Humanoid Gymnastics

Tuen Yue Tsui
University of Pennsylvania
tytsui@seas.upenn.edu

Puen Xu
University of Pennsylvania
puenxu@seas.upenn.edu

***Abstract*—Inspired by gymnastics, this work focuses on developing a sequence of motions for a humanoid robot: walking, transitioning to running, leaping to grasp a horizontal bar, and performing a swing-up as if executed by a skilled gymnast. Due to time constraints, we develop and analyze each motion individually, leaving the integration of these movements for future work. The codes are available at github.**

I. Introduction

Gymnastics has inspired the development of new robots that mimic human movements. For instance, an acrobot is a simplified model of a gymnast on a bar, where the passive first joint represents the gymnast's hands, and the actuated second joint models the hip. [1] Another example is that Hyon et al. developed a gymnastic robot capable of performing various floor exercises, including jumping, somersaults, and back handsprings. [2] Gymnastics also facilitates the implementation of new algorithms for robots to perform dynamic maneuvers. Hodgins and Raibert examined the dynamics of the forward flip and demonstrated how a planar biped running machine can execute this gymnastic maneuver. [3]

Inspired by gymnastics, in this paper, we focus on developing a sequence of motions for a humanoid robot: walking, transitioning to running, leaping to grasp a horizontal bar, and performing a swing-up as if executed by a skilled gymnast. Due to time constraints, we develop and analyze each motion individually, leaving deliverables on the integration of these movements.

II. Background

A. Humanoid Robot Model

Our work focuses on a planar seven-link humanoid robot as depicted in Fig. 5 (left). The generalized coordinates are defined as $[x, z, \theta, q_1, q_2, q_3, q_4, q_5, q_6]^T \in \mathbb{R}^9$, where (x, z, θ) represents the robot position and orientation in the xz-plane, q_1, q_3, and q_5 are the shoulder, hip and knee joint angles on the left side, respectively, and q_2, q_4, and q_6 correspond to the joint angles on the opposite side. The state vector is thus given by $[q, v]^T \in \mathbb{R}^{18}$ where $v = \dot{q}$. The robot has actuators on each of the six joints, and the actuator input space is denoted as $u = [u_1, u_2, u_3, u_4, u_5, u_6]^T \in \mathbb{R}^6$.

In the swing-up setting, the robot hands are attached to a horizontal bar as shown in Fig. 5 (right). The robot thus no longer possesses the floating base and loses 3 Degrees of Freedom in position and orientation in the xz-plane. In the meantime, the robot arms form a closed kinematic chain and loses 1 DoF of motion. Furthermore, we need to consider the angle between the robot arm and the horizontal bar in the plane, which adds 1 DoF to the model. As a result, the generalized coordinates becomes $[q_1, q_2, q_3, q_4, q_5, q_6]^T \in \mathbb{R}^6$. Noticing that although the robot has 6 actuators, both shoulder actuators are applying torque on the same DoF, reducing one dimension of the input space. Accordingly, u is given by $u = [u_1, u_2, u_3, u_4, u_5]^T \in \mathbb{R}^5$.

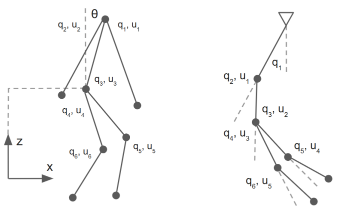

Fig. 1. Humanoid Robot Model

B. Operational Space Control

In operational space control (OSC), a robot is designed to execute desired trajectories or forces in the task space coordinate system. For example, OSC involves controlling the position and orientation of the end-effector of a robot arm as well as the center of gravity of a legged robot. [4]

The general problem in OSC for trajectory tracking involves designing a control policy $u = f_{\text{Control}}(q, \dot{q}, x_d, \dot{x}_d, \ddot{x}_d)$ that controls the robot along a joint space trajectory $q(t), \dot{q}(t)$ such that the system tracks a desired trajectory in task space $x_d(t), \dot{x}_d(t), \ddot{x}_d(t)$. [4]

C. Energy Shaping Control

Energy shaping control is used on underactuated systems such as acrobots and cart-pole systems. It involves using partial feedback linearization (PFL) to simplify or cancel out terms in the nonlinear dynamics equations and certain Lyapunov function to prove stability. [5]

To establish swing-up control of an acrobot, Spong used collocated PFL to simplify the dynamics,

$$m_{11}\ddot{q}_1 + h_1 + \phi_1 = -m_{12}u, \quad u = -k_2\dot{q}_2 - k_1 q_2 - \bar{u}$$

then chose addition control $\bar{u}$ to swing up the second link, where $\bar{u} = \tilde{E}\dot{q}_1$, $\tilde{E} = E - E_d$, and E_d represents the desired potential energy at the upright state of the acrobot. After the acrobot enters a state within a neighborhood of the upright configuration, a separate controller is used that achieves local asymptotic stability to the desired equilibrium. [6]

III. METHODS

In this work, we use operational space control for the walking, leaping and running gaits of the biped robot. In the following three subsections, we illustrate our method to obtain desired center of mass trajectory and swing-foot trajectory per each gait for the OSC to track. In the last subsection, we discuss applying energy shaping control for the swing-up maneuver of the humanoid.

A. Walking

For walking, we built upon the HW6 code, which uses an angular momentum linear inverted pendulum controller that generates footstep and center of mass trajectories to be tracked by the OSC. [7] We updated the robot's URDF to include arms and adjusted the torso angle tracking gain to account for changes in the upper body to establish stable walking of the humanoid robot.

Furthermore, to mimic human behaviors, we designed a feedback linearization controller to ensure that the robot's arms move in sync with the opposite side leg's motion (e.g., the left arm and right leg follow the same motion, and vice versa). Given that the humanoid robot is underactuated, we cannot design a feedback linearization controller directly. Alternatively, we used a "fake" plant model that retains the actuated arm dynamics but removes the unactuated floating base. For arm swinging, the fully underactuated humanoid robot is simulated, but within each simulation loop, the control input for the arms is computed using the fully-actuated model.

B. Leaping

Leaping is the prerequisite for many visually striking gymnastic actions. It is crucial for running and any other movements that involve a flight-then-land phase. The first challenge of these kinds of action is the involvement of impact dynamics. In our methods, we simplify the ground dynamics with a spring damper system. Another challenge presented is the discrete trajectory optimization across different states, as different finite states have different constraints.

1) Finite State Machine: We use four phases to describe the leaping action. The humanoid will first follow the stance phase of walking (lifting one leg), crouch to store potential energy, leap, and finally reach the ground after a short flight phase.

TABLE I
OPERATIONAL SPACE CONTROL PARAMETERS FOR LEAPING

State	Translational Jacobian	Friction	Impact	Contact
Crouch	$\mathbf{J}_v < 0$	Yes	No	Yes
Stance	$\mathbf{J}_v > 0$	Yes	No	Yes
Flight	$\mathbf{J}_v = 0$	No	No	No
Land	$\mathbf{J}_v < 0$	Yes	Yes	Yes

2) Desired Foot Placement: We implemented the Raibert Algorithm [8] to calculate the desired foot placement. The algorithm is defined as follows:

$$p_{\text{foot}} = p_{\text{com}} + k_{\text{raibert}}(v_{\text{com}} - v_{\text{desired}}).$$

where p_{com} is the position of center of mass, k_{raibert} is the raibert constant, v_{com} represents the velocity of center of mass, and v_{desired} is the desired velocity.

3) Impact Dynamics: Since ground dynamics is highly complex, we used a spring-damper system to approximate and simulate the ground impact forces experienced during landing. The impact force is modeled as a combination of a spring force proportional to the penetration depth and a damping force proportional to the relative velocity of the contact point along the normal direction to the ground. This is expressed mathematically as follows:

$$F_{\text{impact}} = k_{\text{spring}} \cdot \delta + d_{\text{damper}} \cdot v_{\text{relative}}.$$

where:

- F_{impact} is the total impact force.
- k_{spring} is the spring constant that governs the stiffness of the virtual ground.
- $\delta = \max(0, -p_z)$ represents the penetration depth, calculated as the negative z coordinate of the position of the foot of the position of the foot, clamped to zero to ensure no tensile force.
- d_{damper} is the damping coefficient that controls the energy dissipation rate during impact.
- v_{relative} is the relative velocity of the stance foot in the vertical (z) direction.

The stance foot velocity is computed using the translational Jacobian J_v at the contact point, such that $v_{\text{stance}} = J_v \cdot v$, where v represents the generalized velocities of the system. The relative velocity in the normal direction is then extracted as $-v_{\text{stance},z}$.

To apply the calculated impact force to the system, the change in velocity (Δv) is computed using the inverse dynamics mass matrix M and the force projection via the Jacobian transpose:

$$\Delta v = M^{-1} \cdot J_v^\top \cdot F_{\text{impact}}.$$

Finally, the updated velocities of the system are computed as:

$$v_{\text{new}} = v + \Delta v.$$

This approach provides a computationally efficient approximation of ground impact dynamics while maintaining physically plausible behaviors.

4) The Center of Mass Trajectory:

a) Crouch Phase: The crouch phase models the lowering of the center of mass (CoM) trajectory. During this phase, the CoM height decreases smoothly, preparing the system for the subsequent stance and flight phases. The trajectory during this phase is calculated as a linear interpolation between the end position and the begin position.

b) Stance Phase: During the stance phase, we apply the Spring-Loaded Inverted Pendulum (SLIP) model [9] to describe the CoM dynamics. The relative position and velocity of the CoM with respect to the stance foot are expressed in polar coordinates:

$$r = \sqrt{(\Delta x)^2 + (\Delta z)^2}, \quad \theta = \arctan 2(\Delta x, \Delta z),$$

$$\dot{r} = \frac{\Delta x \cdot v_x + \Delta z \cdot v_z}{r}, \quad \dot{\theta} = \frac{\Delta x \cdot v_z - \Delta z \cdot v_x}{r^2}.$$

We calculate the radial and angular accelerations according to the SLIP dynamics:

$$\ddot{r} = r\dot{\theta}^2 - \frac{k}{m}(r - l_0) - \frac{c}{m}\dot{r} - g\cos(\theta), \quad \ddot{\theta} = -\frac{2\dot{r}\dot{\theta}}{r}.$$

where k is the spring constant, c is the damper constant, and l_0 represents the length of legs.
Velocities are updated incrementally as:

$$\dot{r}_{i+1} = \dot{r}_i + \ddot{r}_i \cdot \Delta t, \quad \dot{\theta}_{i+1} = \dot{\theta}_i + \ddot{\theta}_i \cdot \Delta t,$$
$$r_{i+1} = r_i + \dot{r}_i \cdot \Delta t, \quad \theta_{i+1} = \theta_i + \dot{\theta}_i \cdot \Delta t.$$

These are then converted to Cartesian coordinates:

$$x = x_{\text{foot}} + r\sin(\theta), \quad z = z_{\text{foot}} + r\cos(\theta),$$
$$\dot{x} = \dot{r}\sin(\theta) + r\dot{\theta}\cos(\theta), \quad \dot{z} = \dot{r}\cos(\theta) - r\dot{\theta}\sin(\theta).$$

The stance phase trajectory is discretized into 100 time steps, and a piecewise cubic Hermite polynomial is used to interpolate the CoM positions and velocities.

c) Flight phase: The flight phase follows a ballistic trajectory dictated by gravity. The CoM position and velocity evolve as:

$$x(t) = x_0 + v_{x0} \cdot t, \quad y(t) = y_0 + v_{y0} \cdot t,$$
$$z(t) = z_0 + v_{z0} \cdot t - \frac{1}{2}gt^2.$$

The velocity updates are:

$$v_x(t) = v_{x0}, \quad v_y(t) = v_{y0}, \quad v_z(t) = v_{z0} - g \cdot t.$$

The CoM trajectory at the end of the flight phase is:

$$\mathbf{p}_{\text{end}} = [x_0 + v_{x0}T, y_0 + v_{y0}T, \max(z_0, z_0 + v_{z0}T - 0.5gT^2)],$$
$$\mathbf{v}_{\text{end}} = [v_{x0}, v_{y0}, v_{z0} - g \cdot T].$$

The positions and velocities are interpolated using a piecewise cubic Hermite polynomial to ensure a smooth transition between the phases.

d) Landing Phase: The landing phase assumes a negligible change in CoM position due to the short impact duration (0.001 seconds in our method). Only the velocity is updated to account for the ground reaction force. The impact force is calculated as:

$$F_{\text{impact}} = k_{\text{spring}} \cdot \delta + d_{\text{damper}} \cdot v_{\text{relative}},$$

where δ is the penetration depth and v_{relative} is the velocity of the contact point relative to the ground. Using the mass matrix M and Jacobian J_v, the velocity change is:

$$\Delta v = M^{-1} J_v^\top F_{\text{impact}}.$$

The updated CoM velocity becomes:

$$v_{\text{com, after}} = v_{\text{com, before}} + J_{\text{com}} \Delta v.$$

The position remains constant, while the velocities are interpolated for smooth continuity using a piecewise cubic Hermite polynomial.

5) Swing Foot Trajectory: The swing foot trajectory generation is categorized into three primary phases: flight, landing, and stance. Each phase incorporates specific control strategies to achieve smooth and efficient foot placement.

a) Crouch Phase: We assume no swing foot displacement during the crouch phase, so the foot position remain the same as beginning.

b) Stance Phase: During the stance phase, the swing foot remains stationary relative to the stance leg. The current swing foot position $\mathbf{Y}_0$ is updated to the desired foot placement $\mathbf{Y}_2$, which is calculated by Raibert Algorithm [8] through an intermediate position $\mathbf{Y}_1$:

$$\mathbf{Y}_1 = \frac{\mathbf{Y}_0 + \mathbf{Y}_2}{2}, \quad \mathbf{Y}_1[2] = \mathbf{Y}_0[2] + h_{\text{clearance}}.$$

The velocity at the start ($\mathbf{v}_0$), midpoint ($\mathbf{v}_{\text{mid}}$), and end ($\mathbf{v}_f$) are defined as:

$$\mathbf{v}_{\text{mid}} = \frac{\mathbf{Y}_2 - \mathbf{Y}_0}{t_{\text{end}} - t_{\text{start}}},$$

where $\mathbf{v}_0 = [0, 0, 0]$ and $\mathbf{v}_f = [0, 0, 0]$. The trajectory is interpolated using cubic Hermite polynomials to achieve smooth transitions.

c) Flight Phase: During the flight phase, the swing foot transitions from the liftoff position $\mathbf{Y}_0$ to the desired foot placement $\mathbf{Y}_2$, passing through an intermediate point $\mathbf{Y}_1$. The intermediate point is defined as:

$$\mathbf{Y}_1 = \frac{\mathbf{Y}_0 + \mathbf{Y}_2}{2}, \quad \mathbf{Y}_1[2] = \mathbf{Y}_0[2] + h_{\text{clearance}},$$

where $h_{\text{clearance}}$ is the foot clearance height to avoid obstacles. The desired foot placement $\mathbf{Y}_2$ is calculated using the desired velocity and Raibert's heuristic [8]. The initial, intermediate, and final velocities ($\mathbf{v}_0$, $\mathbf{v}_{\text{mid}}$, $\mathbf{v}_f$) are determined as:

$$\mathbf{v}_{\text{mid}} = \frac{\mathbf{v}_0 + \mathbf{v}_f}{2},$$

where $\mathbf{v}_0$ is the velocity at liftoff, and $\mathbf{v}_f = [v_{x,f}, v_{y,f}, v_{z,f}]$. The swing foot trajectory is also constructed using cubic Hermite polynomials.

d) Landing Phase: The landing phase assumes the swing foot maintains its position at the computed desired foot placement. Due to the negligible duration of impact (0.001 seconds), the position remains constant, while the velocity is updated post-impact based on the ground reaction force. A zero-order hold is employed to maintain the swing foot at $\mathbf{Y}_2$.

C. Running

Similar to leaping, running also involves a flight phase and an impact phase. Therefore, the implementation on these states are almost identical.

1) Finite State Machine: We used six phases to describe a running cycle.

2) Trajectory Optimization: We follow the same trajectory optimization pipeline for both center of mass and swing foot as leaping.

TABLE II
OPERATIONAL SPACE CONTROL PARAMETERS FOR RUNNING

State	Jacobian	Friction	Impact	Contact
Left Stance	$\mathbf{J}_v > 0$	Yes	No	Yes
Flight After Left	$\mathbf{J}_v = 0$	No	No	No
Impact After Left	$\mathbf{J}_v < 0$	Yes	Yes	Yes
Right Stance	$\mathbf{J}_v > 0$	Yes	No	Yes
Flight After Right	$\mathbf{J}_v = 0$	No	No	No
Impact After Right	$\mathbf{J}_v < 0$	Yes	Yes	Yes

D. *Swing-up*

Given time constraints, we simplified the robot model b treating it as an acrobot, assuming that the leg joints can loc and remain straight. Using this model, we applied partial feed back linearization and energy shaping control, as outlined i Spong's paper, to swing the robot toward the upright positio To stabilize the robot at the upright position, we designed a LQR controller that linearizes the dynamics around this poi to compute the cost-to-go matrix. When the cost-to-go valu falls below a specified threshold, the controller switches to the LQR. Last but not least, the consideration of using the full humanoid dynamics is discussed in the future work section.

IV. RESULTS

The simulations for each motion are shown below. The full video can be found via this link.

A. *Walking*

We simulate the walking gait in Drake with an initial configuration of $q_0 = [0, 0.8, 0, 0, 0, \theta, \theta, -2\theta, -2\theta]$, where $\theta = -\arccos(0.8)$ for 5 seconds.

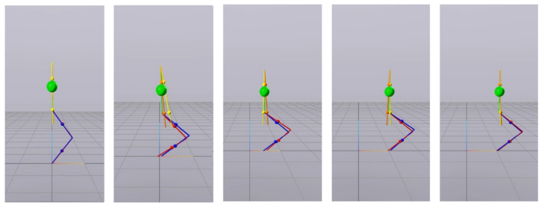
Fig. 2. Humanoid Walking

B. *Leaping*

We follow the same configuration as walking.

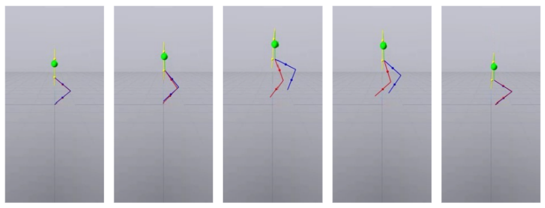
Fig. 3. Humanoid Leaping

C. *Running*

We follow the same configuration as walking. Although we are still working on running, we have the desired CoM trajectory as shown in Fig. 4.

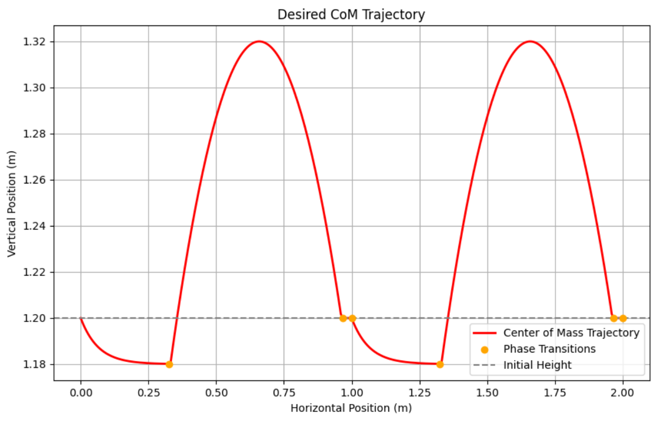

Fig. 4. CoM Trajectory

D. *Swing-up*

We simulate the swing-up motion in Drake with an initial configuration of $q_0 = [0, 0, 0, 0, 0, 0]$ for 20 seconds.

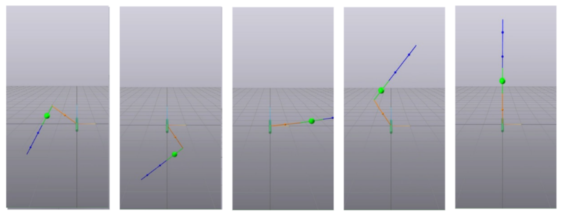
Fig. 5. Humanoid Swing-Up

V. FUTURE WORK

A. *Humanoid Swing-up*

For swing-up, instead of modeling the humanoid as an acrobot, we could adopt the torque-coupled four-link control approach described in the Zhang et al. paper to take advantage of the underactuated system's dynamics. [1]

B. *Motion Integration*

For running, future work may include combining SLIP dynamics with whole body control. To integrate leaping and swing-up, a strategy must be developed for the humanoid to jump and grasp the horizontal bar, potentially using a claw mechanism to facilitate the transition. One possible approach is to solve an optimization problem that minimizes the distance between the robot's hands and the bar (e.g., uses OSC to control the robot hand move toward horizontal bar). Once the distance falls below a specified threshold, the claw mechanism would be programmed to close and secure the bar.

References

[1] A. Zhang, J. Qiu, C. Yang, and H. He, “Stabilization of underactuated four-link gymnast robot using torque-coupled method,” *International Journal of Non-Linear Mechanics*, vol. 77, pp. 299–306, 2015.

[2] S.-H. Hyon, N. Yokoyama, and T. Emura, “Back handspring control of a multi-link gymnastic robot,” *IFAC Proceedings Volumes*, vol. 37, no. 14, pp. 73–78, 2004.

[3] M. H. Raibert and J. Hodgins, “Biped gymnastics,” *Dynamically Stable Legged Locomotion*, vol. 79, 1988.

[4] J. Peters and S. Schaal, “Learning operational space control.” in *Robotics: Science and Systems*, vol. 10, 2006.

[5] R. Tedrake, *Underactuated Robotics*, 2023. [Online]. Available: https://underactuated.csail.mit.edu

[6] M. W. Spong, “Energy based control of a class of underactuated mechanical systems,” *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 2828–2832, 1996.

[7] Y. Gong and J. Grizzle, “Angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-based controller,” *arXiv preprint arXiv:2008.10763*, 2020.

[8] M. Raibert, *Legged Robots that Balance*, ser. Artificial Intelligence Series. MIT Press, 1986. [Online]. Available: https://books.google.com/books?id=EXRiBnQ37RwC

[9] R. Blickhan, “The spring-mass model for running and hopping,” *Journal of Biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, 1989.