Impact-Invariant Control: Maximizing Control Authority During Impacts

William Yang and Michael Posa

***Abstract*—When legged robots impact their environment, they undergo large changes in their velocities in a short amount of time. Measuring and applying feedback to these velocities is challenging, further complicated by uncertainty in the impact model and impact timing. This work proposes a general framework for adapting feedback control during impact by projecting the control objectives to a subspace that is invariant to the impact event. The resultant controller is robust to uncertainties in the impact event while maintaining maximum control authority over the impact-invariant subspace. We demonstrate the improved performance of the projection over other commonly used heuristics on a walking controller for a planar five-link-biped. The projection is also applied to jumping, box jumping on to a platform 0.4 m tall, and running controllers for the compliant 3D bipedal robot, Cassie. The modification is easily applied to these various controllers and is a critical component to deploying on the physical robot. The supplementary video is available at this link.**

I. Introduction

Handling the making and breaking of contact lies at the core of controllers for legged robots. Recent advances in the modeling and planning of these contacts have enabled legged robots to walk reliably in select environments [1]. This progress directs the focus of the field toward developing legged robots capable of increasingly agile motions. However, these agile motions often require interacting with the environment with non-negligible impact events, something our current controllers are incredibly sensitive to. When a robot's foot makes contact with the world, the foot is brought quickly to a stop by a large contact impulse. Large contact forces and rapidly changing velocities hinders accurate state estimation. Coupled with the relatively poor predictive performance of our contact models [2] [3] [4], this combination of large state uncertainty and poor models makes control especially difficult.

Roboticists have attempted to improve the robustness of legged locomotion to these impact events by addressing the reference trajectories as well as the controllers that track those trajectories. For example, the open-loop swing-leg retraction policy has been shown to have inherent stability to varying terrain heights [5]. Qualitatively similar motions were also found independently through robust trajectory optimization [6] [7] [8]. While designing more robust trajectories shows promise, the challenge of designing controllers to track these often discontinuous trajectories still remains.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1845298

The authors are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA {yangwill, posa}@seas.upenn.edu

Fig. 1: Cassie is able to execute agile motions with non-negligible impacts like jumping (top), box jumping (bottom-left), and running (bottom-right) using impact-invariant control.

Tracking a discontinuous trajectory is problematic due to the unavoidable difference between the reference trajectory and actual system caused by even minuscule differences in impact timing. These differences cause feedback control efforts to spike and optimization constraints to be violated [9], leading to instabilities.

While these controller spikes can be reduced through strategies such as reducing controller gains and contact constraints around the impact event [10] [11], these heuristic methods do not address the fundamental challenge of tracking discontinuous trajectories. A strategy that does attempt to directly address this challenge is termed reference spreading control [12] [13]. This method leverages contact detection and extending the reference trajectories to ensure that a valid reference trajectory exists despite mismatches in impact timing. An extension to reference spreading control [14] eliminates the time-dependency through vector fields. However, all of these methods rely on turning off all velocity feedback while the impact is still resolving.

Another impact-aware control formulation is [9], which incorporates anticipated impacts into a robust control for-

mulation. When evaluating velocity constraints near impacts, the authors consider both the pre- and post-impact velocities. By doing so, they are able to avoid infeasibilities due to mismatches in impact timing. The primary distinction is that [9] assumes impacts resolve instantaneously, but impacts on real systems can take tens of milliseconds to resolve [15]. Our proposed method makes no assumptions on the magnitude or duration of impact forces, which allows our method to handle when the impact is still resolving.

Alternative solutions focus instead on avoiding impact events altogether. While impacts do not exist for frequently used templates such as the linear inverted pendulum (LIP) and the spring-loaded inverted pendulum (SLIP), impacts will manifest when embedding these templates onto physical robots with non-negligible mass in the legs. Furthermore, it is neither possible nor desirable to avoid impacts for more agile motions such as running or jumping. Thus, handling non-trivial impacts in a robust manner is essential to the development of more agile legged robots.

In this work, we propose a method for tracking discontinuous trajectories across impacts that directly avoids jumps in tracking error. We achieve this by projecting the tracking objectives down to a subspace where they are invariant to the impact event. Gong and Grizzle [16] shared an important insight about angular momentum about the contact point, noting that it is invariant to impacts at that contact point. Inspired by this, we generalize this property and extend it to include the entire invariant subspace, which we term the impact-invariant subspace. A preliminary version of this article was presented at the International Conference on Intelligent Robots and System 2021 [17]. This work makes the following contributions, where extensions from the preliminary version are noted:

- The primary contribution of this paper is the identification and understanding of a subspace of velocities that are invariant to contact impulses.
- Leveraging this subspace, we propose a general method to adapt controller feedback to only regulate that subspace, which improves robustness to impact uncertainty.
- We demonstrate our impact-invariant controller across a range of walking, running and jumping motions, greatly expanded from the previous conference version. Included in these examples, shown on the physical bipedal Cassie robot, are a 40 cm jump up onto a box and bipedal running. To the best of our knowledge, this is the first example of a model-based running controller for Cassie.

II. Background

We briefly introduce preliminary material and notation from both multibody dynamics and optimization-based control.

A. Rigid Body Dynamics

We use both the planar biped Rabbit [18] and the 3D compliant bipedal robot Cassie. Both legged robots are modeled using conventional floating-base Lagrangian rigid-body dynamics. Cassie has physical springs on its heel and knee joints; for the purposes of modeling and control we treat these springs as rigid. However, we do model the springs when evaluating our results in simulation.

The robot's state $x \in \mathbb{R}^{2n_q} = [q; \dot{q}]$, described by its positions $q \in \mathbb{R}^{n_q}$ and velocities $\dot{q} \in \mathbb{R}^{n_q}$, is expressed in generalized floating-base coordinates[1]. The dynamics are derived using the Euler-Langrange equation and expressed in the form of the general manipulator equation:

$$M(q)\ddot{q} + C(q, \dot{q}) = g(q) + Bu + J_\lambda(q)^T \lambda, \tag{1}$$

where $M \in \mathbb{R}^{n_q \times n_q}$ is the mass matrix, $C \in \mathbb{R}^{n_q}$ and $g \in \mathbb{R}^{n_q}$ are the Coriolis and gravitational forces respectively, $B \in \mathbb{R}^{n_q \times n_u}$ is the actuator matrix, $u \in \mathbb{R}^{n_u}$ is the vector of actuator inputs, and $J_\lambda \in \mathbb{R}^{n_q \times n_c}$ and $\lambda \in \mathbb{R}^{n_c}$ are the Jacobian of the active contact and holonomic constraints and the corresponding constraint forces respectively.

B. Rigid Body Impacts

In this paper, we model the complex deformations and surface forces that occur when a legged robot makes contact with a surface using a rigid-body contact model. This contact model does not allow deformations; instead, impacts are resolved instantaneously. Therefore, the configuration remains constant over the impact event and the velocities change instantaneously according to the contact impulse Λ:

$$M(\dot{q}^+ - \dot{q}^-) = J_\lambda^T \Lambda, \tag{2}$$

where $\dot{q}^+$ and $\dot{q}^-$ are the pre- and post-impact velocities, J_λ is the Jacobian for the active constraints of the new contact mode, and Λ is the impulse sustained over the impact event. If we include the standard constraint that the new stance foot does not move once in contact with the ground (purely inelastic collision and no-slip condition):

$$J_\lambda \dot{q}^+ = 0, \tag{3}$$

Λ can be solved for explicitly, determining the post-impact state x^+ purely as a function of the pre-impact state x^-:

$$q^+ = q^-, \tag{4}$$

$$\dot{q}^+ = (I - M^{-1} J_\lambda^T (J_\lambda M^{-1} J_\lambda^T)^{-1} J_\lambda) \dot{q}^-. \tag{5}$$

This reset map is conventionally enforced as a constraint between hybrid modes separated by an impact event for trajectory optimization of legged robots.

C. Operational Space Control

We use an operational space controller (OSC) to track and stabilize the reference trajectories for the various motions explored in this paper. An OSC is a model-based inverse dynamics controller that tracks a general set of task or output space accelerations by solving for dynamically consistent inputs, ground reaction forces, and generalized accelerations [19] [20]. For an output position $y(q) = \phi(q)$ and corresponding output velocity $\dot{y} = J_y(q)\dot{q}$, where $J_y(q) = \frac{\partial \phi}{\partial q}$, the

[1]For notational simplicity, we assume $\dot{q} = v$, where v are the velocities. For 3D orientation, as is the case for Cassie, this requires a straight-forward extension to use quaternions.

commanded output accelerations $\ddot{y}_{cmd}$ are calculated from the feedforward reference accelerations $\ddot{y}_{des}$ with PD feedback:

$$\ddot{y}_{cmd} = \ddot{y}_{des} + K_p(y_{des} - y) + K_d(\dot{y}_{des} - \dot{y}) \tag{6}$$

The objective of the OSC is then to produce dynamically feasible output accelerations $\ddot{y}$ given by:

$$\ddot{y} = \dot{J}_y \dot{q} + J_y \ddot{q},$$

such that the instantaneous output accelerations of the robot are as close to the commanded output accelerations as possible. This controller objective can be nicely formulated as a quadratic program (QP):

$$\min_{u,\lambda,\ddot{q}} \quad \sum_{i}^{N} \tilde{\ddot{y}}^T W_i \tilde{\ddot{y}} + \|u\|_W^2 + \|\ddot{q}\|_W^2 + \|\lambda\|_W^2 \tag{7}$$

$$\text{s.t.} \quad M\ddot{q} + C = g + Bu + J_\lambda^T \lambda \tag{8}$$

$$J_\lambda \ddot{q} + \dot{J}_\lambda \dot{q} = 0 \tag{9}$$

$$\mu\lambda_z \geq |\lambda_x| \tag{10}$$

$$\mu\lambda_z \geq |\lambda_y| \tag{11}$$

$$\lambda_n \geq 0, \tag{12}$$

where i denotes the particular output being tracked (e.g., center of mass or foot position) and W_i are corresponding weights on the tracking objectives. The QP seeks to minimize the acceleration tracking error, $\tilde{\ddot{y}} = \ddot{y}_i - \ddot{y}_{i_{cmd}}$, for the weighted sum across all tracking objectives. Additional regularization costs can be added to avoid non-unique solutions if the problem is underspecified. Eq. (8) is the dynamics constraint, where J_λ is the Jacobian for the active constraints for the current mode, and λ are the corresponding constraint forces. Eq. (9) enforces the holonomic constraints, and a linear approximation of the friction cone constraint is given by Eq. (10), Eq. (11), and Eq. (12). Here, λ_z is the normal component of the contact force and λ_x, λ_y are the tangential components expressed in the robot frame. The QP can be solved quickly (>1000 Hz), even for complex robots such as Cassie.

III. Impact Invariance

In this section, we present the key ideas of this work. In Section III-A, we explain the common challenges that arise when applying feedback during impacts. We then propose a solution in Section III-B, which also introduces the concept of the impact-invariant subspace. Finally, in Section III-C and Section III-D, we explain the practical details of how to implement the subspace in the context of an operational space controller.

A. Challenges of Control During Impacts

To motivate the concept of the impact-invariant subspace, we begin by highlighting and describing the difficulties of applying feedback control during an impact event. For the sake of simplicity, we consider a feedback controller with constant feedback gains that controls an output $y : \mathbb{R}^{n_q} \to \mathbb{R}^{n_y}$ to track a time-varying trajectory $y_{des}(t) : [0, \infty) \to \mathbb{R}^{n_y}$ by driving the tracking error $\tilde{y}(t) = y_{des}(t) - y(t)$ to zero. This is commonly accomplished with control law $u = u_{ff} + u_{fb}$

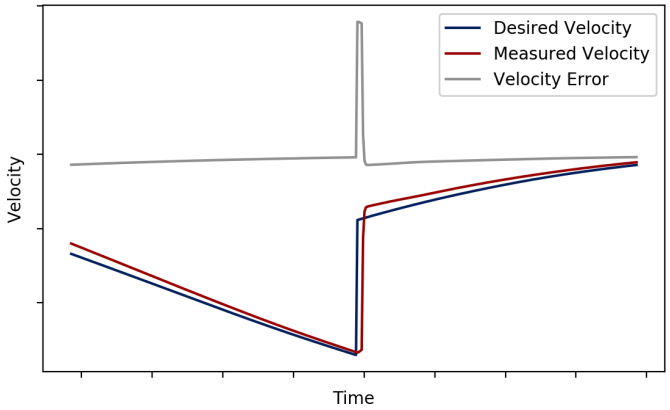


Fig. 2: Illustration of a system that undergoes an impact event. The desired velocity plan correctly includes the discontinuity as predicted by rigid body impact laws and the measured velocity is being properly regulated to match the desired plan. However, due to the mismatch in impact time, the velocity error inevitably spikes during the impact event.

where u_{ff} is the feedforward controller effort required to follow the reference acceleration $\ddot{y}_{des}$ and u_{fb} is the PD feedback component given by:

$$u_{fb}(t) = K_p \tilde{y}(t) + K_d \dot{\tilde{y}}(t). \tag{13}$$

The reference trajectory $\dot{y}_{des}(t)$ for systems that make contact with their environment has discontinuities at the impact events in order to be dynamically consistent with Eq. (5). Therefore, in a short time window around an impact event, there will be a discontinuity in the reference trajectory $\dot{y}_{des}(t)$ at the nominal impact time and a near-discontinuity in the actual robot state when the system $\dot{y}(t)$ makes contact with the ground as shown in Fig. 2. Because the robot configuration is approximately constant over the impact event, the change in controller effort is governed by the change in velocity error:

$$\Delta u \approx K_d \Delta \dot{y}, \tag{14}$$

thus any mismatches in impact timing will unavoidably result in spikes in the feedback error signal, which has been similarly noted in [12].

Remark 1: The previous example assumes the jump in the reference trajectory is time-based. Although it is possible to formulate trajectories with event-triggered jumps, these methods require detection, which for state-of-the-art methods still have delays of 4-5 ms [21]. Moreover, in reality, impacts are not resolved instantaneously but rather over several milliseconds to tens of milliseconds. In this time span, it is not clear which reference trajectory to use, as using either trajectory will output a large tracking error.

Note that a large tracking error, shown in Fig. 2, results from only a small difference in impact timing, yet the controller will respond to the large velocity error and introduce controller-induced disturbances. Furthermore, this sensitivity to the impact event is amplified by the large contact forces that impair state estimation and result in likely inaccurate velocity measurements due to necessary filtering.

The key challenges of applying feedback during impacts can thus be summarized as: impacts are brief moments of

high uncertainty where our references are poorly defined and our measurements are inaccurate.

B. Impact-Invariant Subspace

The key insight in resolving the problem of control during impacts is inspired by [16], in which Gong and Grizzle delineate desirable properties of angular momentum about the contact point. They highlight that it is invariant over impacts on flat ground, meaning that it is continuous over the impact event despite it being a function of velocity.

The concept of an impact-invariant subspace is a generalization of this property. We observe that there is a space of velocities that, like angular momentum about the contact point, are continuous through impacts for *any* contact impulse. By switching to track only these outputs in a small time window around anticipated impacts, we avoid controller-induced disturbances from uncertainty in the impact event. While angular momentum $\in \mathbb{R}^3$ or $\mathbb{R}^2$ for planar systems such as Rabbit, the impact-invariant subspace $\in \mathbb{R}^{n_q - n_c}$, where n_q is the dimension of generalized velocities and n_c is the dimension of the contact impulse of the impact event. For Rabbit walking, this space is $\in \mathbb{R}^{7-2}$. For Cassie, each foot on the ground imposes a contact constraint of dimension 5 and the four bar linkage on each leg provides a distance constraint of dimension 1 that is always active[2]. Therefore the impact-invariant subspace is $\in \mathbb{R}^{18-7}$ for impacts with a single foot (walking, running) and $\in \mathbb{R}^{18-12}$ for impacts with both feet (jumping). A direct benefit of this higher dimensional space is the higher degree of possible control, which enables more agile or energetically efficient motions.

The impact-invariant subspace is defined as the nullspace of $(M^{-1}J_\lambda^T)^T$ or left nullspace [22] of $M^{-1}J_\lambda^T$, where J_λ again is the Jacobian for the active constraints. Thus a basis $P(q) \in \mathbb{R}^{(n_q - n_c)\times n_q}$ for this nullspace is such that:

$$PM^{-1}J_c^T\Lambda = 0 = P(\dot{q} - \dot{q}^-). \tag{15}$$

This creates the intended effect, that is, for any contact impulse Λ, the impact-invariant velocities will be unchanged. To project the generalized velocities down to the impact-invariant subspace, we can create an orthonormal projection matrix $Q(q) \in \mathbb{R}^{n_q \times n_q} = P^T P$. In practice, we can compute this projection matrix as $Q = I - M^{-1}J_\lambda^T(J_\lambda M^{-T}M^{-1}J_\lambda^T)J_\lambda M^{-T}$.

To illustrate the benefit on a *physical robot*, the joint velocities for Cassie executing a jumping motion right when it lands are shown in Fig. 3. Details of the jumping controller and experiments are given in Section VI. Observe that the projected joint velocities are significantly smoother than the original joint velocities.

C. Implementation for Task Space Tracking

The objective of impact-invariant control is to apply control on the subspace where the changes in the velocity introduced via the impact map, $M^{-1}J_\lambda^T$, do not appear. In practice, we accomplish this by modifying our controllers to eliminate the component of the velocity error that is within the subspace spanned by the impact map $M^{-1}J_\lambda^T$ by solving the following optimization problem:

$$\min_{\lambda} \quad \left\|\dot{y}_{des} - J_y(\dot{q} + M^{-1}J_\lambda^T\lambda)\right\|_2 \tag{16}$$

$$\text{subject to:} \quad J_h(\dot{q} + M^{-1}J_\lambda^T\lambda) = 0, \tag{17}$$

where J_h is the Jacobian for the holonomic constraints that are unambiguously active during impact[3]. This reduces the total error by subtracting a correction term $M^{-1}J_\lambda\lambda$ that by construction is in the range of $M^{-1}J_\lambda^T$. We must include the constraint forces for all active constraints, as contact

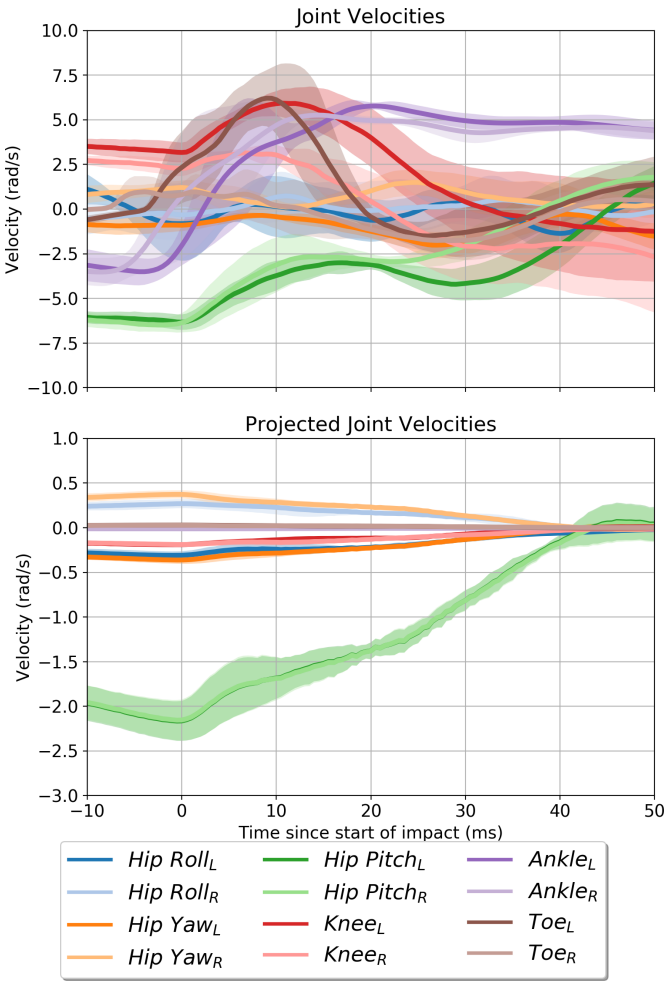


Fig. 3: Demonstration of the impact-invariant projection on joint velocity data from eight consecutive jumping experiments on the physical Cassie robot. Joint velocities (top) during the landing event change rapidly. By projecting the same joint velocities to the impact-invariant subspace (bottom), the values are more consistent and more amenable for feedback control. Note, the change in joint velocities primarily occurs within a time span of only 10 - 20 ms. The L and R subscripts indicate the left and right leg respectively.

[2]Explanation of Cassie contact constraint dimension. We model Cassie's feet as two point contacts on the same rigid body. Each point on the ground imposes a contact constraint of dimension 3; because the contact points are on the same body, one dimension is redundant.

[3]Unambiguously active constraints refer to constraints that are active both before and after the impact event. For example, the four-bar linkage constraint is always active and for walking with a double stance phase, the contact constraint for the current stance foot is active both before and after the other foot makes contact.

forces from the impact event will cause corresponding reaction forces, which should still satisfy kinematic feasibility of the velocities enforced by Eq. (17). Notice that this optimization problem is an equality constrained QP, which we are able to solve this in closed form:

$$\begin{bmatrix} \lambda^* \\ \mu^* \end{bmatrix} = \begin{bmatrix} A^T A & B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} A^T(\dot{y}_{des} - \dot{y}) \\ J_h \dot{q} \end{bmatrix}, \tag{18}$$

where $A = J_y M^{-1} J_\lambda^T$, $B = J_h M^{-1} J_\lambda^T$, and μ^* is the Lagrange multiplier for the holonomic constraint. Notice, this is a least squares problem which is a projection. The benefit of explicitly computing the error correction $M^{-1} J_\lambda^T \lambda^*$ is that it becomes trivial to smoothly blend in the projection. Applying λ^* back into the OSC formulation, we combine the correction and the measured velocity to define the projected generalized velocities as:

$$\dot{q}_{proj} = \dot{q} + M^{-1} J_\lambda^T \lambda^*. \tag{19}$$

We then define the projected output velocity, $\dot{y}_{proj}$, and projected output space velocity error, $\dot{\tilde{y}}_{proj}$, as:

$$\dot{y}_{proj} = J_y \dot{q}_{proj} \tag{20}$$

$$\dot{\tilde{y}}_{proj} = \dot{y}_{des} - \dot{y}_{proj}. \tag{21}$$

The projected output space velocity error $\dot{\tilde{y}}_{proj}$ is then used in the OSC feedback law (Eq. (6)) to create the impact-invariant feedback law:

$$\ddot{y}_{cmd} = \ddot{y}_{des} + K_p(y_{des} - y) + K_d(\dot{y}_{des} - \dot{y}_{proj}). \tag{22}$$

Remark 2: Here we formulate this projection for general task-space objectives. Joint-space objectives are included in task-space objectives, and thus this method can also easily be applied to joint PD controllers.

D. Activating the Impact-Invariant Projection

Because impact-invariant control essentially ignores errors in the space of impacts, we should only use it when we anticipate impacts. In practice, we do this by only activating the projection in a time window near anticipated impact events. To avoid introducing discontinuities when activating the projection, we blend in the correction, $M^{-1} J_\lambda^T \lambda^*$ using a sigmoid function $\alpha(t)$ visualized in Fig. 4.

$$\alpha(t) = \begin{cases} 0 & |t - t_{switch}| > 1.5T \\ \exp(\frac{(t - t_{switch} + T)}{\tau}) & t \leq t_{switch} \\ \exp(\frac{(t_{switch} - t + T)}{\tau}) & t > t_{switch} \end{cases} \tag{23}$$

where t is the current time, t_{switch} is the nominal impact time given by the reference trajectory, T is the duration of the projection window, and τ is the time constant. We then use $\alpha(t)$ to modify (19) to be:

$$\dot{q}_{proj} = \dot{q} + \alpha(t) M^{-1} J_\lambda^T \lambda^*.$$

Note our choice for the blending function $\alpha(t)$ is arbitrary; any monotonic continuous function with a range $\in [0, 1]$ can accomplish a similar purpose. Additionally, while $\alpha(t)$ is a function of time, we can also activate/blend in the impact-invariant projection purely as a function of the robot's state. For example, we can use the distance between the foot or end effector and the environment in place of $t - t_{switch}$.

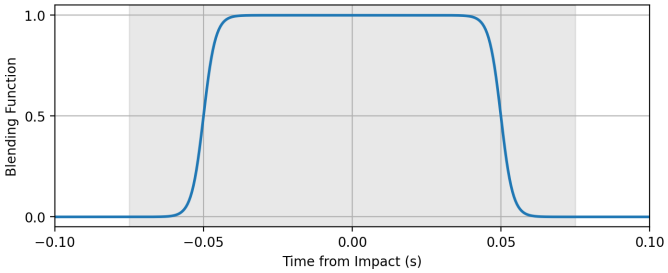


Fig. 4: Blending function for the impact-invariant projection for a window T of 50 ms and time constant τ of 2 ms.

IV. Experimental Setup

To showcase the advantages of using the impact-invariant subspace, we apply the aforementioned projection on the following examples:

- In simulation, for a joint-space inverse dynamics walking controller for the planar five-link biped Rabbit [18], we compare impact-invariant control to a default controller that makes no adjustment near impacts and to a controller that applies no-derivative feedback near impact.
- In simulation, we perform a basic jump as well as more dynamic jumps such as a long jump, box jump, and down jump using an operational space jumping controller for the 3D bipedal robot Cassie. On hardware, we validate the jump and box jump controllers with impact-invariant control.
- In simulation, we compare an operational space running controller for Cassie that uses impact-invariant control with a controller that applies no-derivative feedback near impact. We validate our running controller on hardware.

The reference walking and jumping trajectories were computed offline by solving a constrained trajectory optimization problem on the respective full order models. The problems were transcribed using DIRCON [23] and solved using IPOPT/SNOPT [24] [25]. The various jumping trajectories are distinguished by the constraints imposed:

- The normal jump trajectory was constrained to have the robot pelvis reach an apex height of 0.15 m above its initial starting height and to have both feet reach 15 cm of clearance above the ground at the apex.
- The long jump trajectory has the feet land 0.7 m ahead of their initial position.
- The box jump trajectory has the feet land on a box that is 0.5 m tall and 0.3 m in front of the starting configuration.
- The down jump trajectory has the feet land on a platform 0.5 m below and 0.3 m in front of the starting configuration.

Traces of the reference trajectories are shown in Fig. 6. The same jumping trajectories were used in both simulation and on the physical robot. The reference trajectories for the running controller were generated using common simple models explained in detail in VII-B. Hardware-specific implementation details of the jumping and running controller as well as the full set of are included in Appendix A.

V. Five-Link Biped Walking Controller

1) Experimental Setup: To demonstrate the directional nature of the impact-invariant projection, we apply the projection to a joint space inverse dynamics walking controller for the planar biped Rabbit in simulation with the hip and knee joint angles in both legs as the outputs. We generate a periodic walking trajectory using trajectory optimization and perturb the swing foot vertical velocity by 0.1 m/s at the start of the trajectory so that the robot makes contact away from the nominal impact time. To evaluate robustness, we compare the post-impact velocity error in the joints of both the swing and stance leg for three variations of the joint space controller:

- No adjustment: this is a standard joint space controller that makes no special considerations with regards to the impact event other than switching contact modes at the nominal impact time.
- No derivative feedback ($K_d = 0$) applied in a window 25 ms before and after the nominal impact time.
- Impact-invariant projection applied in a window 25 ms before and after the nominal impact time.

We add the additional comparison to a controller with no derivative feedback to demonstrate the control authority that the impact-invariant controller retains. While both applying no derivative feedback and using the projection seek to reduce the sensitivity to large velocity errors at the impact event, the projection maintains the maximum control feedback which leads to better tracking performance.

2) Results: Shown in Fig. 5, the controller using the impact-invariant projection is able to achieve the best tracking performance out of the three controllers for *both* legs. It has better tracking performance than the default controller for the joint velocities of the impacting leg by not overreacting to the impact event as shown by the controller efforts. At the same time, it has better tracking performance than the controller with no derivative feedback for the joint velocities of the non-impacting leg by appropriately regulating the velocities in those joints.

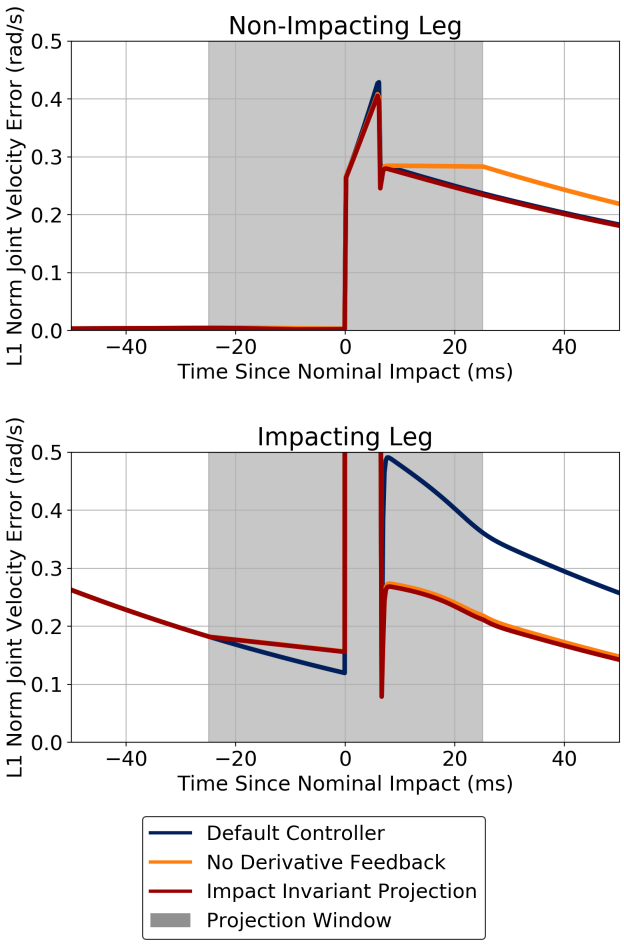


Fig. 5: The joint velocity tracking errors are shown for the impacting leg (top) and the non-impacting leg (bottom) for three strategies. The controller that utilizes the impact-invariant projection is shown to be robust to the mismatch in impact timing as evidenced by lower tracking error compared to the default controller. The impact-invariant controller is also able to maintain full control authority over the joints in the non-impacting leg compared to the controller that applies no derivative feedback in the same window. The time window where no-derivative feedback and the impact-invariant projection are active is shown in grey.

VI. Cassie Jumping Controller

Next, we evaluate the performance of the impact-invariant projection on a jumping controller for Cassie. We chose to look at jumping due to the richness of the impact event: the robot cannot accurately estimate its state [26] when it is in the air and must make impact with the ground with non-zero velocity.

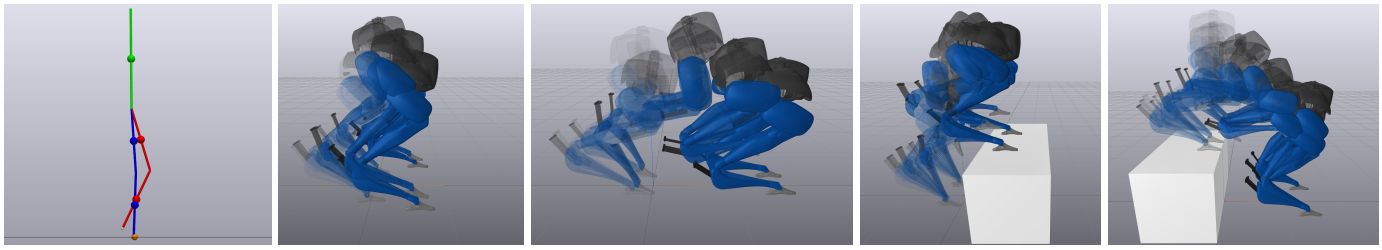
A. Controller Formulation

1) Finite State Machine: We decompose the jumping motion into three states CROUCH, FLIGHT, LAND, and switch between states at fixed times as computed from the reference trajectories.

2) Tracking Objectives: The tracking objectives are listed in Table I. We define tracking objectives such as the foot and pelvis position relative to other points on the robot to reduce sensitivity to errors in state estimation. During the CROUCH and LAND states, the active objectives are $[r, \psi, \beta_L, \beta_R] \in \mathbb{R}^8$. During the FLIGHT state, the active objectives are $[y_L, y_R, \beta_L, \beta_R, \phi_L, \phi_R] \in \mathbb{R}^{10}$. The active tracking objectives per mode are also illustrated in Fig. 7. Similar outputs are used in another jumping controller for Cassie [27].

TABLE I: Tracking objectives for the jumping controller. All values are expressed in the world frame.

Symbol	Description
$r \in \mathbb{R}^3$	Pelvis position
$\psi \in SO(3)$	Pelvis orientation
$y_L, y_R \in \mathbb{R}^3$	Foot position relative to pelvis
$\beta_L, \beta_R \in \mathbb{R}$	Hip yaw angle
$\phi_L, \phi_R \in \mathbb{R}$	Toe joint angle



(a) Walk (b) Jump (c) Long jump (d) Box jump (e) Down jump

Fig. 6: Reference trajectories generated using full model trajectory optimization.

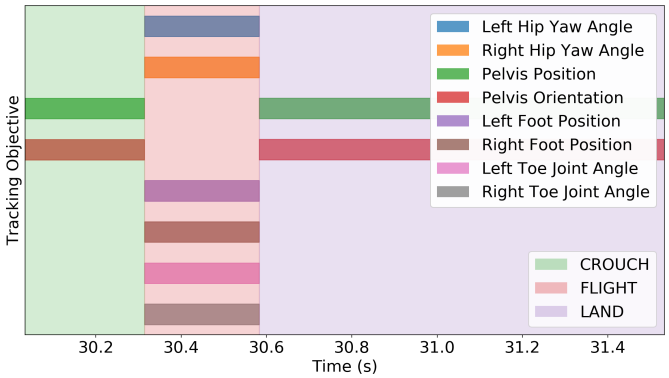


Fig. 7: Active tracking objectives per mode for the jumping controller executing the jump trajectory.

B. Simulation Analysis of Impact-Invariant Control

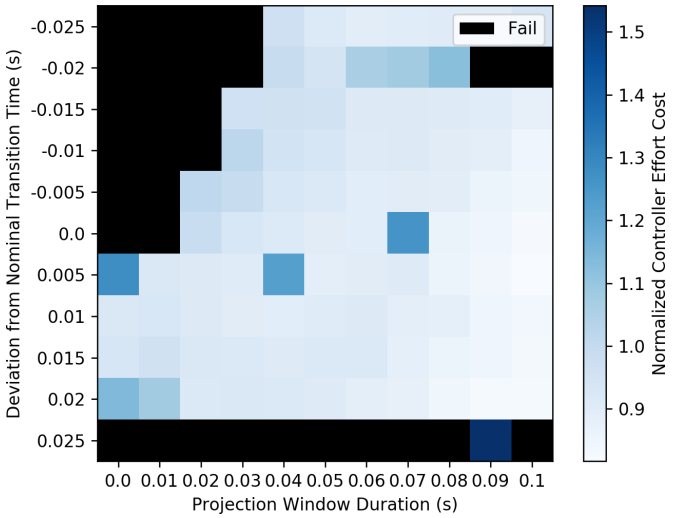
Timing the switch from the FLIGHT state to the LAND state is critical to stabilize the jump. Impact-invariant control reduces sensitivity to the impact timing and thus enables a greater margin of error. To evaluate this effect, for all the jumping motions we test a range of transition times $[-0.025s, 0.025s]$ and durations of the projection duration $[0.0s, 0.1s]$ to empirically evaluate the stable regions for the controller. In addition to testing whether the robot successfully controls the jump, we also evaluate the overall control input cost:

$$J_u = \int_{t_0}^{t_f} u^T W_u u dt, \tag{24}$$

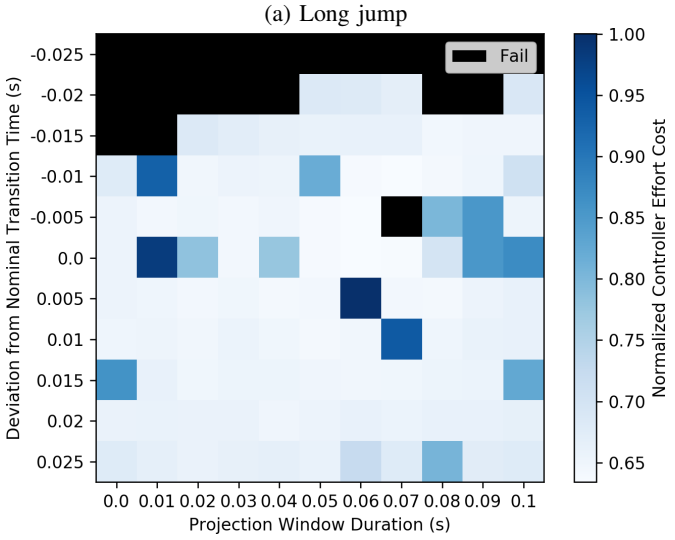
where W_u is the same regularization weight on u we use in the OSC. Five experiments per pair of transition times and projection durations are evaluated using the Drake simulator [28], and the results with J_{mot} normalized so that the largest value is 1 are reported in Fig. 8. The region of stable jumps increases as the projection window duration increases, while there is not a significant effect on the input cost. We see a more noticeable improvement from the impact-invariant controller in the long jump than the down jump. We theorize that this is because the long jump motion is more difficult to stabilize and therefore the improved robustness to the impact event has a greater marginal effect.

C. Hardware Results

Experiments using the controller with and without the impact-invariant projection were both consistently able to



(a) Long jump



(b) Down jump

Fig. 8: Simulation effort costs for (a) long jump and (b) down jump evaluated over a range of fixed transition times and projection window durations. Five trials are evaluated for each pair of transition times and projection durations. The controller without the impact-invariant projection corresponds to a projection window duration of 0.0 s. The regions where the robot fails to stabilize for over half the trials are marked as "Fail". As the projection window increases, the range of successful transition times increases. This effect is more pronounced for the long jump.

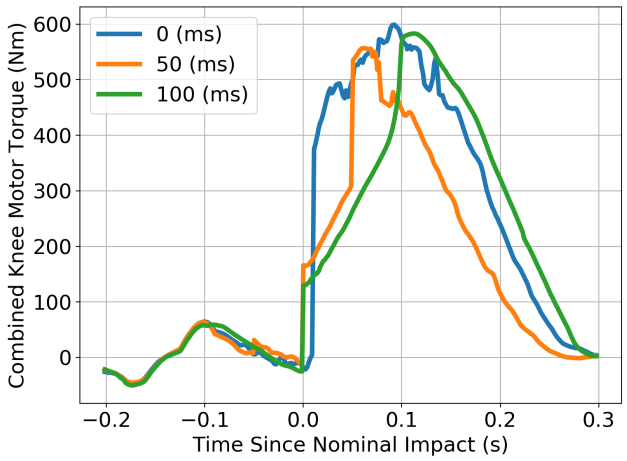


Fig. 9: Motor efforts on Cassie executing the default jumping motion. The combined knee motor torques commanded by the jumping controller are shown for three different durations of the impact-invariant projection.

successfully complete the basic jump. Snapshots of the jumping motion are shown in Fig. 1. As seen in Fig. 3, the joint velocities change rapidly during the impact event. By projecting the velocity error of the outputs (position and orientation of the pelvis) to the impact-invariant subspace, we avoid overreacting to these rapid velocity changes in a principled manner. The effects of this can be seen in Fig. 9, where the change in knee motor efforts, particularly at the impact event, are significantly reduced when using the impact-invariant projection. We choose to show the knee motor efforts because they exhibit the largest change at the impact event due to their role in absorbing the weight of the pelvis at impact. This smoothing out of the commanded motor efforts is similar to what we see in simulation. The jumping motions for the controller both with and without the impact-invariant projection are included in the supplementary video.

Among the other jumping trajectories, we chose to test the box jump trajectory on hardware as it is least likely to damage the robot. We positioned 16 in (∼0.4 m) tall wooden boxes in front of the robot and executed the same tracking controller using a 50 ms projection duration. Although the reference jumping trajectory we optimized was for a box height of 0.5 m, the robustness afforded by the impact-invariant controller enabled the controller to successfully achieve the jump. A frame of the controller executing the box jump is shown in Fig. 1 and is also included in the supplementary video.

Approximately 20 trials were conducted to tune the controller parameters in order to achieve the first successful jump. Due to fear of damaging the robot, we did not conduct enough trials to report a reliability metric for the box jumping controller on hardware.

VII. Cassie Running Controller

Jumping motions have a significant, but singular, impact event. In contrast, running makes impacts with the environment at every stride. Because each stride leads immediately into the next, consistent tracking performance is essential to achieving stable running. For this reason, we also develop a running controller to showcase the benefits of the impact-invariant projection. The impact-invariant space for running is also larger because running makes impact with its environment with only a single foot at a time.

A. Controller Architecture

We track a SLIP-like pelvis trajectory and Raibert stepping generated footstep trajectories [29] using the same OSC as the jumping controller. A diagram with the key elements is shown in Fig. 10.

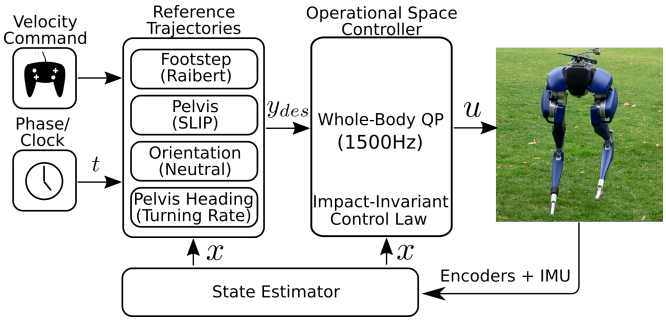


Fig. 10: Key elements of the running controller diagram.

B. Reference Trajectory Generation

1) Mode timing: We use variable stance and flight durations to construct reference trajectories and determine contact mode switches. The nominal stance and flight durations, T_s and T_f respectively, are given in Table IV. However, we allow for variations in the timing to better align with the predicted touchdown and liftoff. The upcoming transition times for the full gait cycle are computed at each mode switch using the following heuristics:

$$T_s^* = \text{clip}\Big(\frac{l}{y_{SLIP}} T_s, (1-\sigma_s)T_s, (1+\sigma_s)T_s\Big)$$

$$T_f^* = \text{clip}\Big(\dot{y}_{SLIP} + \sqrt{\dot{y}_{SLIP}^2 - 2g(l - y_{SLIP})}, (1-\sigma_f)T_f, (1+\sigma_f)T_f\Big),$$

where we clip the modified stance and flight durations to minimize timing changes in response to large disturbances. The modified stance duration computes the ratio of the rest length to the SLIP length to roughly scale the liftoff time, and the flight duration predicts the time to touchdown assuming a ballistic trajectory.

2) SLIP-like pelvis trajectory: Inspired by the extensive literature on SLIP, we use a SLIP-like reference for the pelvis motion during stance {LS, RS}. We achieve a spring-like behavior by regulating the pelvis position relative to the current stance foot to a constant target height l and using the OSC gains to achieve the desired dynamics.

$$\ddot{y}_{SLIP,cmd} = K_p(l - y_{SLIP}) + K_d(\dot{y}_{SLIP}). \quad (25)$$

An important note is that we tune K_p and K_d to achieve the desired dynamics and not to achieve the best tracking.

TABLE II: Tracking objectives for the running controller. All values are expressed in the world frame.

Symbol	Description
$L_{SLIP} \in \mathbb{R}^3$	Pelvis position relative to the stance foot
$\psi \in SO(3)$	Pelvis Orientation
$y_L, y_R \in \mathbb{R}^3$	Foot position relative to pelvis
$\beta_L, \beta_R \in \mathbb{R}$	Hip yaw angle
$\phi_L, \phi_R \in \mathbb{R}$	Toe joint angle

We adopt this simpler approach over tracking to true SLIP dynamics because the true dynamics are more difficult to regulate due to additional parameters and being purely an acceleration reference.

3) Footstep trajectories: Regulating the center of mass velocity is achieved through foot placement. While there are possible variations for choosing where to place the foot [30] [16], the basic principle behind all the stepping strategies is stepping in the direction that you are falling. We choose to regulate the running velocity by planning footsteps with the widely recognized Raibert footstep control law [29]:

$$y_{ft} := \begin{bmatrix} y_{ft,x} \\ y_{ft,y} \\ y_{ft,z} \end{bmatrix} = \begin{bmatrix} K_x(v_{des,x} - v_x) \\ K_y(v_{des,y} - v_y) \\ -l \end{bmatrix}, \tag{26}$$

where K are the Raibert stepping gains, v_{des} are the desired velocities as commanded by the operator, and v is the current pelvis velocity computed by the state estimator. x and y in this context denote the sagittal and lateral directions respectively. y_{ft} then defines the target footstep location relative to the pelvis.

With the end footstep location $y_{ft,2}$ specified, we can generate a trajectory for the swing foot given its initial position $y_{ft,0}$ at liftoff to the final desired location. We specify all the reference trajectories as piecewise quadratic polynomials, so with the additional degrees of freedom we add a waypoint $y_{ft,1}$ so that the trajectory roughly resembles the swing leg retraction profile observed in both numerical optimization [6] [5] and biology [31]. The full set of constraints defining the piecewise quadratic polynomial are as follows:

$$\begin{aligned} h &= [0, 0.6(T_s + 2T_f), (T_s + 2T_f)]^T \\ \sigma_0(h[0]) &= y_{ft,0} \\ \sigma_0(h[1]) &= y_{ft,1} \\ \sigma_1(h[1]) &= y_{ft,1} \\ \sigma_1(h[2]) &= y_{ft,2} \\ \dot{\sigma}_0(h[1]) &= \dot{\sigma}_0 h[1] \\ \ddot{\sigma}_0(h[1]) &= \ddot{\sigma}_0 h[1] \end{aligned}$$

where $y_{ft,0}$ is the initial foot position at liftoff, $y_{ft,2}$ is the desired footstep location at touchdown Eq. (26), and $y_{ft,1} = y_{ft,0} + 0.8(y_{ft,0} + y_{ft,2}) + [0, 0, d]^T$ is the waypoint we introduce to specify foot clearance. An illustration of the trajectory profile is shown in Fig. 11.

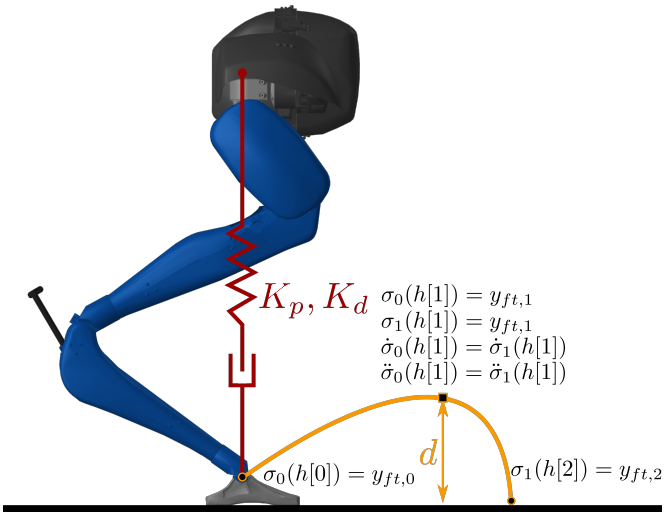


Fig. 11: Illustration of key references for the running controller. The target height and swing foot trajectory with leg retraction profiles are shown in red and yellow respectively.

C. Reference Tracking

1) Finite State Machine: We use a time-based finite state machine (FSM) using the timings from Section VII-B1 to specify the active contact mode and generate the clock signal for the reference trajectories. The set of finite states is {LS, LF, RS, RF}; L and R correspond to the left and right legs respectively and S and F correspond to Stance and Flight. We distinguish between the two air phases {LF, RF} to prescribe different tracking priorities for the different legs. The nominal durations for stance and flight deployed on hardware are reported in Table IV.

2) Tracking Objectives: The tracking objectives for the running controller are reported in Table II. Although L_{SLIP} is defined as the position $\in \mathbb{R}^3$ of the pelvis relative to current stance foot expressed in the world frame, we only track the vertical component.

During left stance (LS), the active vector of tracking objectives is $[L_{SLIP}, y_R, \psi, \phi_R, \beta_R] \in \mathbb{R}^9$. For right stance (RS), the tracking objectives are the same, just mirrored for the other leg. During the aerial modes LF, RF, the active tracking objectives are $[y_L, y_R, \psi, \beta_L, \beta_R, \phi_L, \phi_R] \in \mathbb{R}^{13}$. The active tracking objectives per mode are also illustrated in Fig. 12.

Note, the dimension of the tracking objectives in flight, 13, is greater than the total degrees of actuation, 10. Therefore, during flight, the control formulation is overspecified and perfect tracking cannot be achieved. We choose to leave the problem overspecified as opposed to leaving out either the pelvis orientation or one of the foot targets because we found that trading off between multiple tracking objectives led to better performance on hardware.

3) Turning: We use the identity quaternion as the desired pelvis orientation and zero as the the desired angular velocity. By setting the task-space gains to $K_p = \text{diag}[150, 200, 0]^T$ and $K_d = \text{diag}[10, 10, 5]^T$, the robot operator can specify the desired yaw velocity and achieve basic turning.

D. Simulation Analysis of Impact-Invariant Control

In simulation, we achieve maximum forward velocities of over 3 m/s as shown in Fig. 13. To reach the maximum velocity, we linearly ramp the desired velocity from 0 m/s to 3 m/s and back down to 0 m/s.

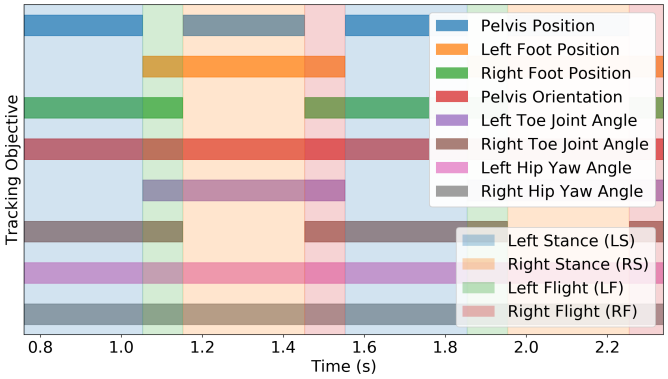


Fig. 12: Active tracking objectives per mode for the running controller.

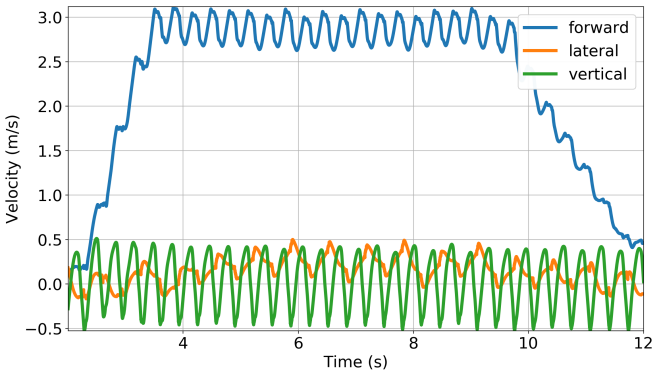


Fig. 13: Simulated robot tracking a target velocity that is linearly ramped up and back down to 0. The forward velocity shows the deceleration/acceleration profile associated with SLIP dynamics. Lateral velocity shows signs of instability at higher velocities.

The projection is an essential component for our framework to achieve stable running. In simulation, the controller without the projection is immediately unstable when the foot touches the ground. We also compare the impact-invariant controller with a controller where we set the velocity feedback gains to 0 during the same projection window. We refer to this controller as no-derivative. Both controllers are tasked with tracking a forward velocity command and the average errors over 20 secs are reported in Table III. The errors are computed as the position error at touchdown. While the no-derivative controller has only slightly worse tracking performance on individual tracking objectives, the minor tracking discrepancies have a non-negligible effect on overall motion. For this reason, the no-derivative controller is around 0.5 m/s slower than the impact-invariant controller with the same velocity command.

E. Hardware Results

We are able to achieve stable running, with the ability to command forward velocities and turn, on the physical robot using the same gains used in simulation. We tested the robot both indoors and outdoor on grass and a turf field. The robustness of the impact-invariant control to early and late impacts on the physical robot is shown in Fig. 14. Videos of the experiments with the physical robot are included in the supplementary video. Relevant controller parameters of the running controller are provided in Table IV.

TABLE III: Average tracking error of the impact-invariant controller and no-derivative controller for running with a constant forward velocity command.

Objective	Impact-Invariant Error	No-Derivative Error
Foot forward position	6.6 cm	6.6 cm
Foot lateral position	**3.6 cm**	4.4 cm
Foot vertical position	**3.9 cm**	5.6 cm
Hip yaw angle	**0.026 rad**	0.034 rad
Toe joint angle	0.030 rad	**0.020 rad**

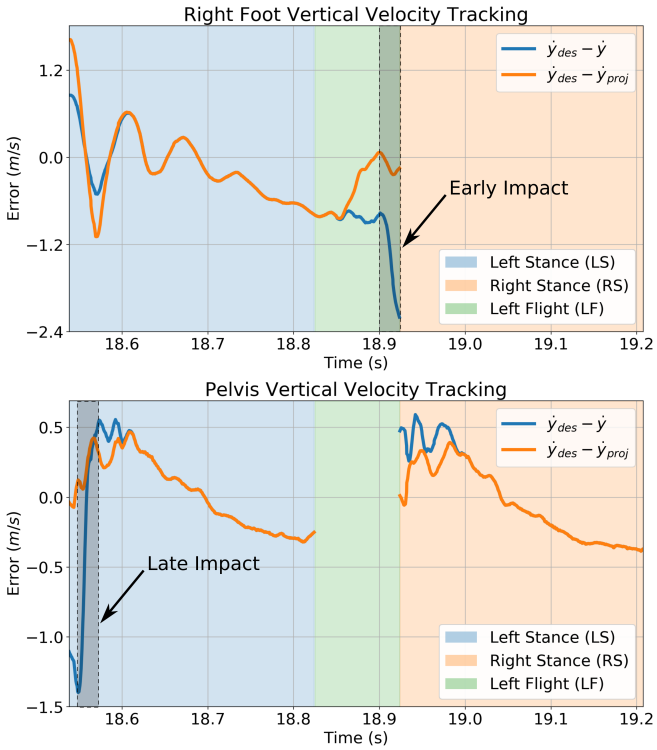


Fig. 14: Velocity tracking errors from the running controller on the physical Cassie robot. During the experiment, the robot makes impact *late* with its left foot, which initially results in a large tracking error for the pelvis target. The robot also make impact *early* with its right foot, which results in a large tracking error for the right foot target. In both cases, the impact-invariant projection protects the controller from overreacting to the mismatch in impact timing.

VIII. Discussion

In this paper, we introduce a general strategy that enables controllers to be robust to uncertainties in the impact event without sacrificing control authority over unaffected dimensions. The strategy makes use of an easy-to-compute modification of how the velocities of the robot enter the controller. Specifically, we project the velocity error into a subspace that is invariant to the impact event. This projection completely eliminates sensitivity of the controller to potential contact impulses, while minimally deviating from the original controller.

TABLE IV: Relevant running controller parameters

Parameter	Value
Projection window T	0.050 s
Blend time constant τ	0.005 s
Stance duration T_s	0.30 s
Flight duration T_f	0.09 s
Pelvis target height l	0.85 m
Foot clearance d	0.2 m

We demonstrate through examples with legged robots that the impact-invariant projection is robust to the impact event, while achieving better tracking performance compared to alternative methods. The modification can easily be applied to controllers for complex bipeds such as Cassie, and the modification enables highly dynamic motions such as jumping and running.

A. Recovered Control Authority

We only apply the projection in a small time window around an impact. An alternative to the projection is to instead turn off all derivative feedback as is done during the contact transition mode [13]. If impacts are infrequent and short, the additional benefit gained from using the impact-invariant projection is minimal. However, for motions where impacts are frequent such as running, the additional benefit can be substantial. Our running controller uses a projection window of 50 ms on both sides of the impact event for a total of 100 ms when the projection is active. The nominal stepping period of the running controller is 0.39 s, so the projection is active for over 25% of the time. Even tighter window durations cover a non-negligible duration of the entire motion.

B. Introducing Additional Controller Invariances

The impact-invariant framework enables robustness to a very particular source of uncertainty: impacts. It seems straightforward to extend this idea to eliminate sources of uncertainty from the control law, but on further examination impacts may be a very specific case of where this method would be useful. The key observation of this work is that impacts are *brief* periods of *high uncertainty* that enter the dynamics in a highly structured manner. Therefore, a reasonable approach is to effectively ignore the uncertainty in the short amount of time when the magnitude of the uncertainty is at its greatest. Other sources of uncertainty such as model differences or uncontrollable elements such as physical springs do not have these attributes. We cannot ignore model differences as they permeate the entire dynamics and are persistent throughout, and while springs only enter the dynamics at a single joint, their oscillations do not resolve in a short amount of time.

C. Future Work

Although the examples featured in this work focus on legged locomotion, the method is general and can be applied to other rigid body systems with impacts, such as manipulation. For future work, we plan to investigate how this method can be applied to high-speed grasping. A complementary avenue of future research lies in how we can leverage other sensor modalities such as tactile sensing as a method to have a less conservative bound on impact uncertainty by working with partial sensor feedback.

Acknowledgements

We thank Yu-Ming Chen and Brian Acosta for their countless hours of help with hardware testing and helpful discussions. We thank Alp Aydinoglu for helpful discussions about the math.

References

[1] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *arXiv preprint arXiv:2211.11644*, 2022.

[2] M. Halm and M. Posa, "Modeling and analysis of non-unique behaviors in multiple frictional impacts," in *Robotics: Science and Systems*, 2019.

[3] N. Fazeli, S. Zapolsky, E. Drumwright, and A. Rodriguez, "Fundamental limitations in performance and interpretability of common planar rigid-body contact models," in *International Symposium on Robotics Research (ISRR)*. Springer, 2020, pp. 555–571.

[4] C. D. Remy, "Ambiguous collision outcomes and sliding with infinite friction in models of legged systems," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1252–1267, 2017.

[5] A. Seyfarth, H. Geyer, and H. Herr, "Swing-leg retraction: a simple control model for stable running," *Journal of Experimental Biology*, vol. 206, no. 15, pp. 2547–2555, 2003.

[6] H. Dai and R. Tedrake, "Optimizing robust limit cycles for legged locomotion on unknown terrain," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 1207–1213.

[7] K. Green, R. L. Hatton, and J. Hurst, "Planning for the unexpected: Explicitly optimizing motions for ground uncertainty in running," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1445–1451.

[8] J. Zhu, N. J. Kong, G. Council, and A. M. Johnson, "Hybrid event shaping to stabilize periodic hybrid orbits," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 01–07.

[9] Y. Wang and A. Kheddar, "Impact-friendly robust control design with task-space quadratic optimization," in *Robotics: Science and Systems (RSS)*, 2019.

[10] S. Mason, N. Rotella, S. Schaal, and L. Righetti, "Balancing and walking using full dynamics lqr control with contact constraints," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 63–68.

[11] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, *et al.*, "No falls, no resets: Reliable humanoid behavior in the darpa robotics challenge," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 623–630.

[12] M. Rijnen, E. de Mooij, S. Traversaro, F. Nori, N. van de Wouw, A. Saccon, and H. Nijmeijer, "Control of humanoid robot motions with impacts: Numerical experiments with reference spreading control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4102–4107.

[13] J. J. van Steen, N. van de Wouw, and A. Saccon, "Robot control for simultaneous impact tasks via quadratic programming-based reference spreading," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 3865–3872.

[14] J. J. van Steen, A. Coşgun, N. van de Wouw, and A. Saccon, "Dual arm impact-aware grasping through time-invariant reference spreading control," *arXiv preprint arXiv:2212.00877*, 2022.

[15] B. Acosta, W. Yang, and M. Posa, "Validating robotics simulators on real-world impacts," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6471–6478, 2022.

[16] Y. Gong and J. Grizzle, "Angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-based controller," *arXiv preprint arXiv:2008.10763*, 2020.

[17] W. Yang and M. Posa, "Impact invariant control with applications to bipedal locomotion," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5151–5158.

[18] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. Westervelt, C. C. De Wit, and J. Grizzle, "Rabbit: A testbed for advanced control theory," *IEEE Control Systems Magazine*, vol. 23, no. 5, pp. 57–79, 2003.

[19] P. M. Wensing and D. E. Orin, "Generation of dynamic humanoid behaviors through task-space control with conic optimization," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3103–3109.

[20] L. Sentis and O. Khatib, "Control of free-floating humanoid robots through task prioritization," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 1718–1723.

[21] 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.

[22] G. Strang, *Introduction to linear algebra*, vol. 3.

[23] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1366–1373.

[24] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[25] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[26] S. H. Jeon, S. Kim, and D. Kim, "Online optimal landing control of the mit mini cheetah," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 178–184.

[27] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3821–3828.

[28] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[29] M. H. Raibert, H. B. Brown Jr, and M. Chepponis, "Experiments in balance with a 3d one-legged hopping machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[30] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1620–1626.

[31] M. A. Daley and A. A. Biewener, "Running over rough terrain reveals limb control for intrinsic stability," *Proceedings of the National Academy of Sciences*, vol. 103, no. 42, pp. 15 681–15 686, 2006.

[32] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4057–4062.

[33] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020.

[34] G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim, "Contact model fusion for event-based locomotion in unstructured terrains," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[35] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

APPENDIX

TABLE V: Full controller gains for the Cassie jumping controller deployed on hardware. Weight and gain matrices are diagonal matrices, represented here as vectors to be concise.

Symbol	Description	Value
μ	Friction coefficient	0.6
T	Projection window	0.05 s
τ	Blend time constant	0.005 s

OSC Objective	W	K_p	K_d
Toe joint angle	0.01	1500	10
Hip yaw angle	2.5	100	5
Pelvis [x, y, z]	$[20, 2, 20]$	$[40, 50, 40]$	$[7.5, 5, 5]$
Pelvis [roll, pitch, yaw]	$[10, 5, 1]$	$[150, 200, 150]$	$[10, 10, 5]$
Foot [x, y, z]	$[10, 100, 10]$	$[125, 50, 150]$	$[2.5, 2.5, 0]$

TABLE VI: Full controller gains for the Cassie running controller deployed on hardware.

Symbol	Description	Value
μ	Friction coefficient	0.6
T	Projection window	0.05 s
τ	Blend time constant	0.005 s
l	Pelvis target height	0.85 m
T_s	Stance duration	0.3 s
T_f	Flight duration	0.09 s
σ_s	Stance duration variance	0.2
σ_f	Flight duration variance	0.1
	Footstep lateral offset	0.04 m
d	Foot clearance	0.2 m
K_x	Raibert footstep sagittal feedback	0.01
K_y	Raibert footstep lateral feedback	0.3

OSC Objective	W	K_p	K_d
Toe joint angle	0.01	1500	10
Hip yaw angle	2.5	100	5
Pelvis [x, y, z]	$[0, 0, 5]$	$[0, 0, 115]$	$[0, 0, 5]$
Pelvis [roll, pitch, yaw]	$[10, 5, 1]$	$[150, 200, 0]$	$[10, 10, 5]$
Foot [x, y, z]	$[10, 100, 10]$	$[125, 75, 75]$	$[5, 5, 5]$

A. Hardware Setup

All processing is done on the Intel NUC 11 computer onboard Cassie. Note we swapped the original Intel NUC onboard Cassie for a newer generation for better performance. We run the state estimator and the controllers asynchronously as separate processes, and communication between processes is handled using LCM [32], while user commands are sent through the radio remote.

1) State Estimator: We use the contact-aided invariant EKF developed in [33] to estimate the floating-base pelvis state. Although we do not directly use contact detection in our controllers, the state estimator utilizes the current contact mode estimate in the measurement update. We achieve this using a generalized-momentum observer, similar to the method used in [34], to estimate the contact force at each foot. We then set a threshold of 60 Nm on the contact normal force to define contact. We observe that this has a faster response and better accuracy over detecting contact using spring deflections. The state estimator runs at 2000 Hz.

2) Controller Implementation: We write our controllers using Drake [28] for the systems framework and all multibody calculations. In addition to the weights and gains for the tracking objectives given in Table V Table VI, we add small regularization weights to all the decision variables for better solver performance. To set different tracking priorities during the flight phase for the running controller, we linearly ramp the weight for the foot tracking objective from 0.5 to 4 times the specified value across the duration of the trajectory. Although the tracking objectives are expressed in the world frame, we compute the errors in the robot yaw frame in order to have the gains operate on the sagittal and lateral directions rather than the world x and y directions. Finally, we solve the OSC QP using a minor modification of the Drake OSQP interface [28] [35] at 1500 Hz.