

Balancing Control and Pose Optimization for Wheel-legged Robots Navigating High Obstacles

Junheng Li, Junchao Ma, and Quan Nguyen

Abstract—In this paper, we propose a novel approach on controlling wheel-legged quadrupedal robots using pose optimization and force control via quadratic programming (QP). Our method allows the robot to leverage the whole-body motion and the wheel actuation to roll over high obstacles while keeping the wheel torques to navigate the terrain while keeping the wheel traction and balancing the robot body. In detail, we first present a linear rigid body dynamics with wheels that can be used for real-time balancing control of wheel-legged robots. We then introduce an effective pose optimization method for wheel-legged robot’s locomotion over steep ramp and stair terrains. The pose optimization solves for optimal poses to enhance stability and enforce collision-free constraints for the rolling motion over stair terrain. Experimental validation on the real robot demonstrated the capability of rolling up on a 0.36 m obstacle. The robot can also successfully roll up and down multiple stairs without lifting its legs or having collision with the terrain.

I. INTRODUCTION

The recent technological and theoretical developments in both robot design and controls have allowed the world to witness many successful and highly-autonomous legged robots. With such hardware and software advancements, researchers in the robotics field are now facing a challenge to develop mobile legged robots that can conduct given tasks fully autonomously and that the control framework can perform robustly in terrains with uneven surfaces with obstacles [1]. Glancing over the development of legged robots in the last decade, many bipedal and quadruped robots have demonstrated outstanding maneuverability and dynamic locomotion in unknown terrain and have proven to have great potential to be controlled autonomously (e.g. ATRIAS [2], Cassie [3], MIT Cheetah 3 [4], ANYmal quadruped [5], and Boston Dynamics Spot autonomous exploration mission [6]). However, energy efficiency in the robot hardware remains one of the most important condition that determines whether a mobile robot can perform real-life task that require extended period of time while maintaining highly dynamical locomotion, such as rescue and disaster-response missions [7]. Legged robots rely on gait sequence and proper foot placement to overcome obstacles and uneven surfaces, which is a more effective method in rough terrain locomotion compared to only wheeled system [8], while wheeled systems generally have far less energy consumption

This work is supported by USC Viterbi School of Engineering startup funds.

The authors are with the Department of Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, CA 90089. email: junhengl@usc.edu, junchaom@usc.edu, quann@usc.edu



Fig. 1: Snapshot of our wheel-legged robot rolling up a 0.30 m obstacle. Experiment and simulation video: <https://youtu.be/s68-tvb1tb4>.

and faster speed to maneuver on an even surface [9]. Hence, the hybrid system of both wheels and legs could leverage the advantages from both worlds, enabling maneuverability in rough terrain, energy efficiency, and speed (e.g. [10] and [11]).

A. Related Work

Recent development in the control of wheel-legged robots shows promising results. The wheel-legged quadruped ANYmal [12] utilizes a kinodynamic model in whole-body model-predictive control for robust locomotion control. The wheel-legged bipedal robot Ascento [13] adopts the whole-body dynamics by using linear-quadratic regulator (LQR) and Zero-moment Point (ZMP) in balance control. Our approach in controlling wheel-legged robots leverages the wheel traction to traverse challenging terrains (i.e., instead of stepping over a high obstacle, we take advantage of the whole body motion and wheel actuation to roll over the obstacle). Note that the obstacle height a legged robot can step over is limited, our approach could allow the robot to roll over an obstacle that is higher than the robot’s nominal standing height. This work also includes pose optimization to solve for optimal and collision-free poses for the robot rolling up high stairs or obstacles.

Force-based QP balancing control has successful implementations on quadruped robot MIT Cheetah 3 [4] and allows the robot to balance even after performing very dynamical and aerial tasks such as jumping [14]. The idea of modifying simplified dynamics in QP balance controller has also gained success in controlling mobile legged robots in the past (e.g. [15]). In our work, we develop this control paradigm for wheel-legged robots by considering the wheel

dynamics and terrain slope in our model. We also adopt this for the rolling task instead of just standing balance.

Trajectory optimization frameworks in motion planning have allowed legged robots to achieve highly-dynamic locomotion. For example, [16], [14], [17], and [18] all utilize NLP solver to find optimal trajectories for a certain task. One important constraint in such optimization frameworks is the utilization of robot dynamics, either full body dynamics or simplified centroidal dynamics. In [19], a NLP parameter optimization is used to find the optimal driving force in achieving highly dynamic locomotion. In wheel-legged robot's motion planning, our approach with pose optimization uses kinematic constraints instead to guarantee collision-free with the terrain and solve for favorable configurations to maintain balance. The advantage of this framework is that the robot can leverage and adapt to the obstacle's shape and height to overcome it rather than simply stepping or rolling over it. Thus, it allows the robot to overcome an obstacle with a height that exceeds its nominal standing height. In our pose optimization framework, we choose not to use dynamics or Inverse Kinematics (IK) to solve for joint angles by relative foot position (i.e. kinodynamics models employed in [12], [20], and [21]), instead, we directly use joint angles, body center of mass (CoM) location in 2D, and body pitch angle as the only optimization variables and use Forward Kinematics (FK) to constrain the relative foot position and collision-avoidance in a favorable pose, which allows us to have a much faster solving time. We consider the dynamics of the robot in our force-based feedback controller for real-time motion planning and control. The optimal pose will then be used in combination with a balance controller using QP-based force control to maintain balance and desired pitch angle. Pose optimization also happens to resonate with the crawling mode introduced in [22] and [23], in which the crawling mode in rolling is proven to be a superior option in slope-climbing. Thanks to only few critical poses being needed, the computation intensity is dramatically scaled-down compared to full trajectory optimizations. Unlike the approach in [11] by adapting wheel-legged robot's posture in rough terrain with feedback control, or the approach in [24] by adding passive suspension for pose adapting, our approach of finding the optimal poses or robot configurations based on the terrain map.

B. Contributions

The main contributions of the paper are as follows:

- We introduce a new rigid body dynamics with wheels dynamics that can be effectively used for force-based balancing control of wheel-legged robots.
- Our proposed pose optimization method with kinematic and collision-free constraints only requires solving very few critical poses in a task that consists of high obstacles while maintaining wheel traction with the terrain. (only 2 poses are needed to solve in a single-stair task)
- The pose optimization is very efficient due to its small problem size. The solved optimal poses at a certain

location can be linearly interpolated to obtain the joint trajectory at any given time during the task.

- We use a hybrid control framework that includes force-based QP and joint PD control to track optimal poses in order to achieve stable locomotion of wheel-legged robots navigating terrain with high obstacles.
- Experimental validation on the real robot demonstrated the capability of rolling up on a 0.36 m obstacle. The robot can also successfully roll up and down multiple stairs without lifting its legs or having collision with the terrain.

The rest of the paper is organized as follows. Section II introduces the wheel-legged robot model, its physical parameters. The overview of our control architecture is highlighted in Section III. Section IV presents the rigid body dynamics with wheels used in balancing control for wheel-legged robots, force-based QP control formulation, and rolling control. Section V introduces the pose optimization and its problem formulations, constraints, and real-time pose planning. Section VI highlights some experimental results with the proposed approach.

II. ROBOT MODEL AND HARDWARE

In this section, we introduce the model of our wheel-legged robot and its physical parameters. Fig. 1 presents the hardware assembly of the wheel-legged robot in motion. Our wheel-legged robot consists of a robot trunk, four sets of 3 degrees of freedom(DoF) legs. Each leg consists of the thigh, calf, and wheel. In total, the wheel-legged robot has 12 actuators. Fig. 2 shows the leg assembly and joint definitions. Table I includes all physical parameters of the robot. Our robot is built from the Unitree A1 quadruped robot with Unitree A1 actuators. The A1 actuator is a torque-controller motor that can provide 21.0 rad/s maximum angular speed and 33.5 Nm maximum torque output.

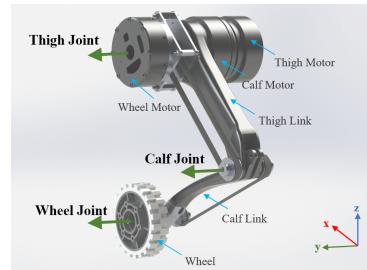


Fig. 2: **Leg Configuration.** The link and joint configuration of wheel-legged robot leg, rendered in SolidWorks.

III. OVERVIEW

Having presented the robot model and hardware, in this Section, we highlight the overview of our control architecture and critical parts of our proposed framework. The control architecture and block diagram are present in Fig. 3. We use QP force-based balance controller to maintain balance in various tasks. This balancing control only commands the thigh and calf joint torques. For controlling the wheels, the rolling control enables the robot to maneuver with

TABLE I: Robot Physical Parameters

Parameter	Symbol	Value	Units
Mass	m	11.84	kg
Body Inertia	I_{xx}	0.0214	$\text{kg} \cdot \text{m}^2$
	I_{yy}	0.0535	$\text{kg} \cdot \text{m}^2$
	I_{zz}	0.0443	$\text{kg} \cdot \text{m}^2$
Body Length	l_b	0.247	m
Body Width	w_b	0.194	m
Body Height	h_b	0.114	m
Thigh and Calf Lengths	l_1, l_2	0.2	m
Wheel Radius	R_{wheel}	0.05	m

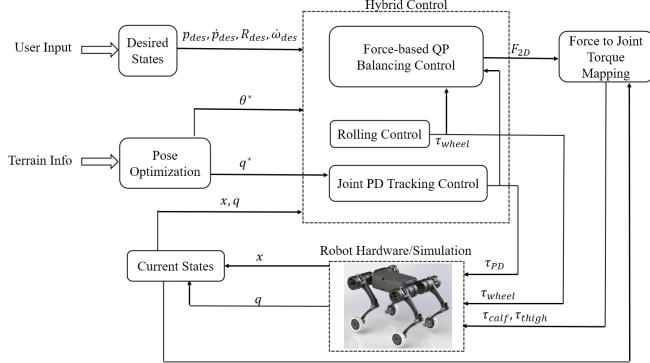


Fig. 3: **Control Architecture.** The block diagram for our proposed approach.

wheel traction and yaw on command. The details of the balancing control will be explained in Section IV. In order to control our robot to roll over challenging terrains (e.g., stairs, high obstacles, or steep ramps), we also propose a pose optimization framework to solve for optimal configuration of the robot that is collision-free with the terrain while maintaining a good support region for the robot to keep the body balanced. Section V will explain in more detail this method. The desired pitch angle is fed into the balance controller to maintain balance during motion. And the joint angles are tracked by joint PD control to manipulate the robot pose.

IV. FORCE-BASED BALANCE CONTROL

A. Rigid-body Dynamics with Wheels

In this section, we introduce a simplified dynamics model for wheel-legged robots that can be used effectively in real-time feedback control for balancing the robot body. Single rigid-body dynamics are commonly used for controlling quadruped robots [25], [14]. However, when the robot highly leverages the wheels in traversing uneven terrain, it is critical to take into account the impact of wheel traction forces on the robot dynamics. In our work, the simplified dynamics model takes wheel dynamics into consideration. Our control framework has been divided into balancing control and rolling control. The balancing controller output is mapped to only the thigh and calf torques, and the wheel torque is controlled by the rolling controller. The two decoupled control methods are connected by combining the wheel dynamics (Fig. 4a)

with the simplified rigid body dynamics of the quadruped robot to become a hybrid linear dynamics model (Fig. 4b), which is to be used in QP balancing controller. Considering the wheel dynamics in angular motion while on the ground, shown in Fig. 4a:

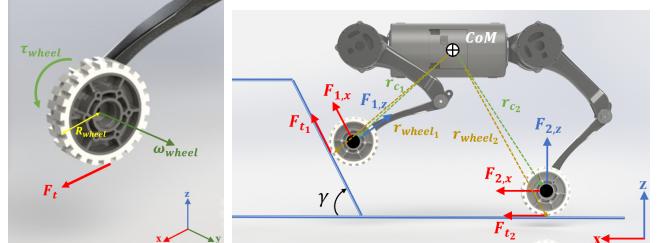


Fig. 4: **Robot Dynamics** Wheel-legged Robot Dynamics

$$\mathbf{I}_{wheel} \cdot \dot{\omega}_{wheel} = \tau_{wheel_i} - \mathbf{F}_{t_i} \cdot \mathbf{R}_{wheel}, \quad (1)$$

where \mathbf{I}_{wheel} is the rotation inertia of the wheel, $\dot{\omega}_{wheel}$ is the angular acceleration of the wheel, τ_{wheel_i} is the wheel torque of the i^{th} leg from control input, and \mathbf{F}_{t_i} is the wheel traction force of the i^{th} leg at ground contact point on the rim of the wheel. We choose to neglect the very small rotation inertia value of the wheel dynamics to obtain a linear relation of the wheel torque and traction force, equation (1) is then reduced to:

$$\mathbf{F}_{t_i} = \frac{\tau_{wheel_i}}{R_{wheel}}. \quad (2)$$

When the robot's wheel contact point is on an aggressive slope, it is important to take into consideration of the change in coordinate frames of ground reaction forces as well as friction constraints. Fig. 4b illustrate one scenario when the front legs of the robot are on the slope with a positive angle γ , notice that the ground reaction force in z-direction has to be normal to the terrain while the ground reaction force in the x-direction has to be parallel to the terrain. The simplified dynamics of the robot can be written as:

$$\mathbf{A}_c \cdot \mathbf{F} + \mathbf{A}_{wheel} \cdot \mathbf{F}_{wheel} = \mathbf{b}, \quad (3)$$

where

$$\mathbf{A}_c = \begin{bmatrix} \mathbf{R}_\gamma & \dots & \mathbf{I}_2 \\ (\mathbf{R}_\gamma^T \mathbf{r}_{c1}) \times & \dots & \mathbf{r}_{c4} \times \end{bmatrix} \quad (4)$$

$$\mathbf{A}_{wheel} = \begin{bmatrix} \mathbf{R}_\gamma & \dots & \mathbf{I}_2 \\ (\mathbf{R}_\gamma^T \mathbf{r}_{wheel1}) \times & \dots & \mathbf{r}_{wheel4} \times \end{bmatrix} \quad (5)$$

$$\mathbf{b} = \begin{bmatrix} m(\ddot{\mathbf{p}}_c + \mathbf{g}) \\ \mathbf{I}_w \dot{\boldsymbol{\omega}} \end{bmatrix}. \quad (6)$$

\mathbf{R}_γ is 2D rotation matrix by the slope angle γ ,

$$\mathbf{R}_\gamma = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) \\ -\sin(\gamma) & \cos(\gamma) \end{bmatrix} \quad (7)$$

The term \mathbf{F} is a force vector containing 2D ground reaction forces at the center of each wheel, $\mathbf{F} = [\mathbf{F}_{1,x}, \mathbf{F}_{1,z}, \dots, \mathbf{F}_{4,x}, \mathbf{F}_{4,z}]^T$. Similarly, \mathbf{F}_{wheel} is a column vector containing the wheel traction forces obtained from equation (2), $\mathbf{F}_{wheel} = [\mathbf{F}_{t_1}, 0, \dots, \mathbf{F}_{t_4}, 0]^T$. Note that the wheel only contributes force in the direction of the ground.

In equation (4) and (5), \mathbf{r}_{c_i} is the vector of distance from trunk CoM to center of the i^{th} wheel and \mathbf{r}_{wheel_i} is the distance from CoM to ground contact point of the i^{th} wheel, $i = 1, \dots, 4$, $\mathbf{r}_{c_i} \times = [\mathbf{r}_{c_i,z}, -\mathbf{r}_{c_i,x}]$, and $\mathbf{r}_{wheel_i} \times = [\mathbf{r}_{wheel_i,z}, -\mathbf{r}_{wheel_i,x}]$. In equation (6), $\ddot{\mathbf{p}}_c$ is the linear acceleration of the robot CoM in 2D (x and z-direction), \mathbf{g} is the gravity vector in 2D, \mathbf{I}_w is the rotation inertia of the robot body in the world frame, and $\dot{\omega}$ is the angular acceleration of the robot body around y-axis.

Similarly, if the robot's rear wheels are in contact with a sloped surface, the formulation in equation(4) and (5) should be modified to reflect the change of frames of the corresponding ground reaction forces.

B. QP Formulation of Balance Controller

Since the dynamics model 3 presented in the previous section is linear, we can incorporate the dynamics constraints in a quadratic program (QP) as follow. The formulation of this balance controller is inspired by [25] which was developed for quadruped robots using a single rigid body dynamics. In this work, we adopt this principle using our model of rigid body dynamics with wheels for our wheel-legged robots. The balancing control employs a PD control policy of the robot body CoM position. It also makes sure the inequality constraints such as force saturation and friction constraints are stratified in the optimal solution.

In this work, we will design a controller that tends to drive the robot dynamics to the following desired dynamics that follows a PD control law:

$$\begin{bmatrix} \ddot{\mathbf{p}}_{c,des} \\ \dot{\omega}_{des} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{p,p}(\mathbf{p}_{c,des} - \mathbf{p}_c) + \mathbf{K}_{d,p}(\dot{\mathbf{p}}_{c,des} - \dot{\mathbf{p}}_c) \\ \mathbf{K}_{p,\omega}(\theta_{des} - \theta) + \mathbf{K}_{d,\omega}(\omega_{des} - \omega) \end{bmatrix}. \quad (8)$$

The right-hand side of equation (8) contains user input command in terms of desired CoM position, velocity, pitch angle θ_{des} and angular velocity. The left-hand side can then be used to represent the desired \mathbf{b} matrix in the dynamics equation (3):

$$\mathbf{b}_{des} = \begin{bmatrix} \mathbf{m}(\ddot{\mathbf{p}}_{c,des} + \mathbf{g}) \\ \mathbf{I}_w \dot{\omega}_{des} \end{bmatrix}. \quad (9)$$

Then, we can obtain the desired dynamics by driving the left-hand side of the dynamics equation (3) to:

$$\mathbf{A}_c \cdot \mathbf{F} \rightarrow \mathbf{b}_{des} - \mathbf{A}_{wheel} \cdot \mathbf{F}_{wheel} \quad (10)$$

where the value of \mathbf{F}_{wheel} is dependent on wheel torques, controller by rolling controller in Section IV-C.

Equation (10) can be obtained by the following quadratic program:

$$\mathbf{F}_{opt} = \min_{\mathbf{F} \in \mathbb{R}^8} \mathbf{D}^T \mathbf{S} \mathbf{D} + \alpha \| \mathbf{F} \|^2 + \beta \| \mathbf{F} - \mathbf{F}_{opt,prev} \|^2 \quad (11)$$

$$\text{s.t. } \mathbf{C}\mathbf{F} \leq \mathbf{d} \quad (12)$$

where, $\mathbf{D} = \mathbf{A}_c \mathbf{F} + \mathbf{A}_{wheel} \mathbf{F}_{wheel} - \mathbf{b}_{des}$. Equation (11) is the cost function of this QP problem, its main goals are driving robot CoM location close to the desired command, minimizing the optimal force \mathbf{F}_{opt} , and filtering the difference of optimal force at the current time step and previous

optimal force $\mathbf{F}_{opt,prev}$. These three tasks are weight by S , α , and β to determine the task priorities. Equation (12) summarizes the friction cone constraint and saturation of computed ground reaction force.

The resulting optimal force inputs from the QP problem in equation (11) and (12), $\mathbf{F}_{opt} = [\mathbf{F}_{1,x}, \mathbf{F}_{1,z}, \dots, \mathbf{F}_{4,x}, \mathbf{F}_{4,z}]^T$ are then mapped to the thigh and calf joint torques for each leg by:

$$\tau_{QP,i} = \mathbf{J}_i^T \begin{bmatrix} \mathbf{F}_{i,x} \\ \mathbf{F}_{i,z} \end{bmatrix}, \quad (13)$$

where \mathbf{J}_i is the leg Jacobian matrix of i th leg.

C. 3D Wheel Rolling Control

While the QP force control provides balance and stability to the wheel-legged robot during motion, the forward velocity and yaw control of the robot can be realized by leveraging the rolling motion of the wheels. With a given CoM velocity command, the wheel torque is calculated using the following feedback law:

$$\tau_{wheel} = \mathbf{K}_{d,wheel}(\dot{q}_{wheel,des} - \dot{q}_{wheel}), \quad (14)$$

where \dot{q}_{wheel} is the measurement of the wheel joint angular velocity, and

$$\dot{q}_{wheel,des} = \frac{\dot{p}_{c_x,des}}{R_{wheel}}, \quad (15)$$

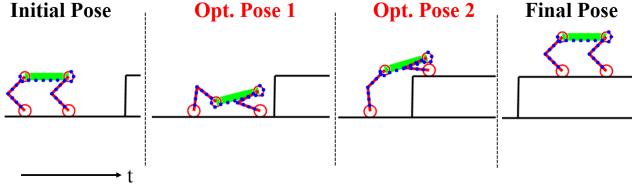
with $\dot{p}_{c_x,des}$ being the desired forward velocity. On top of this rolling control based on the input linear velocity command, the controller can also track a desired yaw speed command during rolling motion. This is achieved by assigning a difference $\Delta \dot{q}_{wheel,des}$ in commanded angular speed to the left and right wheel joints, to achieve a feedback turning control. And $\Delta \dot{q}_{wheel}$ is adjusted by a yaw-speed ($\dot{\psi}$) controller,

$$\Delta \dot{q}_{wheel} = \mathbf{K}_{d,\dot{\psi}}(\dot{\psi}_{des} - \dot{\psi}). \quad (16)$$

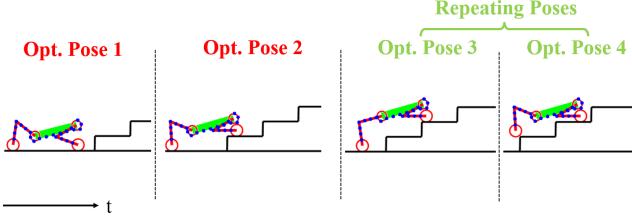
The combination of QP force-based balance control and rolling control allows the wheel-legged robot to have stable dynamic locomotion over uneven terrain by taking the advantage of wheel rolling traction.

V. POSE OPTIMIZATION

In the previous sections, we have explained the architecture of the hybrid control method that enables stable locomotion of wheel-legged robots only leveraging the wheel rolling motion. However, with only balancing control and wheel rolling control, the robot is unable to pass more complex terrains such as terrain with a very steep slope and tall staircases, such as an example shown in Fig. 6. We propose a pose optimization method based on robot kinematics and can solve for optimal poses for a given task that consists of rolling over high obstacles while ensuring collision-free constraints with the terrain. Pose optimization is a motion planning technique that can be used with terrain map information as input. This approach will allow the robot to roll over tall obstacles that exceed the robot's nominal height.



(a) Four main poses during a single-stair terrain problem. The initial and final poses are known. Pose 1 and Pose 2 are solved by optimization.



(b) Optimal pose sequence for a multi-stair terrain problem. Pose 3 and 4 are repeated for each additional stair, assuming stairs have uniform rise and run.

Fig. 5: Pose Optimization Result Illustrations (a) Single-stair Task. (b) Multiple-stair task. The collision model is represented in blue dots in illustrations

A. Problem Description

Our approach, in the task where the wheel-legged robot needs to climb up a high stair, focuses on manipulating the robot to maintain and transform between different poses in order to create a large stability region while the robot is on a slope and while the body is at a significant pitch angle. In order to decrease the problem computation expense, the pose optimization only needs to compute two optimal poses at certain positions in a single-stair obstacle task. As Fig. 5a has shown, the two pose locations have the largest kinematic changes during this transition from ground to the upper surface. To ensure these critical poses are collision-free, we simplified the collision model of the robot to 25 possible collision points placed across the robot trunk and limbs. The collision model is also illustrated in Fig. 5a and 5b. The total number of the points is determined by trial-and-error from simulation results and is the middle ground of a well-constrained problem vs. computation time.

The optimization method also has great potential of extending its usage to more complex terrains such as multiple-stair obstacles. Two additional poses are added to the pose sequence with every additional stair. When the multi-stair terrain has uniformed stair runs and rises, the additional poses can be repeated for each additional stair without the requirement to solve repetitively, illustrated in Fig. 5b.

B. Nonlinear Programming Problem (NLP) Formulation

The formal NLP of the pose optimization is defined as follows,

$$\text{Minimize : } \mathbf{J}_i = (\mathbf{p}_c^{ref} - \mathbf{p}_c)^T \mathbf{Q} (\mathbf{p}_c^{ref} - \mathbf{p}_c) \quad (17)$$

$$\text{Find : } \mathbf{X}_i = [\mathbf{x}_c, \mathbf{z}_c, \boldsymbol{\theta}, \mathbf{q}]_i^T \quad (18)$$

$$\text{Subject to : } \mathbf{p}_{wheel}(\mathbf{X}_i) = \text{terrain}(\mathbf{p}_c) \quad (19)$$

$$p_{rw,x}(\mathbf{X}_i) \leq p_{rh,x}(\mathbf{X}_i) \quad (20)$$

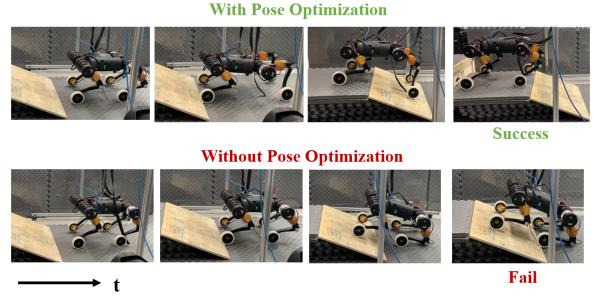


Fig. 6: Wheel-legged Robot Rolling over a Ramp. Experiment snapshots of rolling over a ramp with a height of 0.25m and slope angle of 30°, with and without pose optimization.

$$\text{InCollision}(\mathbf{p}_{cm}, \text{terrain}(\mathbf{p}_c)) = \text{False} \quad (21)$$

$$\mathbf{q}_{min} < \mathbf{q} < \mathbf{q}_{max} \quad (22)$$

The objective of the pose optimization is to find the optimal pose i at pose location $\mathbf{p}_c^{ref} = [\mathbf{x}_c^{ref}, \mathbf{z}_c^{ref}]^T$, the reference pose location is resulted from the terrain information. Hence, the cost function \mathbf{J}_i aims to find the closest possible location that satisfy the given NLP constraints. In this optimization framework, we use kinematic constraints, therefore, a feasible pose solution \mathbf{X}_i should contain CoM 2D locations \mathbf{x}_c and \mathbf{z}_c , body pitch angle $\boldsymbol{\theta}$, and limb joint angles $\mathbf{q}^* = [q_1, q_2, q_3, q_4]^T$. \mathbf{Q} is a diagonal weighting matrix. It is necessary to allow CoM z-direction location delta $\mathbf{z}_c^{ref} - \mathbf{z}_c$ to have certain flexibility in order to solve for the most optimal poses, in another word, we choose the weight in the z-direction location delta to be much smaller than that of x-direction location delta.

The optimization problem is subjected to several nonlinear constraints, shown in equation (19) to (22). The rim of each wheel is defined as the ground contact geometry whose geometry location \mathbf{p}_{wheel} can be derived by FK with optimization variables \mathbf{X}_i . The rim of the wheel is constrained by equation(19) to be on the terrain in each pose. In equation(20), we constrain x-direction of the rear wheel ground contact location $\mathbf{p}_{rw,x}$ to be less than that of rear hip $\mathbf{p}_{rh,x}$. Both of these locations can be derived by FK with \mathbf{X}_i . This will allow a larger support region in optimal poses, to prevent the robot from falling backward due to significantly large pitch angle during the task. Equation(21) can be implemented by integer programming in NLP. A custom function `InCollision` based on one point-line interception is applied here to determine whether the collision model is in contact with the terrain model (i.e. collision model should always be above terrain). The location of the collision model point cloud \mathbf{p}_{cm} is determined by FK. Lastly, in equation(22), the joint angles \mathbf{q}^* in the optimization variable are bounded by the physical joint limits of the hardware platform.

C. Real-time Pose Planning

After the optimal poses are solved for a task, we use real-time pose planning in order to command desired joint angle and pitch angle online. The joint angle and pitch

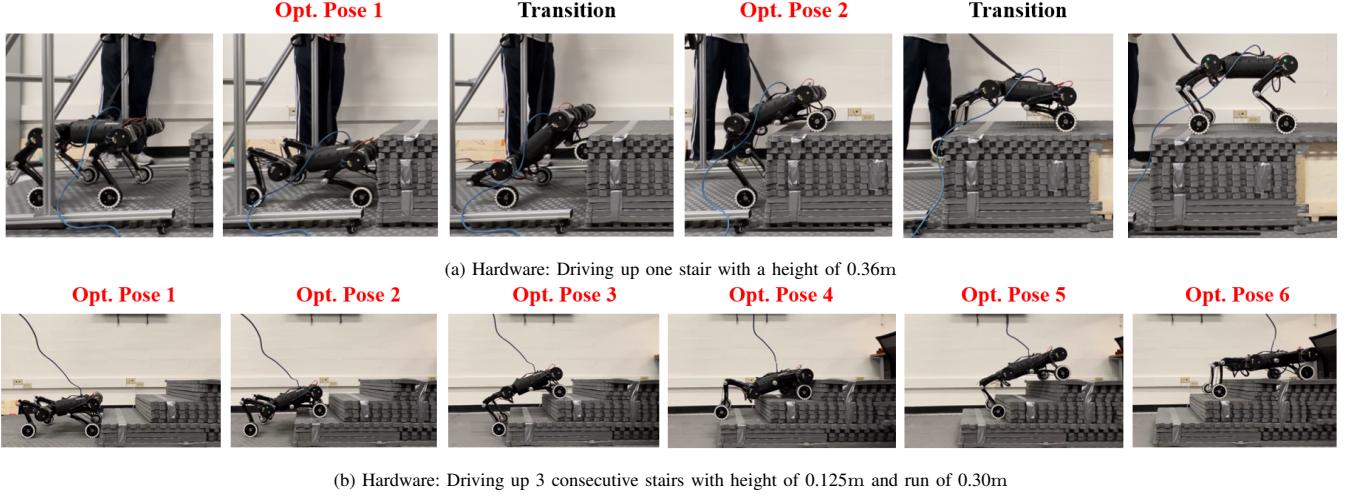


Fig. 7: Hardware Experiment Snapshots.

angle trajectory is linearly interpolated from initial pose to intermediate optimal poses and then to final pose. The general interpolation equation at time t from pose i (\mathbf{q}_i and θ_i) to pose $i+1$ (\mathbf{q}_{i+1} and θ_{i+1}) with a transition phase Δt is as follows,

$$\mathbf{q}_{des}(t) = \mathbf{q}_i + \frac{(\mathbf{q}_{i+1} - \mathbf{q}_i) \cdot (t - t_{0,i})}{\Delta t} \quad (23)$$

$$\theta_{des}(t) = \theta_i + \frac{(\theta_{i+1} - \theta_i) \cdot (t - t_{0,i})}{\Delta t} \quad (24)$$

Where $t_{0,i}$ is the initial time of transition from pose i to pose $i+1$. Since the optimization outputs only optimal joint and pitch angles, a tracking controller is needed to enable the robot to perform certain pose at desired location and timing. The optimal poses \mathbf{q}^* are tracked by a joint PD controller, while the optimal pitch angle θ^* is tracked by QP balancing control. The timing of each pose \hat{t}_1 and \hat{t}_2 is estimated by the current average wheel joint speed $\bar{\dot{q}}_{wheel}$ and terrain stair slope angle γ and height h ,

$$\hat{t}_1 = t + \frac{(\Delta x - \mathbf{R}_{wheel})}{\mathbf{R}_{wheel} \bar{\dot{q}}_{wheel}} \quad (25)$$

$$\hat{t}_2 = \hat{t}_1 + \frac{(\frac{h}{\sin(\gamma)} + \mathbf{R}_{wheel})}{\mathbf{R}_{wheel} \bar{\dot{q}}_{wheel}} \quad (26)$$

t is the current timing at the start of estimation.

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{QP} + \mathbf{K}_{p,q}(\mathbf{q}^* - \mathbf{q}) + \mathbf{K}_{d,q}(\dot{\mathbf{q}}^* - \dot{\mathbf{q}}) \quad (27)$$

The tracking controller works alongside QP force-based balance control to balance the robot while the robot is rolling over high obstacles. This approach is also a well-established tracking controller with motion planning (e.g. [14]). The resulting control input $\boldsymbol{\tau}$ in terms of joint torques for thigh and calf joints is a combination of torque from balance controller $\boldsymbol{\tau}_{QP}$ and tracking controller, as shown in equation (27).

VI. EXPERIMENT RESULTS

In this section, we present highlighted hardware experiment results with the proposed approach.

The pose optimization framework can be implemented by many popular modern NLP solvers. We have implemented and executed the pose optimization in MATLAB fmincon Sequential Quadratic Programming (SQP) solver for our simulation and hardware experiments. The offline computation time for a single-stair pose optimization task is in the range of 0.3s to 0.5s. As a benchmark, the PC hardware platform we use for offline motion planning has an AMD Ryzen 5-5600X CPU clocked at 4.65GHz. We expect the computation cost to be scaled down further when the pose optimization is implemented in a C++ based solver such as IPOPT in the future.

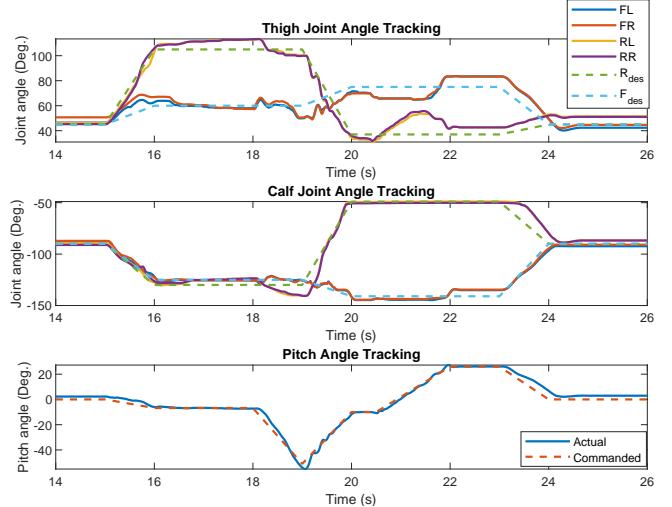


Fig. 8: Hardware Experiment Plots. Joint angle and pitch angle tracking plots in single-stair obstacle task with a height of 0.36m, with pose optimization.

We have successfully validated our proposed approach on the real robot hardware. In hardware experiments, our method with pose optimization has also shown its advantages compared to without pose optimization. Fig. 6 shows snapshots of experiment results comparing the performance of these two approaches. The robot is commanded to roll

up a very simple 0.25m high ramp with a slope angle of 30°. It is observed that using the nominal quadruped pose (without pose optimization), the CoM location falls towards the back of the leg support region and causes significant pitch angle error. In addition, the leg configuration does not favor traversing over slopes due to the collision of rear calf joints and the ground. Whereas, with our proposed approach using pose optimization, the CoM location stays centered at the support region and body pitch angle is minimal.

We demonstrated our proposed approach in single-stair and 3-stair obstacle experiments. Our hardware successfully achieved climbing up a 0.36m stair. Nominal quadruped robot gait cannot climb up such a high stair because it is constrained by the nominal height of the robot. As a reference, the nominal standing height of our wheel-legged robot is 0.33m. Snapshots of this experiment are shown in Fig. 7a. The pitch and joint angle tracking plots of this successful single-stair experiment with pose optimization are shown in Fig. 8. This also demonstrates the effectiveness of our balancing control presented in this paper. We have also validated the versatility of our pose optimization framework in a 3-consecutive-stair task, snapshots are shown in Fig. 7b. Each stair has a stair rise of 0.125m and stair run of 0.30m.

VII. CONCLUSIONS

In conclusion, we proposed an effective approach of balancing the 12 DoF wheel-legged robot with QP force-based control that employs modified simplified dynamics that considers the effects of wheel dynamics. By leveraging the wheel traction in high obstacle terrain locomotion, we have also proposed an optimization method in motion planning that solves for favorable poses during stair-climbing tasks. The optimal poses are tracked by a joint PD tracking controller, along with QP balancing controller. In hardware implementation, the robot is capable of climbing up on a 0.36 m stair (higher than robot nominal height). The versatility of the pose optimization framework is validated through successful multi-stair task experiments and proven to have superior performance as compared to normal quadruped poses during such tasks.

REFERENCES

- [1] S. Kajita and B. Espiau, “Legged robot,” 2008.
- [2] Q. Nguyen, A. Agrawal, X. Da, W. C. Martin, H. Geyer, J. W. Grizzle, and K. Sreenath, “Dynamic walking on randomly-varying discrete terrain with one-step preview,” in *Robotics: Science and Systems*, vol. 2, no. 3, 2017.
- [3] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, “Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway,” in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4559–4566.
- [4] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, “Mit cheetah 3: Design and control of a robust, dynamic quadruped robot,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.
- [5] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, “Anymal-a highly mobile and dynamic quadrupedal robot,” in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.
- [6] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick *et al.*, “Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2518–2525.
- [7] S. Seok, A. Wang, M. Y. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, “Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot,” *Ieee/asmc transactions on mechatronics*, vol. 20, no. 3, pp. 1117–1129, 2014.
- [8] D. Pongas, M. Mistry, and S. Schaal, “A robust quadruped walking gait for traversing rough terrain,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1474–1479.
- [9] M. Schwarz, T. Rodehutkors, M. Schreiber, and S. Behnke, “Hybrid driving-stepping locomotion with the wheeled-legged robot momaro,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5589–5595.
- [10] C. Grand, F. Benamar, F. Plumet, and P. Bidaud, “Stability and traction optimization of a reconfigurable wheel-legged robot,” *The International Journal of Robotics Research*, vol. 23, no. 10-11, pp. 1041–1058, 2004.
- [11] C. Grand, F. BenAmar, F. Plumet, and P. Bidaud, “Decoupled control of posture and trajectory of the hybrid wheel-legged robot hylos,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 5111–5116.
- [12] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, “Whole-body mpc and online gait sequence generation for wheeled-legged robots,” *arXiv preprint arXiv:2010.06322*, 2020.
- [13] V. Klemm, A. Morra, L. Gulich, D. Mannhart, D. Rohr, M. Kamel, Y. de Viragh, and R. Siegwart, “Lqr-assisted whole-body control of a wheeled bipedal robot with kinematic loops,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3745–3752, 2020.
- [14] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, and S. Kim, “Optimized jumping on the mit cheetah 3 robot,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7448–7454.
- [15] J. Li and Q. Nguyen, “Force-and-moment-based model predictive control for achieving highly dynamic locomotion on bipedal robots,” *arXiv preprint arXiv:2104.00065*, 2021.
- [16] B. Katz, J. Di Carlo, and S. Kim, “Mini cheetah: A platform for pushing the limits of dynamic quadruped control,” in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6295–6301.
- [17] G. Bellegarda and K. Byl, “Trajectory optimization for a wheel-legged system for dynamic maneuvers that allow for wheel slip,” in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 7776–7781.
- [18] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, “Gait and trajectory optimization for legged systems through phase-based end-effector parameterization,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.
- [19] Y. Luo, Q. Li, and Z. Liu, “Design and optimization of wheel-legged robot: Rolling-wolf,” *Chinese journal of mechanical engineering*, vol. 27, no. 6, pp. 1133–1142, 2014.
- [20] V. S. Medeiros, E. Jelavic, M. Bjelonic, R. Siegwart, M. A. Meggiolaro, and M. Hutter, “Trajectory optimization for wheeled-legged quadrupedal robots driving in challenging terrain,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4172–4179, 2020.
- [21] M. Chignoli, D. Kim, E. Stanger-Jones, and S. Kim, “The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors,” in *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2021, pp. 1–8.
- [22] G. Besseron, C. Grand, F. B. Amar, F. Plumet, and P. Bidaud, “Locomotion modes of an hybrid wheel-legged robot,” in *Climbing and Walking Robots*. Springer, 2005, pp. 825–833.
- [23] F. B. Amar, C. Grand, G. Besseron, and F. Plumet, “Performance evaluation of locomotion modes of an hybrid wheel-legged robot for self-adaptation to ground conditions,” in *ASTRA'04, 8th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, 2004.
- [24] L. Ni, F. Ma, and L. Wu, “Posture control of a four-wheel-legged robot with a suspension system,” *IEEE Access*, vol. 8, pp. 152790–152804, 2020.
- [25] M. Focchi, A. Del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, “High-slope terrain locomotion for torque-controlled quadruped robots,” *Autonomous Robots*, vol. 41, no. 1, pp. 259–272, 2017.