micromachines

MDPI

Article

Dynamic Balancing of Humanoid Robot with Proprioceptive Actuation: Systematic Design of Algorithm, Software, and Hardware

Yan Xie [1], Jiajun Wang [1], Hao Dong [1], Xiaoyu Ren [1], Liqun Huang [1] and Mingguo Zhao [2,3,*]

[1] Beijing Research Institute of UBTECH Robotics, Beijing 100084, China
[2] Department of Automation, Tsinghua University, Beijing 100084, China
[3] Beijing Innovation Center for Future Chips, Beijing 100084, China
[*] Correspondence: mzhao@tsinghua.edu.cn

Abstract: For humanoid robots, maintaining a dynamic balance against uncertain disturbance is crucial, and this function can be achieved by coordinating the whole body to perform multiple tasks simultaneously. Researchers generally accept hierarchical whole-body control (WBC) to address this function. Although experts can build feasible hierarchies using prior knowledge, real-time WBC is still challenging because it often requires a quadratic program with multiple inequality constraints. In addition, the torque tracking performance of the WBC algorithm will be affected by uncertain factors such as joint friction for a large transmission ratio proprioceptive-actuated robot. Therefore, the balance control of physical robots requires a systematic solution. In this study, a robot control system with high computing power and real-time communication ability, UBTMaster, is implemented to achieve a reduced WBC in real time. Based on these, a whole-body control scheme based on task priority for the dynamic balance of humanoid robots is implemented. After realizing the joint friction model identification, finally, a variety of balancing scenarios are tested on the Walker3 humanoid robot driven by the proprioceptive actuators to verify the effectiveness of the proposed scheme. The Walker3 robot exhibits excellent balance when multiple external disturbances occur simultaneously. For example, the two feet of the robot are subjected to tilt and displacement perturbations, respectively, while the torso is subjected to external shocks simultaneously. The experimental results show that the dynamic balance of the robot under multiple external disturbances can be achieved by using strictly hierarchical real-time WBC with a systematic design.

Keywords: whole-body control; hierarchical optimization; humanoid robot balance; proprioceptive actuation

check for updates

Citation: Xie, Y.; Wang, J.; Dong, H.; Ren, X.; Huang, L.; Zhao, M. Dynamic Balancing of Humanoid Robot with Proprioceptive Actuation: Systematic Design of Algorithm, Software, and Hardware. *Micromachines* **2022**, *13*, 1458. https://doi.org/10.3390/mi13091458

Academic Editors: Zhangguo Yu and Marco Ceccarelli

Received: 1 August 2022
Accepted: 25 August 2022
Published: 2 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Scholars have studied humanoid robots for decades and have made significant progress in robot mobility, dexterity, and intelligence [1–3]. However, making humanoid robots work or interact with humans in a human-friendly environment faces substantial challenges. First, maintaining balance is one of the most fundamental skills of legged robots.

Due to the real-time requirements of the balance algorithm, scholars widely used the balance control method based on a simplified model in their early work. They simplified the robot into a single rigid body mounted on the top of the inverted pendulum, and the control target was its center of mass [4]. Nechev et al. [5] proposed a three-link planar model, which can include more information. Because the model's accuracy is improved, these methods, including ankle or hip joint, will be more feasible for a physical robot. However, due to the failure to consider the motion of all joints, they will still give unnatural solutions in some cases.

Whole-body control (WBC) has been paid more and more attention by robotics in the past two decades because it makes full use of the redundant degrees of freedom of robots to complete multiple tasks simultaneously. For example, the humanoid robot has various degrees of freedom to realize balance as a high-priority task in the WBC framework. Therefore, WBC has gradually become a basic control scheme for the humanoid robot. Dietrich et al. [6] have categorized the existing WBC methods into: null-space projection-based WBC (NSP-WBC), weighted quadratic program-based WBC (WQP-WBC), and hierarchical quadratic program-based WBC (HQP-WBC).

In redundant manipulator control, the NSP-WBC realizes the task hierarchy through null-space projection [7]. Kajita et al. [8] extended this idea to the walking control of the HRP-2 robot for the first time. However, this method is limited because it cannot consider inequality dynamic constraints such as joint torque limitations, which are highly necessary to the robot's safety.

The WQP-WBC handles this problem by formulating the tasks and constraints as a quadratic optimization problem. As a result, it can find the optimal solution that minimizes the task errors while satisfying the constraint conditions. This method has been applied to the Atlas robot to execute multi tasks during the DARPA robotics challenge [9–11]. However, the WQP-WBC cannot guarantee a strict task hierarchy. The soft hierarchy realized by tuning the task weight becomes a limitation when the tasks conflict.

The HQP-WBC has been devised to combine the task hierarchy and inequality constraints together [12–14]. The basic idea is to construct each layer as a quadratic optimization problem and solve them in a sequence where the lower-priority QPs cannot disturb the higher-priority QPs. As a result, the inequality constraints in the QP stack gradually as the priority decreases, thus leading to a time-consuming problem. Ref. [15] proposed an efficient way by introducing null-space projection to reduce the computational cost and implemented it on a torque-controlled robot Sarcos with a 1 KHz control loop. Similarly, ref. [16] implemented this method on a quadruped robot ANYmal showing a natural adaption to the terrain while walking. In short, the HQP-WBC has been a standard whole-body motion generation tool for torque-controlled robots.

In recent years, proprioceptive actuation has become a mature technology widely used in scenarios requiring back-driveability, such as legged robots. Engineers can compensate for the torque loss in the reducer online through the predetermined friction model and online parameter identification, such as the high dynamic locomotion of the MIT cheetah robot [17]. However, when the gear reduction ratio increases, the friction model becomes more complex and has stronger nonlinearity and uncertainty. Therefore, more comprehensive offline modeling and parameter identification processes, and even an online identification process, are required.

Besides the high-bandwidth characteristic of actuators, the overall performance of the robot highly depends on the control frequency. For example, ref. [18] has noticed a phenomenon in the torque-controlled robot, Mercury, that increasing the control frequency from 1 kHz to 1.5 kHz will provide a more significant posture and foot position control bandwidth. This operation puts forward a higher demand for the real-time computation of WBC. Moreover, a real-time WBC is preferred as other time-consuming techniques such as model predictive control (MPC) is usually embedded into the control framework.

This paper focuses on the dynamic balance control of a humanoid robot with proprioception actuators through algorithm, software, and hardware system integration. We first customize a prioritized hierarchy of tasks and constraints for rejecting multi-disturbances. Then, a reduced whole-body control is implemented in real-time by UBTMaster, a control system designed to provide computationally efficient WBC software and powerful computing hardware. This unique real-time computing system is not available in other robotic systems, which ensures the effective implementation of the control algorithm. Next, the model identification process covers the joint friction and model inaccuracy issues. Finally, plenty of experiments on various balancing scenarios are implemented on a robot Walker3 with proprioceptive actuation, and the performance is discussed.

2. Control Approach

Walker3 is a humanoid robot with two legs and a torso, as shown in Figure 1. The robot has a height of 1.6 m and weighs 43 kg. An inertial measurement unit (IMU) is mounted on the torso for state estimation, and two six-axis force sensors are installed on the soles to measure each foot's center of pressure (CoP). Each leg has six electrical motors in series. The motors use a gear reducer to enlarge the output torque, and the gear reduction ratio ranges from 50 to 100. The actuators are controlled in real time using the EtherCAT communication protocol.

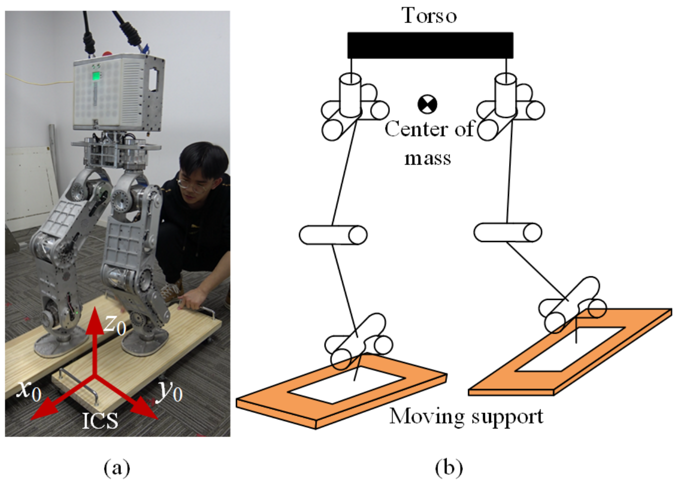


Figure 1. Walker3 humanoid robot with 12 actuated DoFs (**a**) and its kinematic model (**b**).

An online task planner is proposed to adjust the task trajectory. The basic idea of this planner is to ensure the foot ZMP resides in the safe region as much as possible. Then, the desired CoM trajectory is tracked by a reduced whole-body controller coupled with a hierarchical optimization solver. The hierarchy of tasks and constraints is divided into four layers according to their priorities. A quadratic optimization solves each layer, and the null-space projection guarantees the strict hierarchy among layers. After solving a sequence of QPs, the whole-body controller outputs the optimized joint torques.

Joint torques are turned into the current commands for a robot with proprioceptive actuation. The desired joint velocity commands obtained through the numerical integration of desired joint accelerations are also considered here to improve the performance of the joint-level control. The whole control architecture is illustrated in Figure 2. The kinematics solver estimates the robot's state with the sensory data of joints and IMU.

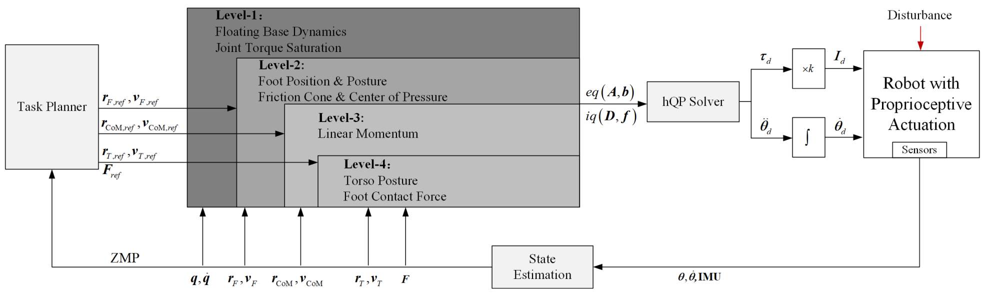


Figure 2. Overview of the control architecture.

3. Tasks and Constraints in Dynamic Balancing

3.1. Task Planner

In dynamic environments, the robot needs to properly tune the desired motion. The task planner presented here can make real-time adjustments to the task trajectory according to the ZMP state.

Unlike the method in [15], we have no extra sensors to obtain information on the moving support, apart from the force sensors to detect the contact between the foot and support. When a robot is stably resting on a moving support, the ZMP of each foot must reside inside its support polygon. Therefore, the desired motion of the foot can be set using the following rule: if the measured ZMP is inside the safe region of the support polygon, we adjust the desired position, posture, and velocity of the foot to its current state $r_{F,ref} = r_F$ and $v_{F,ref} = v_F$.

In addition, the desired motion of CoM should be adjusted along with the foot desired motion. The desired horizontal position is located in the middle of two feet:

$$r_{\mathrm{CoM},ref}(x,y)=\Big(r_{LF,ref}(x,y)+r_{RF,ref}(x,y)\Big)/2 \tag{1}$$

The desired vertical position is set to

$$r_{\mathrm{CoM},ref}(z) = \Big(r_{LF,ref}(z)+r_{RF,ref}(z)\Big)/2+\mathrm{C} \tag{2}$$

where C is a constant value depending on the robot stand pose. The desired velocity is set to the average velocity of two feet

$$v_{\mathrm{CoM},ref}=\Big(v_{LF,ref}+v_{RF,ref}\Big)/2 \tag{3}$$

In our planner, the desired motion of the torso and foot contact force remains unchanged throughout the balance control. The desired motion of torso is set to $r_{T,ref}$=0, $v_{T,ref}$=0 and the desired vertical contact force is set to $F_{LF,ref}(z)$=$F_{RF,ref}(z)$=$mg/2$, where m is the total mass of the robot.

3.2. *Tasks and Constraints Hierarchy*

When multi-tasks have to be performed simultaneously, handling the conflicts among these objectives is crucial. The prioritized hierarchy strategy has been widely adopted in redundant robots. Motion solvers will accomplish the lower-priority tasks under the prerequisite that the higher-priority tasks are implemented first. For example, balance is always considered a top-layer priority task for a humanoid robot. As a result, the robot tends to sacrifice its posture under disturbance to ensure the feet fully contact the ground. Likewise, physical constraints concerning humanoid robot safety should be the highest priority. Table **??** specifies the hierarchy of tasks and constraints.

Table 1. Table task constraint hierarchy.

Level	Task	Task Dimensions	Constraint	Constraint Dimensions
1	Floating base dynamics	6	Joint torque saturation	12
2	Foot position and posture	12	Center of pressure and Friction cone	18
3	Linear momentum	3		
4	Torso posture and Foot contact force	15		

3.2.1. Floating Base Dynamics

Humanoid robots, typical floating-based systems, are an example of underactuated systems due to their partial actuation when interacting with the environment. The configuration of a humanoid robot is represented by generalized coordinates $q = \begin{bmatrix} q_f^{\mathrm{T}} & q_a^T \end{bmatrix}^{\mathrm{T}}$, q_f represents the position and orientation of the robot free-floating body and q_a represents the n-actuated joints of the robot. When the robot is in contact with the environment, the dynamic equation of the system can be fully described by

$$S_f M\ddot{q} + S_f C + S_f G = S_f J^{\mathrm{T}} F \tag{4}$$

where M is the generalized inertia matrix, C is the nonlinear vector including Coriolis and centrifugal forces, G is the gravity vector. F is the contact force vector, and J is the Jacobian matrix of the contact point. $S_f = \left[\begin{array}{cc} I & 0 \end{array}\right]$ is a matrix selecting the free-floating joints. Thus, actuated torque vector τ is eliminated from the dynamic equation. Instead, choosing a different selection matrix $S_a = \left[\begin{array}{cc} 0 & I \end{array}\right]$ will derive a linear function between τ and $\ddot{q}$, F. Due to such linear dependence, the whole-body dynamics of $n+6$ dimensions are simplified as the floating base dynamics of six dimensions. Adopting floating base dynamics is crucial to reduce the optimization time and help implement the 1 KHz control loop.

The dynamic equation is essential for a physical multi-rigid-body system as the highest-priority task. Once the equation holds, the movements of the system are physically feasible. Following the dynamics equation, the second-priority task is the foot position and posture control. A good task control performance will guarantee good contact between foot and support, which is a premise for the contact force. Linear momentum control, the third priority task, has been proven to be essential for a good balance by regulating the state of CoM [19]. Finally, in the lowest-priority task, we prefer to have the torso posture control and foot contact force control on the same level. This is because we have only 30 optimization variables (including 18 for $\ddot{q}$ and 12 for F). While there are 6, 12, and 3 variables for dynamic equation, foot position, posture task, and linear momentum task, respectively, only 9 free variables are left for the torso posture and foot contact force control.

3.2.2. Operational Space Tasks

Operational space tasks such as foot position and posture control, linear momentum control, and torso posture control can be phrased as:

$$J\ddot{q} + \dot{J}\dot{q} = a \tag{5}$$

where J is the Jacobian matrix of a specific task, and a is the task desired acceleration which can be determined by a feedforward and feedback control law. The $\dot{J}\dot{q}$ term is related to the robot state. In particular, the Jacobian matrix of the linear momentum task is also called the centroidal momentum matrix. It can be calculated using an efficient O(n) algorithm based on the generalized inertia matrix M [20].

3.2.3. Robot Safety Constraints

Given the physical limitations of the robot, several safety issues must be appropriately concerned. The joint torque saturation constraint $\tau_{\min} \leq \tau \leq \tau_{\max}$ is especially important for generating control commands that are valid on a robot.

A stable contact between foot and support is an essential precondition for generating a six-dimensional contact force vector, which means the foot cannot tilt or slide relative to the support. Stable contact can be ensured from two aspects. One is the center of pressure constraint. The center of pressure at each foot must not exceed the foot's support polygon boundary. The other is the friction cone constraint, which requires that the foot contact force stays inside the friction cones. The cones are approximated as pyramids here, so the constraint can be expressed as linear inequality.

4. Real-Time WBC

4.1. Reduced Hierarchal Whole-Body Control

The tasks can be formulated as equalities, and constraints can be formulated as inequalities. Therefore, the tasks and constraints in the same level can be stacked vertically into the form.

$$\left\{ \begin{array}{l} A_i x - b_i = 0 \\ D_i x - f_i \leq 0 \end{array} \right. \tag{6}$$

where A_i is the ith task matrix, b_i is the ith task reference vector, D_i is the ith constraint matrix, f_i is the ith constraint boundary vector, and $x = \begin{bmatrix} \ddot{q}^{\mathrm{T}} & F^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ is the optimal variables. The goal of this level is to find $\ddot{q}$ and F that satisfy these objectives as well as possible. The solution under such a linear inequality constraint can be solved through quadratic optimization. The tasks and constraints in different levels need to be optimized in a strict prioritized order. Solving level p yields an optimal solution x_p^*. In order to ensure the strict prioritization of tasks, the solution of level $p+1$ can be found in the null space of all higher-priority tasks $N_p = N_{p-1}\left(I - \hat{A}_p^{\#}\hat{A}_p\right)$. N_{p-1} is the null space of all tasks from level 1 to $p-1$. $\left(I - \hat{A}_p^{\#}\hat{A}_p\right)$ is the null space of task in level p. $\hat{A}_p = A_p N_{p-1}$ describes the task matrix of level p projected into the null space of all higher-priority tasks. The solution of level $p+1$ can be expressed as $x_{p+1}{=}x_p^* + N_p u_{p+1}$, where u_{p+1} is an arbitrary vector lying in the row space of N_p. Substituting x_{p+1} into the QP problem in level $p+1$ yields

$$\begin{aligned}
\min_{u_{p+1}}. \quad & \left\| A_{p+1}\left(x_p^* + N_p u_{p+1}\right) - b_{p+1} \right\|^2 \\
\text{s.t.} \quad & D_{p+1}\left(x_p^* + N_p u_{p+1}\right) - f_{p+1} \leq 0 \\
& D_p\left(x_p^* + N_p u_{p+1}\right) - f_p \leq 0 \\
& \vdots \\
& D_1\left(x_1^* + N_p u_{p+1}\right) - f_1 \leq 0
\end{aligned} \tag{7}$$

All the higher-priority constraints are stacked into the optimization to ensure the strict prioritization of constraints. Then, the recursive algorithm is used to solve the QP of each layer according to the priority order.

The slack variables are introduced initially to turn the hard constraint into a soft one. In our case, however, we notice that the optimized slack variables are always zero, which means that the solver can find the optimal result without violating the hard constraints. Therefore, the slacks are excluded from the optimization variables to reduce the computational complexity. As a result, the slack variables are omitted in the optimization problem, different from the general formulation in [15].

WBC software is developed based on C++ to implement the above algorithm effectively. Figure 3 depicts the architecture of the WBC software. The software contains four basic classes: RobotDynamics, Task, Constraint, and Wbc. These classes provide the basic interfaces for user development, and the derived classes of Walker3 are developed in this software. The following part will describe these classes in detail.

The RobotDynamics class contains the member variables related to the kinematics and dynamics of a robot, such as the number of the generalized joints, contact forces, inertia matrices, Coriolis and centrifugal vectors, gravity vectors, selection matrices, Jacobian matrices, etc.

For the Walker3 robot, it is implemented by a subclass named RobotDynamics_Walker3. The model structure of Walker3 can be constructed in the subclass directly or loaded from URDF files. Calling the calcWbcDependence()function can obtain all the required kinematics and dynamics parameters. The open-source rigid body dynamics library (RBDL), a highly efficient C++ library with some essential rigid body dynamics algorithms [21], is used here. The task and constraint classes are constructed according to Equation (6) with their member variables, including the task's or constraint's name, priority, dimension, matrix, reference vector (boundary vector), and the DoF of variables. Each task or constraint of Walker3 is implemented by a subclass, thus forming a task or constraint library. Using the update (const RobotDynamics &) function will update the member variables with calculated kinematics and dynamics parameters.

The Wbc class, the software's core, contains the pointers of the other three types. It can manage the tasks' addition, deletion, and adjustment operations and constraints. In its implementation, two subclasses based on different algorithms are developed here. One is named WqpWbc, which forms all the tasks and constraints as one quadratic optimization problem [10]. The other is HqpWbc, which implements the hierarchical quadratic optimization mentioned before.

This software avoids lots of redundant codes and improves efficiency development. Meanwhile, developers also build the dynamic model of some robots and their corresponding tasks' and constraints library. Users can also develop their robots without rewriting the WBC solver code.

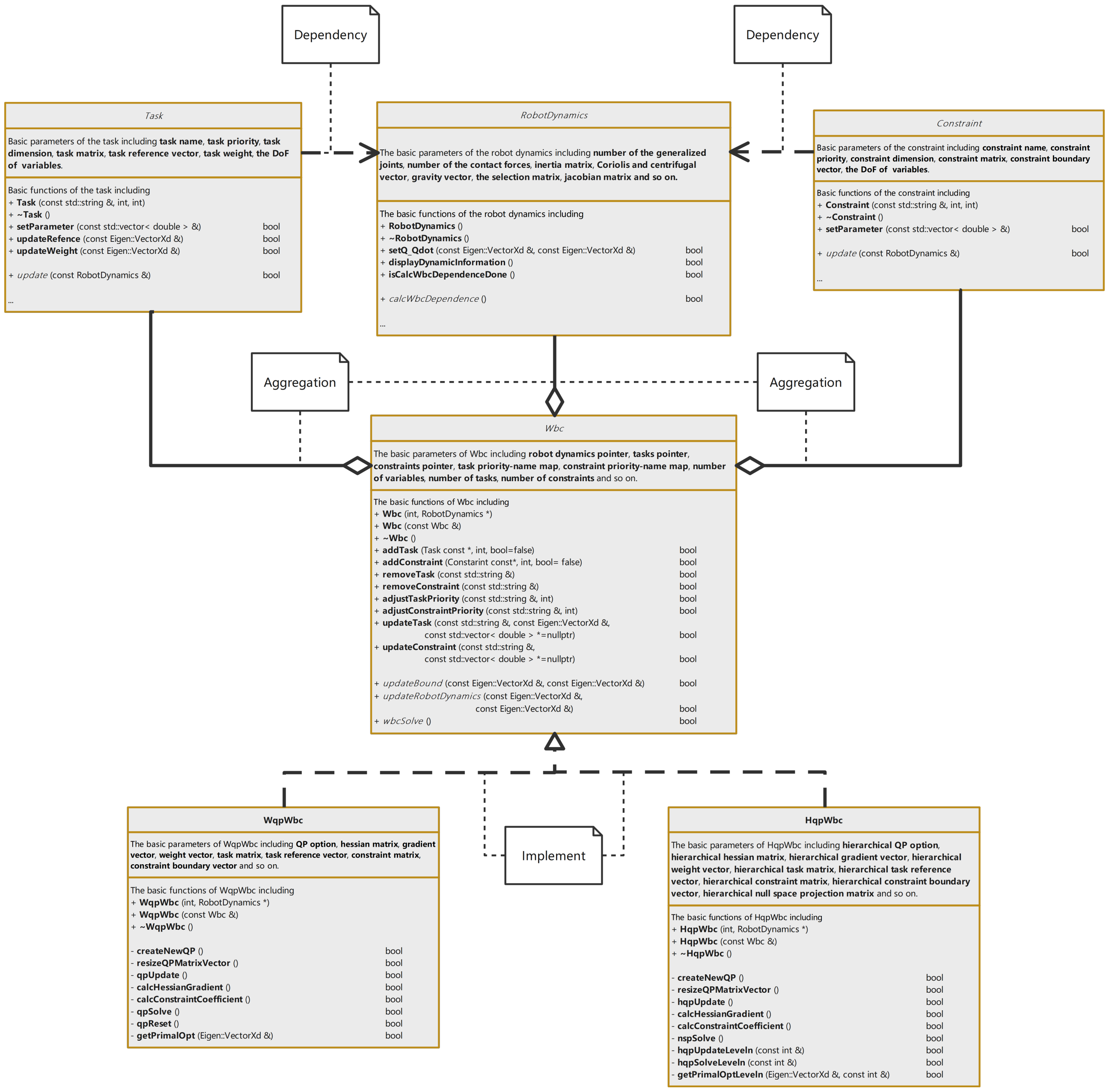

Dependency
Dependency
Task
Basic parameters of the task including task name, task priority, task dimension, task matrix, task reference vector, task weight, the DoF of variables.
Basic functions of the task including
+ Task (const std::string &, int, int)
+ ~Task ()
+ setParameter (const std::vector< double > &) bool
+ updateRefence (const Eigen::VectorXd &) bool
+ updateWeight (const Eigen::VectorXd &) bool
+ update (const RobotDynamics &) bool
...
RobotDynamics
The basic parameters of the robot dynamics including number of the generalized joints, number of the contact forces, inertia matrix, Coriolis and centrifugal vector, gravity vector, the selection matrix, jacobian matrix and so on.
The basic functions of the robot dynamics including
+ RobotDynamics ()
+ ~RobotDynamics ()
+ setQ_Qdot (const Eigen::VectorXd &, const Eigen::VectorXd &) bool
+ displayDynamicInformation () bool
+ isCalcWbcDependenceDone () bool
+ calcWbcDependence () bool
...
Constraint
Basic parameters of the constraint including constraint name, constraint priority, constraint dimension, constraint matrix, constraint boundary vector, the DoF of variables.
Basic functions of the constraint including
+ Constraint (const std::string &, int, int)
+ ~Constraint ()
+ setParameter (const std::vector< double > &) bool
+ update (const RobotDynamics &) bool
...
Aggregation
Aggregation
Wbc
The basic parameters of Wbc including robot dynamics pointer, tasks pointer, constraints pointer, task priority-name map, constraint priority-name map, number of variables, number of tasks, number of constraints and so on.
The basic functions of Wbc including
+ Wbc (int, RobotDynamics *)
+ Wbc (const Wbc &)
+ ~Wbc ()
+ addTask (Task const *, int, bool=false) bool
+ addConstraint (Constarint const*, int, bool= false) bool
+ removeTask (const std::string &) bool
+ removeConstraint (const std::string &) bool
+ adjustTaskPriority (const std::string &, int) bool
+ adjustConstraintPriority (const std::string &, int) bool
+ updateTask (const std::string &, const Eigen::VectorXd &, const std::vector< double > *=nullptr) bool
+ updateConstraint (const std::string &, const std::vector< double > *=nullptr) bool
+ updateBound (const Eigen::VectorXd &, const Eigen::VectorXd &) bool
+ updateRobotDynamics (const Eigen::VectorXd &, const Eigen::VectorXd &) bool
+ wbcSolve () bool
WqpWbc
The basic parameters of WqpWbc including QP option, hessian matrix, gradient vector, weight vector, task matrix, task reference vector, constraint matrix, constraint boundary vector and so on.
The basic functions of WqpWbc including
+ WqpWbc (int, RobotDynamics *)
+ WqpWbc (const Wbc &)
+ ~WqpWbc ()
- createNewQP () bool
- resizeQPMatrixVector () bool
- qpUpdate () bool
- calcHessianGradient () bool
- calcConstraintCoefficient () bool
- qpSolve () bool
- qpReset () bool
- getPrimalOpt (Eigen::VectorXd &) bool
Implement
HqpWbc
The basic parameters of HqpWbc including hierarchical QP option, hierarchical hessian matrix, hierarchical gradient vector, hierarchical weight vector, hierarchical task matrix, hierarchical task reference vector, hierarchical constraint matrix, hierarchical constraint boundary vector, hierarchical null space projection matrix and so on.
The basic functions of HqpWbc including
+ HqpWbc (int, RobotDynamics *)
+ HqpWbc (const Wbc &)
+ ~HqpWbc ()
- createNewQP () bool
- resizeQPMatrixVector () bool
- hqpUpdate () bool
- calcHessianGradient () bool
- calcConstraintCoefficient () bool
- nspSolve () bool
- hqpUpdateLeveln (const int &) bool
- hqpSolveLeveln (const int &) bool
- getPrimalOptLeveln (Eigen::VectorXd &, const int &) bool

Figure 3. The architecture of WBC software.

4.2. High-Performance Master Control System

The WBC software is embedded in a modular master control system named UBT-Master, as shown in Figure 4a. It is designed with the characteristics of real-time solid computation, extensible computing capability, and a configurable interface. As a result, it can realize the different combination configurations of typical hardware platforms, such as ARM, GPU, X86, and DSP, and expand the computing capability in different scenarios. For example, the X86 basic edition can perform 100 GFLOPS per second with the Intel Core i7-7600U.

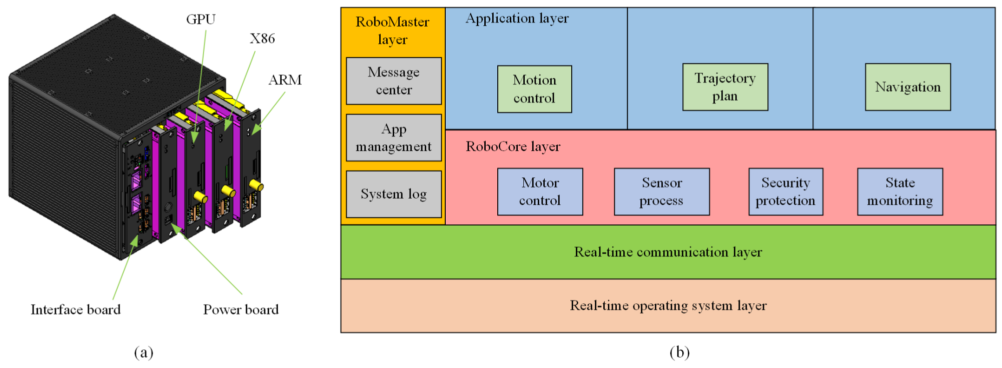


Figure 4. The modular master control system (**a**) and its software architecture (**b**).

The software architecture is illustrated in Figure 4b. The real-time operating system based on the PREEMPT_RT kernel serves real-time applications that process data as it comes in, typically without buffer delays. It ensures that the application's task must be carried out within the defined time constraints. In the real-time communication layer, we apply the high-speed real-time bus communication protocol EtherCAT for short data update times (also called cycle times; $\leq$100 us) with low communication jitter (for precise synchronization purposes; $\leq$1 us). These two aspects can control the time jitter on a microsecond level.

In the upper layer, the roboCore runs in real time and isolates the applications from the hardware platform. As a result, users can develop their applications to meet specific requirements.

5. Proprioceptive Actuation with a Big Reduction Ratio

5.1. Joint-Level Control

Given the inputs, WBC software will output the optimized joint torque. However, the lack of a torque sensor in proprioceptive actuation does not allow direct torque control. A common workaround for this problem is to utilize an admittance coupling to convert joint torque to joint velocity [22]. However, considering that the bandwidth of admittance control will limit the torque tracking performance, we utilize direct current control.

The joint current can be approximated as a linear function of joint torque due to the negligible torque loss in the reducer for proprioceptive actuation with a small reduction ratio. However, the reducer has significant static friction with a large reduction ratio. This stiction translates into joint torque stiction of up to 5 Nm. Consequently, joint friction torque compensation is essential. Moreover, the joint velocity obtained by integrating the optimized joint acceleration can also be added to the current command as a kinematic compensation term to improve the joint impedance [11].

In this research, the final control law for the joint current is calculated as:

$$i^{\mathrm{cmd}} = k_i\left(\tau^{\mathrm{opt}} + k_f\tau^f + \tau^{\dot{q}}\right) \tag{8}$$

where τ^{opt} is the optimized joint torque, τ^f is the joint friction compensation torque, k_f is the corresponding friction compensation coefficient, $\tau^{\dot{q}}$ is the joint kinematic compensation torque. τ^{opt},τ^f, and $\tau^{\dot{q}}$ can be expressed as:

$$\tau^{\mathrm{opt}} = S_a M\ddot{q}^{\mathrm{opt}} + S_a(C+G) - S_a J^{\mathrm{T}} F^{\mathrm{opt}} \tag{9}$$

$$\tau^f = \begin{cases} F_c + F_v\left(\int \ddot{q}^{\mathrm{opt}}dt - \dot{q}^*\right), & \int \ddot{q}^{\mathrm{opt}}dt \geq \dot{q}^* \\ \int \ddot{q}^{\mathrm{opt}}dt\frac{F_c}{\dot{q}^*}, & -\dot{q}^* \leq \int \ddot{q}^{\mathrm{opt}}dt \leq \dot{q}^* \\ -F_c + F_v\left(\int \ddot{q}^{\mathrm{opt}}dt + \dot{q}^*\right), & \int \ddot{q}^{\mathrm{opt}}dt \leq -\dot{q}^* \end{cases} \tag{10}$$

$$\tau^{\dot{q}} = k_{\dot{q}} \left(\int \ddot{q}^{\mathrm{opt}} dt - \dot{q} \right) \tag{11}$$

where τ^{opt} is reorganized from the full dynamics. The joint friction compensation torque τ^{f} is modeled as two parts: Coulomb and viscous friction. F_c is the Coulomb friction and F_v is the viscous friction coefficient. $\dot{q}^{*}$ is a user-defined value to prevent a sudden jump in joint friction compensation torque as the motor rotates reversely. $k_{\dot{q}}$ is a gain acting on the difference between the desired joint velocity $\int \ddot{q}^{\mathrm{opt}} dt$ and the measured joint velocity $\dot{q}$.

5.2. Model Identification

Model identification is an effective method used for obtaining a robot's dynamic parameters. Besides the concerned dynamic parameters such as links' mass, inertia, and center of mass, the joint friction model can also be incorporated into the linearized dynamic equation [23]. Here, the Coulomb–viscous friction model is preferred due to its linear expression.

The main process can be divided into three parts.

5.2.1. Linearization of Dynamic Equation

The recursive Newton-Euler equation is used to reorganize the joint torque τ as a linear function of dynamic parameters π (including joint friction parameters), given that $\tau = Y\pi$. Y is the identification matrix and can be uniquely determined by joint motion q, $\dot{q}$, and $\ddot{q}$.

5.2.2. Optimal Excitation Trajectory

The excitation trajectory is parameterized first by a finite Fourier series function and then optimized for the minimum of a user-defined cost function [24] while satisfying the constraint conditions. The series and base frequency in the Fourier series are set at 5 and 0.1 Hz, respectively. The condition number of the Y matrix is closely relative to the mean square error of identification results and thus selected as the cost function.

5.2.3. Dynamic Parameters Optimization

An optimization problem is constructed to find optimal dynamic parameters π which can minimize the error between the measured and the predicted joint torque by linear dynamic equation.

In addition to the motors lacking torque sensors, a prior calibration of the current–torque coefficient can help to obtain approximate joint torque through the measured joint current.

Figure 5 compares four sets of joint torque in the left leg. The red dotted line indicates the measured torque through the joint current. The blue solid line indicates the predicted torque through identified dynamic parameters, while the green dotted line indicates the predicted torque through identified with base dynamic parameters. The black dotted line indicates the theoretical torque calculated by parameters obtained from the 3D model. There is a large error between the measured and theoretical torque. The main reason is that the joint friction torque is not considered in the dynamics equation calculation.

Not surprisingly, the error between the measured and predicted torque is very small. Meanwhile, several torque jumps are measured when the motor changes its rotating direction. Thanks to the Coulomb friction term in our friction model, the predicted torque curve follows the measured one closely. It directly proves that the identified dynamic parameters can reflect the actual dynamic characteristics and can be used to model the physical control system.

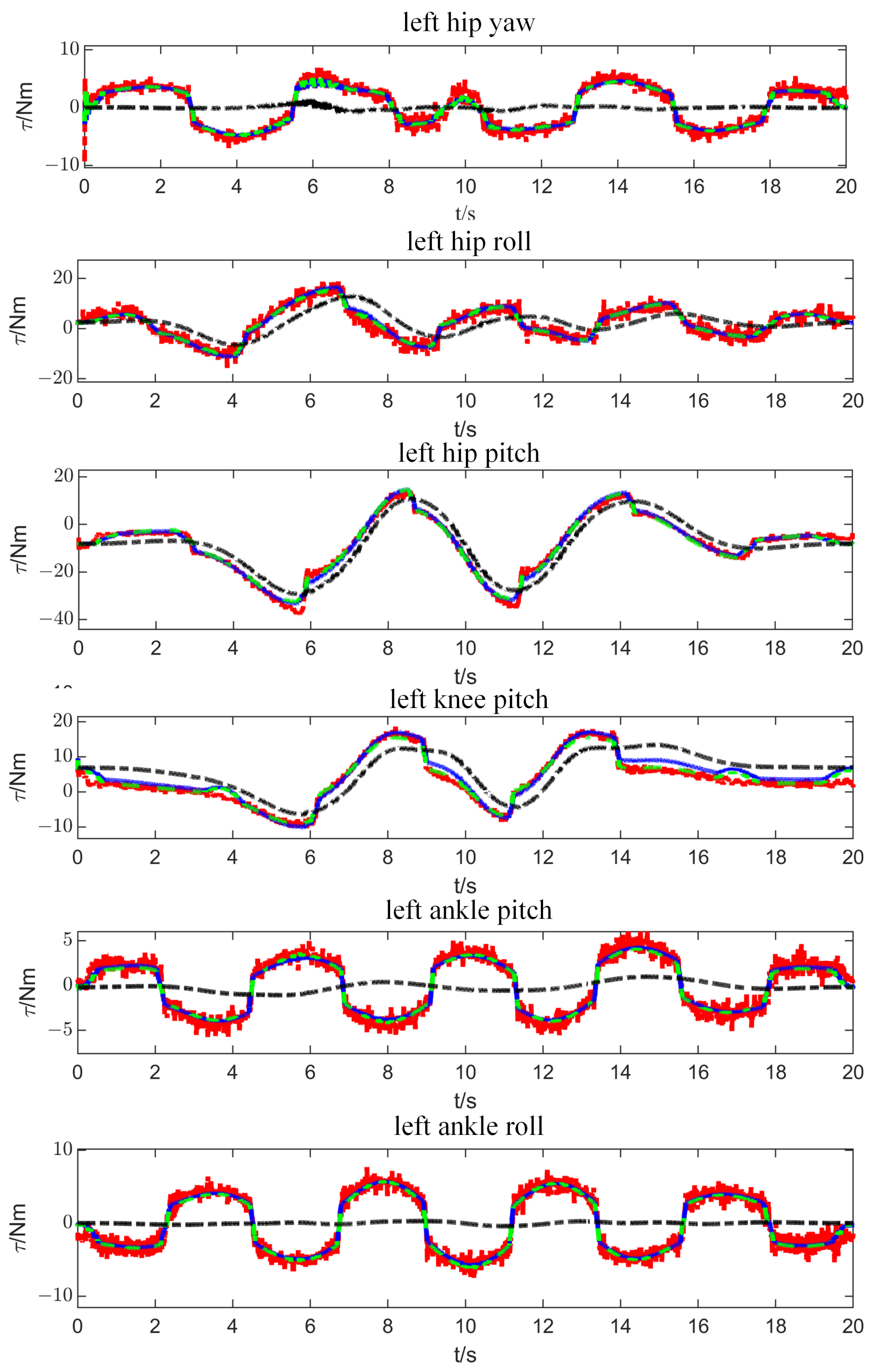


Figure 5. The comparison of joint torque in the left leg.

6. Experimental Results and Discussion

The control approach, as mentioned above, is experimentally evaluated on the Walker3 humanoid. In addition, the balance performance is evaluated in different scenarios: push recovery on the ground, balancing on a seesaw, and push recovery on two moving skateboards. A summary of experimental videos is available at https://youtu.be/g79tWSATmhA (accessed on 21 August 2022).

6.1. Push Recovery on the Ground

The robot is subjected to impulses from X and Y directions while standing on the ground. A ball weighing 5 kg is used to generate impulses at the robot's torso. The ball's momentum with its known mass and velocity can quantify the impulses. Figure 6 shows a series of snapshots when the robot is stroked along the X axis and Y axis. Eight impulses along the X axis and seven impulses along the Y axis are exerted on the robot.

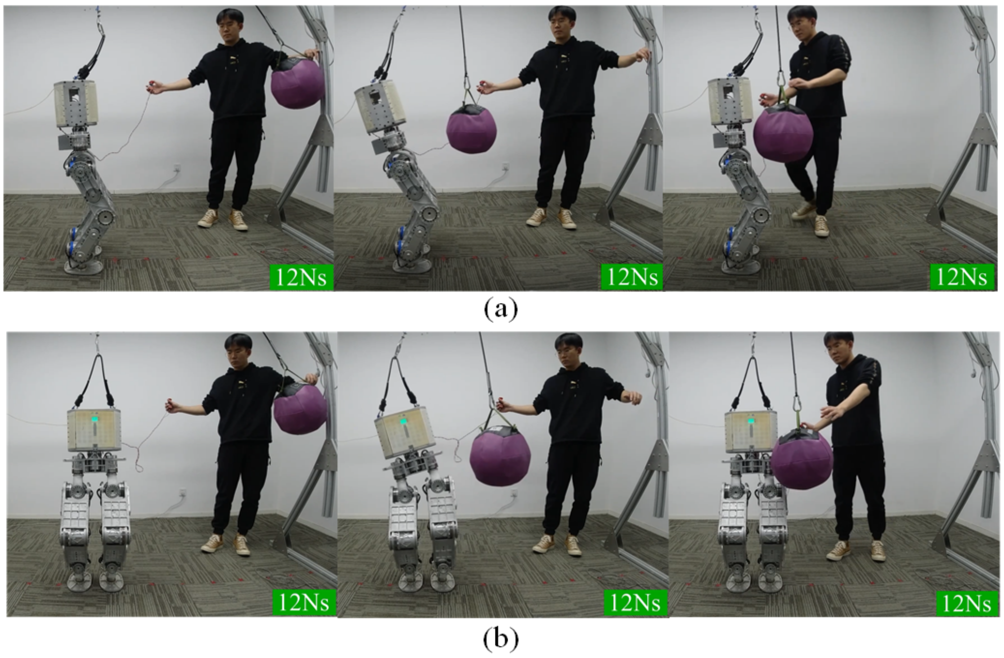


Figure 6. The balancing behavior in push recovery scenario along the X axis (**a**) and the Y axis (**b**).

For the push recovery along the X axis, the index and magnitude of impulses are listed in Table 2.

Table 2. The magnitude of impulses along the X axis.

Index	1	2	3	4~8
Magnitude	8 Ns	10 Ns	11 Ns	12 Ns

Several key features of the tasks are drawn in Figure 7. First, it can be seen that the peak of the foot pitch angle gradually increases as the impulses increase. The foot pitch angle reaches up to 1.5° when the impulse reaches its maximum 12 Ns. Theoretically, the foot pitch angle should be zero because the CoP constraint has been considered in the hierarchical optimization. Such a small pitch angle is acceptable given a carpet between the foot and the ground.

Figure 7b draws the CoM position along the X axis. The CoM position fluctuates in the range of −29~56 mm under the continuous impulses. Figure 7c draws the pitch angle of the torso. The robot tries to rotate the upper body to preserve the foot posture and the CoM position, which looks similar to a human rotating its trunk to maintain balance. The maximum value of the torso pitch angle is 25.7°. Increasing the impulse will cause a balance failure due to the torso pitch angle exceeding the joint angle limits.

A tiny stability error exists for these three tasks in Figure 7, although the tasks' feedback control law works. Such a control command resulting from the tiny error will not drive the joint to move. In the video, the robot behaves in a way that it cannot recover to its original state after the disturbances.

Figure 8a compares the measured ZMP with the optimized one. The measured ZMP is calculated using the force sensor, while the optimized ZMP is calculated using the optimized foot contact force. Ideally, the measured ZMP should follow the optimized ZMP closely, but there is a slight difference. It means that all motors in the robot cannot generate the required optimized foot contact force. The main reason for that is the joint torque error due to the limited identification accuracy of the joint friction model in Section 5.2.

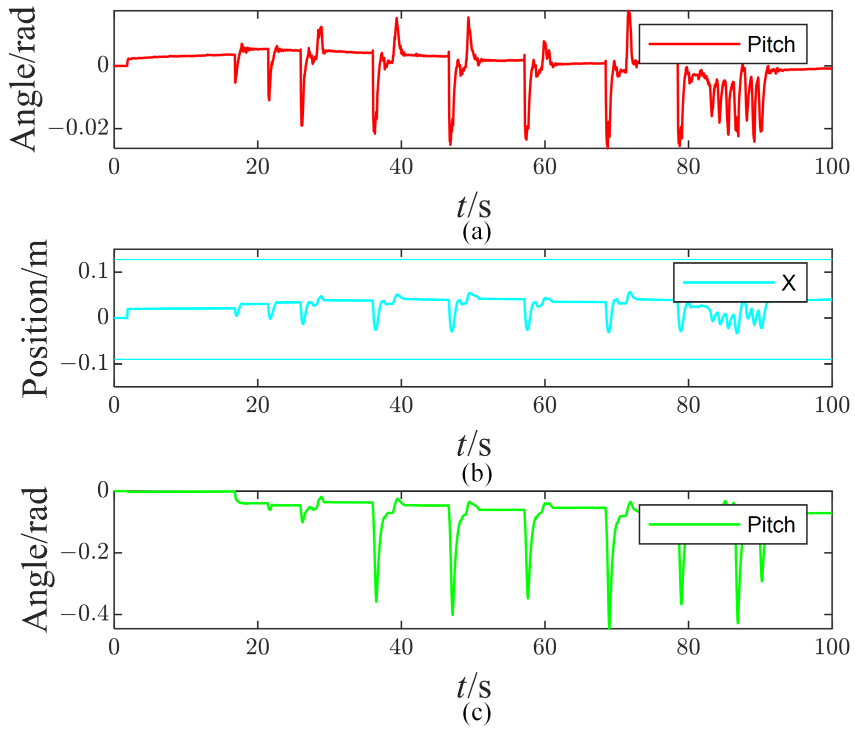


Figure 7. The measured pitch angle of the right foot (**a**), CoM position along the X axis (**b**), and the pitch angle of the torso (**c**).

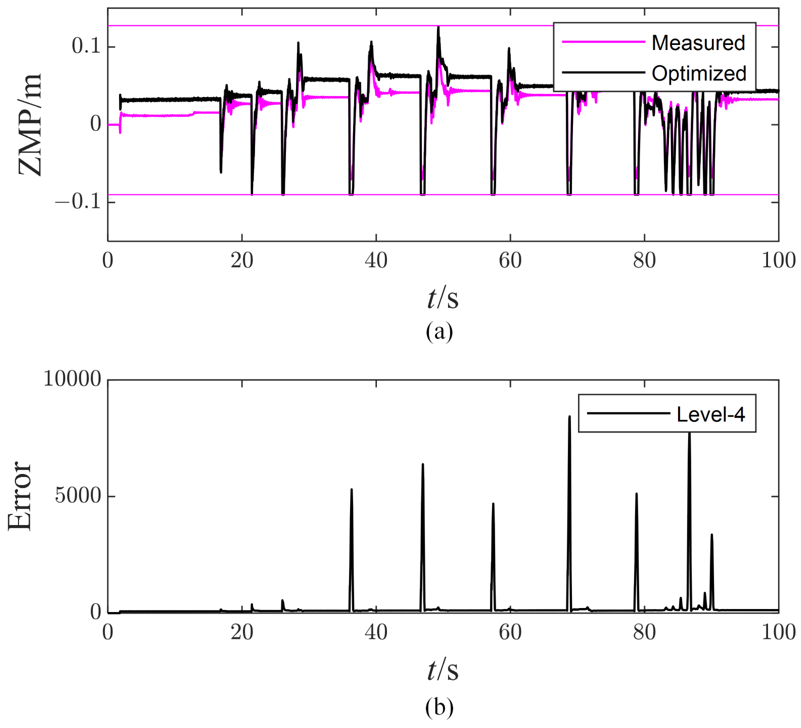


Figure 8. The measured and optimized ZMP of right foot (**a**) and the task error of the lowest priority task (**b**).

In addition, there are several times that the optimized ZMP reaches up to its boundary. This indicates a strong linear relationship between the optimal foot contact force, leading to the dimensionality reduction in optimal variables. As a result, the robot tends to sacrifice the lowest-priority task due to a lack of DoFs. As proof, Figure 8b plots the task error $error = \|Ax - b\|^2$ of the lowest-priority task. It can be seen that the task error is small enough when the first three impulses act on the robot. However, there will always be a sharp peak with the magnitude of 10^3 as long as the impulses increase to 12 Ns. The huge task error will deteriorate the control performance. The robot is originally designed to show its torso compliance according to the PD parameters, but the compliance characteristic cannot be ensured due to the task error. The inappropriate compliance will enlarge the amplitude of the torso pitch angle, which further limits the performance of push recovery. These push recovery test results show that the hQP-WBC method can handle multi-tasks well according to their priority and thus improve the robustness of the robot under environmental disturbance.

6.2. Balancing on a Seesaw

The robot balances the inclination disturbances along the X axis and Y axis on a seesaw. Figure 9 shows how the robot adapts to the inclined surface to maintain balance, and Figure 10 plots the measured orientation and angular velocity of the right foot. Unfortunately, no additional IMU is mounted on the seesaw to measure its real movement. Nevertheless, the estimated foot state approximates the seesaw because the ZMP resides in the foot polygon throughout the test.

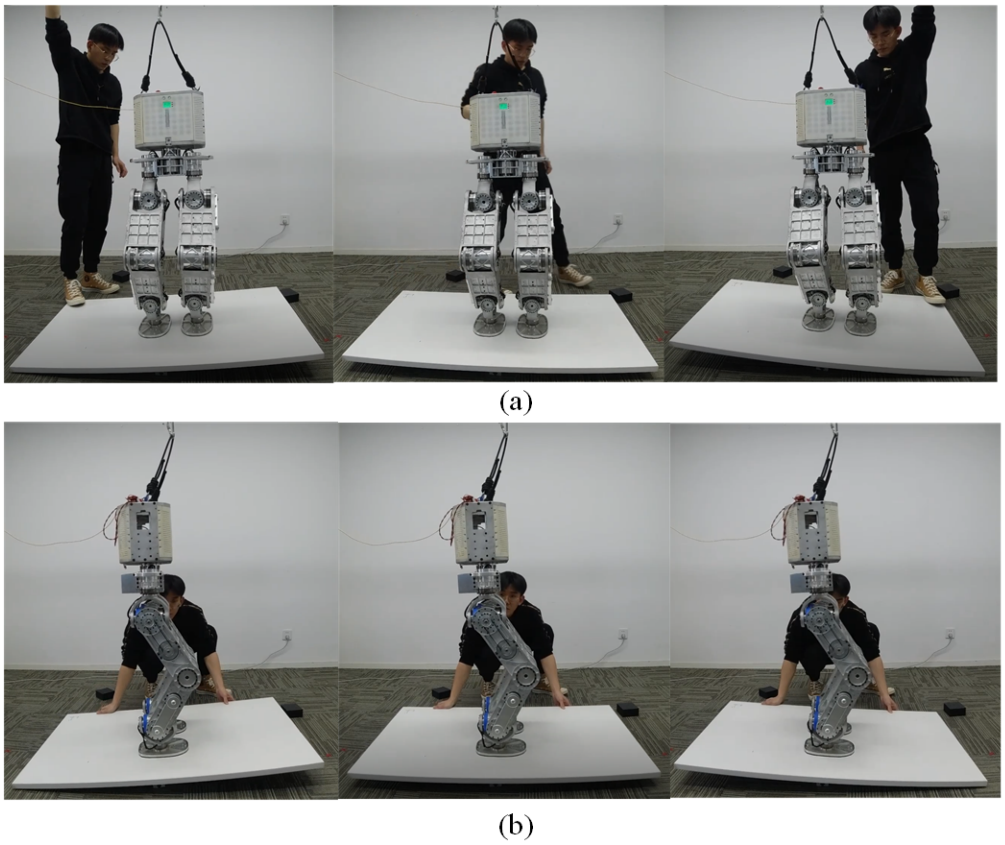


Figure 9. The balancing behaviors when the seesaw rotates along the X axis (**a**) and the Y axis (**b**).

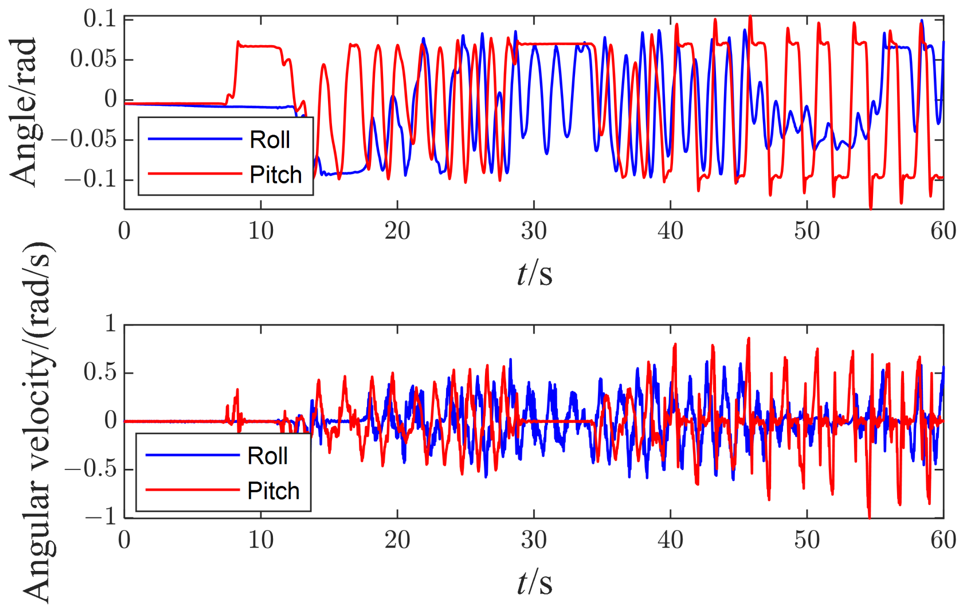


Figure 10. The measured orientation and angular velocity of the right foot.

The amplitude of inclined angles along both axes reaches up to 6°, as shown in Figure 10. Meanwhile, the maximum angular velocity along the Y axis is 1 rad/s, slightly larger than that on the X axis (0.6 rad/s). The reason mainly relies on the joint friction's minor influence on the balance performance when the seesaw inclines along the Y axis. The robot needs to modulate its ankle pitch joint to adapt to the inclined seesaw along the Y axis while modulating its ankle roll joint and the length of both legs to adapt to the inclined seesaw along the X axis. Here, a low-pass filter has processed the angular velocity data with a cutoff frequency of 20 Hz.

Figure 11 shows the response of the right foot's ZMP during the disturbances. The components of ZMP along the X and Y axes did not exceed the constrained boundary defined by foot geometry. All these prove that the task planner works well, and the robot can adjust the tasks' target to resist the disturbance from the seesaw.

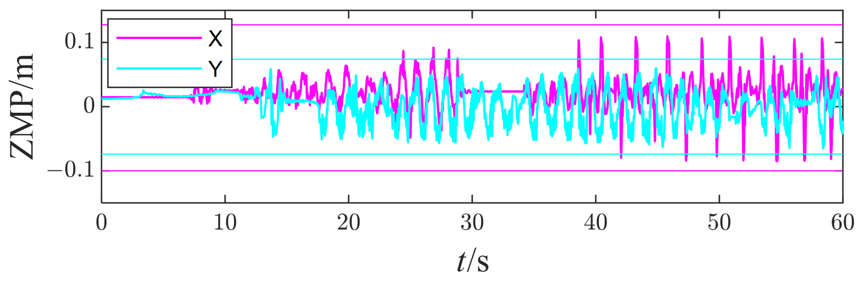
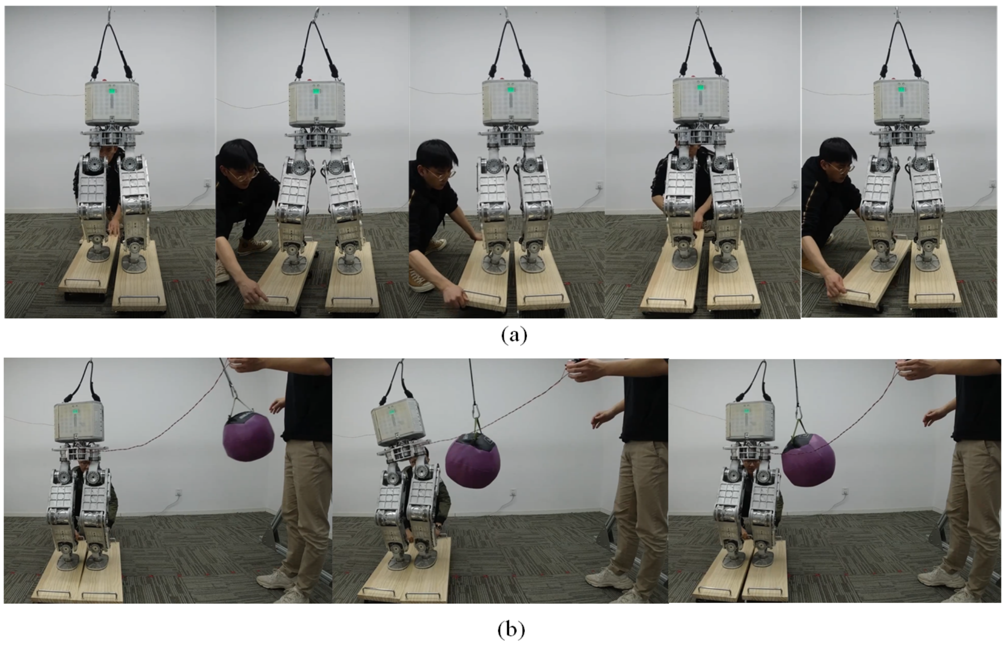


Figure 11. The measured ZMP of the right foot.

6.3. Push Recovery on Two Moving Skateboards

The final experiment in balance maintenance on two moving skateboards is shown in Figure 12. The two feet of the robot rest on two moving skateboards separately and suffer inclination and shift disturbances independently. The right-moving skateboard is actuated by hands to translate along the X and Y axes and rotate along the X, Y, and Z axes. At the same time, the left one is locked (Figure 12a) to evaluate the disturbance rejection capability of the robot. Figure 13 plots the measured velocity of the right foot with each direction tested separately. The shift disturbance along the Z axis is indirectly measured by rotating the skateboard about the Y axis, where the foot will rise as the tilt angle increases. The robot can resist the moving skateboard disturbance with the maximum velocities 0.94 m/s, 0.89 m/s, and 0.47 m/s along the X, Y, and Z axes, and the maximum angular velocities 1.8 rad/s, 1.4 rad/s, and 0.5 rad/s along the X, Y, and Z axes.


(a)

(b)

Figure 12. The balancing behaviors when the right support moves in all directions (**a**) and the balancing behaviors in push recovery on the moving support scenario (**b**).

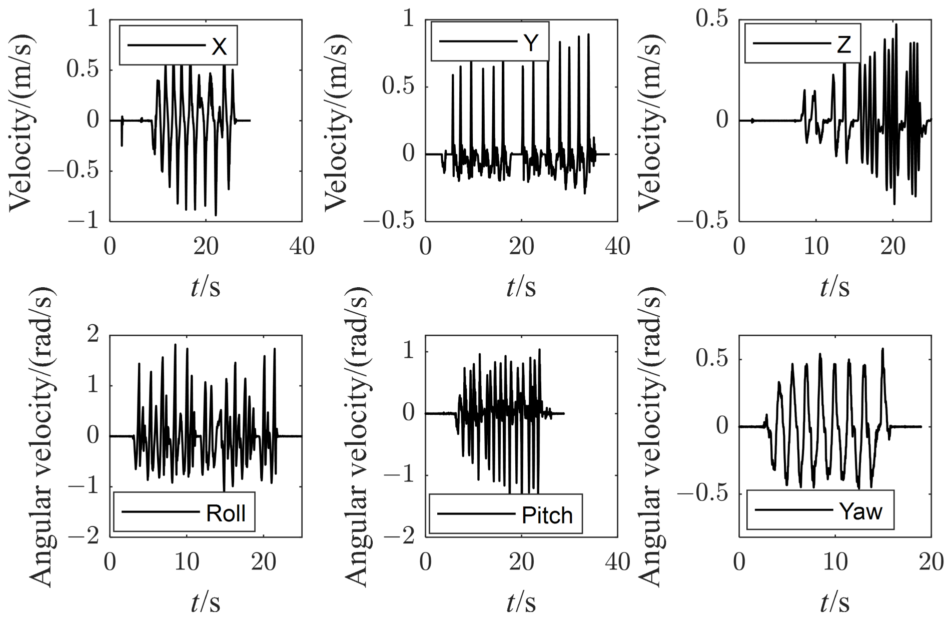


Figure 13. The measured velocity of the right foot with each direction.

The robot can also maintain balance when the two skateboards have different inclination angles and translate back and forth without phase velocity. Meanwhile, when 8Ns impulses along the Y axis are exerted on the robot as the skateboards keep moving, the robot generates a large torso rotation to keep balance (Figure 12b).

The onboard computer needs to solve the hierarchical optimization, which contains four quadratic optimization problems in 1 ms, to achieve a 1 kHz control loop. Figure 14 plots the computation time of the whole algorithm, including the state estimation, trajectory planning, whole-body control, and joint-level control parts. The average and the maximum computation times are 0.363 ms and 0.639 ms, respectively. As the most time-consuming portion, the computation time of the whole-body control part is also plotted here in the yellow line. The average computation time is 0.321 ms, which takes up about 88 percent of the time, leaving 0.042 ms for the other parts. We must solve the quadratic optimization and the null-space projection matrix sequentially in the whole-body control part. About 72% of the computation time is used for quadratic optimization, and the left 28% is used for the null-space projection matrix. The quadratic optimization is solved through a C++ open-source QP solver, qpOASES [25], which implements the active set algorithm. The null-space projection matrix must calculate the pseudo-inverse of the matrix first, and the complete orthogonal decomposition algorithm implemented in the Eigen matrix library is used here.

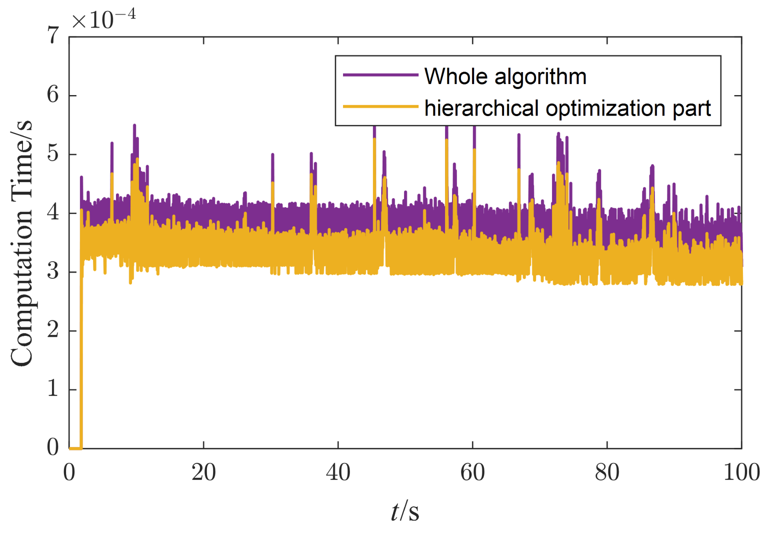


Figure 14. The computation time of the algorithm.

7. Conclusions

This paper aims to make the proprioception-actuated humanoid robot capable of dynamic balance. For this purpose, tasks and constraints are assigned in a hierarchy of task priorities.

- A real-time computation is achieved through computationally efficient WBC software and a reduced hierarchical whole-body control scheme.
- UBTMaster, a modular control system with real-time communication and powerful computing capabilities, is designed.
- The key dynamic parameters are identified to deal with the nonlinear friction and imprecision of the model of the robot.

Results show that the predicted torque is close to the measured, and the average residual is less than 1 Nm.

After fully considering these aspects in a system, the balance performance of the humanoid robot Walker3 is tested in various scenarios. The robot can be balanced with continuous impulses on the X and Y axes up to 12 Ns. Like human behavior, the robot tends to rotate its upper body to maintain foot posture and CoM position. In addition, we found reasons to limit push recovery performance. When the optimal variable reaches the constraint boundary, dimensionality reduction will cause the system to sacrifice the task with the lowest priority. Because of the strict hierarchy, high-priority tasks are not affected. An effective solution is to provide more redundant degrees of freedom, such as adding arms to the robot.

When Walker3 stands on the seesaw, it can actively adapt to the tilted surface to maintain balance. Different from [15], the state of the seesaw is estimated without an additional IMU and then used to update the trajectory of the task. The experimental results show that the inclination angle along the two axes reaches 6°. Meanwhile, the maximum angular velocities of the X axis and Y axis are 0.6 rad/s and 1 rad/s, respectively, which is about 1.7~2.8 times the performance of the torque control robot COMAN [26].

To further exploit the robot's adaptability to uncertain perturbations, we placed Walker3 on two moving skateboards and applied tilt and displacement perturbations. The maximum velocities of the robot in the X, Y, and Z axes are 0.94 m/s, 0.89 m/s, and 0.47 m/s, respectively. The maximum angular velocities of the X, Y, and Z axes are 1.8 rad/s, 1.4 rad/s, and 0.5 rad/s, respectively. When the two skateboards have different inclination angles and no phase velocity in front and back translation, the robot can even resist 8 Ns impulse.

The results show that the proprioception–actuation robot can perform quite well as the torque-controlled robot under a strict hierarchical structure, real-time calculation, and careful joint friction treatment. We hope to improve the balancing framework through intelligent planning to cope with the greater disruption of future work. Furthermore, the results show that the introduction of online predictive control technology will significantly improve the robustness of the robot.

At present, our research focuses on the effectiveness of the proposed method. Our research goal is to focus on the solution of biped robot dynamic balance problem from the perspective of algorithm, software, and hardware, and the adopted Walker is used to test this solution. In terms of the performance of the robot, Walker robot needs to increase its arms to improve its balance performance more fully, which is comparable to, or goes beyond, the Atlas robot.

Author Contributions: Conceptualization, Y.X. and M.Z.; methodology, Y.X.; software, Y.X. and J.W.; validation, Y.X. and H.D.; formal analysis, Y.X., X.R. and L.H.; investigation, Y.X.; resources, H.D.; data curation, Y.X.; writing—original draft preparation, Y.X.; writing—review and editing, Y.X. and M.Z.; visualization, Y.X.; supervision, M.Z.; project administration, M.Z; funding acquisition, M.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science and Technology Innovation 2030-Key Project under grant 2021ZD0201402.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Mikolajczyk, T.; Mikołajewska, E.; Al-Shuka, H.F.N.; Malinowski, T.; Kłodowski, A.; Pimenov, D.Y.; Paczkowski, T.; Hu, F.; Giasin, K.; Mikołajewski, D.; et al. Recent Advances in Bipedal Walking Robots: Review of Gait, Drive, Sensors and Control Systems. *Sensors* **2022**, *22*, 4440. [CrossRef] [PubMed]
2. Mikolajczyk, T.; Fas, T.; Malinowski, T.; Romanowski, L. Prototype Model of Walking Robot. *Appl. Mech. Mater.* **2014**, *613*, 21–28. [CrossRef]
3. Ficht, G.; Behnke, S. Bipedal Humanoid Hardware Design: A Technology Review. *Curr. Robot. Rep.* **2021**, *2*, 201–210. [CrossRef]
4. Li, Z.; Zhou, C.; Tsagarakis, N.; Caldwell, D. Compliance control for stabilizing the humanoid on the changing slope based on terrain inclination estimation. *Auton. Robot.* **2016**, *40*, 955–971. [CrossRef]
5. Nenchev, D.N. Reaction null space of a multibody system with applications in robotics. *Mech. Sci.* **2013**, *4*, 97–112. [CrossRef]
6. Dietrich, A.; Ott, C.; Albu-Schäffer, A. An overview of null space projections for redundant, torque-controlled robots. *Int. J. Robot. Res.* **2015**, *34*, 1385–1400. [CrossRef]
7. Nakamura, Y.; Hanafusa, H.; Yoshikawa, T. Task-priority based redundancy control of robot manipulators. *Int. J. Robot. Res.* **1987**, *6*, 3–15. [CrossRef]
8. Kajita, S.; Kanehiro, F.; Kaneko, K.; Fujiwara, K.; Harada, K.; Yokoi, K.; Hirukawa, H. Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453), Las Vegas, NV, USA, 27–31 October 2003; Volume 2, pp. 1644–1650.

9. 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. [CrossRef]
10. Feng, S.; Whitman, E.; Xinjilefu, X.; Atkeson, C.G. Optimization-based full body control for the darpa robotics challenge. *J. Field Robot.* **2015**, *32*, 293–312. [CrossRef]
11. Koolen, T.; Bertrand, S.; Thomas, G.; de Boer, T.; Wu, T.; Smith, J.; Englsberger, J.; Pratt, J. Design of a Momentum-Based Control Framework and Application to the Humanoid Robot Atlas. *Int. J. Hum. Robot.* **2016**, *13*, 1650007. [CrossRef]
12. De Lasa, M.; Mordatch, I.; Hertzmann, A. Feature-based locomotion controllers. *ACM Trans. Graph. TOG* **2010**, *29*, 1–10. [CrossRef]
13. Kanoun, O.; Lamiraux, F.; Wieber, P.B. Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *IEEE Trans. Robot.* **2011**, *27*, 785–792. [CrossRef]
14. Escande, A.; Mansard, N.; Wieber, P.B. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *Int. J. Robot. Res.* **2014**, *33*, 1006–1028. [CrossRef]
15. Herzog, A.; Rotella, N.; Mason, S.; Grimminger, F.; Schaal, S.; Righetti, L. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Auton. Robot.* **2016**, *40*, 473–491. [CrossRef]
16. Bellicoso, C.D.; Gehring, C.; Hwangbo, J.; Fankhauser, P.; Hutter, M. Perception-less terrain adaptation through whole body control and hierarchical optimization. In Proceedings of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), Cancun, Mexico, 15–17 November 2016; pp. 558–564.
17. Wensing, P.M.; Wang, A.; Seok, S.; Otten, D.; Lang, J.; Kim, S. Proprioceptive Actuator Design in the MIT Cheetah: Impact Mitigation and High-Bandwidth Physical Interaction for Dynamic Legged Robots. *IEEE Trans. Robot.* **2017**, *33*, 509–522. [CrossRef]
18. Kim, D.; Zhao, Y.; Thomas, G.; Fernandez, B.R.; Sentis, L. Stabilizing Series-Elastic Point-Foot Bipeds Using Whole-Body Operational Space Control. *IEEE Trans. Robot.* **2016**, *32*, 1362–1379. [CrossRef]
19. Lee, S.H.; Goswami, A. A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Auton. Robot.* **2012**, *33*, 399–414. [CrossRef]
20. Wensing, P.M.; Orin, D.E. Improved computation of the humanoid centroidal dynamics and application for whole-body control. *Int. J. Hum. Robot.* **2016**, *13*, 1550039. [CrossRef]
21. Featherstone, R. *Rigid Body Dynamics Algorithms*; Springer: Berlin/Heidelberg, Germany, 2014.
22. Dietrich, A.; Wimbock, T.; Albu-Schaffer, A.; Hirzinger, G. Reactive Whole-Body Control: Dynamic Mobile Manipulation Using a Large Number of Actuated Degrees of Freedom. *IEEE Robot. Autom. Mag.* **2012**, *19*, 20–33. [CrossRef]
23. Gaz, C.; Magrini, E.; De Luca, A. A model-based residual approach for human-robot collaboration during manual polishing operations. *Mechatronics* **2018**, *55*, 234–247. [CrossRef]
24. Siciliano, B.; Khatib, O. *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2016.
25. Ferreau, H.J.; Kirches, C.; Potschka, A.; Bock, H.G.; Diehl, M. qpOASES: A parametric active-set algorithm for quadratic programming. *Math. Program. Comput.* **2014**, *6*, 327–363. [CrossRef]
26. Li, Z.; Tsagarakis, N.G.; Caldwell, D.G. Stabilizing humanoids on slopes using terrain inclination estimation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 4124–4129.