p(t)$

1 **begin**
2 Desired acceleration trajectory
$$\ddot{q}_x^d(t) \leftarrow \frac{d^2 q_x^d(t)}{dt^2}$$
3 Acceleration trajectory resulting from additional constraint shape trajectory
$$\ddot{q}_x^{ac}(t) \leftarrow \Gamma\Big(q_s^{ac}(t), \dot{q}_s^{ac}(t), \ddot{q}_s^{ac}(t)\Big) \text{ (Eq. 9)}$$
4 Net desired acceleration trajectory
$$\ddot{q}_x^{net}(t) \leftarrow \ddot{q}_x^d(t) + \ddot{q}_x^{ac}(t)$$
5 Initialize the time-invariant linear map
$$K_{q_x} \leftarrow K_{q_x}^0 \text{ (Eq. 24)}$$
6 **repeat**
7 Planned shape trajectory
$$q_s^p(t) \leftarrow K_{q_x} \ddot{q}_x^{net}(t)$$
8 Resulting acceleration trajectory
$$\ddot{q}_x^p(t) \leftarrow \Gamma\Big(q_s^p(t), \dot{q}_s^p(t), \ddot{q}_s^p(t)\Big) \text{ (Eq. 9)}$$
9 Calculate cost function
$$J \leftarrow \sum_{t=0}^{t_f} \left\| \ddot{q}_x^p(t) - \ddot{q}_x^d(t) \right\|_2^2 dt$$
10 Update the time-invariant linear map
$$\Delta K \leftarrow \text{OptimizerUpdate}(J,\ K_{q_x})$$
$$K_{q_x} \leftarrow K_{q_x} + \Delta K$$
11 **until** $J < J_{thres}$ or $\Delta K < \Delta K_{thres}$
12 **return** Planned shape trajectory $q_s^p(t)$
13 **end**

trajectory $\ddot{q}_x^d(t)$ in the position space, the unactuated shape configurations have to achieve motions that compensate for $\ddot{q}_x^{ac}(t)$, and achieve $\ddot{q}_x^d(t)$. Therefore, in this case, the planned shape trajectory $q_s^p(t)$ is chosen to be

$$q_s^p(t) = K_{q_x} \ddot{q}_x^{net}(t), \tag{35}$$

where $\ddot{q}_x^{net}(t) = \ddot{q}_x^d(t) - \ddot{q}_x^{ac}(t)$ is the net desired acceleration trajectory that the planner uses for planning unactuated shape trajectories. The linear map K_{q_x} in Eq. 35 is obtained using the optimization procedure described in Sec. IV-B, and it is initialized to $K_{q_x}^0$ in Eq. 24 with the weight matrix W chosen such that the shape variables without additional constraints are chosen over those with additional constraints.

The overall shape trajectory planner is presented in Algorithm 1. The Steps 6−11 of the algorithm are implemented using nonlinear least-squares optimizers like Nelder-Mead simplex (Nelder and Mead 1964) and Levenberg-Marquardt (Levenberg 1944) algorithms.

D. Control Architecture

The shape trajectory planner presented above assumes that there exists controllers that can accurately track the planned shape trajectories. For the ballbot with arms, the shape configurations include arm and body angles. As described in Sec. III-C2, the desired body angle trajectories are tracked using the PID balancing controller, while the desired arm angle trajectories are tracked using a combination of feedforward and PID feedback controllers. Tracking desired position trajectories $q_x^d(t)$ by tracking planned shape trajectories $q_s^p(t)$ is open-loop as there is no feedback on the position configurations. This open-loop procedure cannot ensure good tracking of desired position trajectories when the system starts at wrong initial conditions. Moreover, on real robots, there are more issues such as modeling uncertainties, unmodeled dynamics, nonlinear friction effects and noise that will inhibit a good position trajectory tracking performance. A feedback position tracking controller is used to overcome all these issues as shown in Fig. 4.

The feedback position tracking controller tracks planned position trajectories $q_x^p(t)$, which are feasible approximations to desired position trajectories $q_x^d(t)$. The feedback position tracking controller is a Proportional-Derivative (PD) controller that outputs compensation shape trajectories $q_s^c(t)$, which compensate for the error in tracking planned position trajectories $q_x^p(t)$. The planned shape trajectories $q_s^p(t)$ and compensation shape trajectories $q_s^c(t)$ are combined to produce the desired shape trajectories $q_s^d(t)$ as follows:

$$q_s^d(t) \quad = \quad q_s^p(t) + q_s^c(t). \tag{36}$$

It is important to note that the feedback position tracking controller is different from the shape trajectory tracking controller as shown in Fig. 4. The feedback position tracking controller is capable of handling wrong initial conditions, uncertainties in the environment like uneven

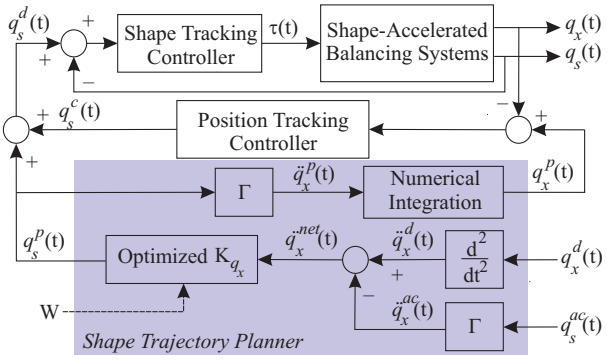


Fig. 4. Control architecture with the shape trajectory planner.

floors and even small gradients. It is also capable of handling small disturbances, and ensures good tracking of the planned position trajectories $q_x^p(t)$.

E. Characteristics of Desired Position Trajectories

The shape trajectory planner presented in this paper requires that the desired position trajectories $q_x^d(t)$ must be at least of differentiability class C^2, so that the desired acceleration trajectories $\ddot{q}_x^d(t)$ exist and are continuous. However, it is preferred to have $q_x^d(t)$ be of differentiability class C^4 so that the planned shape trajectories $q_s^p(t)$ and their first two derivatives ($\dot{q}_s^p(t)$, $\ddot{q}_s^p(t)$) that depend on them exist and are continuous.

Moreover, the desired position trajectories must satisfy acceleration bounds that depend on the shape configurations used to achieve these motions. The planner plans shape trajectories that are linearly proportional to desired acceleration trajectories in the position space, and will fail to achieve good tracking if it is outside this linear region. The acceleration bounds on desired position trajectories are used to limit the motions of a shape-accelerated balancing system to this linear neighborhood, and are also used to account for actuator saturations. The nonlinear map $\Gamma'(q_s)$ in Eq. 15 of the ballbot with arms as a function of the body angle and as a functon of the arm angle are shown in Fig. 5.

Figure 5(a) shows that the nonlinear map $\Gamma'(q_s)$ is approximately linear w.r.t. the body angle upto a body lean of $30°$, which is the lean at which the cylindrical structure of the body makes contact with the floor. Therefore, the nonlinear map $\Gamma'(q_s)$ is approximately linear for the entire range of body angle values of interest. However, in this work, the acceleration bounds are limited to 2 m/s^2 in order to avoid actuator saturations and large accelerations. This acceleration bound corresponds to a maximum body lean of $10°$ as shown by the highlighted region in Fig. 5(a).

For an arm with 1 kg mass at its end, Fig. 5(b) shows that the nonlinear map $\Gamma'(q_s)$ is approximately linear w.r.t. the arm angle upto $55°$, which corresponds to a ball acceleration of 0.082 m/s^2 as shown by the highlighted region. This value is used as the acceleration bound while using arm motions to achieve desired motions on the floor. Larger masses at the end of the arms will result in larger accelerations in the position space. However, for arms with 1 kg masses, it can be seen that significantly larger accelerations can be achieved using body motions than using arm motions. The linear approximation of the nonlinear map $\Gamma'(q_s)$ works well within these bounded regions, and these acceleration bounds are large enough to accommodate the navigation needs of a balancing mobile robot like the ballbot. These bounds were used for all the experimental results presented in Sec. V.

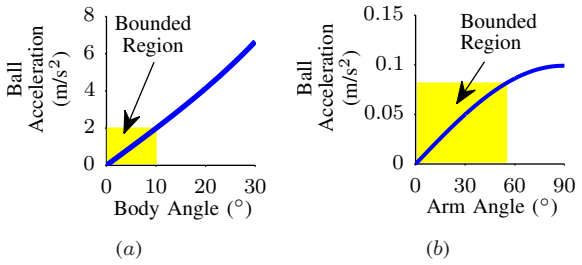


Fig. 5. Nonlinear function of ball acceleration $vs.$ shape configuration for the ballbot with arms: (a) Body Angle; (b) Arm Angle.

F. Performance Comparison against Direct Collocation Methods

Direct collocation methods (Hargraves and Paris 1987; von Stryk 1993; von Stryk and Bulirsch 1992) have emerged as popular numerical techniques to generate feasible trajectories for nonlinear systems. The state and control trajectories are discretized into finite collocation points, and the trajectory generation is solved as an optimization problem subject to nonlinear constraints given by the equations of motion of the system.

Table II compares the performance of PROPT (Rutquist and Edvall 2010), a fast optimal control platform for MATLAB that uses direct collocation methods against that of the shape trajectory planner presented in this paper on four different shape-accelerated balancing systems listed. The trajectory optimization was performed using the SNOPT solver (Gill et al. 2005) on PROPT. The shape trajectory planner presented in this paper was implemented using the *lsqnonlin* function in MATLAB, which uses the Levenberg-Marquardt Algorithm (LMA) (Levenberg 1944) for optimization. Here, the objective was to minimize the sum of squared error in tracking a desired straight line position space motion of 2 m in 5 s with a functional tolerance of $<10^{-3}$ m^2. The PROPT implementation used 101 collocation points for each state, which were necessary for generating reasonably smooth trajectories.

The shape trajectory planner presented in this paper is able to generate feasible trajectories at $28-72$ times faster speeds than PROPT on a standard Core2-Duo processor. The computation times listed are average values over 10 runs. This speed is not surprising as the optimization is performed on a much smaller parameter space compared to that of the direct collocation method. Moreover, the shape planner uses only the dynamic constraint equations, whereas PROPT uses all the equations of motion as constraints. The position

International Journal of Robotics Research (IJRR)
Volume 32, Issue 11, pp. 1323 – 1341, September 2013

TABLE II
PERFORMANCE COMPARISON

System (No. of states)	Computation Time (s)		Speed Factor	Position Tracking RMSE (m)	
	PROPT	Shape Planner		PROPT	Shape Planner
Planar cart-pole (4)	27.581	**0.482**	**57**×	0.1143	**0.0211**
Planar ballbot (4)	20.729	**0.485**	**43**×	0.1154	**0.0391**
3D ballbot without arms (8)	27.362	**0.991**	**28**×	0.1159	**0.0478**
3D ballbot with arms (16)	955.6529	**13.318**	**72**×	0.1148	**0.0462**

trajectories obtained from the shape planner have RMSE that are about $2-5$ times smaller than those obtained from PROPT.

It is important to note that the direct collocation results presented in Table II were obtained with the state trajectories initialized to zero. However, when the direct collocation methods were initialized to the planned shape trajectories obtained from the shape trajectory planner, they improved the feasible configuration trajectories and achieved lower tracking errors. The direct collocation methods also performed better when the state trajectories were initialized to the shape trajectories obtained from transforming the desired acceleration trajectories using the linear map $K^0_{q_s}$ in Eq. 24. Therefore, the key point is that the direct collocation methods perform better when the state trajectories have good initial guesses. The shape trajectory planner can also be used to provide such an initial guess to the direct collocation methods.

It is also important to note that direct collocation methods can be used for any arbitrary dynamic system, whereas the shape trajectory planner presented in this paper is restricted to shape-accelerated balancing systems. The computation times presented in Table II are for a MATLAB implementation of the shape trajectory planner, and a well optimized C/C++ implementation can provide the results an order of magnitude faster, which allows real-time planning on the robot.

V. EXPERIMENTAL RESULTS WITH THE BALLBOT

The shape trajectory planner and the control architecture presented in Sec. IV were experimentally validated on the ballbot with arms shown in Fig. 1(b). The arms had 1 kg masses at their ends for the experiments presented in this section. The balancing controller was used to track the desired body angle trajectories, while the trajectory tracking controller for the arms used the computed torque method (Murray et al. 1994) for feedforward terms and a PID position controller for feedback control (Nagarajan et al. 2012). Different weight matrices were picked to select and relatively weigh the body and arm motions.

Unless mentioned otherwise, the desired position trajectories are nonic (ninth degree) polynomials that satisfy all the preferred characteristics of desired position trajectories discussed in Sec. IV-E. For all the experimental results presented in this section, the chosen desired position trajectories cannot be exactly tracked and hence, the shape trajectory planner finds the feasible trajectories that approximate the desired motion. The ball position tracking error presented in this section is the error between the actual ball position and planned (feasible but approximate) ball position trajectories, and is due to modeling uncertainties and nonlinear friction effects in real robot experiments. The ball position and velocity data presented in this section were obtained from the encoders on the ball motors, and no extrinsic sensors were used for localization. The videos of the ballbot successfully achieving the experimental results presented here can be found in Extension 1.

A. Pure Body Motion

Figure 6 shows the ballbot without arms successfully tracking a fast, straight line motion of 1.414 m in 4 s with a root-mean-square error (RMSE) of 0.018 m. During this motion, the ballbot reached a peak velocity of 1.18 m/s and a peak acceleration of 1 m/s^2. The planned and compensation body angle trajectories are shown in Fig. 7(a). The compensation body angle trajectory is provided by the feedback position tracking controller, and is summed with the planned body angle trajectory to produce the desired body angle trajectory that is tracked by the balancing controller. The resulting body angle

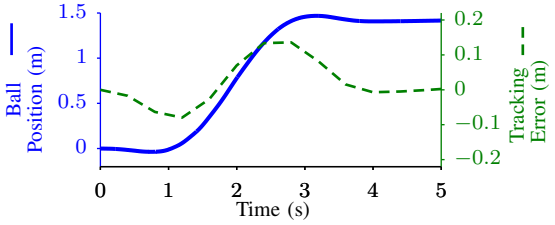


Fig. 6. Pure Body Motion - Tracking a fast, straight line ball motion.

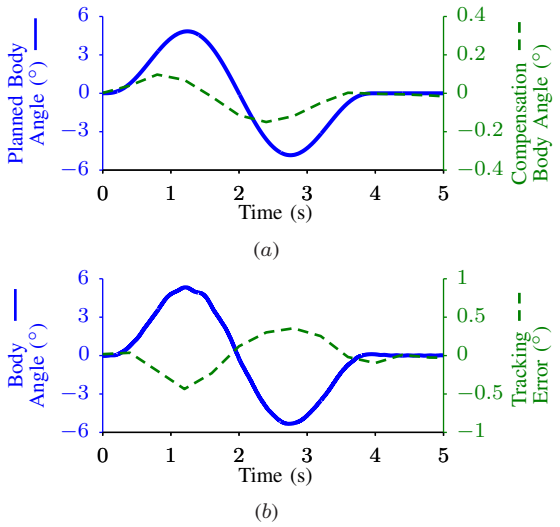


Fig. 7. Tracking a desired fast, straight line ball motion using only body motions: (*a*) Planned and compensation body angle trajectories, (*b*) The resulting body angle trajectory and the tracking error in achieving the desired body angle trajectory, which is a sum of the planned and compensation body angle trajectories.

trajectory and the error in tracking the desired body angle trajectory are shown in Fig. 7(*b*).

Let's compare the straight line tracking performance in Fig. 6 against that in Fig. 8, which is the result of using the LQR controller in (Lauwers et al. 2006) to track a desired straight line motion of 0.707 m in 20 s. This motion is similar to the one shown in Fig. 11 of (Lauwers et al. 2006). Unlike the control architecture shown in Fig. 4, the LQR controller in (Lauwers et al. 2006) is an unified full-state feedback controller that tracks both desired position and body angle trajectories. But it is extremely sensitive to modeling uncertainties and nonlinear friction effects of a soft ball rolling on a hard floor, and it often produces jerky motions. We were unable to achieve successful tracking of motions faster than 0.2 m/s with the ballbot using the LQR controller in (Lauwers et al. 2006). On the other hand, the control architecture (Sec. IV-D) of using a balancing controller to stabilize the shape dynamics and an outer-

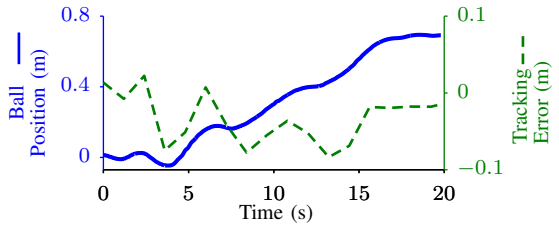


Fig. 8. Tracking a desired straight line ball motion using the LQR controller in (Lauwers et al. 2006).

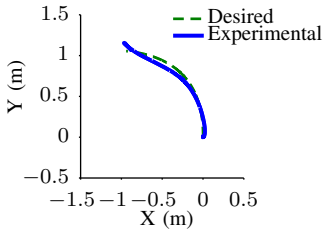


Fig. 9. Pure Body Motion - Tracking a desired curvilinear ball motion.

loop controller to achieve position tracking has been experimentally demonstrated to be robust against modeling uncertainties, nonlinear friction effects and even large disturbances as described in (Nagarajan et al. 2009a,b,c). We were also able to achieve fast and graceful motions as shown in Fig. 6. It is important to note that the shape trajectory planner presented in this paper is also robust to the uncertainties in the actuator model and the nonlinear friction effects since it is dependent only on the dynamic constraint equations.

The ballbot with arms tracking a curvilinear ball motion is shown in Fig. 9. The desired ball motion was chosen to be a Bezier curve of degree $n = 43$ given by

$$\begin{bmatrix} x_w^d(t) \\ y_w^d(t) \end{bmatrix} = \sum_i^n \frac{n!}{i!(n-i)!}\Big(1-t_1\Big)^{n-i} t_1^i p_i, \quad (37)$$

where $p_i = [0,0]^T$ for i =0 to 19; $p_i = [0,1]^T$ for i =20 to 23; $p_i = [-1,1]^T$ for i =24 to 43; $t_1 = t/t_f \in [0,1]$

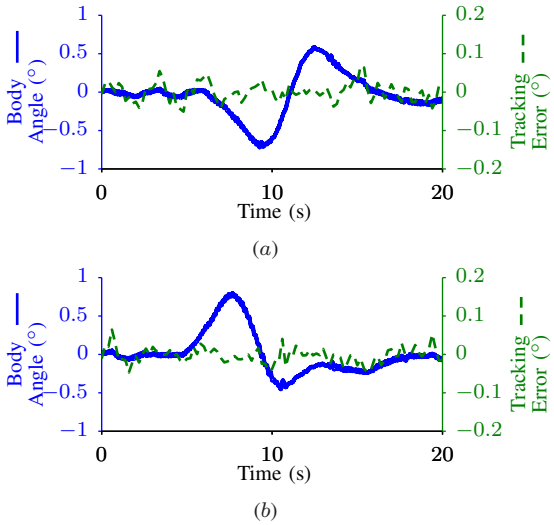


Fig. 10. Tracking a desired curvilinear ball motion using only body motions: (*a*) The resulting X body angle trajectory $\phi_x(t)$ and the tracking error in achieving the desired X body angle trajectory $\phi_x^d(t)$; (*b*) The resulting Y body angle trajectory $\phi_y(t)$ and the tracking error in achieving the desired Y body angle trajectory $\phi_y^d(t)$.

and $t_f = 10$ s. It is important to note that any Bezier curve of degree $n \geq 10$ can be used as a candidate for desired position trajectories as it will satisfy the preferred characteristics of desired position trajectories discussed in Sec. IV-E. The resulting body angle trajectories and the tracking errors are shown in Fig. 10.

For both these experiments, the compensation body angles remained within $\pm 0.15°$, which demonstrates the effectiveness of the shape trajectory planner. Moreover, the arms were maintained at zero angles.

B. Pure Arm Motion

This section presents experimental results of the ballbot with arms achieving desired ball motions using just the arm motions. The arms of the ballbot are lightweight hollow alumnium tubes with 1 kg masses at the ends.

Figure 11 shows the robot tracking a desired straight line motion of 2 m in the forward direction using just the arm motions with a RMSE of 0.035 m. The planned and compensation arm angle trajectories for the left arm are shown in Fig. 12(a), and the desired arm trajectory tracking performance is shown in Fig. 12(b). Similar results were obtained for the right arm. The trajectory tracking performance of the arm controller shown in Fig. 12(b) will be significantly improved with the future arm design, which will avoid the excessive backlash in the current design. The composite frames from a video of the ballbot achieving this motion is shown in Fig. 13. The balancing controller maintained the body angles at zero for these experiments.

Figure 14 shows the ballbot tracking a straight line motion of 1 m in the lateral direction by moving its arms sideways. The right arm is used to initiate the motion as shown in Fig. 15(a), whereas the left arm is used to bring the system to rest as shown in Fig. 15(b). The complete motion is not performed on a single arm in order to avoid self-collision. The desired motion is split into two trajectories, one for the "acceleration-phase" that uses only the right arm, and the other for the "deceleration-phase" that uses only the left arm. Figure 16 shows composite frames of the ballbot achieving the lateral

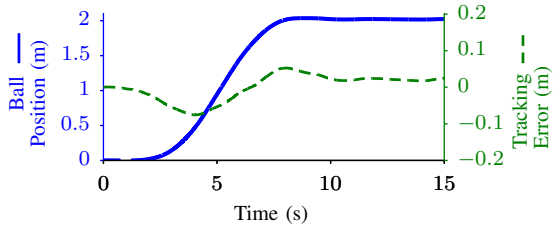


Fig. 11. Pure Arm Motion - Tracking a desired forward straight line ball motion.

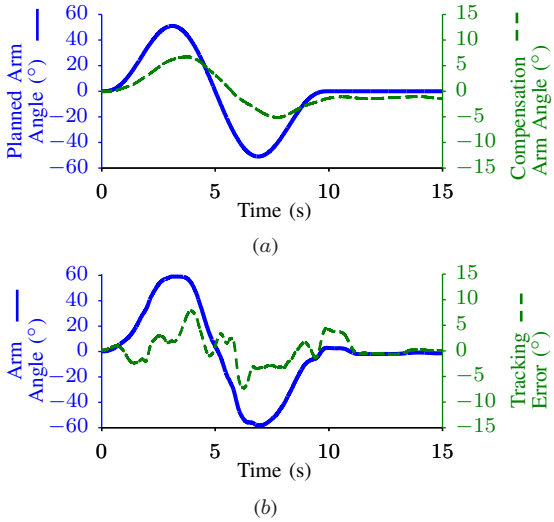


Fig. 12. Tracking a desired straight line ball motion using only arm motions: (a) Planned and compensation left arm angle trajectories, (b) The resulting left arm angle trajectory and the tracking error in achieving the desired left arm angle trajectory, which is a sum of the planned and compensation left arm angle trajectories.

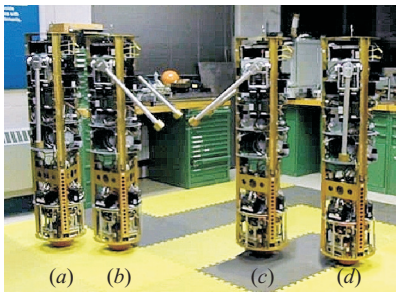


Fig. 13. Composite frames from a video of the ballbot achieving the forward ball motion using only arm motions: (a) the robot starts at rest; (b) the arms are moved forward to accelerate; (c) the arms are moved backward to decelerate; and (d) the robot comes to rest.

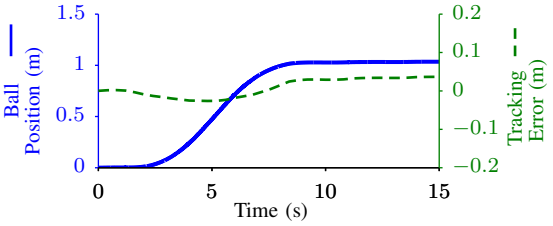


Fig. 14. Pure Arm Motion - Tracking a desired lateral ball motion.

motion using just the arms. For both forward and lateral motions, the compensation arm angles remained within $\pm 5°$ and the balancing controller maintained the body angles within $\pm 0.05°$.

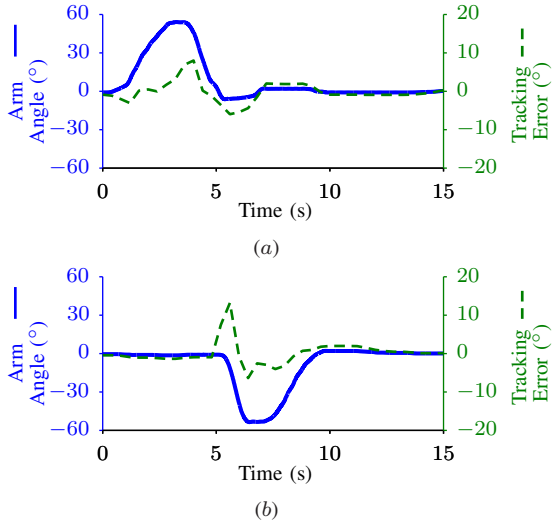


Fig. 15. Tracking a desired lateral ball motion using only arm motions: (*a*) The resulting right arm angle trajectory and the tracking error in achieving its desired trajectory; (*b*) The resulting left arm angle trajectory and the tracking error in achieving its desired trajectory.

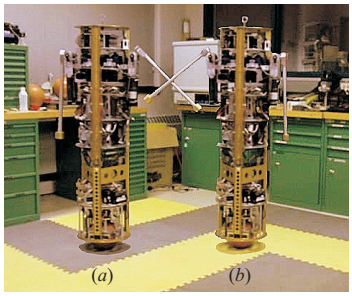


Fig. 16. Composite frames from a video of the ballbot achieving the lateral ball motion using only arm motions: (*a*) the right arm is moved to accelerate; and (*b*) the left arm is moved to decelerate.

C. Arm and Body Motion

Figure 17 shows the ballbot equally using both body and arm motions (50-50) to track a desired 2 m straight line ball motion with a RMSE of 0.04 m. The planned and compensation trajectories for the body angle and the right arm angle are shown in Fig. 18(*a*) and Fig. 19(*a*) respectively. The trajectory tracking performance for

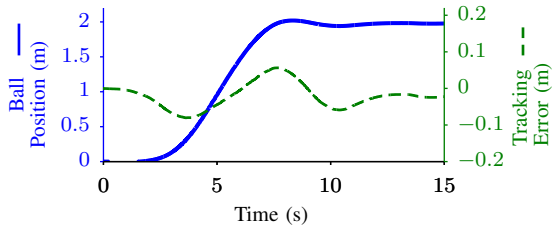


Fig. 17. Arm and Body Motion - Tracking a straight line ball motion.

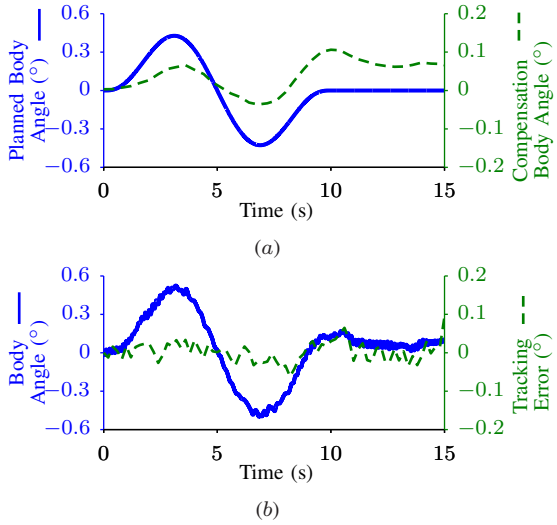


Fig. 18. Tracking a desired straight line ball motion using both body and arm motions: (*a*) Planned and compensation body angle trajectories, (*b*) The resulting body angle trajectory and the tracking error in achieving the desired body angle trajectory, which is a sum of the planned and compensation body angle trajectories.

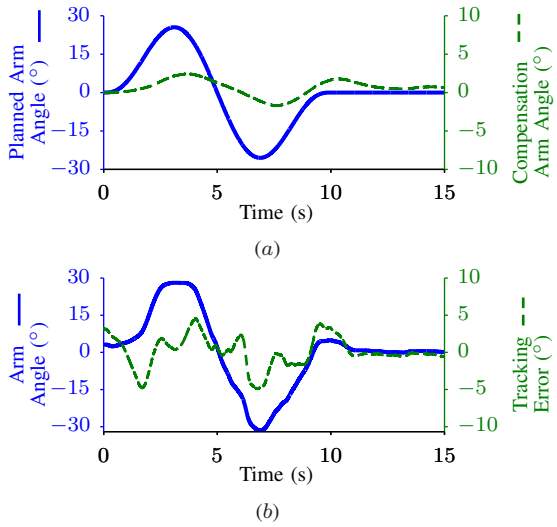


Fig. 19. Tracking a desired straight line ball motion using both body and arm motions: (*a*) Planned and compensation right arm angle trajectories, (*b*) The resulting right arm angle trajectory and the tracking error in achieving the desired arm angle trajectory, which is a sum of the planned and compensation arm angle trajectories.

the body angle and the right arm angle are shown in Fig. 18(*b*) and Fig. 19(*b*) respectively. Similar results were obtained for the left arm. Here, the compensation body angles remained within $\pm 0.06°$, and the compensation arm angles remained within $\pm 5°$.

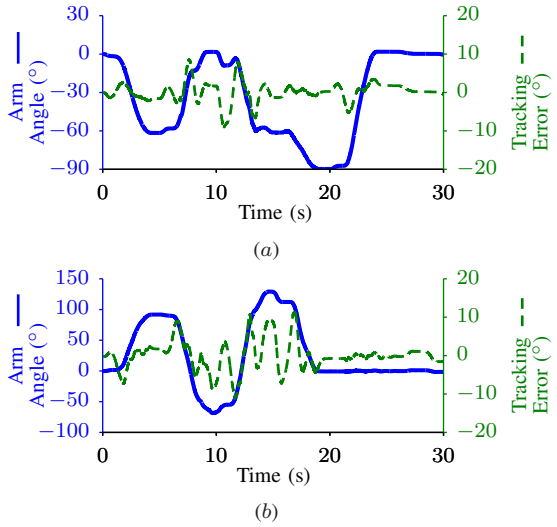


Fig. 20. Tracking the additional arm constraint trajectory: (*a*) X angle for the left arm; (*b*) Y angle for the right arm.

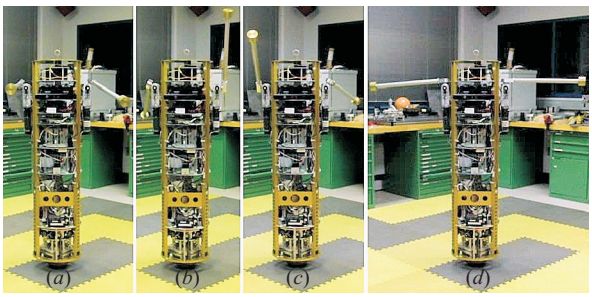


Fig. 21. Selected frames from a video of the ballbot achieving no ball motion while the arms are constrained to trajectories between the four goal configurations (*a*)–(*d*).

D. Constrained Arm Motion

The ballbot with arms was subjected to additional asymmetric constraint trajectories for the arm angles shown in Figs. 20. These arm motions were chosen to be asymmetric so that tracking these arm trajectories will result in the motion of the ball, if not compensated for. Selected frames from a video of the ballbot tracking these constraint trajectories, which consist of four different goal configurations are shown in Fig. 21.

These arm motions were meant to emulate the robot waving its arms randomly. Since the arm angles are constrained to these trajectories, they are unavailable for shape trajectory planning and the shape trajectories were planned only in the space of body angles (Fig. 22) to keep the ball stationary within ±0.04 m of its initial position as shown in Fig. 23.

Figure 24 shows the ballbot with arms tracking a desired straight line motion of 2 m with a RMSE of 0.027 m, while subjected to the additional constraint

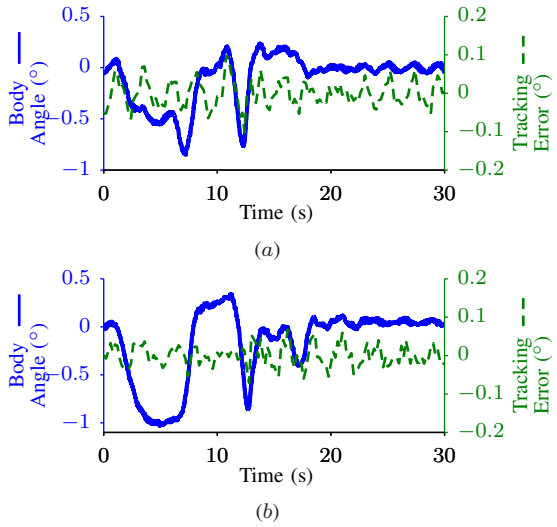


Fig. 22. Tracking a desired body angle trajectory to achieve no ball motion while the arms are constrained: (*a*) X body angle, (*b*) Y body angle.

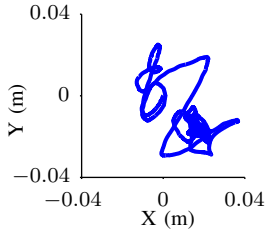


Fig. 23. Constrained Arm Motion - Ball motion while attempting to keep it stationary.

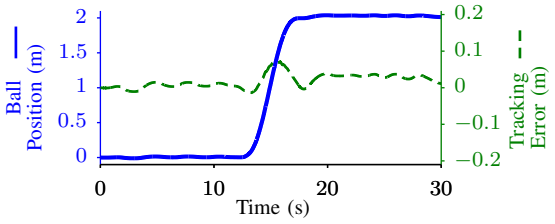


Fig. 24. Tracking a desired straight line ball motion while the arms are constrained to be horizontal.

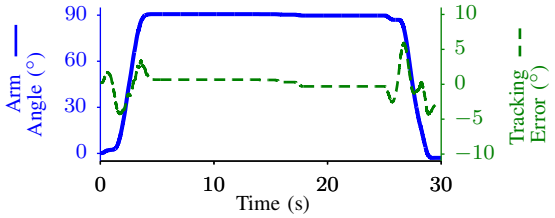


Fig. 25. Tracking the additional constraint trajectory for the left arm.

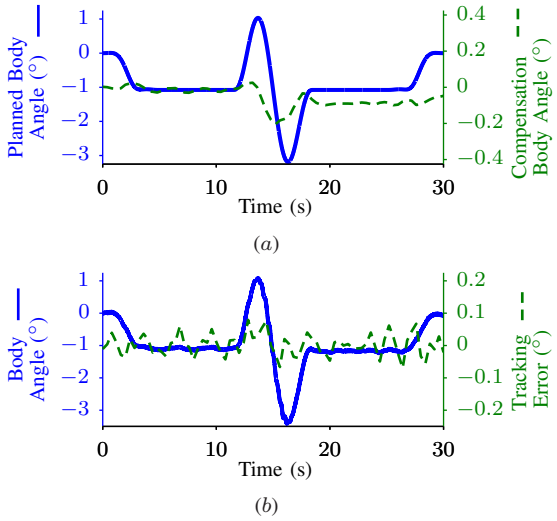


Fig. 26. Tracking a desired straight line ball motion while the arms are constrained to be horizontal: (a) Planned and compensation body angle trajectories, (b) The resulting body angle trajectory and the tracking error in achieving the desired body angle trajectory, which is a sum of the planned and compensation body angle trajectories.

of holding both its arms horizontally forward (90°) as shown in Fig. 25. The constraint arm trajectory consists of three motions, namely, moving the arm from 0° to 90° in the forward direction, holding it at 90° while completing the ball motion of 2 m and finally, moving the arm back from 90° to 0°. This experiment emulates the robot navigating while carrying an object.

The planned and compensation body angle trajectories are shown in Fig. 26(a) and the desired body angle tracking performance is shown in Fig. 26(b). As shown in Fig. 26, the body has to lean back to compensate for the forward held arms, and has to lean forward and backward about this angle to achieve the desired 2 m ball motion. Composite frames from a video of the ballbot performing this motion is shown in Fig. 27.

VI. Conclusions

This paper introduced shape-accelerated balancing systems as a special class of underactuated systems to which balancing mobile robots like the ballbot belong. This paper presented a shape trajectory planner for such systems that uses just the dynamic constraint equations to plan shape trajectories, which when tracked will result in approximate tracking of desired position trajectories. The shape trajectory planning was reduced to an optimization problem of finding a time-invariant linear map (constant gain matrix) that transforms the desired acceleration trajectory in the position space to a planned shape trajectory such that the sum of squared error between

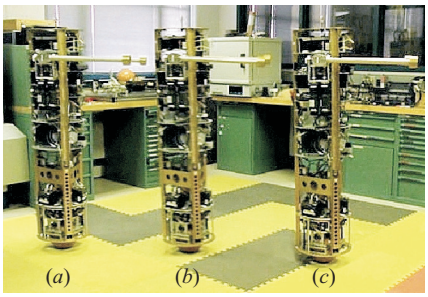


Fig. 27. Composite frames from a video of the ballbot achieving a forward ball motion while the arms are constrained to be horizontal: (a) the body is leaned back to compensate for the arm constraint and then is leaned forward to accelerate; (b) the body is leaned farther back to decelerate; and (c) the robot comes to rest while the body continues to lean back to compensate for the arm constraint.

the acceleration trajectory resulting from tracking the planned shape trajectory and the desired acceleration trajectory in the position space is minimized. The shape trajectory planner was also extended to handle additional constraints on a subset of the shape configurations. The shape trajectory planner was shown to generate feasible state trajectories for shape-accelerated balancing systems at significantly faster speeds (28−72 times) than the trajectory optimization algorithms that use direct collocation methods (Table II).

A feedback position trajectory tracking controller was used in parallel with the shape trajectory planner to achieve better tracking of desired position trajectories. Successful experimental results on the ballbot with arms were presented. The ballbot successfully tracked desired ball motions by tracking pure body motions, pure arm motions, their combinations, and also handled additional constraints on the arms.

VII. Future Work

One needs to evaluate the robustness of the proposed shape trajectory planner and the control architecture to large disturbances. A navigation framework that uses the shape trajectory planner and the control architecture to achieve desired navigation tasks has been presented in (Nagarajan et al. 2013). Several approaches towards automatically choosing weight matrices for a given navigation task can also be explored.

The shape trajectory planner presented in this paper can be extended to plan for a combination of manipulation and navigation tasks. Section V-D presented experimental results of the ballbot successfully achieving desired navigation tasks using only body lean motions while its arms were restricted to additional constraint trajectories, which emulated manipulation tasks. However,

no manipulation tasks were performed. For balancing mobile robots like the ballbot, the navigation and manipulation tasks are tightly coupled. One of the challenges is that the weight of the object that is manipulated plays a significant role in the robot's balance and in its navigation. A state estimator that actively estimates the position of the net center of gravity of the robot and the object is essential for successfully performing such tasks. The shape trajectory planner will have to take into account the manipulation trajectories, and also the dynamics associated with the object's motions. Another challenge is that the robot's body lean motions will affect its manipulation trajectories as the robot will lean with its arms attached to its body. Therefore, both the manipulation planner and the shape space planner must be coupled to successfully achieve both navigation and manipulation tasks.

Acknowledgments

The authors thank George Kantor and Howie Choset for the many useful discussions we have had on this topic. The authors owe great thanks to Byungjun Kim for designing and building the arms for the ballbot and also developing controllers for the same. This work was supported in part by NSF Grants IIS-0308067 and IIS-0535183. The authors also thank the reviewers whose valuable comments and suggestions helped improve the paper significantly.

References

P. Deegan, B. Thibodeau, and R. Grupen. Designing a self-stabilizing robot for dynamic mobile manipulation. *Robotics: Science and Systems - Workshop on Manipulation for Human Environments*, 2006.

P. Deegan, R. Grupen, A. Hanson, E. Horrell, S. Ou, E. Riseman, S. Sen, B.Thibodeau, A. Williams, and D. Xie. Mobile manipulators for assisted living in residential settings. *Autonomous Robots*, 2007.

S. Devasia and B. Paden. Exact output tracking for nonlinear time-varying systems. In *IEEE International Conference on Decision and Control*, volume 3, pages 2346–2355, 1994.

S. Devasia, D. Chen, and B. Paden. Nonlinear inversion-based output tracking. In *IEEE Transactions on Automatic Control*, volume 41, pages 930–942, 1996.

N. Getz. Control of balance for a nonlinear nonholonomic non-minimum phase model of a bicycle. In *American Control Conference*, pages 148 – 151, 1994.

N. Getz and J. K. Hedrick. An internal equilibrium manifold method of tracking for nonlinear nonminimum phase systems. In *American Control Conference*, pages 2241–2245, 1995.

N. H. Getz. Tracking with balance. In *13th IFAC Triennial World Congress*, San Francisco, USA, 1996.

P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–132, 2005.

C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance*, 10(4):338–342, 1987.

R. Hatton and H. Choset. Connection vector fields for underactuated systems. In *Proc. IEEE/RAS-EMBS Int'l Conf. on Biomedical Robotics and Biomechatronics*, pages 451–456, 2008.

L. Havasi. ERROSphere: an equilibrator robot. pages 971–976, 2005.

S. Hirose. *Biologically Inspired Robots: Snake-like Locomotors and Manipulators*. Oxford University Press, Oxford, 1993.

R. Hollis. Ballbots. *Scientific American*, pages 72–78, Oct 2006.

A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 1989.

A. Isidori and C. I. Byrnes. Output regulation of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(2):131–140, 1990.

M. Kumagai and T. Ochiai. Development of a robot balancing on a ball. *Intl. Conf. on Control, Automation and Systems*, 2008.

M. Kumagai and T. Ochiai. Development of a robot balancing on a ball - application of passive motion to transport. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, pages 4106–4111, 2009.

T. Lauwers, G. Kantor, and R. Hollis. One is enough! In *Proc. Int'l. Symp. for Robotics Research*, Oct. 2005.

T. B. Lauwers, G. A. Kantor, and R. L. Hollis. A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, pages 2884–2889, 2006.

K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *The Quarterly of Applied Mathematics*, 2:164–168, 1944.

A. Lewis, J. Ostrowski, R. Murray, and J. Burdick. Nonholonomic mechanics and locomotion: The snakeboard example. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 2391–2397, 1994.

J. E. Marsden and M. J. Hoffman. *Elementary classical analysis*. W.H. Freeman, 1993.

R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Berkeley, 1994.

U. Nagarajan. Dynamic constraint-based optimal shape trajectory planner for shape-accelerated underactuated

balancing systems. In *Proc. Robotics: Science and Systems*, 2010.

U. Nagarajan. *Fast and Graceful Balancing Mobile Robots*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2012. CMU-RI-TR-12-16.

U. Nagarajan, G. Kantor, and R. Hollis. Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot. In *IEEE Int'l. Conf. on Robotics and Automation*, pages 3743–3748, 2009a.

U. Nagarajan, G. Kantor, and R. Hollis. Human-robot physical interaction with dynamically stable mobile robots. *4th ACM/IEEE Int'l. Conf. on Human-Robot Interaction*, 2009b. (Short paper and video).

U. Nagarajan, A. Mampetta, G. Kantor, and R. Hollis. State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot. In *IEEE Int'l. Conf. on Robotics and Automation*, pages 998–1003, 2009c.

U. Nagarajan, B. Kim, and R. Hollis. Planning in high-dimensional shape space for a single-wheeled balancing mobile robot with arms. In *IEEE Int'l Conf. on Robotics and Automation*, pages 130–135, 2012.

U. Nagarajan, G. Kantor, and R. Hollis. Integrated motion planning and control for graceful balancing mobile robots. *The International Journal of Robotics Research, Special Issue on Motion Planning for Physical Robots*, 2013.

Y. Nakamura. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1964.

H. G. Nguyen, J. Morrell, K. Mullens, A. Burmeister, S. Miles, N. Farrington, K. Thomas, and D. Gage. Segway robotic mobility platform. In *SPIE Proc. 5609: Mobile Robots XVII*, Philadelphia, PA, 2004.

R. Olfati-Saber. Nonlinear control and reduction of underactuated systems with symmetry II: Unactuated shape variables case. In *Proc. 40th IEEE Conference on Decision and Control*, pages 4164–4169, 2001.

G. Oriolo and Y. Nakamura. Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators. In *Proc. IEEE Conf. on Decision and Control*, pages 2398–2403, 1991.

J. Ostrowski and J. Burdick. Geometric perspectives on the mechanics and control of robotic locomotion. In *Proc. Int'l Symp. on Robotics Research*, pages 487–504, 1995.

J. Ostrowski and J. Burdick. The geometric mechanics of undulatory robotic locomotion. *International Journal of Robotics Research*, 17:683–701, 1996.

J. P. Ostrowski. Computing reduced equations for robotic systems with constraints and symmetries. *IEEE Trans. on Robotics and Automation*, 15(1):111 –123, 1999.

Rezero. http://www.rezero.ethz.ch, 2010.

Per Rutquist and M. M. Edvall. *PROPT - Matlab Optimal Control Software*. Tomlab Optimization Inc., Pullman, WA, USA, 2010.

E. Shammas, H. Choset, and A. Rizzi. Towards automated gait generation for dynamic systems with non-holonomic constraints. In *Proc. IEEE Int'l. Conf. on Robotics and Automation*, pages 1630–1636, 2006.

E. A. Shammas, K. Schmidt, and H. Choset. Natural gait generation techniques for purely mechanical systems. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 3664–3669, 2005.

E. A. Shammas, H. Choset, and A. A. Rizzi. Towards a unified approach to motion planning for dynamic underactuated mechanical systems with non-holonomic constraints. *International Journal of Robotics Research*, 26:1075–1124, 2007a.

E. A. Shammas, H. Choset, and A. A. Rizzi. Geometric motion planning analysis for two classes of underactuated mechanical systems. *International Journal of Robotics Research*, 26:1043–1073, 2007b.

M. W. Spong. The control of underactuated mechanical systems. In *First Int'l Conference on Mechatronics*, Mexico City, 1994.

M. Stilman, J. Olson, and W. Gloss. Golem Krang: Dynamically stable humanoid robot for mobile manipulation. In *IEEE Int'l Conf. on Robotics and Automation*, pages 3304–3309, 2010.

Y. Takahashi, S. Ogawa, and S. Machida. Step climbing using power assist wheel chair robot with inverse pendulum control. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 1360–65, 2000.

O. von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control, Int'l Series in Numerical Mathematics*, pages 129–143, 1993.

O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37(1):357–373, 1992.

International Journal of Robotics Research (IJRR)
Volume 32, Issue 11, pp. 1323 – 1341, September 2013

APPENDIX A: INDEX TO MULTIMEDIA EXTENSIONS

The multimedia extensions to this article are at: www.ijrr.org.

Extension	Type	Description
1	Video	This video shows the ballbot achieving several desired motions in the position space using pure body motions, pure arm motions, and combinations of the two. It also shows the ballbot achieving desired ball motions while the arms are constrained.