2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Versatile Real-Time Motion Synthesis via Kino-Dynamic MPC with Hybrid-Systems DDP

He Li[1,*], Tingnan Zhang[2], Wenhao Yu[2], Patrick M. Wensing[1]

***Abstract*— Specialized motions such as jumping are often achieved on quadruped robots by solving a trajectory optimization problem once and executing the trajectory using a tracking controller. This approach is in parallel with Model Predictive Control (MPC) strategies that commonly control regular gaits via online re-planning. In this work, we present a nonlinear MPC (NMPC) technique that unlocks on-the-fly re-planning of specialized motion skills and regular locomotion within a unified framework. The NMPC reasons about a hybrid kinodynamic model, and is solved using a variant of a constrained Differential Dynamic Programming (DDP) solver. The proposed NMPC enables the robot to perform a variety of agile skills like jumping, bounding, and trotting, and the rapid transition between them. We evaluated the proposed algorithm with three challenging motion sequences that combine multiple agile skills, on two quadruped platforms, Unitree A1, and MIT Mini Cheetah, showing its effectiveness and generality.**

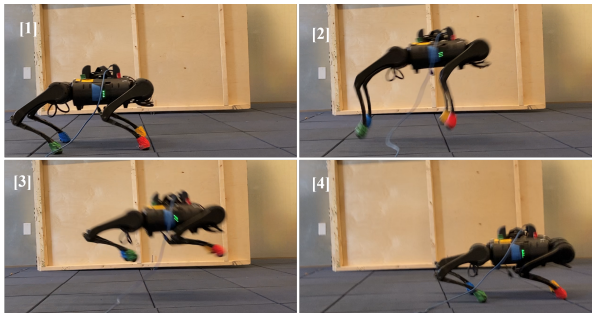


Fig. 1: Result highlight: Leaping forward with the proposed MPC controller on the Unitree A1. The robot can jump up to 0.5 m far (1.4x body lengths) and 0.5 m high (1.9x standing height) from rest. The results section includes more agile motions and diverse gaits, such as continuous jumping and rapid switching between various gaits.

I. Introduction

Quadruped animals show incredible mobility. They exhibit a large variety of locomotion skills including regular walking, trotting, bounding, and more dynamic jumping maneuvers. Transitions between different locomotion skills are quickly devised and smoothly carried out by animals. Achieving the same level of mobility on their robot counterparts has long fascinated robotics researchers but remains a challenge. Existing techniques often employ separate implementations of controllers for regular gaits and specialized motions such as dynamic jumping. In this paper, we present a unified predictive controller for aperiodic locomotion that is capable of performing specialized skills such as jumping and mixed-gait transitions. One of the many motions generated is shown in Fig. 1.

Model Predictive Control (MPC) is a powerful tool for controlling quadruped locomotion. Successful applications of MPC have demonstrated robust gaits in many prior works [1]–[8]. It is known that a trade-off often needs to be made in MPC between computation speed and model complexity. While simplified models are preferred for fast re-planning, more complex models are often needed for versatile and agile motions. Many existing quadruped MPC controllers are based on a Single Rigid Body (SRB) model that ignores the leg motions. Depending on the computational resources that are available on hardware, the SRB can be further simplified in convex MPC formulations [2], [5]. However, convex formulations truncate too many details of the robot, and are thus not suitable for devising agile motions online. The SRB model could be equipped with joint trajectories, resulting in a kinodynamic model [8], [9]. Though the MPC formulated with this model is nonlinear, excellent optimal-structure-exploiting solvers [10], and DDP-based solvers [11] have unlocked its potential for real-time performance, showing adaptability on stepping stones [8], [9]. However, the use of this model for specialized dynamic motions such as jumping remains to be investigated. Whole-body MPC [12]–[15] and contact-implicit MPC [16] have shown promise to unlock complex behaviors, but these controllers are under early development, and are in general slow for agile motions. A recent excellent work [17] shows the promise of Whole-body MPC for agile in-place jumping, but rapid transitions among diverse gaits remain a challenge.

Due to the computational bottleneck, dynamic jumping controllers are often implemented separately from the locomotion controller [18]–[20]. TO is commonly used to synthesize jumping motions [18], [19], [21]. In [18], TO is solved once to find the ground reaction force before the jump, and a Jacobian-transpose relationship is employed to convert the force to joint torques, whereas in [19], [20], a more advanced variational-based controller [22] is employed to produce the torque actuation.

The main contribution of this paper is the use of a hybrid kinodynamic model in an NMPC controller that achieves fast online motion synthesis of both specialized dynamic motions and diverse gaits. This MPC controller, together with a gait library, a leg controller and others, compose a framework that unlocks a large variety of locomotion skills, such as agile jumping and mixed-gaits locomotion, in addition to single-gait motions that are more common in the MPC literature. The performance of the HKD-MPC is

*This research was done while He Li interned with Robotics at Google.
[1] University of Notre Dame, Notre Dame, IN, USA
[2] Robotics at Google, Mountain View, CA, USA

evaluated on two quadruped hardware platforms, Unitree A1 and MIT Mini Cheetah [23], where only minimal changes concerning the inertial parameters and kinematics are made, showing the robustness and transferability of the HKD-MPC. We solve the MPC problem with a constrained variant of a DDP solver, i.e., Hybrid-Systems DDP (HS-DDP) [14]. We show that by using the feedback gain of DDP to warm start the re-planning, and a re-initialization scheme of the dual variable, only a few DDP iterations are needed for the HKD-MPC. We open-source our C++ implementation of the HS-DDP solver for consideration by other groups. A portion of this pipeline was briefly discussed in our previous work [24], where the primary focus was to retarget animal motions that were dynamically feasible. By comparison, this paper demonstrates the first implementation of our HS-DDP solver for use in hardware and demonstrates its ability to control multi-gait and heterogeneous behaviors in real time, all without relying on a dynamically feasible reference trajectory.

II. Hybrid Kinodynamic Model

The HKD model [24] used here is motivated by a previous kinodynamic model in [9] that extends the SRB model with consideration of the leg kinematics. The HKD model differs from [9] by switching the leg kinematic variables based on the leg contact status. If a leg is in stance, the foot location is considered to be fixed, and the leg movement is ignored. If a leg is in swing, the joint angles are controlled with commanded joint velocities, and the foot positions are ignored. A visual illustration of this model is shown in Fig. 2, with its detailed kinematics and dynamics equations below

$$\dot{\boldsymbol{\theta}} = \mathbf{T}(\boldsymbol{\theta})\boldsymbol{\omega} \tag{1a}$$

$$\dot{\mathbf{p}} = \mathbf{v} \tag{1b}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}(-\boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} + \mathbf{R}_B^\top \sum_{j=1}^{4} s_j(\mathbf{p}_{f_j} - \mathbf{p}) \times \boldsymbol{\lambda}_{f_j}) \tag{1c}$$

$$\dot{\mathbf{v}} = \mathbf{g} + \frac{1}{m}\sum_{j=1}^{4} s_j \boldsymbol{\lambda}_{f_j} \tag{1d}$$

$$\dot{\mathbf{p}}_{f_j} = 0 \quad \text{if } j \text{ in stance} \tag{1e}$$

$$\dot{\mathbf{q}}_j = \mathbf{u}_{J_j} \quad \text{if } j \text{ in swing} \tag{1f}$$

where $\boldsymbol{\theta}$ denotes the Euler angles, $\mathbf{p}$ the Center of Mass (CoM) position of the body, $\boldsymbol{\omega}$ and $\mathbf{v}$ respectively are angular and linear velocities of the body, $\mathbf{R}_B$ is the rotation matrix from the world frame to the body frame, $\mathbf{I}$ is the (fixed) rotational inertia relative to the body frame, m is the body mass, $\mathbf{g}$ is the gravity vector in world frame, $\mathbf{p}_f$ and $\boldsymbol{\lambda}_f$ are the foothold location and ground reaction force (GRF) respectively, $s \in \{0, 1\}$ is a binary variable indicating contact status, $\mathbf{q}$ is the joint angle, $\mathbf{u}_J$ is the commanded joint velocity, and $j \in \{1, 2, 3, 4\}$ denotes the leg index. The matrix $\mathbf{T}$ transforms the angular velocity to the rate of change of the Euler angles. The variables $\boldsymbol{\omega}$ and $\mathbf{I}$ are expressed in the body frame, with $\mathbf{p}$, $\mathbf{v}$, $\mathbf{p}_f$, and $\boldsymbol{\lambda}_f$ in the world frame.

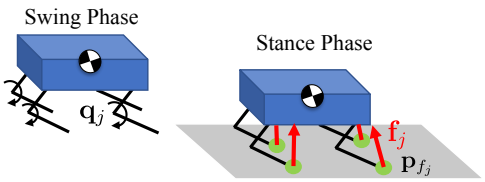


Fig. 2: Illustration of the contact-dependent HKD model. The swing (stance) phase here shows that all four feet are in swing (stance) for clear presentation, whereas in general, the HKD model works for any contact status of the legs individually.

The Eqs. (1a)-(1d) represent the SRB dynamics. The Eqs. (1e)-(1f) constrain the foothold locations and the joint angles at the kinematics level, and are complementary to each other. The change of variable, i.e., reset map, between $\mathbf{q}_j$ and $\mathbf{p}_{f_j}$ takes place at touchdown and takeoff. At touchdown, the foothold location is reset by the joint angle via forward kinematics, i.e.,

$$\mathbf{p}_{f_j}^{+} = \mathrm{FW}_j(\mathbf{q}_j^{-}) \qquad \text{swing} \rightarrow \text{stance}\,, \tag{2}$$

where the superscripts '-' and '+' indicate immediately before and after touchdown, respectively. Note that once the foothold location is reset at touchdown, it remains fixed as enforced by the constraint (1e) until the stance phase is over. At takeoff, the joint angle is reset to a constant value

$$\mathbf{q}_j^{+} = \mathbf{q}_{\text{default}} \quad \text{stance} \rightarrow \text{swing}. \tag{3}$$

This reset map instantaneously changes the joint angles, thus making the swing trajectory infeasible. However, what we really care about is the foothold location at the subsequent stance phase. A Bézier polynomial interpolation is used between foothold locations for swing leg control. Thus, as long as there is sufficient control authority (eq. (1f)) that leads to a good next foothold location, then the immediate change of joint angles will not cause any problem. An alternative approach is to reset the joint angle using the inverse kinematics (IK). The foothold location at take-off, however, may not be kinematically reachable in the middle of optimization. Future work will investigate the IK approach, which results in a feasible swing trajectory, fully taking advantage of joint space optimization and avoiding the need for foot position interpolation.

III. Control Architecture

The HKD-MPC interfaces with other components such as a gait library, reference generator, and leg controller to fully function on the robot hardware. This section overviews the control architecture, illustrated in Fig. 3.

A. Gait Library/Reference Generation

We use a leg-independent phase variable as in [3] to describe periodic gaits, such as trotting and bounding, while using a time-based description for aperiodic gaits such as jumping. Given a random set of gaits, for instance, bounding, trotting, and then jumping, a gait composer is employed to first compute the scheduled contact timings for each leg,

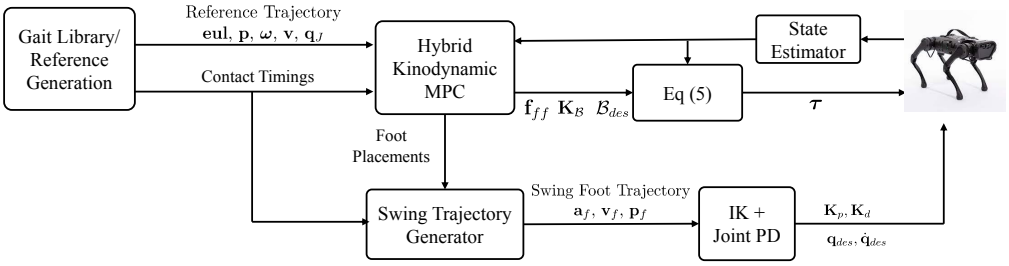


Fig. 3: Overview of the control architecture. The gait library/reference generation provide contact timings and a reference for HKD-MPC. The HKD-MPC optimizes foot placements, GRFs, and feedback gains. The foot placements are interpolated with a cubic polynomial, which is tracked by an IK + joint PD scheme. The MPC runs at 100 Hz, while the state estimator and leg controller run at 500 Hz. We used the state estimator similar to [3] in this work.

and then compose the overall scheduled contact timings that cover all gaits.

The reference generation employs simple heuristics. For each gait cycle, the desired height, forward and lateral velocities are assumed constant, and the x, y positions are obtained by integrating the velocities. Desired angular velocities and Euler angles are assumed zero, though can be other values in practice. An example reference trajectory for the run-jump-run motion tested in this work is shown in Fig. 4. When a leg is in swing, the reference joint angles are assumed constant and take the values at a static standing pose. During stance, the reference for the GRF is as follows

$$\boldsymbol{\lambda}_{ref_j} = m_{total}\mathbf{g} \,/\, \Big(\sum_{i=1}^{4} s_i\Big) \tag{4}$$

where m_{total} is the total mass of the robot, $\mathbf{g}$ the gravity in world frame, and s_i the binary variable indicating the contact status for leg i. The desired relative foot position (used later in (10)) is obtained using Raibert heuristics, similar to [2].

B. Leg Controller

The HKD-MPC produces optimal trajectories for the GRF $\boldsymbol{\lambda}^*$, state $\mathbf{x}^*$, and feedback gains $\mathbf{K}^*$, thanks to the DDP solver as discussed in section V-A. For leg j, the stance controller is implemented as follows

$$\boldsymbol{\tau}_j = \mathbf{J}_j^\top \Big(\boldsymbol{\lambda}_j^* + \mathbf{K}_{\mathcal{B},j}^*(\mathbf{x}_\mathcal{B}^* - \mathbf{x}_\mathcal{B})\Big) \tag{5}$$

where $\mathbf{x}_\mathcal{B} = [\boldsymbol{\theta}, \mathbf{p}, \boldsymbol{\omega}, \mathbf{v}]$ denotes the body state, $\mathbf{J}$ is the leg Jacobian in the hip frame, and $\mathbf{K}_{\mathcal{B},j}^*$ is the sub-matrix in $\mathbf{K}^*$ (from DDP) that maps the body state to the GRF of leg j.

The HKD-MPC also generates optimal foot placements. The swing leg trajectory is obtained by interpolating the current foothold position and next foot placement using a cubic Béizer polynomial. The swing trajectory is then converted to joint space using Inverse Kinematics, and a joint-PD controller is employed for tracking.

IV. Model Predictive Control with HKD Model

A. Problem Formulation

The HKD model is a hybrid system because of the contact-dependent variables and system equations (1e) and (1f). In this research, we assume that the contact status values s_i are pre-scheduled, and can be obtained from a high-level gait library. Denote $\mathbf{x}$ the state variable such that

$$\mathbf{x} = \big[\boldsymbol{\theta}, \mathbf{p}, \boldsymbol{\omega}, \mathbf{v}, y_1, \cdots, y_4\big], \tag{6}$$

where y is a flexible variable that represents $\mathbf{p}_{f_j}$ if leg j in stance and $\mathbf{q}_j$ if leg j in swing. Denote $\mathbf{u}$ the control variable

$$\mathbf{u} = \big[\boldsymbol{\lambda}_{f_1}, \cdots, \boldsymbol{\lambda}_{f_4}, \mathbf{u}_{J_1}, \cdots, \mathbf{u}_{J_4}\big]. \tag{7}$$

The HKD-MPC repeatedly solves a multi-phase TO problem as follows:

$$\min_{\mathbf{u}(\cdot)} \sum_{i=1}^{n} \left[\int_{t_{i-1}^+}^{t_i^-} \ell_i\big(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t)\big)\mathrm{d}t + \Phi_i\big(\mathbf{x}^{[i]}(t_i^-)\big)\right] \tag{8a}$$

$$\text{subject to} \quad \text{HKD model} \quad (1) \tag{8b}$$

$$\text{reset maps} \quad (2), (3) \tag{8c}$$

$$\text{touchdown constraints}, \tag{8d}$$

$$\text{GRF constraints} \tag{8e}$$

where i denotes the phase index where a phase is determined whenever the contact status of any leg changes, ℓ_i and Φ_i denote the running and terminal cost at phase i. The GRF constraints enforce nonnegative normal GRFs, and use a linear approximation of the friction cone to prevent slipping.

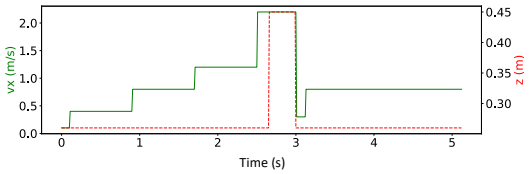


Fig. 4: Reference height z and forward velocity v_x for run-jump-run.

B. Touchdown Constraints

Since the timing in the optimization (8) is fixed, we need to make sure that the foot positions are on the ground at the time that touchdown is scheduled. To do so, an equality constraint is used at the end of a swing phase:

$$\begin{bmatrix}0 & 0 & 1\end{bmatrix} \mathrm{FW}_j(\mathbf{q}_j^-) = 0 \tag{9}$$

C. Cost Function

We use a quadratic cost function that consists of a tracking penalty and foot regularization, similar to our previous work

[24]. For each phase i, the running cost function is

$$l = \int_{t_{i-1}^{+}}^{t_i^{-}} \left\| [\delta\boldsymbol{\theta}^\top \ \delta\mathbf{p}^\top \ \delta\boldsymbol{\omega}^\top \delta\mathbf{v}^\top] \right\|_{\mathbf{Q}_b}^2 + \left\| \bar{\mathbf{S}}\delta\mathbf{q}_J \right\|_{\mathbf{Q}_J}^2 + \|\mathbf{S}\delta\mathbf{p}_f\|_{\mathbf{Q}_f}^2 + \|\mathbf{S}\delta\boldsymbol{\lambda}\|_{\mathbf{R}_\lambda}^2 \, \mathrm{d}t \quad (10)$$

where $\delta\cdot$ represents the deviation, and $\mathbf{Q}_b$, $\mathbf{Q}_J$, $\mathbf{Q}_f$, and $\mathbf{R}_\lambda$ are positive definite weighting matrices. The matrix $\mathbf{S}$ is a diagonal matrix concatenating the contact status of each foot whereas $\bar{\mathbf{S}}$ concatenates the swing status. The terminal cost is defined similarly but without the last term.

V. Online Implementation of HKD-MPC With Hybrid System DDP

In this section, we detail the online implementation of the HKD-MPC with a customized variant of a DDP solver, Hybrid-Systems DDP (HS-DDP) [14]. We first overview the HS-DDP solver, and introduce two schemes for maintaining its fast convergence during replanning in an MPC fashion.

A. HS-DDP

DDP [12], [25] exhibits linear complexity with respect to the length of the planning horizon. The HS-DDP was presented in [14] to advance DDP for hybrid systems where the switching sequence is known. The state-based switching of a hybrid system is triggered when the system state reaches a guard set, which often results in a discontinuous jump in the state space. For instance, the impact of a leg can suddenly reset certain velocities to zero. As another instance, the touchdown reset map (2) may involve a change of variables. The HS-DDP properly handles this state-based switching in two steps: First, terminal constraints are enforced at the end of a phase to guarantee that the guard set is reached at switching. Second, a discontinuous reset map is modeled in the backward sweep of DDP to propagate sensitivities of the value function. The terminal constraints are dealt with via an Augmented Lagrangian (AL) method, which incorporates a linear Lagrange multiplier term and a quadratic penalty term. Previous work [14] implemented HS-DDP in MATLAB as a proof of concept. However, significant efforts have been made for a highly efficient template-based C++ implementation. We open source the code [1] in this paper for consideration by other groups.

B. Schemes of Maintaining Fast Convergence

1) Feedback Gains: At convergence (or early termination with sub-optimality), DDP-type algorithms produce a control law as follows:

$$\mathbf{u} = \mathbf{u}^* + \mathbf{K}^*(\mathbf{x}^* - \mathbf{x}), \quad (11)$$

where $\mathbf{u}^*$ and $\mathbf{K}^*$ are the feedforward control and the feedback gain matrix resulting from the last backward sweep of DDP, and $\mathbf{x}^*$ is obtained by running a forward roll-out. While NLP-solver-based MPC often only applies $\mathbf{u}^*$ to the system in an open loop fashion between each solve, the DDP-based MPC additionally applies $\mathbf{K}^*$ to the system, resulting in feedback between solves. The incorporation of this feedback can handle additional policy lag and provides more robustness [9]. Further, during re-planning, most single-shooting NLP solvers use $\mathbf{u}^*$ from the previous planning to warm start the current optimization. When large disturbances occur, however, naively applying $\mathbf{u}^*$ may result in divergence of the first forward roll-out in the current optimization. We find out that additionally using $\mathbf{K}^*$ to warm start the current optimization could robustify the algorithm and encourage fast convergence.

2) Re-initilization of HS-DDP: Denote z as a decision variable, $f(z)$ a cost function, and $g(z) = 0$ an equality constraint on z. The AL method converts the constrained optimization problem to an unconstrained version as follows:

$$\min_z \quad f(z) + \lambda g(z) + g^2(z)\sigma/2 \quad (12)$$

where λ is the Lagrange multiplier and σ is a penalty coefficient. The multiplier λ gives an estimate of the optimal Lagrange multiplier λ^*. A proper estimate of λ^* matches the constraint sensitivity, leading to a reduction in the constraint violation. With this motivation, we solve the very first optimization to (local) optimality, and during re-planning, reuse the linear multiplier from the previous planning to initialize the current optimization. The penalty coefficient is set to its initial values rather than inheriting from the previous solution, preventing ill-conditioning. Since the re-planning happens fast, the solutions between two consecutive MPC solves are supposed to be close. In practice, we find this strategy enables us to run just three DDP iterations during re-planning for all the hardware experiments.

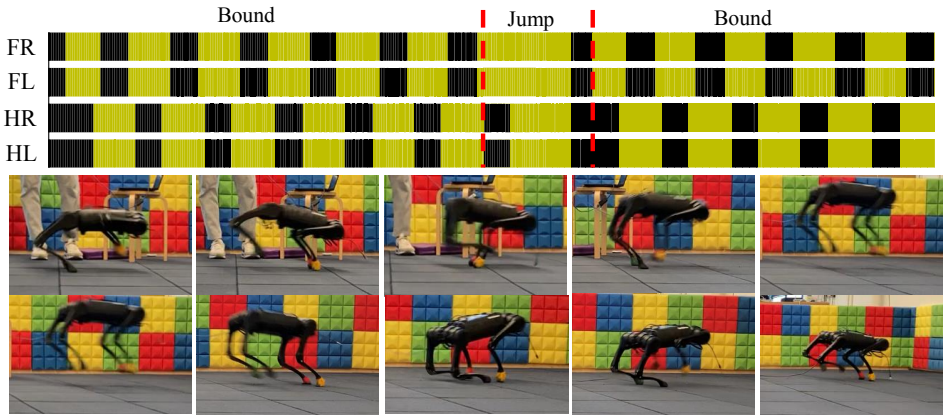
VI. Results

The proposed controller is implemented on two quadruped platforms, Unitree A1, and the MIT Mini Cheetah [23], to perform a variety of agile locomotion skills, beyond regular gaits (e.g., trotting and bounding). A summary of the hardware experiment results is shown in Figures .5 and 6 and in the supplemental video. For all motions tested, the HKD-MPC controller shares an identical configuration (i.e., cost function, constraints, etc) on each quadruped platform. Switching between the platforms only requires changing the robot mass, inertia matrix, and kinematic parameters. The planning horizon used for all motions is 0.5 s, and the integration time step is 0.01 s, resulting in 50 time steps in the optimization. We run the HS-DDP for a maximum of 100 iterations for the first planning, and a maximum of three iterations during re-planning. The MPC controller runs at 100 Hz on a Laptop with an 8-core, 2.5GHz, 16-GB RAM CPU.

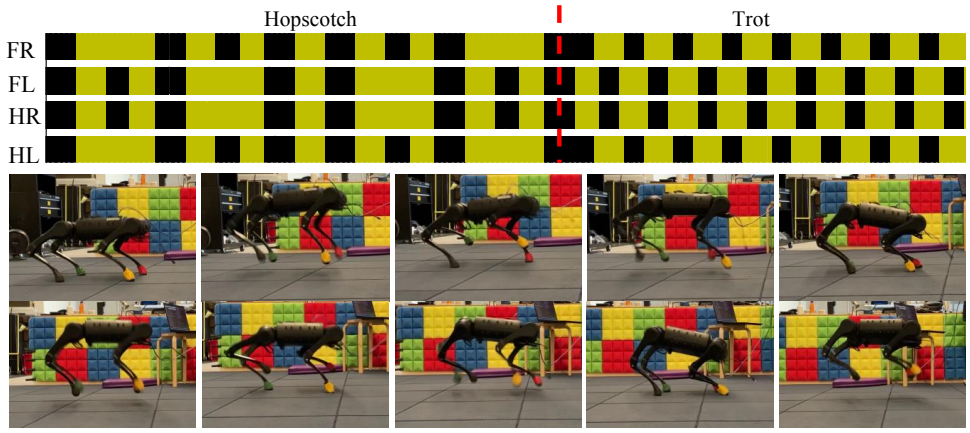
A. Hardware Results

We show three examples of the proposed controller for robustly achieving versatile and agile quadruped locomotion skills. In the first example, the robot is commanded to run, make a jump, and then immediately return to the running gait. In the second example, the robot is commanded to hop alternately using two legs and four legs. In the last example,

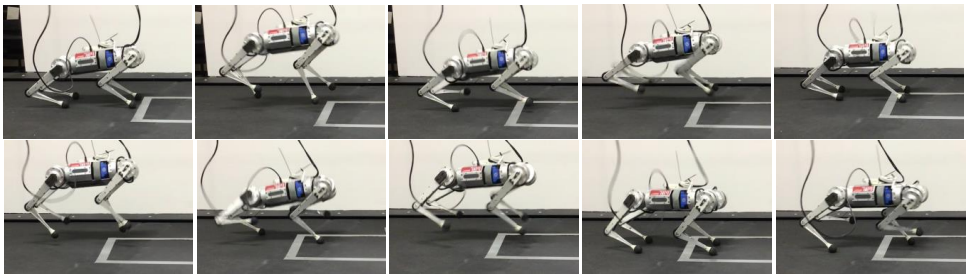
[1] https://github.com/heli-sudoo/HS-DDP-CPP.git



(a) Run, jump, and run, on Unitree A1



(b) Mixed-gaits hopping on Unitree A1



(c) Mixed-gaits hopping on MIT Mini Cheetah

Fig. 5: Time-series snapshots of experimental hardware tests for several irregular agile locomotion skills and the associated contact schedules.

the robot is commanded to continuously jump in place. The readers are encouraged to check the robust trotting and high-speed bounding examples in the accompanying video.

Figure. 5(a) shows the experimental result of the Unitree A1 robot executing the run-jump-run sequence. The robot jumps to about 0.5 m high and 1.0 m far, roughly 1.8x its regular standing height and 2.7x the body length. This type of motion is challenging for three reasons. First, in preparing for the jump, the robot needs to reason about the GRF such that (1) it is large enough to support the air phase (2) it does not create a moment that flips the robot over (3) it avoids slipping. Second, during jumping, the robot needs to reason about proper GRFs and foothold positions for a safe landing. Last, the robot is able to recover the running gait after landing. Previous solutions [18] employed three separate controllers to address the three challenges. The result (Fig. 5) shows that our HKD-MPC controller can cohesively achieve all three tasks in a single MPC controller. The reference trajectory for creating this motion is shown in Fig. 4, which simply sets the desired jumping height and speed, indicating that the proposed HKD-MPC controller is very powerful and robust so that simple heuristic references can work. While these references were human-authored herein, we envision them being provided by a high-level motion planner in the future.

Figure. 5(b) shows the result of Unitree A1 executing the mixed-gait hopping motion, with the associated contact schedule. The robot starts from a standing pose, and hops

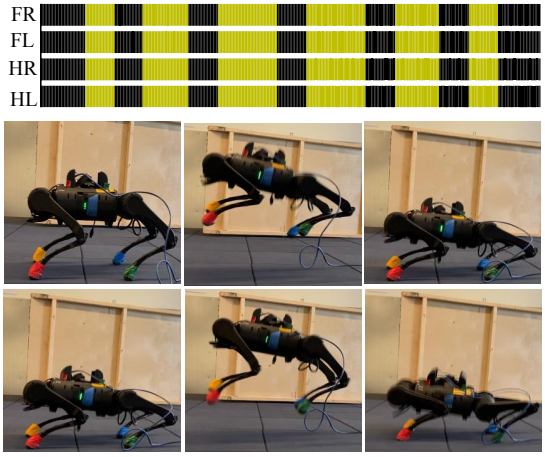


Fig. 6: Time-series snapshots of Unitree A1 performing continuous jumping and the associated contact schedule. Two jumps are shown.

alternately over diagonal legs and four legs, followed by a trotting gait. During hopping, the robot spends 0.3 s on average in each phase. The result demonstrates that the proposed HKD-MPC controller is very reliable so that it enables the robot to seamlessly transition among gaits. We also tested this same motion on the MIT Mini Cheetah, by just changing the mass, inertia matrix, and kinematics of the HKD model. The result is shown in Fig. 5(c), demonstrating the high reliability and easy transferability of the controller.

Figure 6 shows the results of continuous jumping executed on the Unitree A1 robot. The robot starts with a flight phase of 0.2 s, which gradually grows to 0.4 s, at which the robot achieves the maximum height of about 0.48 m. The result further demonstrates the performance of the proposed controller for robustly controlling a variety of agile aperiodic locomotion behaviors.

We benchmark the performance of the HS-DPP solver in simulation for all motions tested in this section. Table. I summarizes the statistics of solve times along the MPC iterations for each motion. The fast solve time of our HS-DDP solver enables us to run the HKD-MPC at 100 Hz.

B. *Effects of DDP Feedback Gains*

In Section V-B.1, we discussed that the feedback matrix in Eq. (11) is helpful to robustify the optimization and encourage fast convergence. In this section, we examine this effect by seeding with/without the feedback gain in the first forward roll-out of DDP during re-planning. We benchmark the performance using the motion of leaping forward as shown in Fig. 1. Figure. 7 depicts the angular velocity of

TABLE I: Solve time statistics (in ms) for different tasks.

Task	mean (ms)	std (ms)	max (ms)	min (ms)
Run-jump-run	5.64	2.3	14.12	3.27
Mixed gaits	5.8	1.27	10.0	2.8
Continuous jump	5.45	1.38	14.7	2.6

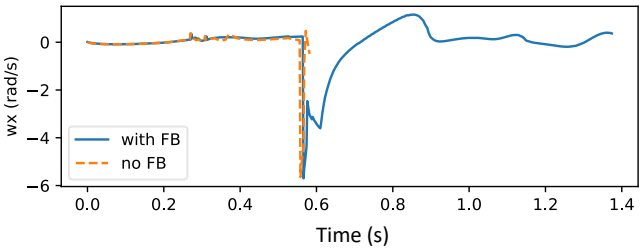


Fig. 7: Comparison of the angular velocity around the x axis with and without enabling the feedback gain in (11) during re-planning.

the body around x that was obtained with and without using the DDP feedback gain. When the feedback gain is not enabled, the state evolution becomes unstable after the robot touchdown. A closer examination reveals that there is a jump in the velocity and angular velocity caused by touchdown, for instance, a 5 rad/s change in w_x. Simply applying the open-loop control of Eq. (11) results in the divergence of the forward roll-out, thus de-stabilizing the optimization.

VII. Conclusions and Future Works

In this paper, we presented a control framework that achieves online motion synthesis of a variety of quadruped locomotion skills. At the core of this framework is the HKD-MPC, which unlocks on-the-fly motion generation for not only regular locomotion gaits, but also heterogeneous behaviors. The proposed framework was tested on the Unitree A1 robot hardware for several challenging motions, i.e., run-jump-run motion, mixed-gaits hopping, continuous jumping, etc. The HKD-MPC is shown to be very robust in the sense that dynamic and complex behaviors are synthesized even with sparse references.

These are several limitations of the current work. First, the initial joint angles of a swing leg are reset to a constant in the optimization stage. The resulting swing trajectory is not kinematically feasible, and thus is not directly used for swing leg control. Future work will use inverse kinematics to reset the swing leg, and a reachability constraint will be used to ensure kinematic feasibility. Further, inequality constraints can be added for collision avoidance. Second, the contact mismatch is not handled in the current framework. For continuous jumping, the body of the robot can sometimes touch the ground due to early contact, where the ground reaction forces are not available. The robot eventually falls down as the mismatch error accumulates. Third, the current solver is based on a single-shooting method, and is thus subject to divergence in the case of large perturbations. Future work will consider a multiple-shooting solver to improve robustness. Last, the sparse reference trajectory is heuristic, and requires some tuning efforts, especially for the running-to-jump gait, where the contact timings, the jumping height, and velocity need to be coordinated. Future work will take a more strategic approach by building a library of short trajectories offline, and synthesizing them online.

VIII. Acknowledgements

The Mini Cheetah is sponsored by the MIT Biomimetic Robotics Lab and NAVER LABS.

REFERENCES

[1] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 1398–1404.

[2] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control," in *Int. Conf. on Intelligent Robots and Sys.*, 2018, pp. 1–9.

[3] 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 *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 2245–2252.

[4] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 2436–2442.

[5] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1154–1171, 2021.

[6] P.-A. Léziart, T. Flayols, F. Grimminger, N. Mansard, and P. Souères, "Implementation of a reactive walking controller for the new open-hardware quadruped solo-12," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5007–5013.

[7] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8352–8358.

[8] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model predictive control," *arXiv preprint arXiv:2208.08373*, 2022.

[9] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 4730–4737.

[10] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[11] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 93–100.

[12] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 4906–4913.

[13] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[14] H. Li and P. M. Wensing, "Hybrid systems differential dynamic programming for whole-body motion planning of legged robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5448 – 5455, 2020.

[15] N. J. Kong, C. Li, and A. M. Johnson, "Hybrid ilqr model predictive control for contact implicit stabilization on legged robots," *arXiv e-prints*, pp. arXiv–2207, 2022.

[16] S. L. Cleac'h, T. Howell, M. Schwager, and Z. Manchester, "Fast contact-implicit model-predictive control," 2021.

[17] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Sola, N. Mansard, and S. Vijayakumar, "A feasibility-driven approach to control-limited ddp."

[18] H.-W. Park, P. M. Wensing, and S. Kim, "Jumping over obstacles with mit cheetah 2," *Robotics and Autonomous Systems*, vol. 136, p. 103703, 2021.

[19] M. Chignoli and S. Kim, "Online trajectory optimization for dynamic aerial motions of a quadruped robot," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7693–7699.

[20] M. Chignoli, S. Morozov, and S. Kim, "Rapid and reliable quadruped motion planning with omnidirectional jumping," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6621–6627.

[21] Z. Zhou, B. Wingo, N. Boyd, S. Hutchinson, and Y. Zhao, "Momentum-aware trajectory optimization and control for agile quadrupedal locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7755–7762, 2022.

[22] M. Chignoli and P. M. Wensing, "Variational-based optimal control of underactuated balancing for dynamic quadrupeds," *IEEE Access*, vol. 8, pp. 49 785–49 797, 2020.

[23] B. Katz, J. Di Carlo, and S. Kim, "Mini Cheetah: A platform for pushing the limits of dynamic quadruped control," in *IEEE Int. Conf. on Robotics and Automation*, 2019, pp. 6295–6301.

[24] H. Li, W. Yu, T. Zhang, and P. M. Wensing, "Zero-shot retargeting of learned quadruped locomotion policies using hybrid kinodynamic model predictive control," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 971–11 977.

[25] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.