

Article

A Unified Control Framework for Self-Balancing Robots: Addressing Model Variations in Wheel-Legged Platforms and Human-Carrying Wheelchairs

Guiyang Xin ^{1,2} , Boyu Jin ³, Chen Liu ⁴ and Mian Jiang ^{5,*}¹ State Key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin 150001, China; guiyang.xin@dlut.edu.cn² School of Biomedical Engineering, Dalian University of Technology, Dalian 116024, China³ College of Future Technology, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511400, China⁴ School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China⁵ School of Mechatronic Engineering and Automation, Foshan University, Foshan 528225, China

* Correspondence: mjiang@fosu.edu.cn

Abstract

Self-balancing robots, with their compact size, are capable of achieving high agility. Small wheel-legged self-balancing robots have demonstrated significant potential across various applications. However, expanding small self-balancing robots to larger sizes to serve as personal transport tools is a more attractive and impactful direction than further miniaturization or confinement to niche laboratory demonstrations. This paper presents the development of a small self-balancing robot, which is then scaled up to a larger version designed to carry human passengers as a self-balancing wheelchair. A unified control framework, built around a shared core of online model-updating LQR for balance and PD for steering, is applied to both robots. This core is supplemented with platform-specific modules, such as a dedicated leg controller for the wheel-legged robot, to handle distinct dynamic maneuvers. The LQR controller is implemented for balance control in both robots. Additionally, a dedicated leg controller is applied exclusively to the small wheel-legged robot to enable dynamic maneuvers, such as jumping. A series of experiments conducted with the final prototypes validate the effectiveness of the control systems and highlight the robots' application potential.

Keywords: self-balancing robots; wheel-legged robots; LQR

Academic Editors: Sungkeun Yoo, Taewon Seo and Taegyun Kim

Received: 8 October 2025

Revised: 16 November 2025

Accepted: 20 November 2025

Published: 22 November 2025

Citation: Xin, G.; Jin, B.; Liu, C.; Jiang, M. A Unified Control Framework for Self-Balancing Robots: Addressing Model Variations in Wheel-Legged Platforms and Human-Carrying Wheelchairs. *Sensors* **2025**, *25*, 7144. <https://doi.org/10.3390/s25237144>

Copyright: © 2025 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

The concept of self-balancing robots has gained considerable attention in both research and industry due to their versatility in various applications, ranging from personal mobility devices to delivery systems and assistive robots. Notable examples include two-wheeled robots (e.g., Segway [1] and JOE [2]) and wheel-legged robots (e.g., Ascento [3] and Handle [4]). While self-balancing robots offer significant agility, it is important to delineate the application domains where they hold a distinct advantage over statically stable alternatives, such as four-wheeled or tracked robots. The primary strengths of self-balancing platforms lie in their compact footprint, zero turning radius, and dynamic obstacle-crossing capabilities. These traits make them ideally suited for operations in tight, human-populated environments like crowded offices, factory floors with narrow aisles, or even the interior of aircraft and trains, where the larger turning circle of a conventional robot would be

impractical. From a theoretical perspective, self-balancing systems are interesting platforms because they are nonlinear, underactuated, and inherently unstable [5–8]. These characteristics make them challenging to control, especially when subjected to external disturbances or model variations.

In this study, we developed two types of self-balancing robots: one is a conventional, small-sized wheel-legged robot, and the other is a larger self-balancing robot capable of carrying a person. The primary challenge in controlling the wheel-legged robot lies in the changing leg dynamics, which result in variations in the system model. In contrast, the challenge in the self-balancing wheelchair is the variation in system equilibrium caused by the unknown weight and center of mass (CoM) of the passenger.

1.1. Related Works

Without model variation, there are mature solutions for the control of self-balancing robots. A backstepping controller dedicated to balance control and a PID controller for velocity tracking were combined to control a two-wheeled self-balancing robot in [9]. To overcome external disturbances, researchers have proposed various optimization methods for tuning PID gains, such as artificial bee colony optimization, particle swarm optimization, grey wolf optimization, and the cuckoo search algorithm [10,11]. However, PID controllers with fixed gains are not ideal for nonlinear systems like self-balancing robots. As a result, adaptive fuzzy PID control [12,13] and neural network-based PID control [14] have been developed to adjust PID gains online, improving system performance under varying conditions.

Model-based controllers, such as LQR and MPC, have demonstrated greater robustness compared to PID controllers [15], particularly for balance control. Self-balancing robots are often modeled as inverted pendulums [2,16], which can maintain an upright position and stable movement despite their inherent instability. However, the mobile inverted pendulum model does not account for wheel inertia. Neglecting wheel inertia can lead to an underestimation of the system's overall inertia, resulting in a controller that is too aggressive and performs poorly in practice. The wheeled inverted pendulum (WIP) model [17,18] provides a more accurate dynamic representation. While LQR controllers improve robustness, their reliance on linearization around the equilibrium state limits their effectiveness to small pitch angles. To address this limitation, nonlinear MPC controllers [15,19–22] have been proposed for extreme dynamic maneuvers. However, these methods are computationally intensive and challenging to implement in real-time, especially when the system model changes dynamically.

Recent advancements in adaptive control and reinforcement learning have also shown promise for self-balancing robots. For instance, ref. [23] proposed an adaptive zero-offset angle identification algorithm to compensate for deviations caused by changes to the robot's center of gravity, while ref. [24] applied deep reinforcement learning to optimize control policies for wheel-legged robots in unstructured environments. Additionally, ref. [25] introduced a disturbance observer-based approach to enhance the robustness of wheeled inverted pendulum systems, and ref. [26] provided a comprehensive review of adaptive control techniques for underactuated balance robots. These approaches highlight the growing interest in combining traditional control methods with modern machine learning techniques to enhance system performance [27,28].

1.2. Overview

In this paper, we propose a unified control framework to address the model variation problem in wheel-legged self-balancing robots and human-carrying self-balancing wheelchairs. The two custom-built self-balancing robots are shown in Figure 1. The small

robot employs a five-bar linkage mechanism as the leg, acting as an active suspension system, enabling the robot to overcome common indoor and urban obstacles such as thresholds, curbs, and uneven pavement; traverse uneven terrain; and even jump. The self-balancing wheelchair utilizes a force/torque sensor to estimate the person's mass and center of mass (CoM). The dynamic models of both robots can be updated in real time. An LQR controller, with online computation, is employed to maintain balance based on the updated dynamic models, while the rotation speed is controlled by a PD controller. Crucially, this same LQR controller also regulates the robot's forward/backward translational speed by tracking a desired velocity command, leveraging its inherent ability to control the entire sagittal-plane state vector. The proposed system is validated through extensive tests, demonstrating its ability to maintain balance, navigate obstacles, and perform dynamic maneuvers such as jumping. The main contributions of the paper are listed as follows.

1. A model updating method with an online computing LQR controller is proposed to solve the model changing problem for both robots, demonstrating versatility across platforms. We integrate force/torque sensor feedback for CoM estimation, enabling automatic recalibration of the equilibrium point. This is critical for passenger-carrying applications, where traditional LQR fails due to unmodeled CoM offsets.
2. Human-carrying, jumping, and obstacle-crossing experiments were conducted to demonstrate the real-world application potential of both robots.

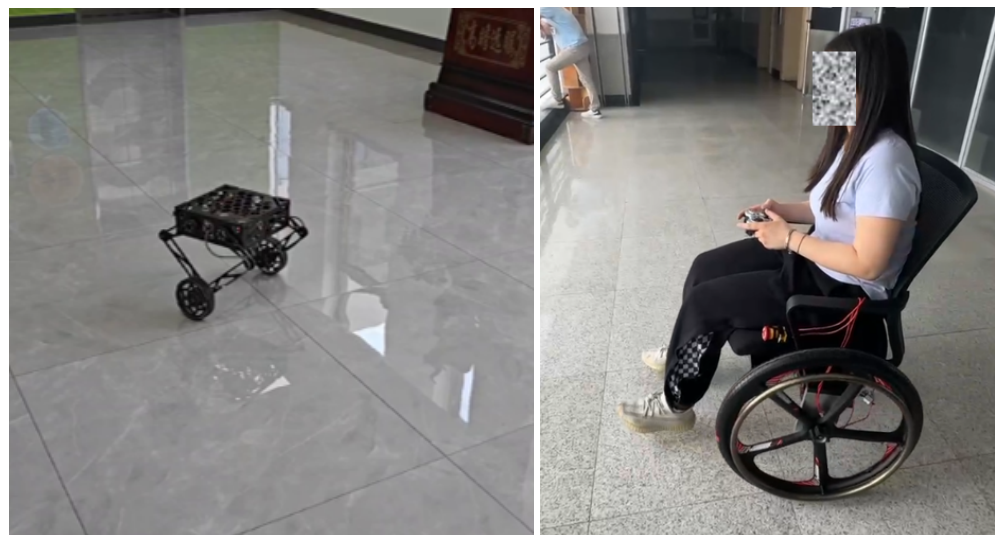


Figure 1. A small wheel-legged self-balancing robot and a large self-balancing wheelchair. See supplementary video [29] for dynamic maneuvers.

The remainder of this paper is organized as follows: Section 2 presents the decoupled dynamic models of the self-balancing robots. The control framework and model update method are described in Section 3. Experimental results and analysis are provided in Section 4. Finally, the conclusions are presented in Section 5.

2. Problem Modification

The dominant part of a self-balancing robot system lies in the sagittal plane, which is responsible for balance control and translational motion. We will present the sagittal plane dynamics in Section 2.1. The rotation dynamics are ignored since a PD controller is used to control the turning motion. The wheel-legged self-balancing robot has a five-bar linkage closed-loop mechanism as the leg mechanism. We present the kinematics and statics of the leg mechanism in Section 2.2.

2.1. Sagittal Plane Dynamics

Self-balancing robots are usually formulated as mobile wheeled inverted pendulum models or cart-pole systems for balance control. However, the cart-pole model cannot include the wheel inertia in the dynamics. In this work, we choose to use the mobile wheeled inverted pendulum model to derive the system dynamics. The two robots can be projected on their sagittal planes, as shown in Figure 2, where the WIP model is a 2D model. The derived WIP model is standard and aligns with the established dynamics in the literature, such as [17]. θ_w and θ_b are the wheel's rotation angle and the inclination angle of the body, respectively. Here, θ_w is a continuous rotation angle, not restricted to a principal value. m_b , I_b , m_w and I_w are the masses and inertias of the body and the wheel of the 2D WIP model. l denotes the length between the wheel axle and the CoM of the body. r is the wheel radius. The coordinates of the wheel and body are denoted by (x_w, z_w) and (x_b, z_b) , respectively. The positions of the system can be given by

$$x_b = l \sin \theta_b + x_w \quad (1)$$

$$z_b = l \cos \theta_b \quad (2)$$

$$x_w = r \theta_w \quad (3)$$

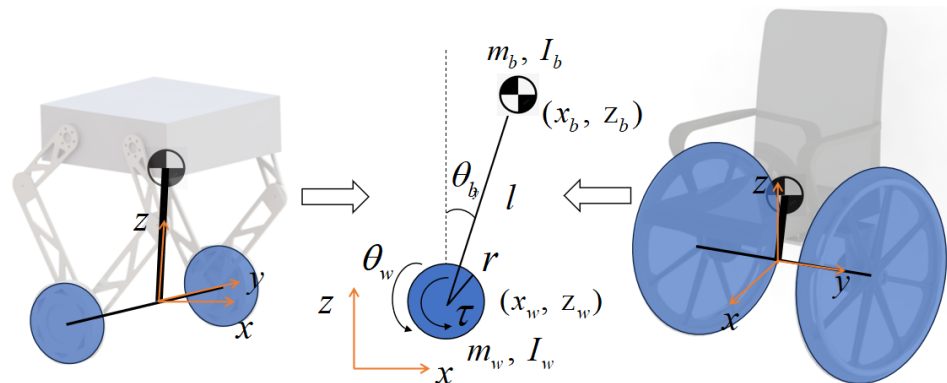


Figure 2. Two-dimensional mobile wheeled inverted pendulum system model projected on the sagittal plane (x-z plane).

The Lagrange's equation of motion is used to derive the dynamics of the system. The kinetics and potential energy are computed as follows:

$$V_b = 0.5m_b(\dot{x}_b^2 + \dot{z}_b^2) + 0.5I_b\dot{\theta}_b^2 \quad (4)$$

$$V_w = 0.5m_w\dot{x}_w^2 + 0.5I_w\dot{\theta}_w^2 \quad (5)$$

$$P_b = m_bg l \cos \theta_b \quad (6)$$

Therefore, the Lagrange function $\mathcal{L}$ is given by

$$\mathcal{L} = V_b + V_w - P_b \quad (7)$$

The dynamics of the WIP model can be represented by

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_b} \right) - \frac{\partial \mathcal{L}}{\partial \theta_b} = -\tau \quad (8)$$

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_w} \right) - \frac{\partial \mathcal{L}}{\partial \theta_w} = \tau \quad (9)$$

By substituting (1)–(3) into (8) and (9), the nonlinear dynamics of the WIP is as follows:

$$M_{22}\ddot{\theta}_b = -M_{12}\ddot{\theta}_w \cos\theta_b + G \sin\theta_b - \tau \quad (10)$$

$$M_{11}\ddot{\theta}_w = M_{12}(\sin\theta_b \dot{\theta}_b^2 - \cos\theta_b \ddot{\theta}_b) + \tau \quad (11)$$

where parameters M_{11} , M_{12} , M_{22} , and G are

$$\begin{cases} M_{11} = (m_b + m_w)r^2 + I_w \\ M_{12} = m_b r l \\ M_{22} = m_b l^2 + I_b \\ G = m_b g l \end{cases} \quad (12)$$

We define the state as $\mathbf{x} = [\theta_b \ \dot{\theta}_b \ \dot{\theta}_w]^\top$. In order to rewrite the dynamics into a discrete-time state space description, we need to perform linearization at the equilibrium point $\theta_b = 0$ by using the Taylor expansion with a time step Δt :

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\tau_k \quad (13)$$

where $\mathbf{A} = \begin{bmatrix} 1 & \Delta t & 0 \\ \frac{M_{11}G\Delta t}{M_{11}M_{22}-M_{12}^2} & 1 & 0 \\ -\frac{M_{12}G\Delta t}{M_{11}M_{22}-M_{12}^2} & 0 & 1 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 0 \\ -\frac{(M_{11}+M_{12})\Delta t}{M_{11}M_{22}-M_{12}^2} \\ \frac{(M_{22}-M_{12})\Delta t}{M_{11}M_{22}-M_{12}^2} \end{bmatrix}$. It should be noted that

A and B must be updated since l and m_b can be changed when the leg length varies or a passenger sits on the wheelchair.

2.2. Leg Kinematics and Statics

Figure 3 shows the leg mechanism of the wheel-legged robot, which is a five-bar linkage close chain actuated by two motors. θ_1 and θ_4 are independently actuated by motors. While closed-form inverse kinematic solutions exist for the five-bar linkage, for our torque control application, we directly employ the Jacobian transpose for force control. Therefore, we derive the forward kinematics and statics to map the end-effector forces to the actuated joint torques.

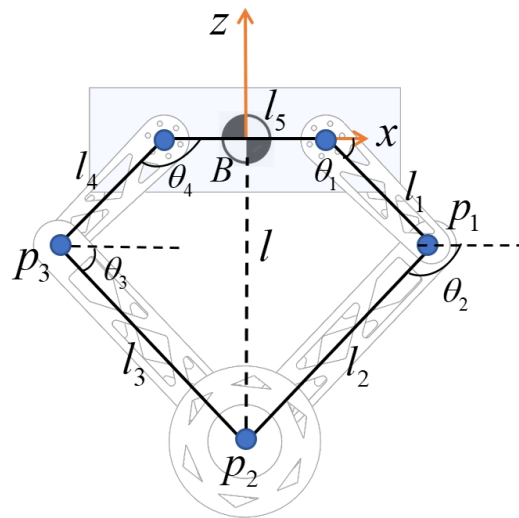


Figure 3. The five-bar linkage model of the robot's leg.

Given θ_1 and θ_4 , the positions of p_1 and p_3 are determined. Then, the position of p_2 in frame $\{B\}$ can be expressed as

$$\begin{cases} x_{p_2} = x_{p_1} + l_2 \cos \theta_2 = x_{p_3} + l_3 \cos \theta_3 \\ z_{p_2} = z_{p_1} + l_2 \sin \theta_2 = z_{p_3} + l_3 \sin \theta_3 \end{cases} \quad (14)$$

where $x_{p_1} = 0.5l_5 + l_1 \cos \theta_1$, $z_{p_1} = l_1 \sin \theta_1$, $x_{p_3} = -0.5l_5 + l_4 \cos \theta_4$, and $z_{p_3} = l_4 \sin \theta_4$. By eliminating θ_3 , we have

$$\theta_2 = 2 \tan^{-1} \frac{B + \sqrt{A^2 + B^2 - C^2}}{A + C} \quad (15)$$

where $A = 2l_2(x_{p_1} - x_{p_3})$, $B = 2l_2(z_{p_1} - z_{p_3})$, and $C = l_2^2 + (x_{p_1} - x_{p_3})^2 + (z_{p_1} - z_{p_3})^2 - l_3^2$.

Substituting (15) into (14) yields the coordinates of p_2 . θ_3 can be computed using the same method.

Taking the derivative with respect to time of (14) and eliminating $\dot{\theta}_3$, we obtain

$$\dot{\theta}_2 = \frac{(\dot{x}_{p_3} - \dot{x}_{p_1}) \cos \theta_3 + (\dot{z}_{p_3} - \dot{z}_{p_1}) \sin \theta_3}{l_2 \sin(\theta_3 - \theta_2)} \quad (16)$$

where $\dot{x}_{p_1} = -l_1 \dot{\theta}_1 \sin \theta_1$, $\dot{z}_{p_1} = l_1 \dot{\theta}_1 \cos \theta_1$, $\dot{x}_{p_3} = -l_4 \dot{\theta}_4 \sin \theta_4$, and $\dot{z}_{p_3} = l_4 \dot{\theta}_4 \cos \theta_4$. By substituting (16) into the derivative of (14), we can get the Jacobian matrix of the five-bar linkage mechanism:

$$\mathbf{J} = \begin{bmatrix} \frac{l_1 \sin(\theta_1 - \theta_2) \sin \theta_3}{\sin(\theta_2 - \theta_3)} & \frac{l_4 \sin(\theta_3 - \theta_4) \sin \theta_2}{\sin(\theta_2 - \theta_3)} \\ -\frac{l_1 \sin(\theta_1 - \theta_2) \cos \theta_3}{\sin(\theta_2 - \theta_3)} & -\frac{l_4 \sin(\theta_3 - \theta_4) \cos \theta_2}{\sin(\theta_2 - \theta_3)} \end{bmatrix} \quad (17)$$

Then, the statics of the five-bar linkage mechanism can be obtained:

$$\boldsymbol{\tau}_{\text{leg}} = \begin{bmatrix} \tau_1 \\ \tau_4 \end{bmatrix} = \mathbf{J}^\top \begin{bmatrix} F_x \\ F_z \end{bmatrix} = \mathbf{J}^\top \mathbf{F} \quad (18)$$

where τ_1 and τ_4 represent the output torques of the two motors that actuate the input links l_1 and l_4 of the five-bar linkage mechanism (see Figure 3). $\mathbf{F}$ is the virtual force vector applied at the foot-end (i.e., the wheel axle point p_2).

3. Control Framework

The control diagram for the robots is shown in Figure 4. The user commands include the steering translational speed v , rotational speed ω , and a height adjustment command which is unique to the wheel-legged robot. The LQR controller maintains balance while tracking the desired translational speed, whereas a PD controller regulates the rotational speed. An additional PD controller manages the base height for the wheel-legged robot. To account for variations in base height and the presence of an unknown passenger, the dynamic models are continuously updated by an algorithm.

3.1. LQR Controller

Since the linearized dynamic model, i.e., (13), has been derived, a LQR control problem can be formulated as follows:

$$\begin{aligned} \min_{\tau_0, \tau_1, \dots, \tau_\infty} \quad & \sum_{k=0}^{\infty} (\mathbf{x}_k^\top \mathbf{Q} \mathbf{x}_k + R \tau_k^2) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{A} \mathbf{x}_k + \mathbf{B} \tau_k \end{aligned} \quad (19)$$

where $\mathbf{Q} \in \mathbb{R}^{3 \times 3}$ represent the weight matrices for the states, and $R \in \mathbb{R}^+$ is a scalar for the control input. The solution of (19) is the well-known optimal feedback control law:

$$\tau_k = -\mathbf{K}(\mathbf{x}_k - \mathbf{x}_d) \quad (20)$$

where $\mathbf{x}_k$ is the current state measured by the state estimator, $\mathbf{x}_d$ is the desired state, and $\mathbf{K}$ is the optimal feedback control gain matrix.

$$\mathbf{K} = \left(R + \mathbf{B}^\top \mathbf{P} \mathbf{B} \right)^{-1} \mathbf{B}^\top \mathbf{P} \mathbf{A} \quad (21)$$

where $\mathbf{P}$ is the unique positive definite solution to the discrete time algebraic Riccati equation:

$$\mathbf{P} = \mathbf{A}^\top \mathbf{P} \mathbf{A} - \mathbf{A}^\top \mathbf{P} \mathbf{B} \left(R + \mathbf{B}^\top \mathbf{P} \mathbf{B} \right)^{-1} \mathbf{B}^\top \mathbf{P} \mathbf{A} + \mathbf{Q} \quad (22)$$

We solve the algebraic Riccati equation by iterating the dynamic Riccati equation of the finite-horizon case until it converges. This approach is termed the online LQR controller.

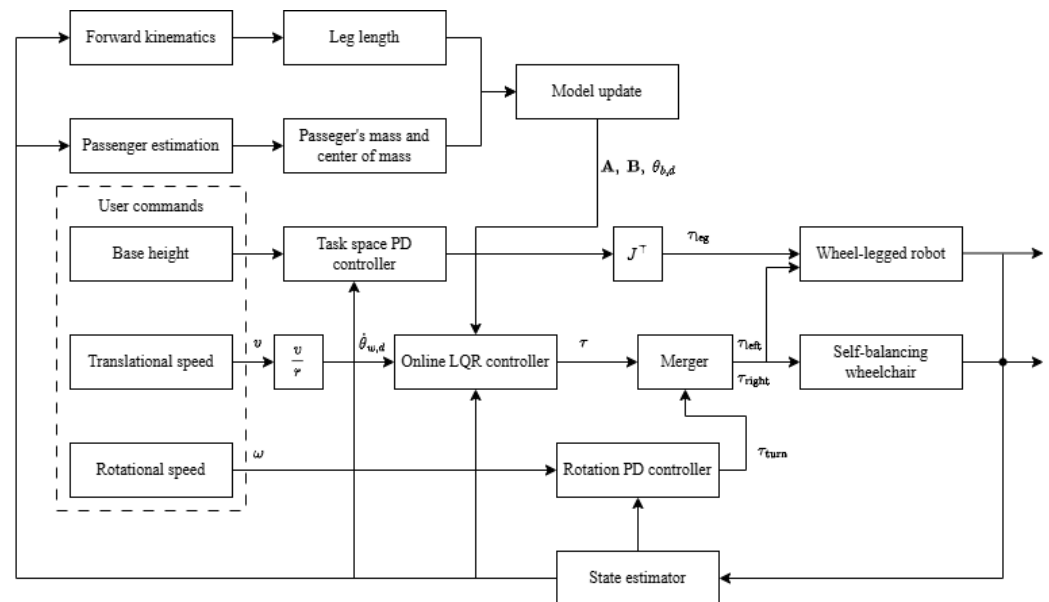


Figure 4. The five-bar linkage model of the robot’s leg. The framework consists of a universal control core and platform-specific control modules. The universal control core consists of an online model-updating LQR controller for balance and translational velocity tracking, while the wheel-legged robot has a task-space PD leg controller for height adjustment.

Note that the desired state $\mathbf{x}_d = [\theta_{b,d} \ \dot{\theta}_{b,d} \ \dot{\theta}_{w,d}]^\top$ typically takes the form $\mathbf{x}_d = [0 \ 0 \ \dot{\theta}_{w,d}]^\top$ to maintain balance. However, the desired inclination angle $\theta_{b,d}$ is no longer zero if there is a horizontal bias in the passenger’s CoM relative to the robot’s CoM. The $\theta_{b,d}$, $\mathbf{A}$, and $\mathbf{B}$ values will be updated in the following subsection.

3.2. Model Update Method

The variation in leg length will cause a change in the CoM for the wheel-legged robot. It is easy to compute the new l and update $\mathbf{A}$ and $\mathbf{B}$ in (13). In contrast, the change in the model caused by the presence of a person in the wheelchair is more complex because the person’s mass and CoM are unknown. More importantly, the new CoM of the system will no longer align with the system’s centerline, as illustrated in Figure 5. The desired inclination angle $\theta_{b,d}$ must be changed since the IMU is fixed on the board. If we do not change the desired inclination angle $\theta_{b,d}$, the system will never settle down.

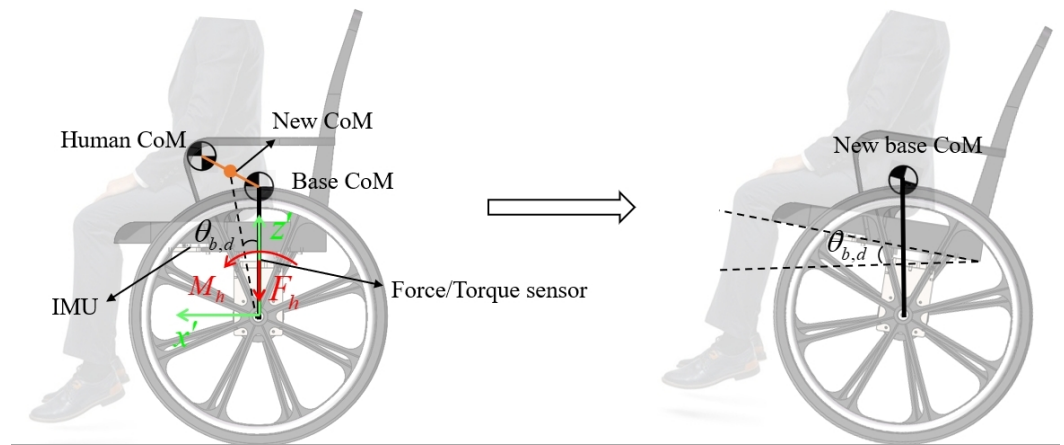


Figure 5. The illustration diagram for the wheelchair model update method.

In our design, a force/torque sensor is used to connect the chair and the base of the wheelchair robot in order to estimate the person's properties. As shown in Figure 5, we can compute the mass and the CoM bias based on the measured force and torques as follows:

$$\begin{cases} m_h = F_h / g \\ x_h = M_h / (m_h g) \end{cases} \quad (23)$$

Assuming that each person's CoM in height is the same as z_h , which is measured by the 3D CAD model, the new CoM can be calculated as follows:

$$\begin{cases} x'_c = \frac{x_h m_h}{m_h + m_b} \\ z'_c = \frac{z_h m_h + z_b m_b}{m_h + m_b} \end{cases} \quad (24)$$

We can update the **A** and **B** values in (13) by substituting m_b with $m_b + m_h$ and l with $l = \sqrt{x_c'^2 + z_c'^2}$, respectively. The desired inclination angle $\theta_{b,d}$ also can be determined by

$$\theta_{b,d} = \tan^{-1} \left(\frac{x'_c}{z'_c} \right) \quad (25)$$

Unlike traditional LQR controllers that use fixed model parameters, the equilibrium adjustment method enables our controller to handle unknown CoM offsets.

3.3. Sensitivity Analysis and Empirical Robustness

To rigorously analyze the stability of the proposed control framework, we examine the closed-loop dynamics under real-time model updates. The core of our control framework relies on the linearized discrete-time state-space model (13) of the self-balancing robot. It is a bounded-input bounded-output (BIBO) system since the system parameters (m_b , m_h , l , CoM) are physically bounded. The following theorem establishes the BIBO stability of the system when the LQR controller is updated online.

The proposed control framework involves a linearized model that is updated online, resulting in a time-varying system. A principled stability analysis for such a system would require a switched-linear or linear time-varying (LTV) framework, which is beyond the scope of this work. Instead, we provide a heuristic sensitivity analysis to offer insight into how parameter variations affect the control system.

Theorem 1 (BIBO Stability Under Model Updates). *Consider the linearized system (13) with the LQR control law (20). If the model parameters (l, m_b) are updated such that*

1. *The CoM deviation $\Delta l = |l_{k+1} - l_k|$ is bounded by $\Delta l_{\max} = \sqrt{\frac{\lambda_{\min}(\mathbf{Q})}{2\|\mathbf{P}\|\|G\|}}$;*
2. *The mass variation $\Delta m_b = |m_{b,k+1} - m_{b,k}|$ satisfies $\Delta m_b < \frac{r^2 I_w}{2l^2(m_w + m_b)}$*

then the closed-loop system is BIBO stable, and the state $\mathbf{x}_k$ converges to a neighborhood of $\mathbf{x}_d$ with ultimate bound:

$$\|\mathbf{x}_k - \mathbf{x}_d\| \leq \frac{\epsilon}{\sigma_{\min}(\mathbf{Q})}, \quad (26)$$

where ϵ accounts for linearization errors and $\sigma_{\min}(\mathbf{Q})$ is the smallest singular value of $\mathbf{Q}$.

Proof. Let $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}_d$ be the tracking error. The Lyapunov candidate $V(\mathbf{e}_k) = \mathbf{e}_k^\top \mathbf{P} \mathbf{e}_k$ satisfies

$$\Delta V = V(\mathbf{e}_{k+1}) - V(\mathbf{e}_k) \leq -\mathbf{e}_k^\top \mathbf{Q} \mathbf{e}_k + 2\|\mathbf{P}\|\|G\|\Delta l\|\mathbf{e}_k\|^2. \quad (27)$$

For $\Delta l < \Delta l_{\max}$, $\Delta V < 0$ ensures convergence and stability. Mass variation bounds follow similarly from (12). $\square$

Lemma 1 (Positive Definiteness of $\mathbf{P}$). *The Riccati solution $\mathbf{P}$ remains positive definite if the length update rate satisfies*

$$\left| \frac{\Delta l}{l} \right| < \frac{1}{2\kappa(\mathbf{A})}, \quad (28)$$

where $\kappa(\mathbf{A})$ is the condition number of $\mathbf{A}$.

3.4. Turning Controller

Since the wheels are controlled by torque commands generated by the LQR controller, rotation speed tracking is directly controlled by a PD controller instead of using differential speed control.

$$\tau_{\text{turn}} = K_p^{\text{turn}}(\omega_d - \omega) + K_d^{\text{turn}}(\dot{\omega}_d - \dot{\omega}) \quad (29)$$

where $\dot{\omega}_d = 0$, ω_d is sent by the users, and ω and $\dot{\omega}$ are measured by the IMU.

The torque commands sent to the left and right wheels can be calculated as follows:

$$\begin{cases} \tau_{\text{left}} = \frac{1}{2}(\tau - \tau_{\text{turn}}) \\ \tau_{\text{right}} = \frac{1}{2}(\tau + \tau_{\text{turn}}) \end{cases} \quad (30)$$

where τ is the result generated by the LQR controller, i.e., (20).

The torque split defined in Equation (31) introduces a potential coupling where the turning torque τ_{turn} could disturb the sagittal-plane balance regulated by the LQR-derived torque τ . To justify the robustness of this decoupled design, we consider the timescale separation between the two control objectives.

The sagittal-plane dynamics of a self-balancing robot are inherently unstable, with a characteristic time constant on the order of a few hundred milliseconds. The LQR controller is therefore designed to be a high-bandwidth controller to stabilize this fast process. In contrast, the rotational dynamics of the robot are stable (behaving similarly to a damped rotational system), and the turning PD controller is tasked with tracking a user's rotational velocity command, which is typically a low-frequency signal.

Due to this separation of timescales, the high-gain LQR controller can rapidly reject the low-frequency disturbances introduced into the sagittal plane by τ_{turn} . The turning controller's output acts as a slow-varying disturbance that the balance loop effectively treats as a quasi-static load change, which it is designed to compensate for through its

integral action (inherent in the state-feedback structure around a non-zero equilibrium). This rationale explains the empirical robustness observed in our experiments.

3.5. Leg Controller

Similarly to the turning controller, the leg positions relative to the robot's base for the wheel-legged robot are also controlled using a PD controller.

$$\mathbf{F} = \mathbf{K}_p^{\text{leg}}(\mathbf{P}_{3,d} - \mathbf{P}_3) + \mathbf{K}_d^{\text{leg}}(\dot{\mathbf{P}}_{3,d} - \dot{\mathbf{P}}_3) \quad (31)$$

where $\mathbf{P}_{3,d} = [p_{3,d}^x \ p_{3,d}^z]^\top$ is the desired position of the wheel axle relative to the base, as shown in Figure 3, $p_{3,d}^x = 0$, and $p_{3,d}^z$ is the given user command, which is important for jumping motion. A jumping trajectory generator is used to generate $p_{3,d}^z$. Then, the torque commands for the leg control motors will be computed by the statics (18). The purpose of the leg is to serve as an active suspension and maneuver generator, while the stability under the resulting dynamic loads is guaranteed by the unified LQR core.

4. Results

In this section, we evaluate the proposed control framework using velocity tracking, traversing steps, and human carrying experiments. The wheel-legged robot also shows its jump ability.

4.1. Experiments of the Wheel-Legged Robot

The wheel-legged robot is controlled by an onboard STM32H723 MCU (ST company, Geneva, Switzerland), which communicates with the six motors via a CAN bus. The six motors have the same peak torque limit, which is 7 Nm. A BMI088 IMU (Bosch company, Gerlingen, Germany) is connected to the MCU. The nominal length l is 0.2 m. The base mass is 2.4 kg, while the wheel mass is 0.7 kg. The radius of the wheel is 0.06 m. A Kalman filter is employed for optimal state estimation by fusing data from the IMU and wheel odometer.

For the wheel-legged robot, the high-level control loop runs on the STM32H723 MCU at 500 Hz, with a worst-case execution time of 1.5 ms, demonstrating the efficiency of the proposed algorithms even on an embedded platform.

4.1.1. Velocity Tracking

In this experiment, the wheel-legged robot successfully tracked the desired translational and rotational velocities. Figure 6 presents snapshots of the robot during translational velocity tracking. Figure 7 illustrates the desired translational velocity along the x -axis and the corresponding desired position, which was derived by integrating the desired translational velocity. It is important to note that the proposed LQR controller was not designed to track the desired position. The comparison between the desired and measured positions is provided to offer readers an intuitive understanding of the reciprocating motion observed in this experiment. Figure 8 demonstrates the rotation control performance, where the desired rotational angle ϕ_d was generated by integrating the desired rotational velocity ω . The results indicate that rotational tracking achieved minimal errors, while the base position remained stable. The near-zero x -displacement in Figure 8 confirms excellent decoupling between rotational and translational motions. This experiment highlights the effective coordination between the LQR and the rotational PD controllers, where $K_p = 30$ and $K_d = 1$.

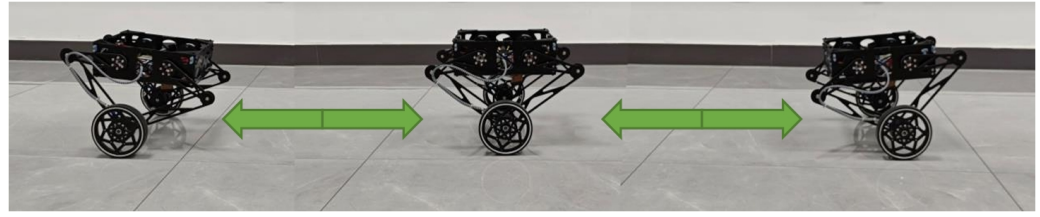


Figure 6. Snapshots of when the wheel-legged robot tracks a reciprocating motion.

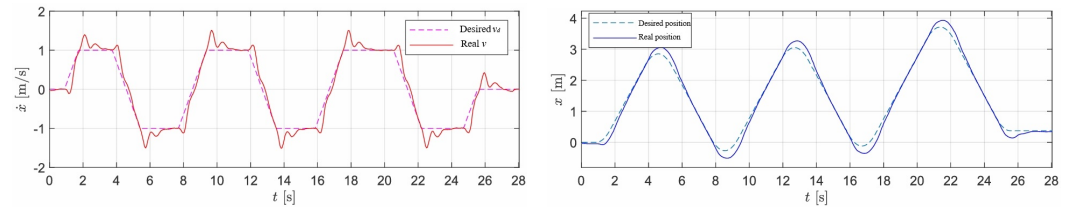


Figure 7. Velocity and position control performance along x axis.

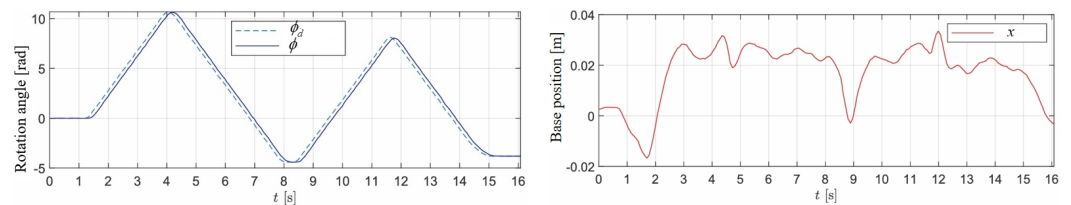


Figure 8. Rotation control performance.

4.1.2. Jumping

In this experiment, the task space PD controller was evaluated through a jumping motion, as depicted in Figure 9. The desired end-effector trajectories were determined through iterative trials and then transformed into the base frame, with real-time leg length variations and ground contact forces recorded in Figure 10. The jumping sequence exhibited three distinct phases:

1. Preparation phase (0.3–0.9 s): The robot lowered its center of mass by retracting the legs, storing elastic energy in the system.
2. Takeoff phase (0.9–1.1 s): Rapid leg extension generated peak ground reaction forces of 38 N, propelling the robot vertically.
3. Flight and landing phases (1.1–1.6 s): The legs fully extended during flight (contact force ≈ 0 N), followed by controlled compression during landing to dissipate impact energy.

The PD controller effectively tracked the desired positions of the wheel axle within the base frame, with stiffness and damping coefficients set to $K_p = 20$ and $K_d = 1$, respectively. The virtual forces generated by the controller were mapped to the active leg motor torques using the Jacobian matrix ((18)), enabling dynamic leg actuation. The robot achieved a jumping height of approximately 0.1 m, as verified through manual measurement (using a tape measure) when state estimation became unreliable during the flight phase. It is worth noting that the PD gains were intentionally kept moderate, as the primary function of the leg in this context is to act as an active suspension system for navigating uneven terrain.

The jumping maneuver explicitly drives the system far from its linearized equilibrium. To quantify this, Table 1 reports the extreme values of the key states recorded during the jump depicted in Figures 9 and 10.

It is critical to note that the LQR controller used was the same single-gain controller designed for the nominal upright position; no gain scheduling or online relinearization was activated during the jump. To assess whether the controller remained effective despite the

large pitch angle, we monitored the quadratic cost term $\mathbf{x}_k^\top \mathbf{Q} \mathbf{x}_k$ relative to its value during steady-state balancing. This cost increased by a factor of 4.5 during landing but rapidly decayed. This indicates that the state entered a region where the linear model was not accurate, but the control law, derived from that model, still provided a stabilizing action that successfully recovered the robot.



Figure 9. Snapshots when the wheel-legged robot is jumping.

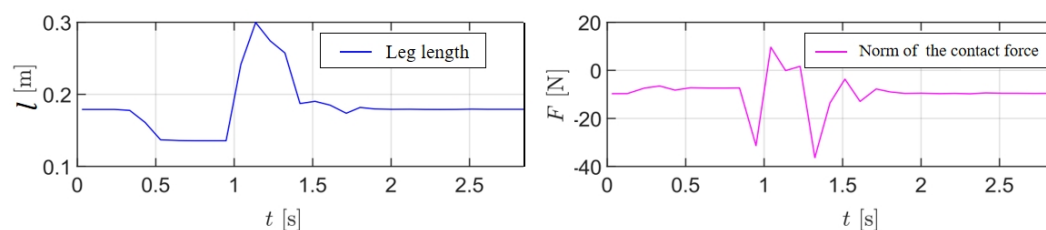


Figure 10. The leg length and contact force during the jumping motion.

Table 1. Maximum state deviations during the jumping experiment.

State Variable	Maximum Value	Phase of Occurrence
Body Pitch Angle θ_b	16.8°	Landing Impact (at 1.4 s)
Body Pitch Rate $\dot{\theta}_b$	142.5°/s	Landing Impact (at 1.4 s)
Wheel Contact Force	38 N	Take-off (at 1.0 s)

4.2. Experiments of the Self-Balancing Wheelchair

The self-balancing wheelchair is driven by two DC servo motors, which are controlled by an Intel NUC computer through an EtherCAT bus. Each motor can generate a maximum torque of 70 Nm. An Xsens MTi 630 IMU sensor (Movella company, El Segundo, CA, USA) is mounted on the chair's base, while a Sunrise M3715B 6-axis Force/Torque sensor (Sunrise Instruments company, Canton, MI, USA) is positioned between the chair and the axle for precise measurement. The base mass m_b is 15.5 kg, while the mass of the wheel is 1.9 kg. The CoM height l is 0.128 m. The radius r of the wheel is 0.3 m. A Kalman filter is also employed to make optimal state estimation by merging the IMU and the wheel odometer.

The high-level control loop, encompassing state estimation, model updating, and LQR control law computation, runs at a deterministic 200 Hz. The data confirms that the worst-case execution time is well within the 5 ms available per cycle, leaving a sufficient computational margin.

4.2.1. Wheelchair Experiments

The velocity tracking experiment was the first conducted to validate the performance of the proposed controller. Figure 11 illustrates the results obtained when the robot, carrying a person weighing 55.1 kg, was tested. The dynamic model was updated by the proposed model estimation algorithm using the force/torque sensor when the person

was sitting down. The estimated mass is 56.12 kg, which is within a 2% relative error of the actual mass (55.1 kg). We deem this error margin reasonable given the potential for small shifts in the passenger's posture and the inherent noise in the force/torque sensor readings during the brief static measurement period. The matrix $\mathbf{Q}$ in the LQR was kept at (20,000, 30, 500), while the matrix $\mathbf{R}$ in the LQR was kept at 1. The controller demonstrates strong performance in tracking step commands, while the inclination angle is rapidly regulated back to the equilibrium point, as shown in Figure 11. Figure 12 depicts real robot experiments where the user sent commands via a joystick. The LQR controller and the PD turning controller ($K_p = 45$ and $K_d = 1$) coordinated effectively while carrying the person. Figure 13 demonstrates the recorded velocity and inclination angle as the wheelchair overcame a 5 cm height step. The wheels made contact with the step at around the 5-s mark, after which the robot successfully surmounted the step and stabilized within 3 s. This experiment highlights the potential application of this human-carrying wheelchair as it demonstrates superior obstacle-crossing capabilities compared to traditional four-wheeled wheelchairs.

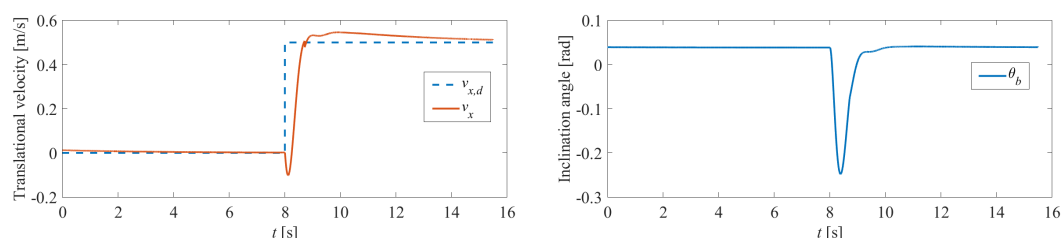


Figure 11. Velocity tracking experiment of the self-balancing wheelchair.



Figure 12. Snapshots of the self-balancing wheelchair carrying a person.

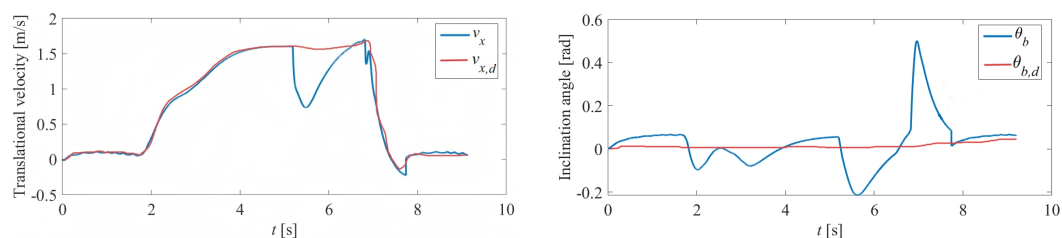


Figure 13. The recorded states as the wheelchair overcame a 5 cm height step. The robot came into contact with the step at 5 s.

4.2.2. Safety System and Experimental Protocol

The operation of a self-balancing wheelchair carrying a human passenger necessitates a rigorous safety-first approach. This subsection documents the safety interlocking, comfort metrics, and experimental criteria employed.

A defense-in-depth strategy was implemented to ensure passenger safety:

1. **Software Safety Limits:** The control software incorporated hard limits on critical states. If the body pitch angle $|\theta_b|$ exceeded 15° or the translational velocity exceeded 1.5 m/s, the controller would cut power to the motors and initiate a controlled braking procedure.
2. **Watchdog Timer:** A dedicated hardware watchdog timer was used to monitor the high-level control loop. If a computation hang or overrun prevented the loop from being complete within its 5 ms deadline, the watchdog would trigger a system halt.
3. **Emergency Stop (E-Stop):** A latching, physical E-stop button was installed and made readily accessible to both the passenger and an external operator. Engaging the E-stop cut all motor power independently of the main computer.
4. **Safety Assistant:** During all experiments, the wheelchair operated within a controlled laboratory space. The passenger was protected by other people in case of tip-over.

4.3. Ablation Studies and Quantitative Analysis

To rigorously evaluate the contribution of each component in the proposed control framework, a series of ablation studies and quantitative tests were conducted on the self-balancing wheelchair platform. All quantitative results are reported as the mean $\pm$ standard deviation across $N = 5$ trials for each condition.

4.3.1. Benefits of Online Model Update

We first isolated the benefits of the online model-updating algorithm. The controller was tested under three passenger conditions: no passenger, a 55 kg passenger, and an 85 kg passenger. We compared the proposed online LQR (model parameters updated via F/T sensor) against a fixed LQR (parameters fixed for the no-passenger condition).

As shown in Table 2, the performance of the fixed LQR degraded severely with increasing passenger mass and became unstable with the 85 kg passenger. In contrast, the online LQR maintained consistently low tracking error and stability across all conditions, demonstrating its critical role in handling large model variations.

Table 2. Ablation study: online vs. fixed LQR.

Passenger Mass	Controller	RMSE of v_x (m/s)	Success Rate
0 kg	Fixed LQR	0.032 ± 0.005	5/5
0 kg	Online LQR	0.029 ± 0.004	5/5
55 kg	Fixed LQR	0.158 ± 0.021	5/5
55 kg	Online LQR	0.035 ± 0.006	5/5
85 kg	Fixed LQR	Unstable (Fell)	0/5
85 kg	Online LQR	0.041 ± 0.007	5/5

4.3.2. Benefits of Pitch Bias Compensation

We further isolated the effect of the equilibrium pitch bias compensation $\theta_{b,d}$. Using the online LQR, we tested with a 55 kg passenger with a known CoM horizontal offset of 8 cm. We compared performance with and without the force/torque-sensor-based $\theta_{b,d}$ to update Equation (25).

The results in Table 3 show that without pitch bias compensation, the system settles at an incorrect equilibrium with a large steady-state pitch error, requiring constant control

effort to maintain this offset posture. Our compensation method successfully corrects the equilibrium, bringing the steady-state pitch to zero and significantly reducing the control effort.

Table 3. Ablation study: with vs. without pitch bias compensation.

Condition	Steady-State Pitch Error (°)	RMS of τ (Nm)
Without $\theta_{b,d}$ compensation	-4.8 ± 0.3	5.2 ± 0.6
With $\theta_{b,d}$ compensation	-0.2 ± 0.1	1.8 ± 0.3

5. Conclusions

This paper presented a unified control framework to address the fundamental challenge of model variations in two distinct self-balancing robot platforms: a wheel-legged robot and a human-carrying wheelchair. The core of our approach lies in the choice of an online model-updating LQR controller as the universal balancing core. This design was strategically selected because the LQR provides a principled and computationally tractable optimal control solution, while the real-time model update mechanism directly compensates for the primary source of uncertainty—changes in mass and center of mass. This contrasts methods that treat model variations as unstructured disturbances, enabling our controller to proactively reconfigure for new operating conditions. The wheel-legged robot achieved a 0.1 m jump height and maintained reasonable velocity error, while the wheelchair overcame a 5 cm step with a 3 s settling time while carrying a 55 kg passenger. The model updating algorithm estimated passenger mass and CoM with high accuracy, enabling dynamic adjustments. These results validate the framework’s robustness and potential for applications in personal mobility and assistive robotics, with future work focusing on optimization for complex environments and enhanced sensor integration.

Future work will focus on several key areas. First, we plan to conduct comprehensive testing in more complex environments. Furthermore, a systematic quantitative comparison of the proposed unified framework against state-of-the-art alternative strategies, such as nonlinear MPC and gain-scheduled LQR, will be undertaken to rigorously benchmark its performance, advantages, and limitations. Also, reinforcement learning control methods, particularly the Actor–Critic reinforcement learning method, may be promising alternative algorithms for the model variation problem.

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

Funding: This work was supported by the State Key Laboratory of Robotics and Systems (HIT) (grant number [SKLRS-2023-KF-16]); the Guangdong Basic and Applied Basic Research Foundation (grant number [2022A1515140156]); the Dalian Science and Technology Innovation Funding (2023JJ12GX017); the Fundamental Research Funds for the Central Universities (DUT25YG257); and the Liaoning Province Basic and Applied Basic Research Foundation (grant number [2023JH2/101600033]).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is available if required by email.

Conflicts of Interest: The authors declare no conflicts 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. Rajendran, V.; Lin, J.F.S.; Mombaur, K. Towards Humanoids Using Personal Transporters: Learning to Ride a Segway from Humans. In Proceedings of the 2022 9th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob), Seoul, Republic of Korea, 21–24 August 2022; pp. 1–8. [\[CrossRef\]](#)
2. Grasser, F.; D’Arrigo, A.; Colombi, S.; Rufer, A. JOE: A mobile, inverted pendulum. *IEEE Trans. Ind. Electron.* **2002**, *49*, 107–114. [\[CrossRef\]](#)
3. Klemm, V.; Morra, A.; Salzmann, C.; Tschopp, F.; Bodie, K.; Gulich, L.; Küng, N.; Mannhart, D.; Pfister, C.; Vierneisel, M.; et al. Ascento: A Two-Wheeled Jumping Robot. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 7515–7521. [\[CrossRef\]](#)
4. Dynamics, B. Introducing Handle. 2017. Available online: <https://www.youtube.com/watch?v=-7xvqQeoA8c> (accessed on 28 February 2017).
5. Spong, M.W. Underactuated Mechanical Systems. In *Control Problems in Robotics and Automation*; Springer: Berlin/Heidelberg, Germany, 1995; pp. 135–150.
6. Westervelt, E.; Grizzle, J.; Koditschek, D. Hybrid zero dynamics of planar biped walkers. *IEEE Trans. Autom. Control* **2003**, *48*, 42–56. [\[CrossRef\]](#)
7. Khalil, H.K. *Nonlinear Systems*; Prentice Hall: Upper Saddle River, NJ, USA, 2002.
8. Zhang, H.; Mohamad Nor, N. Control Strategies for Two-Wheeled Self-Balancing Robotic Systems: A Comprehensive Review. *Robotics* **2025**, *14*, 101. [\[CrossRef\]](#)
9. Thao, N.G.M.; Nghia, D.H.; Phuc, N.H. A PID backstepping controller for two-wheeled self-balancing robot. *Int. Forum Strateg. Technol.* **2010**, *2010*, 76–81. [\[CrossRef\]](#)
10. Erkol, H.O. Optimal $PI^{\lambda} D^{\mu}$ Controller Design for Two Wheeled Inverted Pendulum. *IEEE Access* **2018**, *6*, 75709–75717. [\[CrossRef\]](#)
11. Wang, J. Path Tracking of a Two-Wheeled Self-Balancing Robot Based on Multi-objective Optimization with Artificial Immune Algorithm. In Proceedings of the 2023 7th International Conference on Robotics and Automation Sciences (ICRAS), Wuhan, China, 11–13 June 2023; pp. 12–17. [\[CrossRef\]](#)
12. Wardoyo, A.S.; Hendi, S.; Sebayang, D.; Hidayat, I.; Adriansyah, A. An investigation on the application of fuzzy and PID algorithm in the two wheeled robot with self balancing system using microcontroller. In Proceedings of the 2015 International Conference on Control, Automation and Robotics, Singapore, 20–22 May 2015; pp. 64–68. [\[CrossRef\]](#)
13. Simon, J. Fuzzy Control of Self-Balancing, Two-Wheel-Driven, SLAM-Based, Unmanned System for Agriculture 4.0 Applications. *Machines* **2023**, *11*, 467. [\[CrossRef\]](#)
14. Ha, M.S.; Jung, S. Neural network control for balancing performance of a single-wheel transportation vehicle. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–5. [\[CrossRef\]](#)
15. Kim, S.; Kwon, S. Nonlinear Optimal Control Design for Underactuated Two-Wheeled Inverted Pendulum Mobile Platform. *IEEE/ASME Trans. Mechatron.* **2017**, *22*, 2803–2808. [\[CrossRef\]](#)
16. Fang, J. The LQR Controller Design of Two-Wheeled Self-Balancing Robot Based on the Particle Swarm Optimization Algorithm. *Math. Probl. Eng.* **2014**, *2014*, 729095. [\[CrossRef\]](#)
17. Huang, J.; Guan, Z.H.; Matsuno, T.; Fukuda, T.; Sekiyama, K. Sliding-Mode Velocity Control of Mobile-Wheeled Inverted-Pendulum Systems. *IEEE Trans. Robot.* **2010**, *26*, 750–758. [\[CrossRef\]](#)
18. Yang, Z.; Bian, Z.; Zhang, W. Design and Control of Multi-mode Wheeled-Bipedal Robot with Parallel Mechanism. In Proceedings of the 2023 International Conference on Communications, Computing and Artificial Intelligence (CCCAI), Shanghai, China, 23–25 June 2023; pp. 69–74. [\[CrossRef\]](#)
19. Karthika, B.; Jisha, V. Nonlinear Optimal Control of a Two Wheeled Self Balancing Robot. In Proceedings of the 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, India, 1–3 December 2020; pp. 1–6. [\[CrossRef\]](#)
20. Liu, C.; Qin, K.; Xin, G.; Li, C.; Wang, S. Nonlinear Model Predictive Control for a Self-Balancing Wheelchair. *IEEE Access* **2024**, *12*, 28938–28949. [\[CrossRef\]](#)
21. Rawlings, J.B.; Mayne, D.Q.; Diehl, M. *Model Predictive Control: Theory, Computation, and Design*; Nob Hill Publishing Madison: Madison, WI, USA, 2017; Volume 2.
22. Grne, L.; Pannek, J. *Nonlinear Model Predictive Control: Theory and Algorithms*; Springer Publishing Company, Incorporated: New York, NY, USA, 2013.
23. Su, Y.; Wang, T.; Zhang, K.; Yao, C.; Wang, Z. Adaptive nonlinear control algorithm for a self-balancing robot. *IEEE Access* **2019**, *8*, 3751–3760. [\[CrossRef\]](#)

24. Cui, L.; Wang, S.; Zhang, J.; Zhang, D.; Lai, J.; Zheng, Y.; Zhang, Z.; Jiang, Z.P. Learning-based balance control of wheel-legged robots. *IEEE Robot. Autom. Lett.* **2021**, *6*, 7667–7674. [[CrossRef](#)]
25. Huang, J.; Zhang, M.; Ri, S.; Xiong, C.; Li, Z.; Kang, Y. High-Order Disturbance-Observer-Based Sliding Mode Control for Mobile Wheeled Inverted Pendulum Systems. *IEEE Trans. Ind. Electron.* **2020**, *67*, 2030–2041. [[CrossRef](#)]
26. Nguyen, K.D.; Dankowicz, H. Adaptive control of underactuated robots with unmodeled dynamics. *Robot. Auton. Syst.* **2015**, *64*, 84–99. [[CrossRef](#)]
27. Tang, C.; Abbatematteo, B.; Hu, J.; Chandra, R.; Martín-Martín, R.; Stone, P. Deep reinforcement learning for robotics: A survey of real-world successes. *Annu. Rev. Control Robot. Auton. Syst.* **2025**, *8*, 153–188. [[CrossRef](#)]
28. Yan, C.; Li, X. Research on Stability Control System of Two-Wheel Heavy-Load Self-Balancing Vehicles in Complex Terrain. *Appl. Sci.* **2024**, *14*, 7682. [[CrossRef](#)]
29. Xin, G. Experimental Validation of Unified Control Framework for Self-Balancing Robots. Available online: https://www.youtube.com/watch?v=SM_TTUR_zOo (accessed on 8 March 2025).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.