IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, USA, October 2023.

Dynamic Hybrid Locomotion and Jumping for Wheeled-Legged Quadrupeds

Mojtaba Hosseini[1], Diego Rodriguez[2], and Sven Behnke[1]

***Abstract*— Hybrid wheeled-legged quadrupeds have the potential to navigate challenging terrain with agility and speed and over long distances. However, obstacles can impede their progress by requiring the robots to either slow down to step over obstacles or modify their path to circumvent the obstacles. We propose a motion optimization framework for quadruped robots that incorporates non-steerable wheels and dynamic jumps, enabling them to perform hybrid wheeled-legged locomotion while overcoming obstacles without slowing down. Our approach involves a model predictive controller that uses a time-varying rigid body dynamics model of the robot, including legs and wheels, to track dynamic motions such as jumping. We also introduce a method for driving with minimal leg swings to reduce energy consumption by sparing the effort involved in lifting the wheels. Our method was tested successfully on the wheeled Mini Cheetah and the Unitree AlienGo robots. Further videos and results are available at `https://www.ais.uni-bonn.de/~hosseini/iros2023`**

I. Introduction

Legged robots efficiently navigate irregular terrains and obstacles but at a slow pace and high energy consumption. Conversely, wheeled robots offer speed and energy efficiency on flat terrains but fail to overcome obstacles larger than their wheel radius. To address these issues, we propose a novel locomotion framework blending the benefits of both, depicted in Fig. 1. It uses efficient driving on flat terrains and stepping or jumping to overcome obstacles, improving the robot's speed and energy efficiency without sacrificing versatility. The driving jump motions make use of the robot's dynamics to negotiate obstacles without slowing down. This hybrid approach mitigates the trade-off between efficiency and speed by minimizing leg swings during driving and triggering stepping based on terrain irregularities or user commands. Our novel methods enable dynamic hybrid driving-stepping locomotion for quadrupeds which are capable of jumping by using model predictive control (MPC), taking into account the wheels in terms of their mass and inertia tensor, with main contributions:

1) a dynamic robot controller utilizing body and foot trajectory planning for effective mass, inertia, and center of mass (CoM) trajectory prediction, forming a linear time-varying MPC (LTV-MPC),
2) a novel approach for the online generation of jumping motions on hybrid and non-hybrid robots, and
3) a driving assistant system that curtails leg swings to ensure a natural hybrid driving, while actively responding to disturbances and user gait commands.

[1]Autonomous Intelligent Systems, University of Bonn, Germany `behnke@ais.uni-bonn.de`
[2]Dexterity Inc., USA `diego.rodriguez@dexterity.ai`

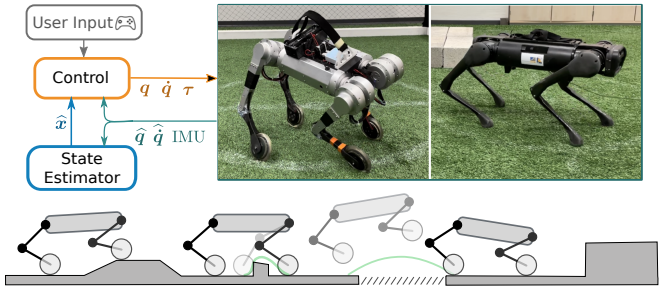

Fig. 1. Top: Using joint positions $\hat{\boldsymbol{q}}$, velocities $\hat{\dot{\boldsymbol{q}}}$, and IMU data, our hybrid framework computes the robot's state vector $\hat{\boldsymbol{x}}$, aiding in the generation of joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$, and torque commands $\boldsymbol{\tau}$ [1]. Bottom: The hybrid robot leverages motorized wheels for swift terrain traversal, steps to free a trapped leg, and performs a driving-jumping motion for terrain obstacles or gaps.

II. Related Work

The study in [2] introduces a whole-body walking controller using Quadratic Programming (QP), yet its perturbation response is limited. Bellicoso et al. [3] designed various quadruped gait patterns but lacked flexibility in real-world scenarios. It is extended in [4] by accommodating the Center of Mass (CoM) position and introducing separate footholds for broader gait applications with full flight phases. The Cheetah 2 robot [5] showcased jump motions to overcome obstacles using an MPC, albeit without perturbation handling. A controller introduced in [6] provides a robust response to disturbances by simplifying the robot's dynamics to a single rigid body with massless legs. This approach assumed constant body mass and inertia over the prediction horizon, ignoring link dynamics. It is improved in [7] to incorporate a Whole-Body Controller (WBC) computing joint commands, tracking commanded body and foot states while optimizing for reaction forces found by MPC.

Hybrid locomotion platforms [8], [9] employ offline Trajectory Optimization (TO) for planning motion on flat terrain. The hybrid driving-stepping methods introduced in [10], [11], [12] operate wheels and walk separately, which is extended in [13] by using a single optimization framework for base and wheel trajectories using linearized ZMP constraints, albeit with limited real-world experimentation and a slow update rate. In [14], an optimization framework incorporates the additional degrees of freedom introduced by the wheels into the motion generation. The optimization problem is split into end-effector and base trajectory planning, similar to [15], to make locomotion planning more manageable. In [16], utility values are assigned to each leg, enabling dy-

namic aperiodic motions; however, jumping is not addressed. Lastly, the bipedal robot Ascento [17] can navigate, balance, and jump over obstacles, but neglects leg link and motor dynamics, which is significant due to the weight of the wheels in relation to the body.

III. Method

Quadruped dynamics involve leg movements that alternate between contact and swing phases. Leg mass does not impact the robot's effective mass during contact, but alters effective mass, inertia, and CoM during the swing phase, especially in hybrid robots with heavy wheels. Hence, leg mass must be modeled in complex actions like bounding, pronking, and jumping. Force exertion on the robot's feet is restricted by ground contact, risk of slippage, and the requirement for non-negative forces along the z axis. For a successful jump, a controller must anticipate contact sequence changes, calculate the restricted ground reaction forces pre-flight, and optimize them for flight duration and safe, minimal-inclination landing. Post-landing, the controller needs to counteract gravity without significant height or orientation deviation. Stability hinges on understanding the imminent foot contact pattern.

Our method develops an MPC to minimize control effort and trajectory deviations by optimizing future state and control sequences. It utilizes a time-varying robot model, encompassing legs' mass, inertia, and CoM, to approximate composite inertia tensor, effective mass, and robot's CoM over the prediction horizon. We employ our prior state estimator [1] that facilitated walking on uneven terrain with robust state estimation including the impact of driving with wheels on the robot's state. This enabled our controller to generate footstep offsets based on gait and perturbations alone, thus allowing smooth integration of walking and driving in our driving assistant application (III-C).

Our control framework, depicted in Fig. 2, utilizes user commands, trajectory, contact, and footstep planners to generate trajectories for the MPC and WBC, set for a configurable horizon length N based on processing power for appropriate control update rate ($N = 10$ in our experiments). The MPC creates reaction force commands aiding the robot to follow reference trajectories, which are further refined using WBC for accurate and quick joint control. The wheel controller then transforms the refined forces into torques considering the wheel's current effective radius, thereby preventing wheel joint rotation against reaction forces, and actively corrects the stance leg's position along the rolling direction [1]. Applicable to both hybrid and non-hybrid quadruped robots, our system enables seamless transitions between arbitrary gaits, jumping motions, and wheel driving, facilitating overcoming height differences or obstacles by jumping without speed reduction, thus enhancing overall locomotion speed.

A. Time-Varying Dynamic Controller

Over a finite time horizon N, the MPC employs the robot's dynamic model to anticipate flight or underactuation

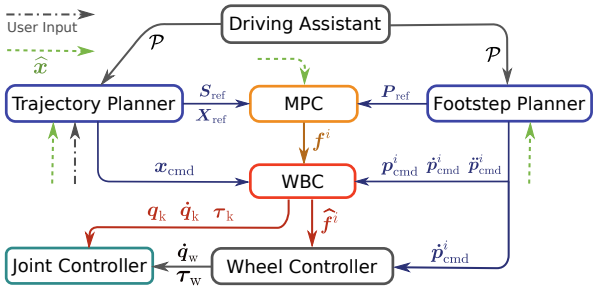

Fig. 2. Control framework diagram. The body state vector $\hat{\boldsymbol{x}}$, with i denoting leg number, is derived using the state estimator. The driving assistant (Section III-C) defines parameter set $\mathcal{P}$ used by the trajectory and footstep planner to produce reference trajectories for state, contact, and foot positions ($\boldsymbol{X}_{\text{ref}}$, $\boldsymbol{S}_{\text{ref}}$, and $\boldsymbol{P}_{\text{ref}}$) and current state commands ($\boldsymbol{x}_{\text{cmd}}$, $\boldsymbol{p}^i_{\text{cmd}}$). The MPC then computes optimal reaction forces $\boldsymbol{f}^i$ from these trajectories, employed in the WBC with state and foot commands to generate joint' position, velocity, and torque commands ($\boldsymbol{q}_{\text{k}}$, $\dot{\boldsymbol{q}}_{\text{k}}$, and $\boldsymbol{\tau}_{\text{k}}$) and refined reaction forces $\widehat{\boldsymbol{f}}^i$. Lastly, the wheel controller calculates the wheels' rotational speed and torque ($\dot{\boldsymbol{q}}_{\text{w}}$ and $\boldsymbol{\tau}_{\text{w}}$) using commanded foot velocities and refined reaction forces.

periods and finds optimal reaction forces for the prediction horizon, ensuring the robot follows the reference trajectory $\boldsymbol{X}_{\text{ref}}$. Our method extends [7] by eliminating massless leg assumptions and integrating rolling wheels into the model for a hybrid quadruped. Despite a computationally demanding time-varying model, our software's parallel implementation achieves higher update rates for the MPC (100 Hz compared to 30 Hz in [7]).

1) Trajectory and Footstep Planner: generates reference trajectories for the body state $\boldsymbol{X}_{\text{ref}} := \begin{pmatrix}\boldsymbol{x}_1 & \dots & \boldsymbol{x}_N\end{pmatrix}$ and the contact state $\boldsymbol{S}_{\text{ref}} := \begin{pmatrix}\boldsymbol{s}_1 & \dots & \boldsymbol{s}_N\end{pmatrix}$ over a N segment prediction window for the MPC to track. At each time step k, the body state $\boldsymbol{x}_k := \begin{pmatrix}\boldsymbol{\theta} & \boldsymbol{p} & \boldsymbol{\omega} & \dot{\boldsymbol{p}}\end{pmatrix}$ contains the orientation, position, the rotational and linear velocity of the robot, while the vector $\boldsymbol{s}_k := \begin{pmatrix}s^1 & \dots & s^4\end{pmatrix}$ denotes all legs' contact status with $s^i \in 1, 0$ for contact and swing respectively. In each control loop, the reference trajectory $\boldsymbol{X}_{\text{ref}}$ is generated based on the commanded stepping and driving velocities, as well as the target orientation, and position of the robot, whereas $\boldsymbol{S}_{\text{ref}}$ is generated according to the gait pattern. The footstep planner forms foot trajectories $\boldsymbol{P}_{\text{ref}}$ over the prediction horizon. Each control step determines the target step offset for each swing foot by:

$$\boldsymbol{p}_{\text{symmetry}} = k_{\text{s}} \tfrac{t_{\text{c}}}{2} \left(\widehat{\boldsymbol{v}}_{\text{g}} + \boldsymbol{v}_{\text{sh}}\right) + k_1 \left(\widehat{\boldsymbol{v}}_{\text{g}} - \boldsymbol{v}^{\text{cmd}}_{\text{g}}\right) + k_2 \boldsymbol{v}_{\text{sh}}, \quad (1)$$

$$\widehat{\boldsymbol{v}}_{\text{g}} = \dot{\boldsymbol{p}} - \dot{\boldsymbol{p}}_w, \quad (2)$$

$$\boldsymbol{v}_{\text{sh}} = \boldsymbol{\omega} \times \left(\mathbf{R}(\boldsymbol{\theta}) \boldsymbol{p}_{\text{sh}}\right), \quad (3)$$

where t_{c} is the contact phase duration, $\boldsymbol{v}^{\text{cmd}}_{\text{g}}$ the gait velocity command, $\dot{\boldsymbol{p}}$ and $\boldsymbol{\omega}$ the robot's estimated linear and angular velocities. k_s sets swing symmetry and defaults to 1, while k_1 and k_2 are the feedback gains for linear and angular motion errors, respectively. Gait velocity $\widehat{\boldsymbol{v}}_{\text{g}}$ (excluding driving velocity) and linear velocity $\boldsymbol{v}_{\text{sh}}$, from the current yaw rate, are estimated by the state estimator. $\boldsymbol{p}_{\text{sh}}$ is the shoulder position in the body frame and $\mathbf{R}(\boldsymbol{\theta})$ the body orientation rotation matrix. $\boldsymbol{p}_{\text{symmetry}}$ applies Raibert's heuristic, reducing leg

extension during stepping and ensuring identical landing and leaving leg angles [18]. Feedback terms enhance control robustness against external pushes altering the robot's velocity ($\dot{\boldsymbol{p}}$), by amplifying the error between $\widehat{\boldsymbol{v}}_{\mathrm{g}}$ and $\boldsymbol{v}_{\mathrm{g}}^{\mathrm{cmd}}$, resulting in a farther target landing step position that counteracts the push.

Equation (1) enhances robustness to external yaw torques from [7], accounting for the induced shoulder linear velocity as the robot turns and allowing hybrid locomotion that dynamically updates symmetrical footstep offsets via k_s. When driving without stepping, $\boldsymbol{v}_{\mathrm{g}}^{\mathrm{cmd}}$ is 0, meaning $\boldsymbol{p}_{\mathrm{symmetry}}$ only depends on external pushes and yaw torques, minimizing leg swings and only stepping when perturbed or legs are stuck.

2) Whole-Body Kinematic Model: At each step k, robot's orientation $\boldsymbol{\theta}_k$, foot positions $\boldsymbol{p}_k^i$, and contact state $\boldsymbol{s}_k$ are inferred from the reference state $\boldsymbol{X}_{\mathrm{ref}}$, feet $\boldsymbol{P}_{\mathrm{ref}}$, and contact $\boldsymbol{S}_{\mathrm{ref}}$ trajectories. Leg joint angles computed from inverse kinematics over the foot positions relative to the body are used in forward kinematics, translating each limb's inertia tensor and CoM to the parent frame to obtain composite inertia and CoM for the entire robot. The mass of the wheel end-effector alters the CoM notably during the swing phase but is negligible in the contact phase. Thus, wheel mass is determined by the contact probability P_c [1], computing the body's effective mass as:

$$m_k = \ M + \sum_{i=1}^{4} (1 - P_c^i) m_w^i, \tag{4}$$

where M is the total weight of the robot excluding the wheels. This process generates trajectories over the prediction horizon for the robot's effective mass m, composite inertia tensor ${}_{\mathcal{B}}\mathbf{I}$, and CoM offset $\boldsymbol{p}^{\mathrm{CoM}}$.

3) Time-Varying Dynamic Model: We extend [7] by predicting a reduced single rigid body at each time step k using the above whole-body kinematic model. The resultant lumped mass model for each k relates the angular momentum rate change to contact points' reaction forces, as per these dynamics in the global coordinate system:

$$m_k \ddot{\boldsymbol{p}}_k = \sum_{i=1}^{n_k} \boldsymbol{f}_k^i - \boldsymbol{g}, \tag{5}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{I}_k \boldsymbol{\omega}) = \sum_{i=1}^{n_k} \boldsymbol{r}_k^i \times \boldsymbol{f}_k^i, \tag{6}$$

where $\boldsymbol{g}$ represents gravity, k the time step, $\ddot{\boldsymbol{p}}$ the robot's acceleration, $\boldsymbol{f}^i$ reaction forces for n contacts, $\mathbf{I}$ and m the composite inertia tensor and effective mass respectively, $\boldsymbol{\omega}$ the body's angular velocity, and $\boldsymbol{r}^i$ the moment arm indicating the expected contact point position $\boldsymbol{p}^i$ relative to the robot's predicted CoM for the i-th leg:

$$\boldsymbol{r}^i = \boldsymbol{p}^i - \boldsymbol{p}^{\mathrm{CoM}}. \tag{7}$$

By assuming small off-diagonal terms for the inertia tensor in (6), the angular acceleration is approximated as:

$$\dot{\boldsymbol{\omega}} \approx \mathbf{I}_k^{-1} \sum_{i=1}^{n_k} \boldsymbol{r}_k^i \times \boldsymbol{f}_k^i, \tag{8}$$

which allows us to define the continuous dynamics of our time-varying model as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_t = \mathbf{A}_t(\boldsymbol{\theta})\, \boldsymbol{x}_t + \mathbf{B}_t \left(\boldsymbol{r}^1, \boldsymbol{r}^2, \boldsymbol{r}^3, \boldsymbol{r}^4, {}_{\mathcal{G}}\mathbf{I}, m \right) \boldsymbol{u}_t, \tag{9}$$

where $\mathbf{A}_t$ and $\mathbf{B}_t$ are the matrices for orientation and translational dynamics defined in [6] and $\boldsymbol{u}_t = \begin{bmatrix} \boldsymbol{f}_t^1 \dots \boldsymbol{f}_t^4 \end{bmatrix}^\top$ is the reaction forces applied by the controller.

Assuming a time-invariant system between steps, we sample approximated matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ at prediction step k using corresponding values. Precomputed arguments of $\mathbf{A}_k$ and $\mathbf{B}_k$ allow linearization of orientation dynamics using known $\boldsymbol{\theta}_k = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$ from $\boldsymbol{X}_{\mathrm{ref}}$. Resulting the rotation matrix $\mathbf{R} = \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_x(\psi)$ which transforms the inertia tensor ${}_{\mathcal{B}}\mathbf{I}_k$ generated from the whole-body kinematic model in body frame to global ${}_{\mathcal{G}}\mathbf{I}_k$, and the angular velocity $\boldsymbol{\omega}$ to body frame with:

$$\dot{\boldsymbol{\theta}} = \mathbf{J}(\boldsymbol{\theta}_k)\boldsymbol{\omega}, \tag{10}$$

$${}_{\mathcal{G}}\mathbf{I}_k = \mathbf{R}(\boldsymbol{\theta}_k)\, {}_{\mathcal{B}}\mathbf{I}_k \mathbf{R}(\boldsymbol{\theta}_k)^\top, \tag{11}$$

where $\mathbf{J}$ is the Jacobian matrix, mapping angular velocity to Euler angle rates for current $\boldsymbol{\theta}_k$, acting as an 'inverse' to the rotation matrix $\mathbf{R}$'s transformation [6].

Through a zero-order hold discrete-time model of the state transition in (9), we compute the discrete-time matrices $\hat{\mathbf{A}}_k$ and $\hat{\mathbf{B}}_k$ by solving the following block matrix exponential:

$$\begin{bmatrix} \hat{\mathbf{A}}_k & \hat{\mathbf{B}}_k \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \exp\left(\Delta t \begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right), \tag{12}$$

where Δt is the duration between each time step. The discrete-time form of the dynamics is then expressed by:

$$\boldsymbol{x}_{k+1} = \hat{\mathbf{A}}_k \boldsymbol{x}_k + \hat{\mathbf{B}}_k \boldsymbol{u}_k, \tag{13}$$

which is the standard state transition form for the convex QP formulation of the MPC. Our MPC adopts this formulation, penalizing deviation from the reference trajectory $\boldsymbol{x}_k$ and control input $\boldsymbol{u}_k$ at each step with penalty matrices, balancing tracking accuracy and control overhead while respecting the constraints. However, inaccuracies in $\hat{\mathbf{B}}_k$ matrices, due to their reliance on commanded reference trajectories, cause divergence from the reference in the presence of disturbances or delays. This deviation grows exponentially over time steps but is mitigated by our MPC's high update rates (about 100 Hz), which recalculates the reference trajectory based on the perturbed robot, effectively countering disturbances [6].

4) Whole-Body Controller: with a frequency of 100 Hz, the MPC is insufficient for controlling dynamic motions. Thus, our system employs a 500 Hz hierarchical whole-body controller [7], modified to follow MPC-generated optimal trajectories rather than reference ones. This modification improves performance during dynamic motions like bounding, pronking, and jumping, which are unmanageable with mere unrealistic reference trajectories.

B. Dynamic Jumping

Executing jumping motions is challenging due to the limited time to generate sufficient lift force while maintaining the desired orientation. This requires accurately predicting the leg transition from contact to swing, and determining the flight phase onset. Without this prediction, appropriate force trajectories cannot be generated for a successful jump. Accurate flight phase timing is critical, enabling maximum jump height through prolonged ground-based force exertion and allowing the state estimator to disengage from contact point corrections during flight [1].

For jump execution, stepping commands are zeroed, and $\boldsymbol{X}_{\text{ref}}$ is adjusted to the commanded jump height h_{cmd}, rotation, and position, enabling aerial turns and movements.

For the online generation of jumping motions, accurately identifying the jump phase is important.

To achieve this, the contact trajectory $\boldsymbol{S}_{\text{ref}}$ is initially configured to maintain contact throughout the predicted jump duration. The MPC module then computes an optimal jump trajectory, which is further analyzed to identify the time step at which the optimal height trajectory attains h_{cmd} or the maximum height feasible kinematically. This time step is utilized to modify $\boldsymbol{S}_{\text{ref}}$, prompting a shift into the swing phase at that instant. This comprehensive process is iterated, accompanied by continual refinement of the contact trajectory up until takeoff, ensuring a successful jump execution.

Post takeoff, $\boldsymbol{S}_{\text{ref}}$ is modified to switch back to contact after the flight duration t_{flight}, as follows:

$$t_{\text{flight}} = \frac{-\hat{\dot{h}}_f - \sqrt{\hat{\dot{h}}_f^2 - 2\left(\hat{h}_f - h_{\text{des}}\right)^+ g}}{g}, \tag{14}$$

where $\hat{h}_f$ and $\hat{\dot{h}}_f$ are the estimated position and velocity of the robot's height at the beginning of the flight phase, h_{des} is the desired robot height when landing and g is the gravity.

During flight, footstep offset is determined via (1) for increased landing stability. Post-flight, after t_{flight} has passed, all legs' $\boldsymbol{S}_{\text{ref}}$ are set for contact, allowing MPC to generate a stabilizing reaction force profile. Then, stepping locomotion resumes for continuous operation and landing regulation.

C. Driving Assistant

The driving assistant aims to preserve the desired kinematics of the robot and minimize unnecessary leg swings when locomotion is not required, i.e. when purely driving a hybrid robot. This results in a natural motion and reduces the energy consumption of the robot due to the effort required to lift the legs, specifically the heavy wheel at the end of each leg of a hybrid robot. However, if a foot encounters an obstacle that is rough or high, the driving is interrupted and the foot must step over the obstacle to continue moving. Despite the user-commanded stepping velocity, the turning commands and disturbances continue to force the robot to step because the non-steerable wheels make the robot turn only by stepping and the disturbances induce the foot command in (1) to account for the symmetric step sizes.

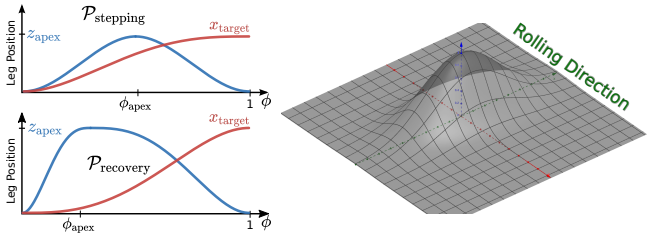

Fig. 3. This figure illustrates the relationship between leg utility u and xy-positional errors, depicted on the right. Given the foot's constraint to roll on the x-axis without y-axis slippage, y-errors, perpendicular to the roll direction, notably degrade the utility function compared to x-errors. On the left, the swing trajectories under $\mathcal{P}_{\text{stepping}}$ (top) and $\mathcal{P}_{\text{recovery}}$ (bottom) are compared. When recovering a slipped or stuck leg, the z swing happens earlier than the xy swings, with a higher ground clearance z_{apex} for obstacle evasion.

Inspired by the concept of leg utility proposed in [16], we developed the Driving Assistant (DA) approach. Leg utility is a quantitative metric that evaluates the effectiveness of a leg during its contact phase with the ground. It ranges from 1, indicating full functionality, to 0, suggesting the leg is incorrectly positioned and is ineffective. This utility metric decreases to near zero immediately after a leg encounters an obstacle. We define the utility function u as:

$$u(\boldsymbol{e}) = \exp\left(\frac{-\boldsymbol{e}_x^2}{2\sigma_x^2} + \frac{-\boldsymbol{e}_y^2}{2\sigma_y^2}\right) \tag{15}$$

where $\boldsymbol{e}$ is the positional error along and perpendicular to the rolling direction, and σ_x^2 and σ_y^2 are variances that determine the impact of the error $\boldsymbol{e}$ on the utility. The graph on the right in Fig. 3 represents the utility function[1].

The $\mathcal{P}$:=$\begin{pmatrix} z_{\text{apex}} & \phi_{\text{apex}} & c_{\text{bezier}} & k_s & d_{\text{gait}} \end{pmatrix}$ vector determines the properties of gait and leg swings. Here, z_{apex} is the peak height of the swing along the z axis at the ϕ_{apex} phase, c_{bezier} defines the curvature of the quartic Bézier swing trajectories, k_s is the symmetry gain in (1), and d_{gait} adjusts the nominal period to $d_{\text{gait}}T_{\text{nominal}}$ and the ratio of contact to swing phase duration to $1/d_{\text{gait}}$. If $d_{\text{gait}} < 1$, the gait is slower, and the leg's stance phase is extended, improving control by reducing underactuated periods.

Setting all $\mathcal{P}$ elements to zero effectively ceases gait and leg swings, inducing pure driving locomotion. This occurs as z_{apex} of 0 prevents vertical leg movement, ϕ_{apex} of 0 gives no time for swing curves, k_s of 0 stops symmetry adjustments, and d_{gait} of 0 leads to full stance duration, inhibiting leg swings. In essence, setting $\mathcal{P}$ to zero immobilizes the legs, making the robot rely solely on driving for locomotion.

The parameter set $\mathcal{P}_{\text{stepping}}$ is designed for stable stepping locomotion resilient to perturbations, while $\mathcal{P}_{\text{recovery}}$ is dedicated to recovering a foot that has slipped or become stuck. The comparative swing trajectories of these parameter sets are displayed in the left-hand graphs of Fig. 3.

The offset between the takeoff and the targeted foot position during the swing phase is given by:

$$\boldsymbol{p}^i_{\text{offset}} := \boldsymbol{p}^i_{\text{symmetry}} - \boldsymbol{p}^i_{\text{takeoff}}, \tag{16}$$

[1]Live graph at: `https://www.geogebra.org/3d/y9hqumya`

where $\boldsymbol{p}^i_{\text{symmetry}}$ is the commanded foot position in (1), and $\boldsymbol{p}^i_{\text{takeoff}}$ is the measured position of the foot at the beginning of the swing. This offset is influenced by three factors: error $\boldsymbol{e}_{\text{pos}}$ between the desired and the estimated body position, which expands when external disturbances occur; gait error $\boldsymbol{e}_{\text{gait}}{=}t_c\boldsymbol{v}^{\text{user}}_g$ arising from the user's gait speed command over leg contact duration t_c; and leg error $\boldsymbol{e}^i_{\text{leg}}$, which accounts for foot misplacement due to rugged terrain or obstacles. Hence, the leg error can be expressed as:

$$\boldsymbol{e}^i_{\text{leg}} = \left|\boldsymbol{p}^i_{\text{offset}}\right| - \left|\boldsymbol{e}_{\text{gait}}\right| - \left|\boldsymbol{e}_{\text{pos}}\right|. \tag{17}$$

By defining the weight vector $\boldsymbol{w}$ for each leg as:

$$\boldsymbol{w}^i = \left[1 - \min\{u\left(\boldsymbol{e}_{\text{gait}}\right), u\left(\boldsymbol{e}_{\text{pos}}\right)\} \quad 1 - u\left({\boldsymbol{e}^i_{\text{leg}}}^+\right)\right], \tag{18}$$

the minimized parameter set $\mathcal{P}$ for i-th leg is formulated as:

$$\mathcal{P}^i = \hat{\boldsymbol{w}}^i \left[\mathcal{P}_{\text{stepping}} \quad \mathcal{P}_{\text{recovery}}\right]^\top, \tag{19}$$

where $\hat{\boldsymbol{w}}^i$ is equal to $\boldsymbol{w}^i$ if $\left\|\boldsymbol{w}^i\right\|_1 < 1$, otherwise normalized.

The DA strives to minimize all elements of $\mathcal{P}$ toward zero for each leg in accordance with respective utility values. Under perfect driving conditions, all utility values in (18) approach 1, yield minimal $\boldsymbol{w}$ gains, thus suspending leg swing and gait execution. Encountering an obstacle precipitates a drop in leg utility (i.e. $u\left({\boldsymbol{e}^i_{\text{leg}}}^+\right)$), prompting the use of $\mathcal{P}_{\text{recovery}}$ to free the leg via extended swing duration and height and early z swing trajectory execution. When both $u\left(\boldsymbol{e}_{\text{gait}}\right)$ and $u\left(\boldsymbol{e}_{\text{pos}}\right)$ approach zero, the legs execute normal swings, primarily directed by $\mathcal{P}_{\text{stepping}}$, to fulfill user commands and counter disturbances.

Differing from [16], which discretely alternates leg modes between pure driving, static walking, and trotting to restore kinematically challenged legs, our method executes dynamic gaits perpetually, accommodating user commands and corrective swings simultaneously. This approach maintains kinematics near their nominal states, avoiding abrupt shifts between driving and leg swinging.

IV. Experimental Results

We tested our method on two quadruped robots: Unitree's AlienGo and the AIS[2]-developed wheeled Mini Cheetah [1]. The Mini Cheetah's wheeled design supports speeds up to 20 km/h, albeit with an added 0.39 kg per leg due to component replacement.

Our implementation builds on the open-source Mini Cheetah control framework, transferred to the ROS ecosystem, and integrated with the MuJoCo multi-body simulator [19]. This simulator accurately models contact dynamics, wheel rolling friction, joint, and rotor dynamics. Videos showcasing the actual and simulated robots' performance are publicly available[3].

[2]Autonomous Intelligent Systems, University of Bonn

[3]https://www.ais.uni-bonn.de/~hosseini/iros2023

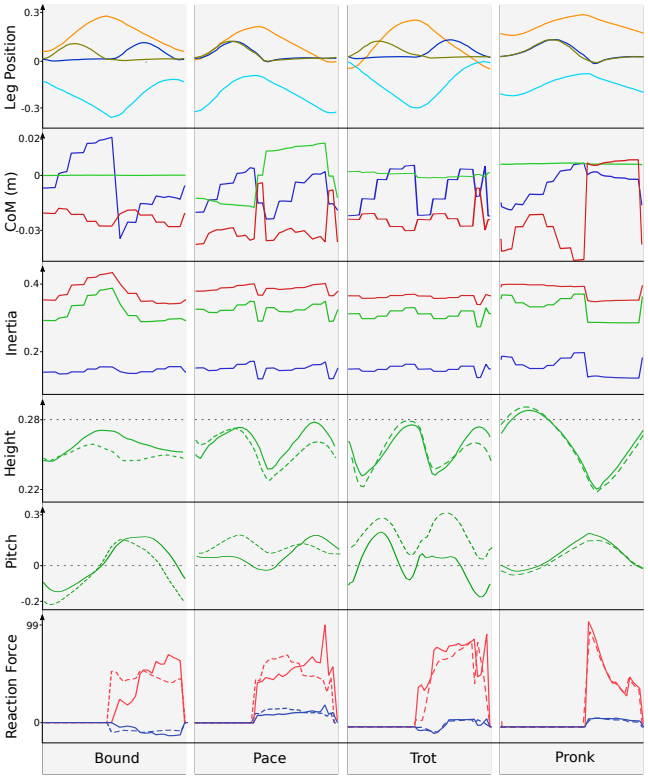

Fig. 4. This figure shows the time-varying rigid-body model of the robot (Section III-A.3) trotting, bounding, pronking, and pacing at a speed of 1.5 m/s. All graphs represent one gait cycle. First row: Foot trajectories with front and rear right foot positions on x-axis in orange and blue, respectively, and corresponding z-axis positions in olive and dark blue. Second row: Predicted CoM of the robot for x, y, and z axes depicted in blue, green, and red, respectively. Third row: Composite inertia trajectory for Ixx (blue), Iyy (green), and Izz (red). Fourth and fifth rows: Robot's height and pitch relative to the reference height of 0.28 m and pitch of 0 rad; Dotted green lines denote constant-model MPC, solid green ones represent our LTV-MPC. Last row: Demonstrates optimal reaction forces on the front-right leg, with z axis (red) and x axis (blue). The dotted and solid lines represent forces generated by the constant-model MPC and LTV-MPC, respectively.

A. Time-Varying Dynamic Controller

This paper assesses the use of a time-varying rigid body dynamics (RBD) model in the MPC. The gait's substantial influence on the robot's effective mass, inertia, and CoM is demonstrated in Fig. 4. In a comparative experiment between the constant-model MPC, which maintains a steady robot model throughout the prediction horizon, and our proposed LTV-MPC across various gaits, the LTV-MPC showed improved control, especially in tracking the robot's height and pitch, by generating an improved reaction force profile. The improved performance was notably observed in bounding and pacing gaits (Fig. 4), but less so in pronking due to the minimal changes in the robot's composite inertia and CoM. Calculating the composite inertia tensor and center of mass requires iterative inverse kinematics over future foot positions and whole-body kinematics for each prediction horizon segment. Despite boosting computational demands, the total cost remains marginal as composite inertia computation is considerably cheaper than solving the QP. Leveraging a time-

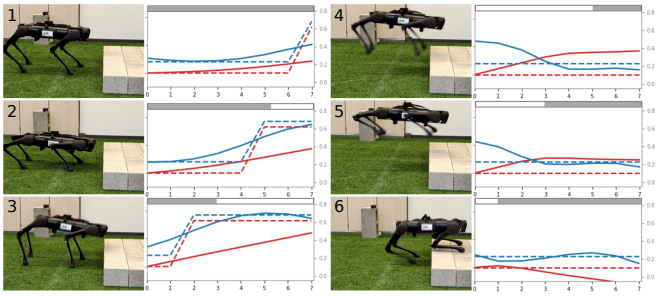

Fig. 5. The AlienGo robot successfully performs a forward jump, landing on a 14 cm high surface. The bar graph, consisting of $N = 8$ segments for the prediction horizon, showcases the scheduled contact state in gray. The reference trajectory is denoted by dotted lines, with the optimal LTV-MPC trajectory in solid lines; x-axis in red, z-axis in blue. Initially (graph 1), legs are assumed grounded throughout the prediction horizon. In graph 2, the previous optimal height trajectory is utilized to adjust the contact reference, yielding a refined optimal trajectory in graph 3. Entering the flight phase (graph 4), the contact reference is adjusted for landing after flight duration t_{flight} is passed. Upon landing, the controller applies reaction forces to all legs for stabilization (graph 6).

varying RBD model and updating the WBC controller at 500 Hz, we still maintain an MPC update rate surpassing 100 Hz, triple that of the open-source Mini Cheetah implementation. This performance improvement stems from our software's hierarchical structure.

B. Dynamic Jumping

This paper examined dynamic jumping motions, encompassing forward and rotational commands, on the AlienGo quadruped. The robot proficiently performed jumps featuring a height of 53 cm, forward displacement of 60 cm, turning motion of 0.8 rad, and a ground clearance of 40 cm. The forward jumping motion and predicted trajectories are displayed in Fig. 5

In the simulation, the Mini Cheetah robot achieved a 35 cm ground clearance during driving-jumping, contrasted with the 8 cm clearance of the physical hardware due to electronic limitations that impeded simultaneous full-torque application on multiple joints, leading to voltage drops and movement failure. Fig. 6 presents simulated driving-jumping snapshots and height graphs for both the robot and the front-right foot.

Both robots executed dynamic jumps from various foot positions, with the hybrid robot uniquely able to convert stepping velocity into driving velocity during jumps. This conversion allowed the legs to stay within kinematic constraints while following the stepping direction and executing the jump. The video supplement showcases these jumps in real-world scenarios.

C. Driving Assistant

The driving assistant is vital in guiding the robot along the wheel rolling direction, promoting a natural driving style, reducing energy usage, and minimizing leg swings by optimizing gait parameters $\mathcal{P}$. It seamlessly adjusts gait characteristics in response to user commands or external disturbances. For instance, when forces or torques affect

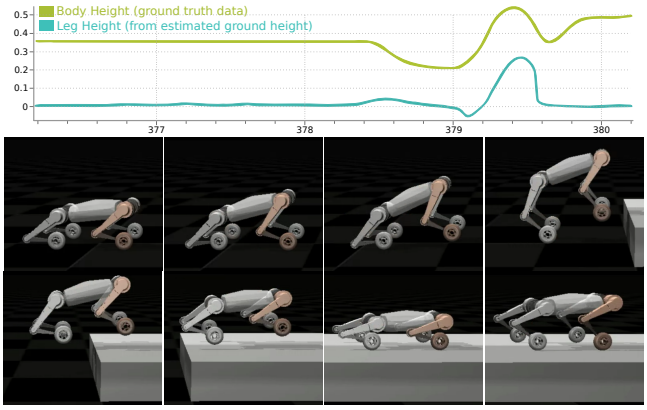

Fig. 6. Driving-jumping motion. The top graph represents the front right foot's estimated height (cyan) and the robot's ground truth body height (yellow). Snapshots are from time 378.6 (top-left) to 379.8 (bottom-right). At time 379.5, leg height resets to zero as the state estimator recognizes the landing plane as new ground.

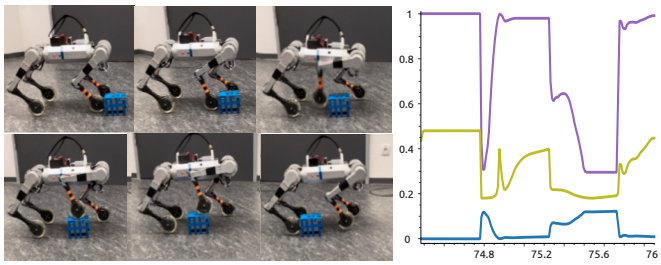

Fig. 7. This figure shows the robot's forward driving motion, guided by the driving assistant, where all feet stay grounded until the front-right wheel hits a large obstacle (top left). The resulting leg's position discrepancy lowers its utility value (shown in purple), triggering a swing with height z_{apex} (in blue) and swing peak phase ϕ_{apex} (in olive), while the other legs stay grounded. The hindered wheel eventually surmounts the obstacle, completing its swing and landing.

wheel-driven motion, leg utilities decrease, triggering greater swing height and duration to correct the robot's trajectory.

This method enhances the robot's agility under perturbations, enabling path following and recovery from stuck feet. As shown in Fig. 7, encountering a large obstacle reduces leg utility (according to (15)), causing an increase in swing height and duration to overcome the obstruction.

V. Conclusion

In conclusion, our novel control framework for quadruped robots enables agile and efficient locomotion and dynamic jumping over rough terrain using a time-varying RBD for the MPC. The controller can be applied to both hybrid and non-hybrid quadruped robots. The driving assistant allows the robot to perform driving and stepping seamlessly while responding to disturbances and user gait commands. Our approach represents a significant step forward in quadrupedal locomotion by demonstrating the ability to jump while driving with wheels for the first time.

In the future, we plan to integrate perceived terrain information into the control framework to determine target foot positions. This will allow us to generate optimized state

trajectories that can follow those positions while maintaining balance.

REFERENCES

[1] M. Hosseini, D. Rodriguez, and S. Behnke, "State estimation for hybrid locomotion of driving-stepping quadrupeds," *IEEE International Conference on Robotic Computing (IRC)*, 2022.

[2] D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 11 2016.

[3] D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 09 2017.

[4] D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, 01 2018.

[5] H.-W. Park, P. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," *Robotics: Science and Systems*, 07 2015.

[6] J. Carlo, P. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control," *IEEE Robotics and Automation Letters (RA-L)*, 10 2018.

[7] D. Kim, J. Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv*, 09 2019.

[8] G. Bellegarda and K. Byl, "Trajectory optimization for a wheel-legged system for dynamic maneuvers that allow for wheel slip," *IEEE Conference on Decision and Control (CDC)*, 12 2019.

[9] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics (TOG)*, 07 2018.

[10] M. Bjelonic, D. Bellicoso, Y. Viragh, D. Sako, F. Tresoldi, F. Jenelten, and M. Hutter, "Keep rollin'-whole-body motion control and planning for wheeled quadrupedal robots," *IEEE Robotics and Automation Letters (RA-L)*, vol. PP, 01 2019.

[11] T. Klamt, D. Rodriguez, L. Baccelliere, X. Chen, D. Chiaradia, T. Cichon, M. Gabardi, P. Guria, K. Holmquist, M. Kamedula, H. Karaoguz, N. Kashiri, A. Laurenzi, C. Lenz, D. Leonardis, E. Mingo, L. Muratore, D. Pavlichenko, F. Porcini, and S. Behnke, "Flexible disaster response of tomorrow: Final presentation and evaluation of the CENTAURO system," *IEEE Robotics and Automation Magazine*, vol. PP, 10 2019.

[12] T. Klamt and S. Behnke, "Planning hybrid driving-stepping locomotion on multiple levels of abstraction," *International Conference on Robotics and Automation (ICRA)*, 2018.

[13] M. Bjelonic, D. Bellicoso, M. Hutter, Y. Viragh, and F. Jenelten, "Trajectory optimization for wheeled-legged quadrupedal robots using linearized ZMP constraints," *IEEE Robotics and Automation Letters (RA-L)*, vol. PP, 01 2019.

[14] M. Bjelonic, P. K. Sankar, C. D. Bellicoso, H. Vallery, and M. Hutter, "Rolling in the deep – hybrid locomotion for wheeled-legged robots using online trajectory optimization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, 2020.

[15] J. Rebula, P. Neuhaus, B. Bonnlander, M. Johnson, and J. Pratt, "A controller for the LittleDog quadruped walking on rough terrain," *IEEE International Conference on Robotics and Automation (ICRA)*, 05 2007.

[16] 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," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7 2021.

[17] V. Klemm, A. Morra, C. Salzmann, F. Tschopp, K. Bodie, L. Gulich, N. Kung, D. Mannhart, C. Pfister, M. Vierneisel, F. Weber, R. Deuber, and R. Siegwart, "Ascento: A two-wheeled jumping robot," in *2019 International Conference on Robotics and Automation (ICRA)*, 05 2019.

[18] M. H. Raibert, H. B. Brown, and M. Chepponis, "Experiments in balance with a 3D one-legged hopping machine," *International Journal of Robotics Research (IJRR)*, vol. 3, 1984.

[19] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.