biomimetics

MDPI

Article

Whole-Body Dynamics-Based Aerial Fall Trajectory Optimization and Landing Control for Humanoid Robot

Weilong Zuo [1,2], Junyao Gao [1,2,*], Jingwei Cao [1,2], Xilong Xin [1,2], Mingyue Jin [1,2] and Xuechao Chen [1,2]

[1] School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 100081, China
[2] Beijing Advanced Innovation Center for Intelligent Robotics and Systems, Beijing Institute of Technology, Beijing 100081, China
* Correspondence: gaojunyao@bit.edu.cn

Abstract: When humanoid robots work in human environments, falls are inevitable due to the complexity of such environments. Current research on humanoid robot falls has mainly focused on falls on the ground, with little research on humanoid robots falling from the air. In this paper, we employ an extended state variable formulation that directly maps from the high-level motion strategy space to the full-body joint space to optimize the falling trajectory in order to protect the robot when falling from the air. In order to mitigate the impact force generated by the robot's fall, during the aerial phase, we employ simple proportion differentiation (PD) control. In the landing phase, we optimize the optimal contact force at the contact point using the centroidal dynamics model. Based on the contact force, the changes to the end-effector positions are solved using a dual spring–damper model. In the simulation experiments, we conduct three comparative experiments, and the simulation results demonstrate that the robot can safely fall 1.5 m from the ground at a pitch angle of 45°. Finally, we experimentally validate the methods on an actual robot by performing a side-fall experiment. The experimental results show that the proposed trajectory optimization and motion control methods can provide excellent shock absorption for the impact generated when a robot falls.

Keywords: fall; trajectory optimization; control; humanoid robot

Citation: Zuo, W.; Gao, J.; Cao, J.; Xin, X.; Jin, M.; Chen, X. Whole-Body Dynamics-Based Aerial Fall Trajectory Optimization and Landing Control for Humanoid Robot. *Biomimetics* **2023**, *8*, 460. https://doi.org/10.3390/biomimetics8060460

Academic Editors: Mingguo Zhao and Biao Hu

Received: 1 August 2023
Revised: 20 September 2023
Accepted: 22 September 2023
Published: 1 October 2023

1. Introduction

Humanoid robots are a tangible manifestation of human technological advancement, making them of great interest to researchers. The world's most advanced humanoid, Atlas, demonstrates flexible motion and manipulation abilities, pointing to the current direction of humanoid robotics development [1]. Business magnate Musk has greatly supported humanoid robotic technology by funding relevant research. As seen in the latest video [2], Optimus can now assist humans in performing tasks in human environments.

At present, humanoid robot research primarily centers on walking, running, jumping, and aiding people in performing duties [3–6]. Due to the complexity of the human environment, robots will inevitably experience falling behavior when completing these tasks. Therefore, designing a structure or control method that can effectively protect robots is very important. According to the law of impulse, when the robot has a certain mass, reducing the impact force of the robot on the ground mainly involves increasing the touchdown time and reducing the landing speed. For the aspect of increasing the ground impact time, Kajita et al. introduced a method using airbags [7]; however, the airbags employed can only be used once and must reinflate before they can be used again, making the method complex and expensive. Similarly, Sung-Hee Lee introduced a way of placing a backpack on the robot's back [8] to change the falling direction after the robot receives an impact, which is suitable for environments wherein surrounding objects do not interfere. In [9], Kakiuchi designed a robot with hard points mounted over its entire body, which looks like a man

wearing a set of shackles and significantly limits the robot's workspace. Inspired by the self-protection of turtles, Nguyen et al. designed a type of shell protector. However, in their latest research, since the robot only had a lower body, the actual fall performance of the whole body has not yet been observed [10]. In the aspect of decreasing the ground impact velocity, researchers use methods for optimizing trajectories and motion control methods. For example, Fujiawara optimized the falling trajectories of robots in different orientations through imitating Ukemi motion by using energy shaping and distribution methods [11]. Luca established a variable-length inverted pendulum model to optimize forward falling trajectories for a robot [12]. In addition, some researchers optimize the the distribution of contact points and the contact sequence, as described in [13,14]. Regarding robot motion control, being able to effectively cushion impact forces is a very important strategy. Mujica proposed a variable admittance control that successfully achieved human–robot collaborative tasks [15]. For the task of adjusting control parameters, ref. [16] proposed a new Lyapunov-based offline self-tuning control method. Similarly to [15,17] presented a compliant adaptive control method that can enable stable and safe physical human–robot interaction during work. In recent research, MIT proposed using model predictive control to mitigate the impact forces of acrobatic robot maneuvers [18]. HaoxiangQi achieved stable control after a robot's high jump landing by optimizing contact forces and using a virtual model [19].

Based on the above references, we can see that current research on robot falls mainly considers robots falling on flat ground. However, research on robots falling from high altitudes is still lacking. Because human exploration desires are endless, people also hope that robots can replace humans to accomplish some high-difficulty actions such as aerial acrobatics and parachuting. But when performing these actions, robots face the risk of falling. Therefore, more research needs to focus on solutions to help robots mitigate the impact of falling from high altitudes in order to achieve complex aerial tasks that are difficult for humans.

Research on falling mitigation for robots mainly focuses on quadruped robots, notably the Mini Cheetah from MIT [20,21]. Research employs offline or online optimization to combine machine learning or neural networks when falling from heights. The results show that by using the method in [20], a robot can fall from a height of 10 feet (about 3 m); while using the method in [21], a robot can fall from around a 2 m height. Another avenue of research is from a bio-inspired perspective, where a tail is added to the quadruped robot to cushion the impact of falling, which is discussed in papers [22,23]. Different from the methods and experimental objectives mentioned above, Francesc proposed an optimization-based reactive landing controller for handling horizontal impacts during falls [24] and showed that a quadruped robot can recover from falls at horizontal speeds up to 3 m/s. In this paper, we adopted a direct mapping relationship from the high-level motion task strategy space to the robot's whole joint space by setting the robot's different state variables and adding the corresponding contact constraint equations. In this method, the high-level action task strategy is utilized as the objective function of the robot, allowing the optimizer to satisfy our desired actions. For instance, when a robot experiences a fall from the air, its high-level motion task strategy space is to minimize contact force, follow a prior trajectory function, and distribute the force evenly across each contact point. Using the optimizer solution method, we optimized the trajectory of a robot falling from the air given the initial and target states. When the robot was in the aerial stage, we used a proportion derivative (PD) method to control the posture of the centroid and endpoints. When the robot was in the landing stage, the target reference force was optimized using centroid dynamics to reduce the impact force by damping control. Figure 1 illustrates the overall framework of our method.

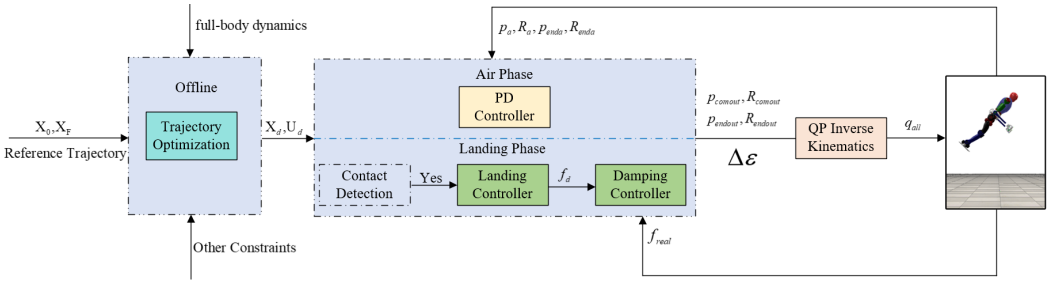

Figure 1. Planning and control framework for a humanoid robot falling from the air. The superscripts "d" and "a" mean the desired value and the actual (measured) value, respectively; X_0 and X_F represent the initial state and the final state, respectively; X_d and U_d represent the expected state and the expected input value, respectively.

The contribution of this work is three-fold:

- An optimized trajectory for the robot falling from the air is generated, enabling the humanoid robot to land smoothly on the ground.
- Direct mapping is adopted from the high-level task strategy space to the joint space, bypassing the need to go through the centroid space as in previous methods.
- Simulation results show that the trajectories optimized after adding contact point information are more human-like compared to those without contact information.

2. Trajectory Optimization

Traditional motion planning methods map from the high-level motion task strategy space to the centroid space and then to the full-body joint space of humanoid robots [25]. The advantage of this method is that the centroid motion model is simple and can easily generate whole-body joint motions. However, due to the oversimplified movements in the centroid space, which cannot consider all possible contacts, and its inability to utilize known information about the robot's spatial posture, we employ an optimization method that maps directly from the high-level motion task strategy space to the full-body joint space of the robot in this paper, as shown in Figure 2.

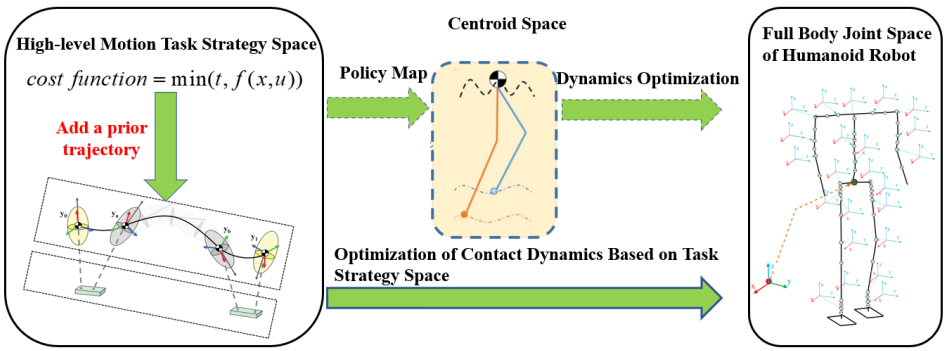

Figure 2. The differences between step-by-step centroid space optimization and unified strategy space planning methods.

2.1. Methods

According to the above-mentioned method, we adopt an approach that works from the high-level task space directly to the joint space, putting the robot's joint angles into the optimization variables. The state variables of existing trajectory optimization contact models mainly focus on the position, direction, velocity, and angular velocity of the center of mass; and since a humanoid robot produces contact forces with the ground during the landing phase, these contact forces also have an impact on the robot's overall state, so this

paper considers the robot's contact forces and the velocity and energy of the contact point and establishes the following form of state variables:

$$x = [p_{com}; \theta_{com}; q_{all}]^T \tag{1}$$

$$\mathrm{X} = [x; \dot{x}; p_c; \dot{p}_c; W_{c1} ... W_{cn}]^T \tag{2}$$

where p_{com} represents the position of the centroid, θ_{com} represents the direction of the centroid, q_{all} represents all joint angles, p_c represents the position of the contact point, $\dot{p}_c$ represents the velocity of the contact point, and W_{ci} represents the work done by the contact force corresponding to the *i*-th contact point.

Here, we represent the control input of the robot as follows

$$\mathrm{U} = [\tau_i; f_1 ... f_n]^T \tag{3}$$

where τ represents the driving torque of the corresponding joint of the robot, and f_n represents the contact force of the n-th contact point. The advantages of using this form of state variables and control inputs are: First, we added the state update equations for the contact point position dimensions, allowing us to leverage partial prior information about contact points from the reference trajectory. Second, the additional terms for contact force work nicely to avoid energy conservation issues while facilitating reference trajectories based on energy changes. To further validate that the state variables proposed in this paper can provide good guidance for trajectory optimization of the robot's falling state, we conducted a comparison in the simulation section to demonstrate its effectiveness.

When a humanoid robot falls from the air, it has no contact with the environment, but when it lands on the ground, parts of its body come into contact with the surroundings. These contact forces affect the robot's base; however, the full-body dynamic model provides equations describing the impact of external forces on the robot's motion. According to reference [26], we can obtain the robot dynamic equation involving all external contact forces as

$$D(x)\ddot{x} + C(x, \dot{x})\dot{x} + G(x) = \tau + \sum_{i=1}^{n} J_{ci}^T f_i \tag{4}$$

where $D(x) \in R^{(N+6)\times(N+6)}$ is the mass inertia matrix, $C(x, \dot{x}) \in R^{(N+6)\times(N+6)}$ is Coriolis force, $G(x) \in R^{(N+6)\times(N+6)}$ is gravity acting on the robot, $\tau \in R^{(N+6)\times 1}$ are the joint torques of the robot, $x \in R^{(N+6)\times 1}$ are the state variables of the robot, $J_{ci} \in R^{(N+6)\times(6n))}$ is the Jacobean matrix of the i-th contact point, and $f_i \in R^{(6n)\times 1}$ is the contact force of the robot at the i-th contact point, where N denotes the number of degrees of freedom. When a robot's body comes into contact with the ground, an acceleration is produced at the point of contact. According to the contact dynamics equation mentioned above, the acceleration at the contact point can be related to the contact force via an expression as follows

$$\ddot{p}_c = \dot{J}_c\dot{x} + J_c D^{-1}(u - C(x, \dot{x})\dot{x} - G(x) - \sum_{i=1}^{n} J_i^T f_i) \tag{5}$$

2.2. Cost Function

In this paper, we aim to achieve the reference target with minimal driving force, and there should be no uneven force distribution at each contact point. Therefore, the objective function can be defined as follows:

$$J = \int_{t_0}^{t_f} w_1(U - U_{ref})^2 + \int_{t_0}^{t_f} w_2(X - X_{ref})^2 + \int_{t_0}^{t_f} w_3(F_c - F_{ref})^2 \tag{6}$$

where U represents the robot's control input; U_{ref} represents the robot's reference input; X and X_{ref} represent the state variables and reference variables mentioned earlier, respectively; F_{ref} represents the reference contact force; and w_1, w_2, and w_3 represent the corresponding

weight matrices. The purpose of setting the third term is that we do not expect the robot to have an uneven distribution of forces after falling down the ground. In optimization, we expect the robot's knees and arms to land, resulting in a F_{ref} value of mg/4. In addition, we expect the robot's center of mass to be 0.8 m above the ground with no slippage at the contact points and with the robot's roll, pitch, and yaw angles to be 0.

2.3. Constraints

The following constraints are crucial for optimizing the trajectory:

(1). At the start of optimizing the robot, the minimum and maximum values need to be specified for the initial state variable X_0; also, the minimum and maximum desired values for the final state variable X_F need to be specified. Additionally, the minimum time t_{min} and maximum time t_{max} for the desired robot motion, the minimum q_{min} and maximum q_{max} values for all joints, and the lower limit U_{min} and upper limit U_{max} for the control inputs need to be specified. To accelerate convergence, an initial guess value is also created, including the motion time t_{guess}, state X_{guess}, and control input U_{guess} for the robot.

(2). For the dynamics, we want the entire motion trajectory to satisfy the full-body dynamical model, as shown in Equation (4).

(3). In terms of kinematics, during the optimization process of the robot, the maximum extension of the robot's end-effector should less than the length of the robot arm or leg and should meet the forward kinematics equation [21].

$$\begin{aligned} r &\leq L_{max} \\ r &= g(q_j) \end{aligned} \tag{7}$$

(4). During the optimization process, we found that when the robot falls in a non-dynamic environment, the end points of the robot's arms or legs are prone to mold piercing when colliding with the ground, as shown in Figure 3. Therefore, it is necessary to set the vertical distance $p_i^z > 0$ of the robot's collision point.

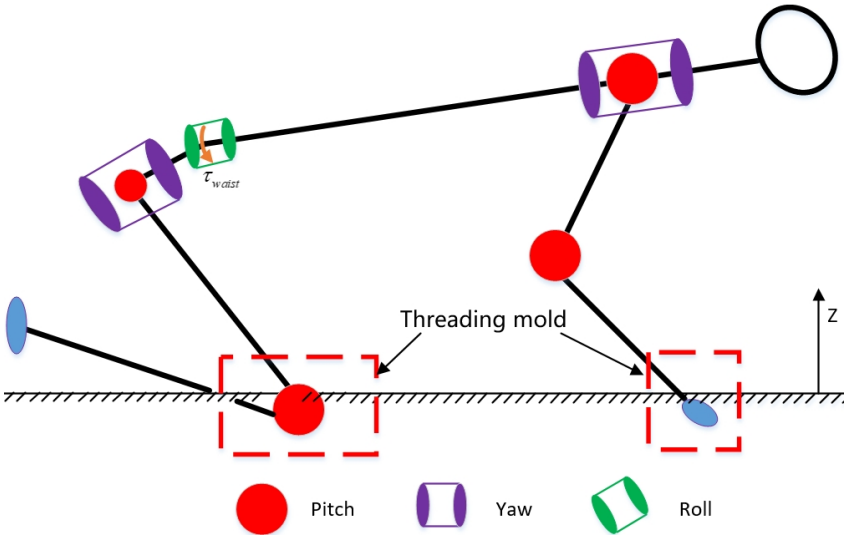

Figure 3. The robot exhibits ground penetration. In the simulation, the robot is in a non-dynamic environment with its arms and knees already penetrating the ground.

(5). When the robot makes contact with the ground, we want no slippage to occur. Based on Posa's contact complementarity constraint equations [27], we can obtain the relationship between the contact position, contact force, and contact velocity, as described below, where ξ is a slack coefficient to encourage convergence.

$$p_c * f_n \leqslant \xi \tag{8}$$

$$\dot{p}_c * f_n \leqslant \xi \tag{9}$$

(6). In [28], the robot's sole was simplified to four support points, with each support point corresponding to a tetrahedron that forms a relatively complex friction cone constraint. In this paper, a simplified model is adopted by approximating each contact location to a single point. Based on the friction relationship, the following equations can be obtained, where F_N and F_f are optimization variables from the inputs above.

$$\mu_f F_N \leq F_f \tag{10}$$

3. Controller

3.1. Air Stage Controller

Through previous trajectory optimizations, we can optimize the state variables and control inputs of the robot during the falling process. These parameters are essential for setting the motion controller. During the air stage, due to differences between the simulation model and the actual physical model, there will be deviations in the position and orientation of the robot's center of mass during motion. Therefore, designing an effective motion controller to minimize these deviations becomes crucial. The PD controller has the function of simple parameters and easy implementation, so we set the following control equations

$$\begin{aligned} \ddot{p}_{comout} &= k_{p_c}(p_a - p_d) + k_{d_c}(\dot{p}_a - \dot{p}_d) \\ \dot{w}_{comout} &= k_{R_c} log(R_a R_d) + k_{w_c}(\dot{\alpha}_a - \dot{\alpha}_d) \end{aligned} \tag{11}$$

$$\begin{aligned} \ddot{p}_{endout} &= k_{p_e}(p_{enda} - p_{endd}) + k_{d_e}(\dot{p}_{enda} - \dot{p}_{endd}) \\ \dot{w}_{comout} &= k_{R_e} log(R_{enda} R_{endd}) + k_{w_e}(\dot{\alpha}_{enda} - \dot{\alpha}_{endd}) \end{aligned} \tag{12}$$

where p represents the position, R represents the orientation, subscript a represents the actual value, subscript d represents the desired actual value, and end represents the endpoint of arms or legs. The variables k_{p_c}, k_{d_c}, k_{R_c}, k_{w_c} represent the corresponding coefficients and have the same meaning as the coefficients in Equation (12).

3.2. Landing Controller

During a robot's walk or run, the center of mass is usually placed at the hip joint center; however, the robot requires a more precise position during falls. Therefore, we use the actual link lengths and mass distributions to solve for the corresponding center-of-mass position.

$$p_a = \frac{\sum_{i=1}^{N_n} m_i p_i(q)}{\sum_{i=1}^{N}(m_i)} \tag{13}$$

where N_n is the total number of links, m_i is the mass of the ith link, and $p_i(q)$ represents the position of the ith link in the world coordinate frame.

Although the aforementioned trajectory optimization method solves for the impact force of the robot's contact points during landing, this is done without considering external disturbances and lacks some robustness. In other words, since the landing time is relatively short, designing a method that can efficiently and quickly respond to such impact forces becomes very important. Model predictive control needs high precision in modeling, but optimization is computationally intensive and time-consuming. Also, different cost functions need to be designed for various scenarios, limiting generalizability. Since a robot's fall from the air and contact with the ground happens very quickly, a fast-responding algorithm is needed. Here, we refer to the landing control method proposed in [29]; according to the Newton–Euler laws of motion, we can obtain the following:

$$\begin{aligned} m\dot{p}_{com} &= F_{all} - mg \\ \dot{L} &= n - c \times F_{all} \end{aligned} \tag{14}$$

where m denotes the mass of the robot, p_{com} is the position of the center of mass, c is the position from the contact point to the center of mass, L is the angular momentum about the center of mass, and F_{all} and n represent the force and moment, respectively, exerted on the robot by the environment, expressed in the world frame. Similar to [30], we approximate the angular momentum equation of the robot's center of mass as follows:

$$L = I_{all}\dot{q} \approx I_{base}w \approx Iw \tag{15}$$

where I_{all} is the angular part of the centroidal momentum matrix; $\dot{q}$ is the angular velocity of all joints, including the floating base; I_{base} is the matrix block corresponding to the base coordinate in I_{all}; w is the angular velocity of the base link; and I is a constant and diagonal approximation of I_{base}. Combining Equations (14) and (15) and neglecting the effect of the moment, we can obtain the following equation:

$$\begin{bmatrix} I \\ c\times \end{bmatrix} f = \begin{bmatrix} m(\ddot{p}_{com} + g) \\ I\dot{w} \end{bmatrix} \tag{16}$$

We set the force and torque when the robot lands as

$$N_d = \begin{bmatrix} m(\ddot{p}_{comout} + g) \\ I\dot{w}_{comout} \end{bmatrix} \tag{17}$$

Let

$$M = \begin{bmatrix} I \\ c\times \end{bmatrix} \tag{18}$$

We can express Equation (16) as a form of a quadratic program (QP), where α_1 and α_2 represent the corresponding weight coefficients, the rightmost term in the equation means that the expected landing impact force of the robot is minimized.

$$\begin{aligned} &\min_{f} \alpha_1 (Mf - N_d)^2 + \alpha_2 f^2 \\ &s.t. \mu * F_N \leqslant F_f \end{aligned} \tag{19}$$

3.3. Spring–Damper Controller

Using the method above, we have optimized for the impact force during the robot's landing. However, since our robot is position controlled, we need to convert this to corresponding joint angles. To do so, we adopt a dual spring–damper model as shown in Figure 4. Where k_1, k_2, D_1, and D_2 are the spring and damper coefficients, respectively.

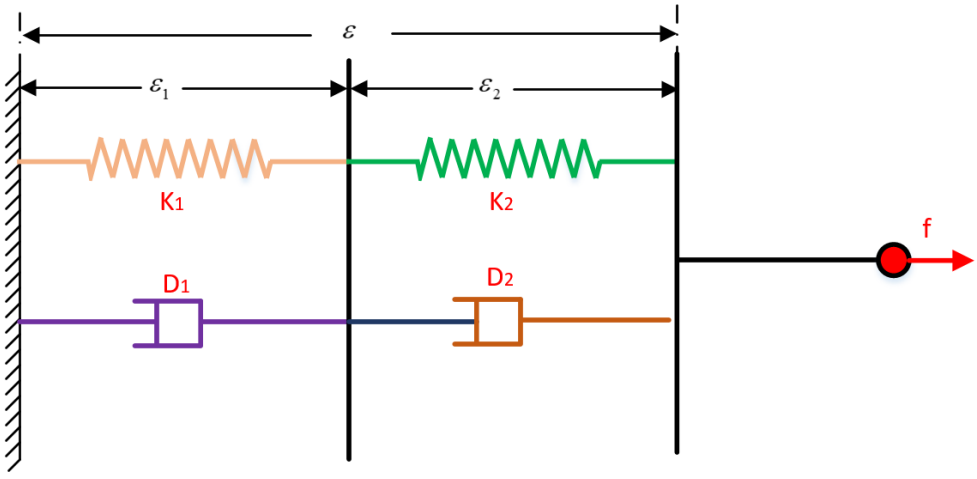

Figure 4. Spring–damper model.

Let ε be the overall deformation of the robot after being subjected to an external force f, where ε_1 and ε_2 are the deformations on the left and right sides, respectively. Then according to Hooke's law, we can obtain:

$$\begin{aligned} \varepsilon &= \varepsilon_1 + \varepsilon_2 \\ f &= k_1\varepsilon_1 + D_1\dot{\varepsilon}_1 = k_2\varepsilon_2 + D_2\dot{\varepsilon}_2 \end{aligned} \tag{20}$$

Applying a Laplace transformation to the second term of Equation (20) yields

$$\begin{aligned} \varepsilon_1 &= \frac{f}{k_1 + D_1 s} \\ \varepsilon_2 &= \frac{f}{k_2 + D_2 s} \end{aligned} \tag{21}$$

Substituting Equation (21) into the first term of Equation (20) and taking the inverse Laplace transform, we obtain

$$k_1k_2\varepsilon + (k_2D_1 + k_1D_2)\dot{\varepsilon} + \ddot{\varepsilon} = f(k_1 + k_2) + \dot{f}(D_1 + D_2) \tag{22}$$

Let the state variable of the spring–damper model be $\chi = \left[f, \varepsilon, \dot{\varepsilon}\right]^T$, where the control input $u = \ddot{\varepsilon}$; then the state equation of this spring–damper model can be written as:

$$\dot{\chi} = A\chi + B\nu \tag{23}$$

$$\frac{d}{dt}\begin{bmatrix} f \\ \varepsilon \\ \dot{\varepsilon} \end{bmatrix} = \begin{bmatrix} \frac{k_1+k_2}{D_1+D_2} & \frac{k_1k_2}{D_1+D_2} & \frac{k_1D_2+k_2D_1}{D_1+D_2} \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{24}$$

Based on Equation (23), we can obtain the relationship between the desired state variable and the desired input as follows:

$$\dot{\chi_d} = A\chi_d + B\nu_d \tag{25}$$

Subtracting Equation (23) from (25), we have

$$\Delta\dot{\chi} = A\Delta\chi + B\Delta\nu \tag{26}$$

The state feedback controller is given by

$$\Delta\nu = -K\Delta\chi \tag{27}$$

where $k = [k_1; k_2; k_3]^T$, which can be obtained through the LQR method. Define the cost function as

$$\begin{aligned} J =& \frac{1}{2}\int_0^\infty (\Delta\chi^T Q\Delta\chi + \Delta\nu^T R\Delta\nu)dt \\ =& \frac{1}{2}\int_0^\infty \Delta\chi^T (Q + K^T RK)\Delta\chi dt \end{aligned} \tag{28}$$

where Q and R are the weight matrices of the state variables X and the input variable u, respectively. Let $K = R^{-1}B^TP$, where P can be obtained through the Riccati equation:

$$A^TP + PA + Q - PBR^{-1}B^TP = 0 \tag{29}$$

Through Equations (28) and (29), we can get the values of the gain coefficient $k = [k_1; k_2; k_3]^T$; then the end change can be expressed as follows:

$$\ddot{\Delta\varepsilon} = -k_1(f_{real} - f) - k_2\Delta\varepsilon - k_3\dot{\Delta\varepsilon} \tag{30}$$

4. Simulation and Experiment

4.1. Simulation Platform

The simulation platform used in this paper is a bipedal robot independently developed by our laboratory, as shown Figure 5. The robot weighs 50 kg in total and has 20 degrees of freedom, including 6 degrees of freedom in the legs, 2 degrees of freedom in the waist, and 3 degrees of freedom in the arms. The specific dimensions and parameters are shown in Table 1.

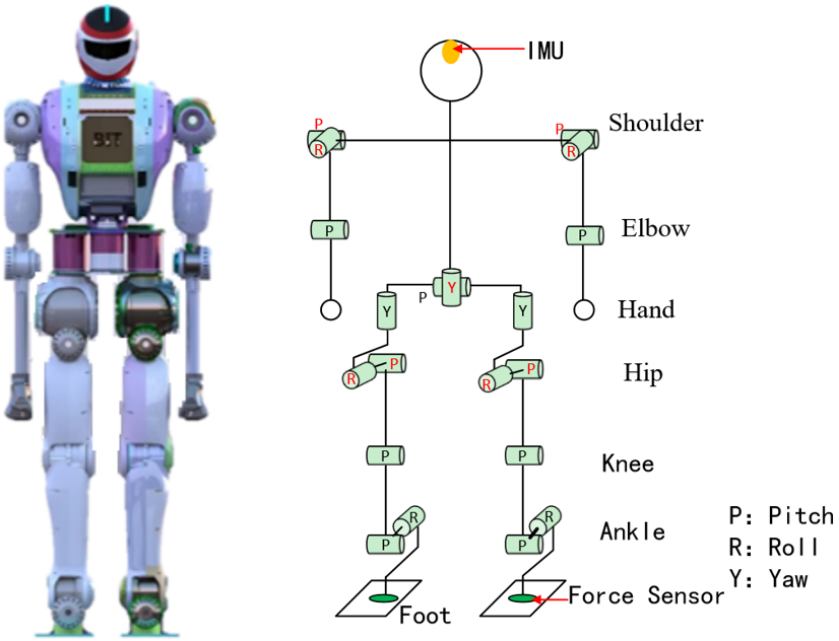

Figure 5. Snapshots of the humanoid robot. The left side represents a three-dimensional view of the robot, and the right side depicts a schematic diagram of the robot's joints.

Table 1. The parameters of our robot.

Parameter	Size	Mass
Thigh	361 (mm)	7.36 (kg)
Shank	330 (mm)	5.12 (kg)
Boom	350 (mm)	4.15 (kg)
Jib	360 (mm)	2.3 (kg)
Others	——	31.07 (kg)
Total Mass	——	50 (kg)

4.2. Trajectory Optimization

The robot's falling trajectory was optimized in MATLAB, while the robot's kinematics and dynamics were generated by the open-source software FROST [31]. The optimization library used consulted Matthew Kelly [32,33]. In the optimization process, we first follow the first constraint introduced in Section 2.3 of the article to give the upper and lower bounds of the robot's state variables, motion time, state input, and so on. To shorten the optimization time and avoid local optima in the first optimization process, we give a free-fall trajectory as an initial trajectory, which is a simple trajectory that does not satisfy the dynamic constraints but can constrain the optimization result to an ideal reliable value range, as shown in Figure 6. Additionally, we selected the trapezoidal method as the interpolation approach. Obviously, the initial reference trajectory from the first optimization did not satisfy the dynamics equations. So after this, we put the optimized reference trajectory into the estimate, performed a second optimization, and repeated this process until the optimal trajectory appeared.

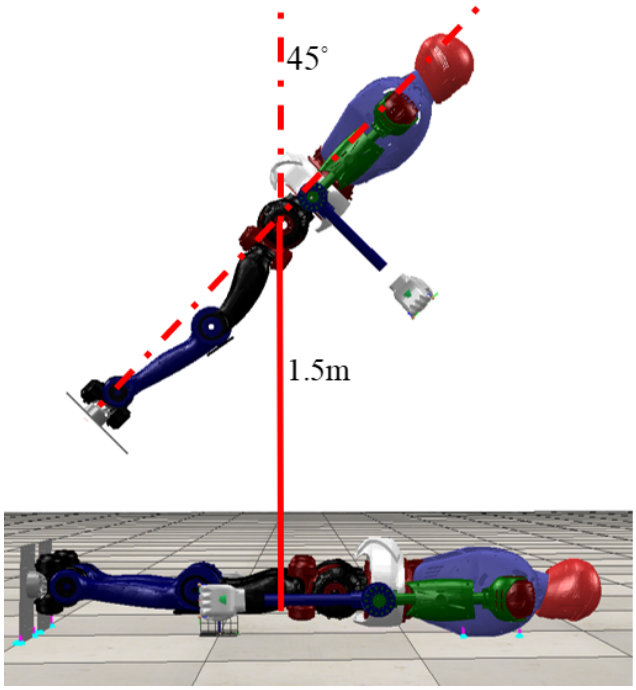

Figure 6. The robot free-falls through the air and in the final state lies flat on the ground.

4.3. Simulation

We perform the simulations using Coppeliasim and MATLAB. In Coppeliasim, the robot's physical engine is Bullet 2.78 and the control period is 5 ms. The initial pitch angle of the robot is 45°, the distance from the ground is 1.5 m, the ground friction coefficient is 0.75, and the other parameters are introduced in Table 2, which appears at the end of the article. We set two different sets of state variables to compare the optimization results: the first set of optimization variables are as shown in Equation (1), and the second set of optimization variables are as shown in Equation (2). The other constraints are kept consistent. In addition, we also simulated the effect of adding the controller to the robot when it landed.

Table 2. The control parameters of the simulation.

Parameter	Value	Parameter	Value	Parameter	Value
w_1	0.1	k_{p_e}	diag([10 850 1760])	k_2	200
w_2	0.8	k_{d_e}	diag([0.005 27 50])	D_1	1500
w_3	0.005	k_{R_e}	diag([5 150 1000])	D_2	1800
ξ	0.03	k_{w_e}	diag([0.005 52.3 65])	Q	diag([1 1 1])
k_{p_c}	diag([10 1000 2000])	α_1	0.52	R	0.0001
k_{d_c}	diag([0.01 12.7 45])	α_2	1	m	50
k_{R_c}	diag([2 100 750])	μ	0.75		
k_{w_c}	diag([0.002 10.0 25])	k_1	210,000		

4.3.1. Simple State Variables

As described above, we used the equation in (1) for the optimization variables. Figure 7 shows the optimized robot motion states obtained using this method. As can be seen from Figure 7(1)–(4), in order to reduce the landing speed, the robot swings its arms backward. Figures 7(5)–(8) show that after the robot lands, its knees quickly touch the ground and its arms also start to find the landing position. Figure 7(8) is the final state, and it can be seen that without the constraint of touchdown information, the robot has problems such as flipping over backward and unbalanced ground contact, and the overall optimized motion exhibits unreasonable phenomena.

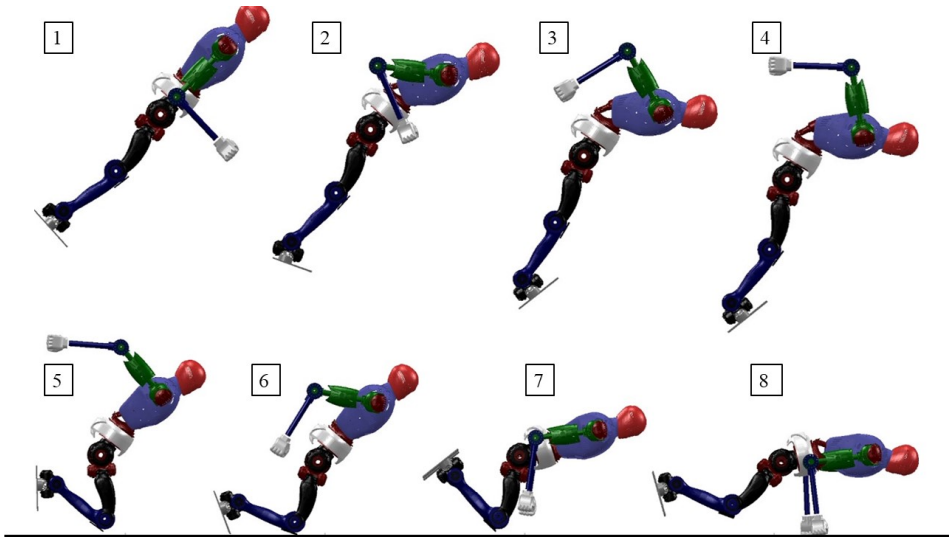

Figure 7. Snapshots of the robot falling from the air without contact point information in the state variables.

4.3.2. Extended State Variables

In the second set of trajectory optimization experiments, we optimized using extended state variables. The optimized robot motion states obtained are shown in Figure 8. Compared to Figure 7, the state variables contain touchdown information. It can be seen that when in the air, the robot stretches its arms and legs straight while bending its waist, which is conducive to adapting when contacting the ground; after the front end of the foot touches the ground, the robot immediately bends its knees to reduce the ground impact while the arms touch the ground to share the pressure. After the robot's hands or knees make contact with the ground, the whole body keeps balance.

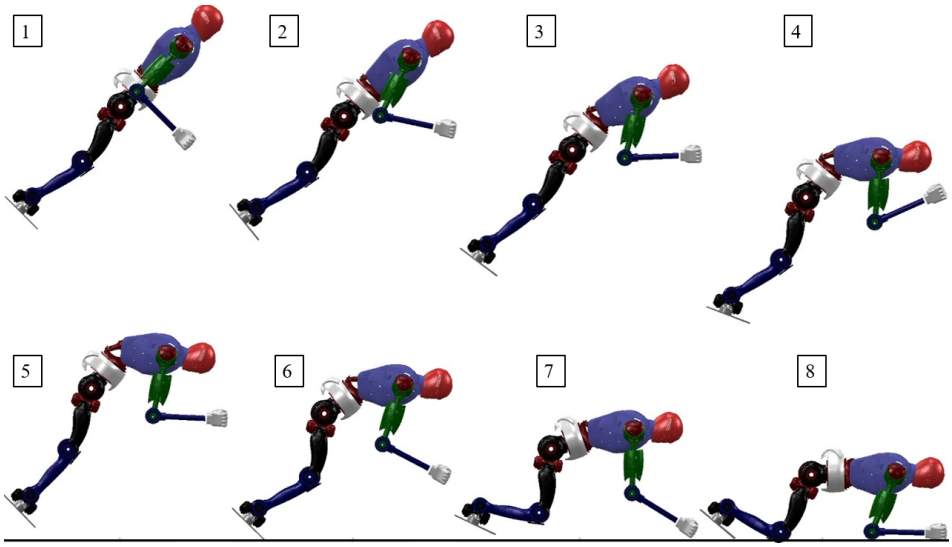

Figure 8. Snapshots of the robot landing without damping controller.

4.3.3. Extended State Variables and Control

Although the trajectories optimized using extended state variables conform to the full-body dynamics model, the impact force during the robot's landing remains large. To solve this problem, we incorporated the motion controller described above. When the robot is moving in the air, simple PD control is used. However, when the robot lands, the landing controller is engaged. Figure 9(5)–(8) shows the effect of it. Once contact with the ground is detected, the robot swings its arms backward, presses down its body, and

moves forward. Near the end of landing buffering, in order to maintain an overall balanced posture, the robot's upper body moves upward and recovers to a four-point landing state.

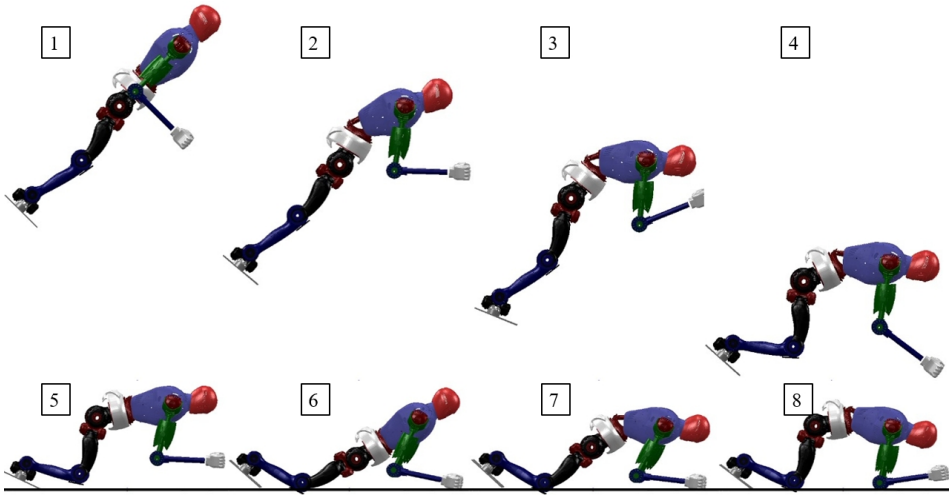

Figure 9. Snapshots of the robot landing with the damping controller.

4.3.4. Graphical Analysis

Due to the significant changes in the z- and y-directions during the robot's fall, this study primarily considers the hip, knee, ankle, waist, shoulder, and elbow joints. Figure 10 shows the joint angle trajectories of the robot joints optimized using the aforementioned trajectory optimization method. It can be observed from the figure that the joint angle trajectories vary smoothly without any anomalous values.

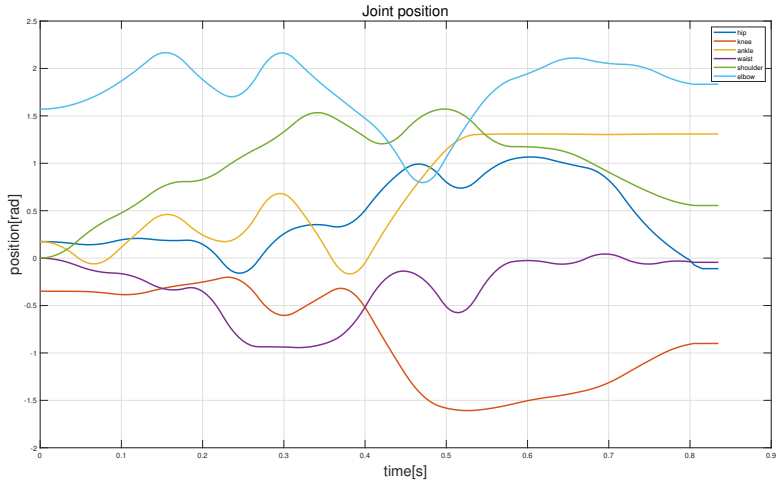

Figure 10. The joint trajectory curve optimized by our method. Owing to the bilateral symmetry between the legs and arms, only the joint angle profiles of the right leg, waist, and arm of the robot are displayed.

Figures 11–13 show schematic diagrams of the robot's actual center-of-mass position, velocity, and orientation, respectively, when falling from the air. The trajectory plots include the cases of extended state variables and control (with control), extended state variables (without control), simple state variables (without contact points), and free-fall motion. The other snapshots are similar. The red curve in Figure 11 shows that after adding the landing controller, the robot's center-of-mass trajectory continues moving downward and then recovers to a stable state, consistent with the motion in Figure 9. In Figure 12, the free-fall motion velocity curve is blue and has a maximum velocity reaching 4.02 m/s. The purple curve indicates optimization with no contact point information in the state

variables: the robot lands after 0.76 s with a maximum velocity of 3.92 m/s. Comparing the red and fluorescent curves (without control), the speed of the robot drops directly from 3.1315 m/s to 0.18185 m/s within 0.04 s after landing, while with control, the velocity decreases from 2.9311 m/s to 1.3363 m/s. This demonstrates that the proposed landing controller provides good shock absorption. Figure 13 shows the orientation of the robot during landing, and it can be seen that after touchdown, the landing controller starts to take effect, demonstrating the effectiveness of the proposed method.

Figure 14 shows a schematic diagram of the impact force on the robot's right hand when striking the ground. Since in free-fall motion we do not want the robot's arms to contact the ground, the impact force is almost 0 during landing. Compared to no landing controller and trajectory optimization without contact point information, the red curve represents the use of control during the fall and has a maximum impact force of 5797 N at the instant of touchdown. Approximately 0.1 s later, a secondary impact occurs, but this time the impact force is 709.2 N, which is less than the 1132 N experienced without a landing controller.

Figure 15 shows the ground reaction force on the robot's right foot: it can be seen that free-fall motion has the most significant impact force. After adding the landing controller, the robot's first impact force is reduced to about 4100 N, and the second impact force is reduced to 2300 N. Comparing without control and without contact points, it can be seen that the impact effects of both are similar. These data demonstrate that the proposed methods can handle ground impact forces effectively.

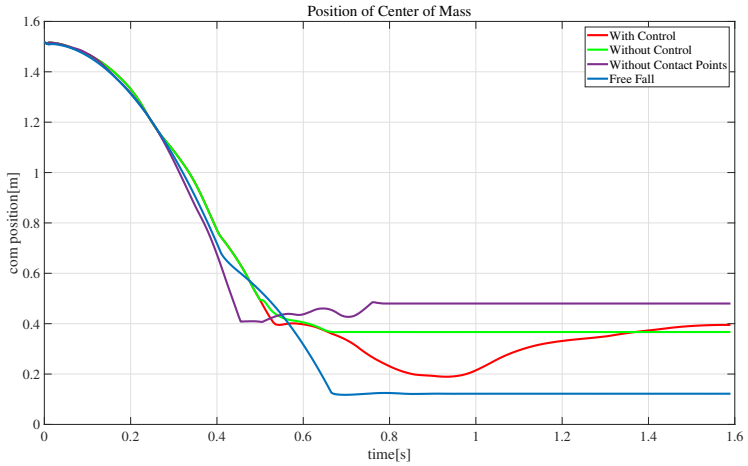

Figure 11. Snapshots of the centroid position of the robot falling from the air.

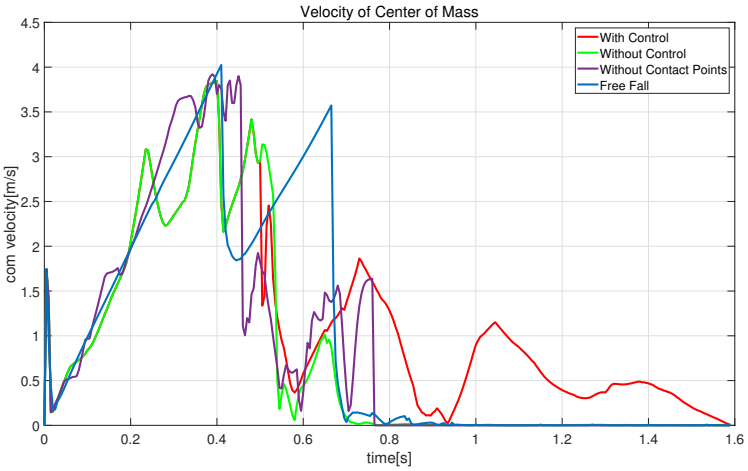

Figure 12. Snapshots of the centroid velocity of the robot falling from the air.

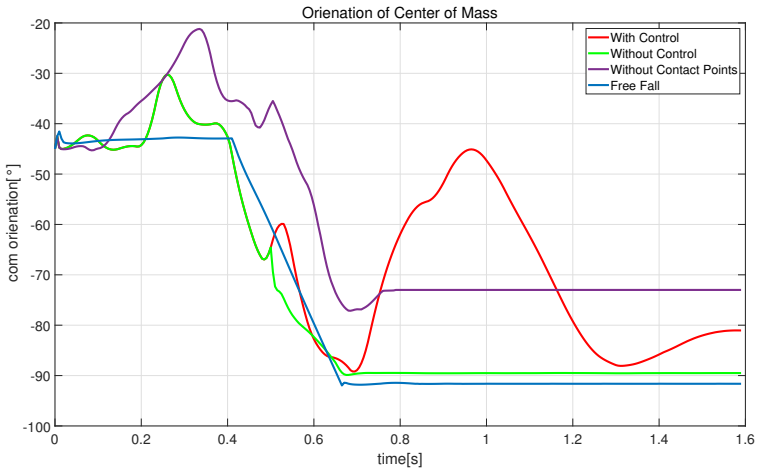

Figure 13. Snapshots of the centroid orientation of the robot falling from the air.

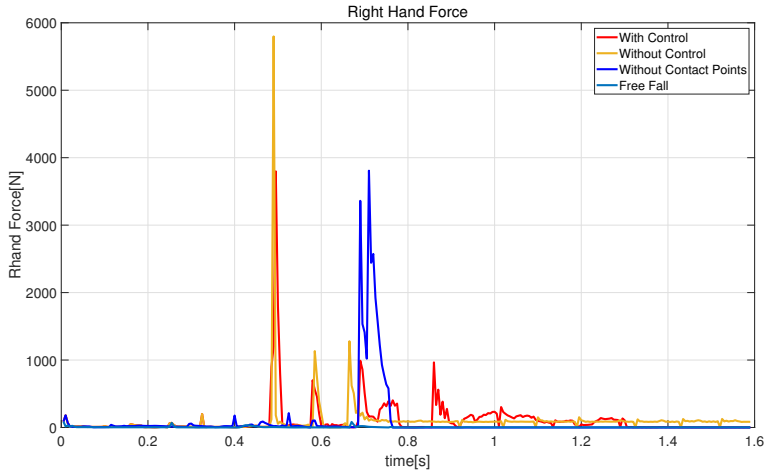

Figure 14. Schematic diagram of the impact force generated by the robot's right hand hitting the ground.

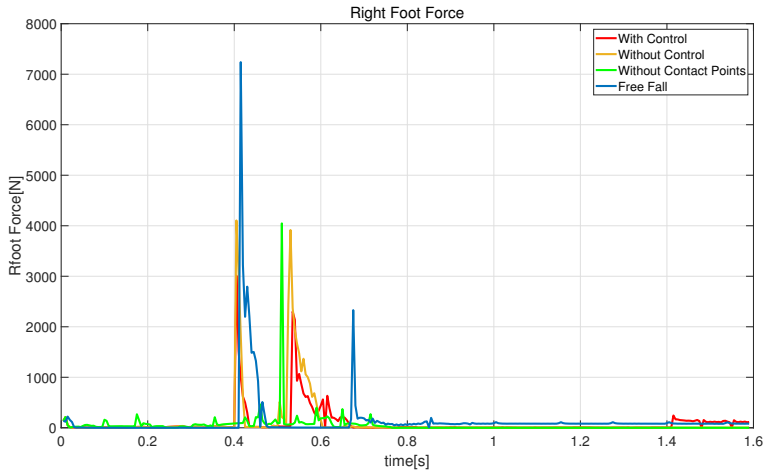

Figure 15. Schematic diagram of the impact force generated by the robot's right foot hitting the ground.

4.4. Experiment

Falling from the air is a very dangerous maneuver for the robot and requires various safety equipment. However, we were unable to complete this experiment at this time since the hardware environment is still being set up. To validate the method proposed in

this paper, we conducted an experiment outdoors using the example of the robot falling forwards to the right. We completed this experiment using trajectory optimization and motion control methods. Figure 16 shows the joint angle profiles of the robot optimized using the aforementioned optimization approach. During the motion, we applied a lateral force of approximately 150 N to the robot for 0.3 s. When the lateral push force exceeded the robot's self-adjustment range, it had to fall. Our robot detected the falling direction and threshold based on the method proposed in [34]. Upon detecting the fall, the robot's right leg quickly lifted up and the waist joint immediately twisted to brace for impact with the ground. Since the robot's hands are quite delicate, to avoid damage during the collision, we swung the arms upwards during the fall. This minimized the impact on the hands; the final effect is shown in Figure 17.

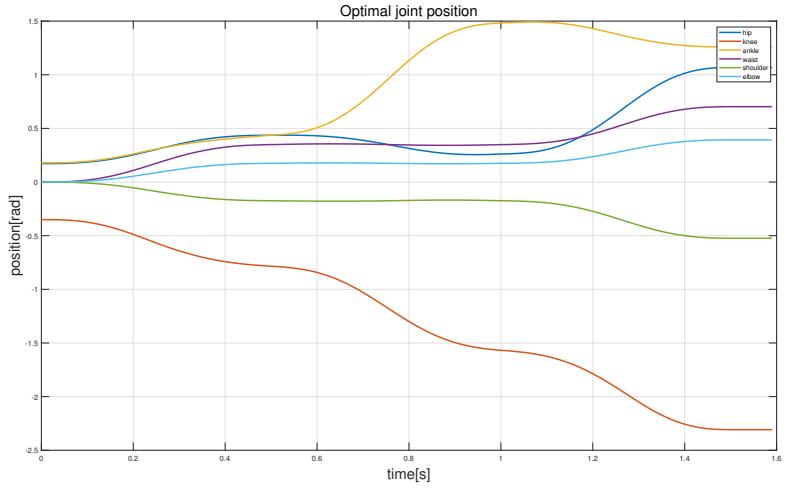

Figure 16. Robot joint position.

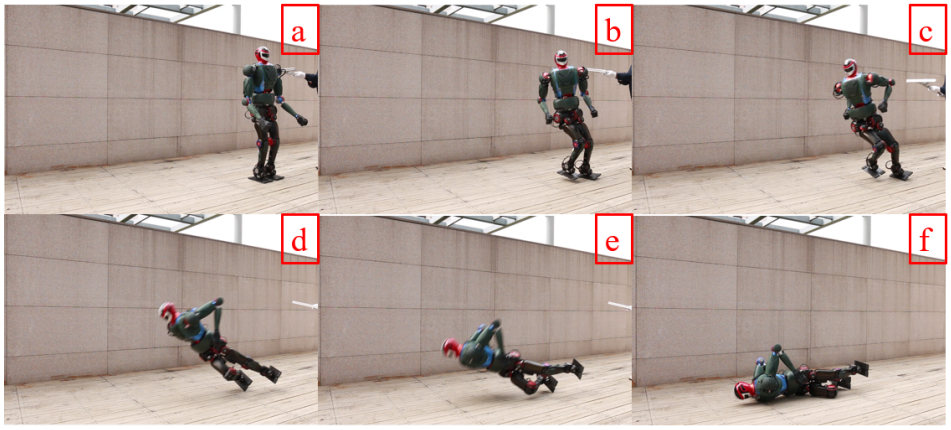

Figure 17. The robot falls on its side, and in Figures (**a**,**b**) an external force is applied to the robot, Figures (**c**,**d**) represents that the robot completes this protection action according to the optimized trajectory and PD control, Figures (**e**,**f**) indicates that the landing controller takes effect after the robot reaches the ground.

5. Conclusions

This paper optimizes a protection trajectory for a robot falling from the air. Compared with traditional trajectory optimization methods, this paper abandons the strategy of humanoid robots of working from the high-level motion task strategy space to the center-of-mass space and then to the whole-body joint space. Rather, we establish a relationship between the high-level motion task strategy space and the whole-body joint space. Moreover, this paper adds the robot's contact point information to the state variables, enabling it to utilize reference contact information to avoid phenomena that do not comply

with contact dynamics during trajectory optimization. A PD controller is added during the robot's flight phase to control the position and direction of the center of mass or the endpoints. During the robot's contact phase, according to the center-of-mass dynamics model, the contact force is optimized. Assuming that force sensors are installed at the endpoint parts of the robot, then according to the actual applied force and the optimized applied force, we can use a damping controller to calculate the movement of the endpoint to finally put the above results into an inverse kinematics optimization based on QP to obtain the joint angle changes required to desired forces. The simulation and hardware experiment results show that by combining trajectory optimization and motion control methods, the robot can safely fall to the ground. Compared to simple free-fall motion or only simple state variables, this method can effectively buffer the impact force.

6. Future Work

In the future, we will implement online motion trajectory optimization of the robot and deploy it to the physical prototype to further prove the effectiveness of this method.

Author Contributions: Conceptualization, W.Z. and J.G.; methodology, W.Z. and J.C.; software, W.Z. and J.C.; validation, W.Z., J.C., X.X. and M.J.; formal analysis, W.Z., J.G. and J.C.; investigation, W.Z.; writing—original draft preparation, W.Z.; writing—review and editing, W.Z.; visualization, M.J.; supervision, X.C.; project administration, X.C. and J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under grants 91748202 and 61973039 and the Beijing Municipal Science and Technology Project under grant Z191100008019003.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feng, S.; Xinjilefu, X.; Atkeson, C.G.; Kim, J. Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals. In Proceedings of the 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, Republic of Korea, 3–5 November 2015; Volume 10, pp. 1028–1035.
2. Elon Musk Reveals New Optimus Robot Video! (2023 Tesla Shareholder Meetinig). Available online: https://www.youtube.com/watch?v=KW3iRzXs940 (accessed on 25 July 2023).
3. Jeong, H.; Lee, I.; Oh, J.; Lee, K.K.; Oh, J.H. A robust walking controller based on online optimization of ankle, hip, and stepping strategies. *IEEE Trans. Robot.* **2019**, *35*, 1367–1386. [CrossRef]
4. Kaneko, K.; Harada, K.; Kanehiro, F.; Miyamori, G.; Akachi, K. Humanoid robot HRP-3. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 2471–2478.
5. Mesesan, G.; Englsberger, J.; Garofalo, G.; Ott, C.; Albu-Schffer, A. Dynamic walking on compliant and uneven terrain using dcm and passivity-based whole-body control. In Proceedings of the 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), Toronto, ON, Canada, 15–17 October 2019; pp. 25–32.
6. Bouyarmane, K.; Chappellet, K.; Vaillant, J.; Kheddar, A. Quadratic programming for multirobot and task-space force control. *IEEE Trans. Robot.* **2019**, *35*, 64–77. [CrossRef]
7. Kajita, S.; Cisneros, R.; Benallegue, M.; Sakaguchi, T.; Nakaoka, S.; Morisawa, M.; Kaneko, K.; Kanehiro, F. Impact acceleration of falling humanoid robot with an airbag. In Proceedings of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), Cancun, Mexico, 15–17 November 2016; Volume 35, pp. 637–643.
8. Lee, S.H.; Goswami, A.; Kaneko, K.; Kanehiro, F. Fall on backpack: Damage minimization of humanoid robots by falling on targeted body segments. Journal of Computational Nonlinear Dynamics. *J. Comput. Nonlinear Dyn.* **2013**, *8*, 021005. [CrossRef]
9. Kakiuchi, Y.; Kamon, M.; Shimomura, N.; Yukizaki, S.; Inaba, M. Develop- ment of life-sized humanoid robot platform with robustness for falling down, long time working and error occurrence. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 689–696.
10. Nguyen, K.; Kojio, Y.; Noda, S.; Sugai, F.; Inaba, M. Dynamic fall recovery motion generation on biped robot with shell protector. *IEEE Robot. Autom. Lett.* **2021**, *6*, 6741–6748. [CrossRef]

11. Subburaman, R.; Lee, J.; Caldwell, D.G.; Tsagarakis, N.G. Online falling-over control of humanoids exploiting energy shaping and distribution methods. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 448–454.
12. Braghin, F.; Henze, B.; Garzon, M.A.R. Optimal trajectory for active safe falls in humanoid robots. In Proceedings of the 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), Toronto, ON, Canada, 15–17 October 2019; pp. 305–312.
13. Ruiz-Del-Solar, J.; Palma-Amestoy, R.; Marchant, R.; Parra-Tsunekawa, I.; Zegers, P. Learning to fall: Designing low damage fall sequences for humanoid soccer robots. *Robot. Auton. Syst.* **2009**, *57*, 796–807. [CrossRef]
14. Ha, S.; Liu, C.K. Multiple contact planning for minimizing damage of humanoid falls. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–3 October 2015; pp. 2761–2767.
15. Mujica, M.; Crespo, M.; Benoussaad, M.; Junco, S.; Fourquet, J.Y. Robust variable admittance control for human–robot co-manipulation of objects with unknown load. *Robot. Comput.-Integr. Manuf.* **2023**, *79*, 102408. [CrossRef]
16. Abadi, A.S.S.; Ordys, A.; Pierscionek, B. Novel off-line self-tuning controller with guaranteed stability. *Int. J. Automot. Technol.* **2023**, *24*, 851–862 [CrossRef]
17. Liu, A.; Chen, T.; Zhu, H.; Fu, M.; Xu, J. Fuzzy variable impedance-based adaptive neural network control in physical human–robot interaction. *Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng.* **2023**, *237*, 220–230. [CrossRef]
18. Chignoli, M.; Kim, D.; Stanger-Jones, E.; Kim, S. The MIT humanoid robot: Design, motion planning, and control for acrobatic behaviors. In Proceedings of the 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), Munich, Germany, 19–21 July 2021; pp. 1–8.
19. Qi, H.; Chen, X.; Yu, Z.; Huang, G.; Liu, Y.; Meng, L.; Huang, Q. Vertical Jump of a Humanoid Robot with CoP-Guided Angular Momentum Control and Impact Absorption. *IEEE Trans. Robot.* **2023**, *39*, 3154–3166. [CrossRef]
20. Kurtz, V.; Li, H.; Wensing, P.M.; Lin, H. Mini cheetah, the falling cat: A case study in machine learning and trajectory optimization for robot acrobatics. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 4635–4641.
21. Jeon, S.H.; Kim, S.; Kim, D. Online optimal landing control of the mit mini cheetah. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 178–184.
22. Tang, Y.; An, J.; Chu, X.; Wang, S.; Wong, C.Y.; Au, K.S. Towards Safe Landing of Falling Quadruped Robots Using a 3-DoF Morphable Inertial Tail. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation, London, UK, 29 May–2 June 2023;
23. Yang, Y.; Norby, J.; Yim, J.K.; Johnson, A.M. Proprioception and Tail Control Enable Extreme Terrain Traversal by Quadruped Robots. *arXiv* **2023**, arXiv:2303.04781.
24. Roscia, F.; Focchi, M.; Del Prete, A.; Caldwell, D.G.; Semini, C. Reactive Landing Controller for Quadruped Robots. *arXiv* **2023**, arXiv:2305.07748.
25. Dai, H.; Valenzuela, A.; Tedrake, R. Whole-body motion planning with centroidal dynamics and full kinematics. In Proceedings of the 2014 IEEE-RAS International Conference on Humanoid Robots, Madrid, Spain, 18–20 November 2014; pp. 295–302.
26. Featherstone, R. *Robot Dynamics Algorithms*; Edinburgh University: Edinburgh, UK, 1987.
27. Posa, M.; Cantu, C.; Tedrake, R. A direct method for trajectory optimization of rigid bodies through contact. *Int. J. Robot. Res.* **2014**, *33*, 69–81. [CrossRef]
28. Cisneros, R.; Benallegue, M.; Morisawa, M.; Kanehiro, F. QP-based task-space hybrid/parallel control for multi-contact motion in a torque-controlled humanoid robot. In Proceedings of the 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), Toronto, ON, Canada, 15–17 October 2019.
29. Nguyen, Q.; Powell, M.J.; Katz, B.; Carlo, J.D.; Kim, S. Optimized jumping on the mit cheetah 3 robot. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 7448–7454.
30. Murooka, M.; Morisawa, M.; Kanehiro, F. Centroidal trajectory generation and stabilization based on preview control for humanoid multi-contact motion. *IEEE Robot. Autom. Lett.* **2022**, *7*, 8225–8232. [CrossRef]
31. Hereid, A.; Ames, A.D. FROST: Fast Robot Optimization and Simulation Toolkit. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017.
32. OptimTraj—Trajectory Optimization for Matlab. Available online: https://github.com/MatthewPeterKelly/OptimTraj (accessed on 10 August 2023).
33. Kelly, M. An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. *SIAM Rev.* **2017**, *59*, 849–904. [CrossRef]
34. Wu, T.; Yu, Z.; Chen, X.; Dong, C.; Gao, Z.; Huang, Q. Falling Prediction based on Machine Learning for Biped Robots. *J. Intell. Robot. Syst.* **2021**, *103*, 1–14. [CrossRef]