Perceptive Mixed-Integer Footstep Control for Underactuated Bipedal Walking on Rough Terrain

Brian Acosta and Michael Posa

***Abstract*—Traversing rough terrain requires dynamic bipeds to stabilize themselves through foot placement without stepping in unsafe areas. Planning these footsteps online is challenging given non-convexity of the safe terrain, and imperfect perception and state estimation. This paper addresses these challenges with a full-stack perception and control system for achieving underactuated walking on discontinuous terrain. First, we develop model-predictive footstep control (MPFC), a single mixed-integer quadratic program which assumes a convex polygon terrain decomposition to optimize over discrete foothold choice, footstep position, ankle torque, template dynamics, and footstep timing at over 100 Hz. We then propose a novel approach for generating convex polygon terrain decompositions online. Our perception stack decouples safe-terrain classification from fitting planar polygons, generating a temporally consistent terrain segmentation in real time using a single CPU thread. We demonstrate the performance of our perception and control stack through outdoor experiments with the underactuated biped Cassie, achieving state of the art perceptive bipedal walking on discontinuous terrain. Supplemental Video: (Short [1], Long [2]).**

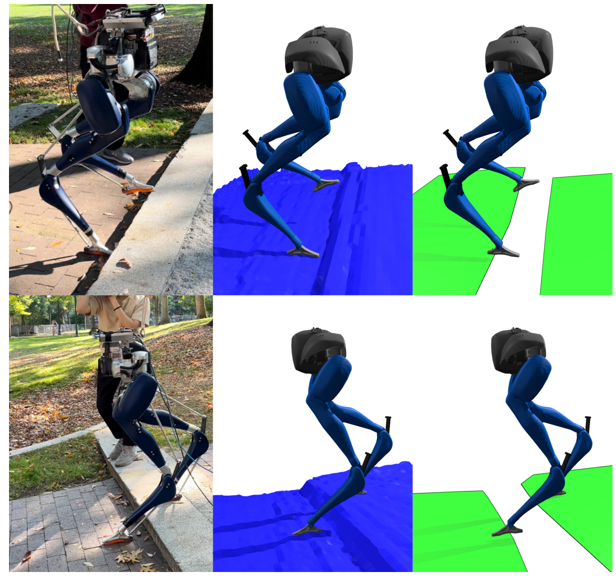


Fig. 1: The bipedal robot Cassie walks up and down brick steps using the perception and control framework developed in this paper. Left: the physical robot and steps. Middle: an elevation map of the steps. Right: a convex decomposition of the safe terrain.

I. Introduction

Bipedal robots can theoretically traverse challenging terrain by breaking contact with the ground to clear obstacles, making them potentially useful for disaster response, planetary exploration, and deployment in cluttered home environments. However, dynamic bipedal walking over rough terrain remains challenging for today's perception and control algorithms. To traverse rough terrain, bipeds must quickly identify safe footstep positions which maintain the robot's balance and make progress in the desired walking direction. This is a highly coupled problem where online terrain estimation is used to control an underactuated hybrid system. Despite the existence of mature techniques for both underactuated walking, and footstep planning over constrained footholds, few works attempt to address both of these problems at once. Often, underactuated gaits are stabilized within a fixed sequence of stepping-stone constraints [3–5], or rough terrain is assumed to be of varying height but without any unsafe footstep positions [6, 7]. Without these combinatorial aspects, the optimal control problem is easier to solve, but we have an incomplete solution for walking on rough terrain.

This paper presents Model Predictive Footstep Control (MPFC), a model-predictive-control-style footstep planner which reasons over many of the relevant decision variables for underactuated walking. In addition to discrete foothold selection, MPFC optimizes over the continuous footstep positions, center of mass trajectory, ankle torque, and gait timing. MPFC is one of the first controllers to simultaneously optimize over the discrete choice of stepping surface and the robot's dynamics in real time [1], and to our knowledge, this paper and its precursor [9] represent the first deployment of such a controller on hardware.

We use binary variables to assign each footstep to a convex foothold [11], providing a straightforward extension of linear-quadratic MPC footstep controllers [12] to discontinuous terrain, with the consequence that optimal control problem graduates in difficulty from a Quadratic Program to a Mixed-Integer-Quadratic Program (MIQP). MIQPs have been used extensively for offline trajectory optimization over broken terrains [13–15], but due to their combinatorial complexity in the planning horizon, they have seen much less use in real-time control. Our controller achieves solve times of less than 10 milliseconds by using a low dimensional, linear dynamics model, planning over a short footstep horizon, and eliminating foothold candidates far from the robot.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1845298. Toyota Research Institute also provided funds to support this work.

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

[1] [8] was published concurrently with the conference version of this paper [9] and uses artificial potentials to snap footsteps onto nearby footholds, and [10] was published shortly after [9], and enforces stepping stone constraints with offline-generated signal-temporal-logic objectives.

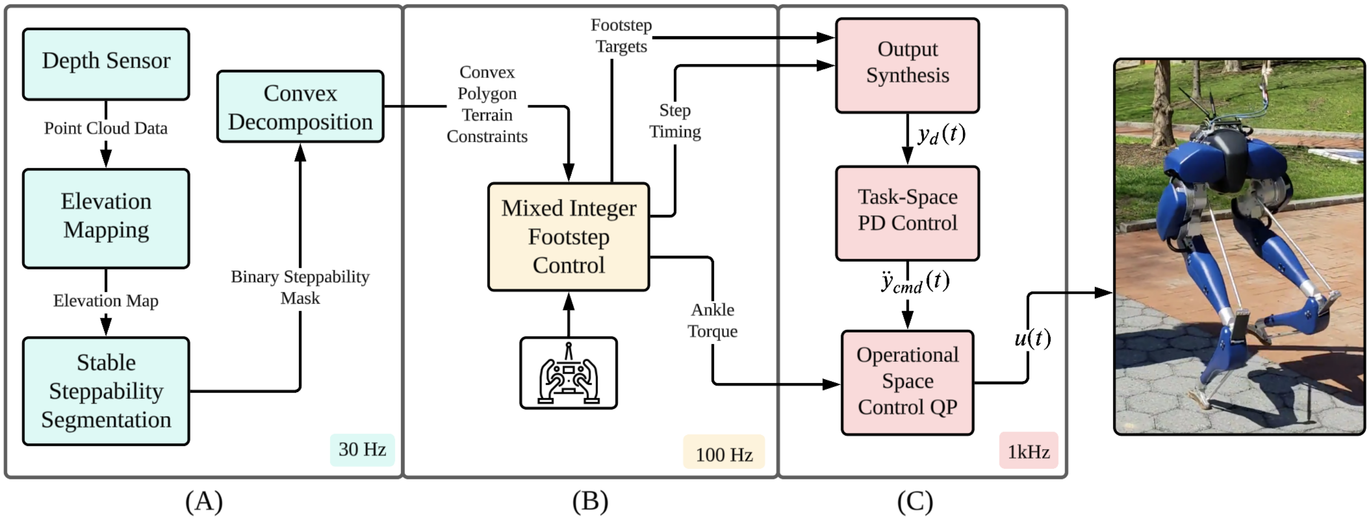


Fig. 2: The perception and control stack proposed in this paper to achieve underactuated walking over discontinuous terrain. Our perception stack (A) generates convex polygon foothold constraints for MPFC, a mixed-integer MPC style footstep planner (B). MPFC sends the next footstep, step timing adaptation, and ankle torque plan to a low-level operational-space-control process (C) which performs kHz level torque control.

A significant barrier to deploying mixed-integer footstep planning methods on hardware is the need for a convex planar polygon decomposition of the terrain around the robot. As we discovered during the hardware experiments for our original mixed-integer footstep planning work [9], and as others have noted in recent literature [16], explicit plane segmentation approaches suffer from poor temporal consistency. This destabilizes online footstep planners with constraints that "flicker" into and out-of existence.

We propose a new approach to terrain segmentation and convex decomposition. We argue that requiring a one-to-one correspondence between foothold contstraints in the controller and real planar polygons in the environment [16, 17] is overly restrictive and brittle. Our approach recognizes that planar polygons are a modeling choice used to support optimization based control, rather than a hard requirement for steppability. By focusing on avoiding terrain which is clearly unsafe, we arrive at a simple algorithm which is robust to non-planar surfaces and more temporally consistent than explicit plane segmentation.

Our terrain segmentation algorithm consists of two stages. The first stage, called Stable Steppability Segmentation (S3), uses local safety criteria and a simple hysteresis mechanism to classify elevation map pixels as safe or unsafe, resulting in a binary steppability mask. The second stage of our segmentation algorithm generates a set of convex polygons approximating the safe terrain identified by S3. We perform approximate convex decomposition[18], then take a convex inner-approximation of the resulting polygons before finally fitting plane paramenters to these convex polygons using the original elevation map. While the S3 implementation in this paper uses intuitive heuristic criteria for steppability classification, the general algorithm supports any number of criteria, allowing for composition with learning-based approaches and higher-level obstacle detectors.

An earlier version of our MPFC footstep controller was presented in [9], with limited hardware results due to the brittleness of explicit plane segmentation. This article extends that work by introducing our new terrain segmentation approach, improving our MPFC formulation, and presenting hardware experiments that demonstrate the capabilities of our perception and control stack.

The primary contributions of this paper are:

1) We present a new terrain segmentation framework which is faster and more temporally consistent than explicit plane segmentation.
2) We propose a new MPFC formulation which jointly optimizes over the robot's discrete choice of stepping surface, footstep plan, ankle torque, and step duration, using the step-to-step ALIP dynamics. Compared to [9], our new formulation simplifies the problem statement, resulting in faster solve times, while adding the ability to optimize over the initial stance duration.
3) The resulting full-stack system is validated with hardware experiments that demonstrate real–time perceptive, dynamic, underactuated walking over constrained footholds.

II. Related Work

Our controller and perception stack build on a number of mature or maturing techniques such as mixed–integer–convex footstep planning, linear-inverted-pendulum based footstep control, and elevation mapping for mobile-robot motion planning. This section reviews these and related topics, focusing on the features of our approach compared to what exists in the literature.

A. Footstep Planning over Rough Terrain

The literature on safe bipedal footstep planning mainly considers humanoid robots with large feet [19], which allow a feasible center of mass trajectory to be planned and tracked for any reasonable footstep plan. These plans can be generated quickly via traditional motion planning approaches like graph search [20] or mixed–integer–convex programming [11]. However, because footsteps are not re-planned at high rates, decoupled approaches have slow walking speeds to avoid violating zero-moment-point constraints [21].

1) Mixed-Integer Footstep Planning: Deits and Tedrake introduced the use of MIQPs for footstep planning in [11] by decomposing safe terrain into a collection of convex polygons, and using integer variables to assign every footstep to a polygon. Tonneau et al. [22] provide a convex approximation of this problem as a linear program, and Song et al. [23] show how both the mixed integer and linear programming formulations can be made more efficient by using a simplified trajectory planner to prune irrelevant footholds. In contrast to our work, these works focus on long horizon footstep planning, and only consider geometric criteria such as workspace constraints, and quasistatic stability criteria, such as the existence of a feasible center of mass trajectory which lies completely above the support polygon.

MIQP footstep planning has also been used for quadruped robots. In [24], Risbourg et al. use the convex relaxation from [22] online to project the desired footstep sequence to the closest convex footholds, subject to kinematic constraints. In [16], Corberes et al. incorporate this footstep planning strategy as an online foothold scheduler at 1-5 Hz with vision in the loop. Due to the low planning rate, and the lack of dynamics constraints in the contact scheduler, they rely on a separate whole body MPC to find feasible robot trajectories. Aceituno-Cabezas et al. [13] formulate a full quadruped trajectory optimization problem using mixed integer constraints for assigning footsteps to footholds and to approximate the nonlinear manifold constraint for 3D rotations. Their trajectory optimization features both kinematic and dynamics constraints, but does not re-plan the footholds in real time.

B. Footstep Control for Underactauted Bipeds

Dynamic walking research assumes minimal ankle actuation, instead viewing walking as controlled falling, where momentum can only be added or removed from the system by stepping to the appropriate spot on the ground. These approaches generally assume flat or constantly sloped ground without obstacles to synthesize reactive stepping controllers based on the linear inverted pendulum [25–27]. This approach regulates walking speed without ankle torque by using foot placement to affect the initial conditions of each single stance phase. Combined with output tracking via inverse-dynamics based whole body torque controllers, this approach has enabled dynamic and robust walking. The Angular Momentum Linear Inverted Pendulum (ALIP) model, in particular, has been shown to accurately describe the bulk motion of walking even for robots with heavy legs [28], and has been used to stabilize walking on sloped terrain [12], synthesize specialized stair climbing controllers [29], and walk on pre-selected constrained footholds [5].

C. Safe Terrain Estimation for Legged Locomotion

Elevation maps are a convenient intermediate terrain representation for legged locomotion due to their ability to fuse multiple sensor streams over time in a compact representation [30–32]. This has lead to a proliferation of algorithms for extracting convex planar polygons from elevation maps via plane segmentation [32, 33]. However, these approaches segment each elevation map independently, leading to issues with temporal consistency [9, 16], especially because elevation mapping is vulnerable to artifacts from drift in the floating base position estimate [34]. To generate temporally consistent polygon constraints in real-time, despite these challenges imposed by legged locomotion, Bin et al. develop a GPU accelerated semantic mapping framework [35] to directly estimate the state of polygonal terrain from depth images. This approach has advantages for stair climbing, where the terrain is known to be planar and precise foot placement is required, but could struggle in outdoor environments where the ground is not perfectly flat.

On unstructured terrain, some works compute heuristic costs from the elevation map to guide planning. McRory et al. encode various traversability costs into a graph search algorithm for humanoid footstep planning [36]. Jenelten et al. add a nonconvex cost on the gradient of the elevation map at the planned stance foot locations in their MPC formulation for quadrupedal walking [37]. These heuristic costs recognize that planar polygons are a modeling choice to support optimization based control, not a necessary condition for steppability. Our terrain segmentation approach adopts a similar philosophy by classifying elevation map cells as steppable or not without regard for global planarity, but still achieves global optimality in the MPC problem by transforming this classification back into mixed-integer convex terrain constraints.

D. Reinforcement Learning for Legged Locomotion

Sim-to-Real reinforcement learning, where control policies are learned in simulation and then deployed in the real world, has seen increasing success in recent years, especially for legged locomotion. These policies can be made very robust and performant through a combination of domain randomization and adaptation. For example, Siekmann et al. learn a blind stair climbing controller for Cassie in [38], and Duan et al. use a similar policy to walk on constrained footholds in [39]. With additional vision modules, they achieve perceptive locomotion over boxy terrain as well [40]. Because reinforcment learning can struggle in scenarios with sparse footholds, Jenelten et al. proposed a hierarchical approach where an MPC footstep planner guides a lower level learned tracking policy [41]. Yu et al. propose the opposite, where reinforcement learning is used learn high level strategies like gait selection and foot placement, and MPC is used to generate and stabilize corresponding full body motions [42]. This strategy is now powering Boston Dynamics' Spot quadruped in industrial use cases [43]. While reinforcement learning is not a focus of this paper, the major subcomponents of our stack (including our segmentation module) could easily be used in one of these hierarchical frameworks.

III. Preliminaries

This section overviews the reduced-order Angular-Momentum Linear Inverted Pendulum (ALIP) model used in MPFC, and the operational space controller used to track MPFC's outputs. We start by reviewing the ALIP dynamics, then we derive the reset map and step-to-step dynamics for a hybrid ALIP model with a finite double stance period. We then provide a linearization of solutions to the ALIP model with respect to time be used for stance timing adaptation. Finally we overview our inverse-dynamics operational space controller for output tracking based on MPFC solutions.

A. ALIP model

The ALIP model (Fig. 3) is an approximation of the horizontal center-of-mass dynamics of the robot during single stance. The ALIP model is similar to Kajita's Linear Inverted Pendulum model [25], but uses angular momentum about the contact point in place of center-of-mass velocity to describe the speed of the robot. Angular momentum about the contact point has the advantage of being relative-degree three to (non-stance ankle) motor torques, compared to relative-degree one for center-of-mass velocity [28], making the predictions of the ALIP model relatively accurate even for robots with heavy legs. We direct the reader to [12] for a derivation of the ALIP dynamics assuming piece-wise planar terrain with a passive ankle. The state of the ALIP model consists of the horizontal position of the center of mass realative to the stance foot, (x_{com}, y_{com}) and the tilting components of the angular momentum of the robot about the contact point (L_x, L_y).

To take full advantage of Cassie's blade foot, we include ankle torque in the sagittal plane, u as an input to the continuous time ALIP model. The dynamics of the ALIP with ankle torque are given by

$$\underbrace{\begin{bmatrix}\dot{x}_{com}\\ \dot{y}_{com}\\ \dot{L}_x\\ \dot{L}_y\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}0 & 0 & 0 & \frac{1}{mH}\\ 0 & 0 & \frac{-1}{mH} & 0\\ 0 & -mg & 0 & 0\\ mg & 0 & 0 & 0\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}x_{com}\\ y_{com}\\ L_x\\ L_y\end{bmatrix}}_{x} + \underbrace{\begin{bmatrix}0\\0\\0\\1\end{bmatrix}}_{B} u \quad (1)$$

where m is the robot's mass, and H is the height of the CoM above the terrain, and all quantities are in the stance frame.

B. Hybrid ALIP Model Based on Foot Placement

To enable control of the ALIP through foot placement, we derive a reset map relating the positions of the robot's feet at touchdown to a discrete jump in the ALIP state. Many walking controllers feature a double stance phase during which weight transfers from one leg to the other. A double stance phase is particularly useful for Cassie, to avoid oscillations caused by rapidly unloading Cassie's leaf springs. To treat the single and double stance phases as a single step in the step-to-step dynamics [26], we derive a reset map from x_-, the ALIP state just before footfall, to x_+, the ALIP state just after liftoff, including a double stance phase of fixed duration, T_{ds}. We start by integrating the double stance dynamics, and then we apply a coordinate change to express the ALIP state with respect to the new stance foot.

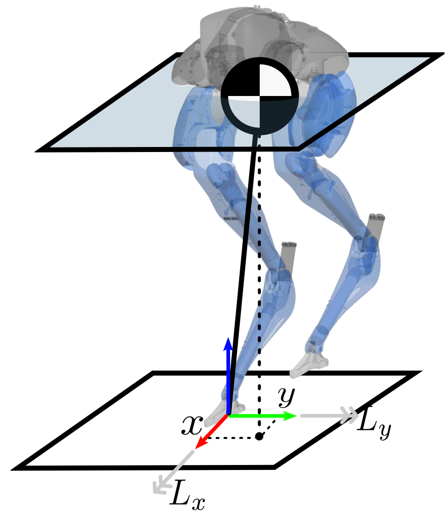


Fig. 3: The ALIP model assumes that the robot's CoM is restricted to a virtual plane above the terrain. The states of the ALIP model are the horizontal CoM positions, and the angular momentum of the robot about the horizontal axes.

During double stance, we leave the ankles passive and treat the center of pressure (CoP) between the two feet as a control input. We then integrate the resulting dynamics with an assumed input trajectory,

$$p_{CoP}(t) = p_- + f(t)(p_+ - p_-) \quad (2)$$

where $p_-, p_+ \in \mathbb{R}^3$ are the pre- and post-touchdown stance foot positions, t is the time since the beginning of double stance, and $f(t) : \mathbb{R} \mapsto [0, 1]$ determines the rate at which weight is transferred to the new stance foot. The CoP enters the ALIP dynamics via the angular momentum transfer formula

$$L_{CoP} = L_{p_-} + (p_{CoP} - p_-) \times m v_{CoM}.$$

To formulate the double-stance dynamics as a linear system, we introduce an assumption to remove the cross-product term:

Assumption. $[(p_{CoP} - p_-) \times m v_{CoM}]_{x,y} \approx 0.$

Justification. The most straightforward justification is that the robot is generally stepping in the direction it is walking, so v_{CoM} is approximately parallel to $p_+ - p_-$. We can also consider that under the ALIP model, $p_{CoP} - p_-$ and v_{CoM} both lie in the ground plane, so $(p_{CoP} - p_-) \times v_{CoM}$ must be normal to this plane. For flat ground, this vector is perpendicular to the world $x - y$ axes, and the x and y components remain small for non trivial slopes (e.g. $\sin(15°) \approx 0.25$).

Under this assumption, the tilting angular momenta about p_- and p_{CoP} are equal. This allows us to treat the CoP as a virtual contact point, yielding

$$\begin{aligned}\dot{L}_x &= -mg(y_{com} - p_{CoP,y}(t))\\ \dot{L}_y &= mg(x_{com} - p_{CoP,x}(t)).\end{aligned} \quad (3)$$

Substituting (2) into (3), we arrive at the continuous dynamics describing the ALIP during double stance:

$$\dot{x} = Ax + \underbrace{\begin{bmatrix} 0_{2\times1} & 0_{2\times1} & 0_{2\times1} \\ 0 & mg & 0 \\ -mg & 0 & 0 \end{bmatrix}}_{B_{CoP}} f(t)(p_+ - p_-) \quad (4)$$

The solution to (4) with the initial condition $x(0) = x_-$ is linear in x_-, p_- and p_+ [44]:

$$x(T_{ds}) = A_r x_- + B_{ds}\left(p_+ - p_-\right). \quad (5)$$

where $A_r = \exp(AT_{ds})$ and

$$B_{ds} = \left(\int_0^{T_{ds}} f(t) e^{A(T_{ds}-t)} dt\right) B_{CoP}. \quad (6)$$

For $f(t) = \frac{t}{T_{ds}}$, i.e. linearly shifting the robot's weight between feet over double stance[2], (6) evaluates to

$$B_{ds} = A_r A^{-1}\left(\frac{1}{T_{ds}} A^{-1}\left(I - A_r^{-1}\right) - A_r^{-1}\right) B_{CoP}. \quad (7)$$

The remainder of the reset map is a coordinate change to express the CoM position as relative to the new stance foot,

$$x_+ = x(T_{ds}) + \underbrace{\begin{bmatrix} -I_{2\times2} & 0_{2\times1} \\ 0_{2\times2} & 0_{2\times1} \end{bmatrix}}_{B_{fp}} \left(p_+ - p_-\right). \quad (8)$$

with the fp subscript denoting "foot placement".

By sequentially applying (5) then (8), we arrive at a reset map from x_- to x_+ which is linear in x_-, x_+, p_- and p_+,

$$x_+ = \begin{bmatrix} A_r & (-B_{ds} - B_{fp}) & \underbrace{(B_{ds} + B_{fp})}_{B_r} \end{bmatrix} \begin{bmatrix} x_- \\ p_- \\ p_+ \end{bmatrix}. \quad (9)$$

C. Step-to-Step ALIP Dynamics

We will also consider step-to-step (s2s) ALIP dynamics. We view the ALIP without ankle actuation as a discrete-time linear time-invariant system by sampling the ALIP state at the end of each (fixed duration of T_{ss}) single stance phase. These dynamics are simply

$$x_{n+1} = A_{s2s} x_n + B_{s2s}(p_{n+1} - p_n) \quad (10)$$

where $A_{s2s} = \exp(A(T_{ss} + T_{ds}))$ and $B_{s2s} = \exp\left(AT_{ss}\right) B_r$.

D. Step Timing Adaptation

We will use the fact that the initial ALIP state is constant to adapt the duration of the initial swing phase as part of the MPFC problem formulation. This has previously been applied to controllers based on the divergent component of motion [4, 45, 46] and instantaneous capture point [27], as these models admit an exact coordinate transform for the initial stance duration to make the touchdown state linear in the transformed variable. The ALIP state space does not admit this coordinate change, so we instead linearize the solution to (1). Given a stance duration T, and an initial ALIP state x_c, the exact solution to (1) with constant ankle torque, u, is

$$x(T) = A_d(T) x_c + B_d(T) u \quad (11)$$

Where $A_d(T) = \exp(AT)$ and $B_d(T) = A^{-1}(A_d(T) - I)B$. We linearize (11) with respect to T and u about a nominal stance time of T^* and ankle torque of 0 to find the ALIP state at the end of the current stance period (and the initial state of the s2s ALIP model), x_0:

$$x_0 = A_d(T^*) x_c + \left.\frac{\partial A_d}{\partial T}\right|_{T^*} (T - T^*) x_c + B_d(T^*) u. \quad (12)$$

E. Operational Space Control

We use operational-space control (OSC) to track outputs such as swing foot position and pelvis orientation, while respecting frictional contact constraints [47]. OSC considers a full-order Lagrangian model of the robot's dynamics:

$$M(q)\dot{v} + C(q, v) = g(q) + Bu + J_\lambda^T \lambda \quad (13)$$

Where q and v are generalized positions and velocities, u are inputs, and λ are forces arising from contacts or other holonomic constraints. Given a set outputs to track, $\{y_i\}$, we define task-space PD controllers,

$$\ddot{y}_{i,cmd} = \ddot{y}_{i,des} + K_p(y_{i,des} - y_i) + K_d(\dot{y}_{i,des} - \dot{y}_i).$$

The goal of OSC is to find dynamically feasible inputs, generalized accelerations, contact forces, and constraint forces, such that the task-space accelerations, $\ddot{y}_i = J_i\dot{v} + \dot{J}_i v$, match the PD controller as closely as possible, while satisfying contact constraints and holonomic constraints. We formulate this as a quadratic program with Lorentz cone constraints on the contact forces:

$$\begin{aligned}
\underset{\dot{v}, u, \lambda_h, \lambda_c, \varepsilon}{\text{minimize}} \quad & \sum_i^N \tilde{\ddot{y}}_i^T W_i \tilde{\ddot{y}}_i + \|u\|_W^2 + \|\dot{v}\|_W^2 + \|\varepsilon\|_W^2 && \text{(14a)} \\
\text{subject to} \quad & M\dot{v} + C = g + Bu + J_h^T \lambda_h + J_c^T \lambda_c && \text{(14b)} \\
& J_h \dot{v} = -\dot{J}_h v && \text{(14c)} \\
& J_c \dot{v} + \varepsilon = -\dot{J}_c v && \text{(14d)} \\
& \lambda_c \in \mathcal{F} && \text{(14e)} \\
& u_{min} \leq u \leq u_{max} && \text{(14f)}
\end{aligned}$$

where λ_c and J_c are the stacked contact forces and contact Jacobians, and $\mathcal{F}$ is the product of the friction cones for each contact point. The contact constraint is treated as a soft constraint by the introduction of a slack variable ε to ensure the problem is always feasible. The holonomic constraint $J_h\dot{v} = -\dot{J}_h v$ represents Cassie's four-bar linkages and fixed joint constraints to model Cassie's leaf spring springs. The task space acceleration errors are $\tilde{\ddot{y}}_i = \ddot{y}_{cmd} - (J_{y,i}\dot{v} + \dot{J}_{y,i} v)$.

[2]Because B_{ds} is decoupled in x and y, our MPFC implementation assumes $f(t) = 1$ for the lateral compenents of the ALIP state, which corresponds to instantaneous weight transfer at the beginning of double-stance. We detail how this helps Cassie track the desired step width in Appendix A.

IV. Mixed Integer Footstep Control

The following section details the formulation of our model predictive footstep controller as an MIQP (Fig. 2B). For the current single stance phase, MPFC can adjust the gait timing and stance-foot ankle torque. To encourage robust footstep choices, in the subsequent stance phases, MPFC can only affect the step-to-step ALIP state through foot placement. The continuous decision variables of MPFC are the step-to step ALIP states, x_n, the foostep positions, p_n, a constant ankle torque during single stance, u, and the duration of the current stance phase, T. We also introduce one binary variable per discrete foothold per stance phase, $\mu_{n,i}$, where $i \in 1 \dots M$ specifies that the binary variable corresponds to $\mathcal{P}_i$, one of M available convex polygon footholds. A diagram of the key MPFC decision variables is show in Fig. 4.

We can now introduce the problem statement of the MPFC (15), and dedicate the rest of this section to elaborating on the cost and constraints. Let x_c be the current ALIP state, with T^* seconds nominally remaining in the current stance period. MPFC is be formulated as:

$$\begin{aligned}
\underset{\mathbf{x},\mathbf{p},\boldsymbol{\mu},u,T}{\text{minimize}} \quad & J_{mpc}(\mathbf{x},\mathbf{p}) + J_{reg}(T,u) && (15a)\\
\text{subject to} \quad & x_0 = A_d x_c + A A_d x_c (T - T^*) + B_d u && (15b)\\
& x_{n+1} = A_{s2s} x_n + B_{s2s}(p_{n+1} - p_n) && (15c)\\
& \mu_{n,i} = 1 \implies p_n \in \mathcal{P}_i && (15d)\\
& \sum_{i\in\mathcal{I}} \mu_{n,i} = 1 && (15e)\\
& \mu_{n,i} \in \{0,1\} && (15f)\\
& \text{CoM, Input,Timing, and Footstep limits}
\end{aligned}$$

A. Cost Design

This section explains how we design the MPFC cost function for walking on rough terrain. Using periodic orbits of the ALIP dynamics as reference trajectories can lead to unintuitive behaviors when crossing gaps in the terrain. To prioritize taking reasonable footsteps, we strongly regularize the relative footstep positions to a nominal stepping pattern. Rather than specifyng a reference ALIP trajectory, we regularize the ALIP trajectory to the affine subspace of all possible period-two (P2) orbits that achieve the desired velocity, v_{des}. This subspace can be expressed as

$$\Pi_n x_n = \Pi_n d_n(v_{des}). \tag{16}$$

We show how to construct the projection matrix Π_n and offset d_n in Appendix B. Our MPC cost is then formulated as

$$\begin{aligned}
J_{mpc}(\mathbf{x},\mathbf{p}) = \sum_{n=1}^{N-1} \big[& (x_n - d_n)^T \Pi_n^T Q \Pi_n (x_n - d_n) + \\
& (\delta p_n - \delta p_n^*)^T R (\delta p_n - \delta p_n^*) \big] + \\
& (x_N - d_N)^T \Pi_N^T Q_N \Pi_N (x_N - d_N)
\end{aligned}$$

where $\delta p_n = p_{n+1} - p_n$, δp_n^* defines the nominal stepping pattern, and Q, R, and Q_N are positive-definite weight matrices. The nominal step sequence is defined by the desired velocity and the step width, l, as

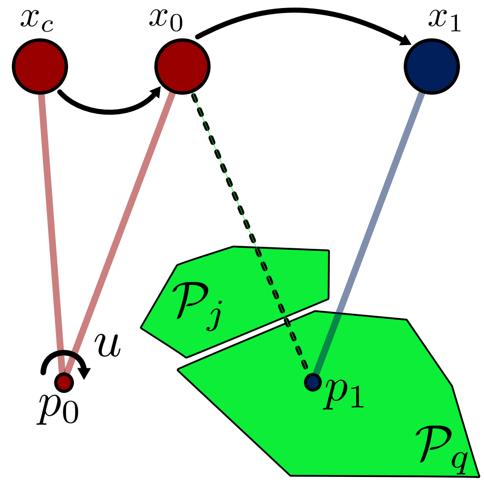


Fig. 4: Illustration of key MPFC decision variables and constraints for a horizon of 2 stance phases. x_c is the current ALIP state, u is ankle torque applied during the current stance phase, x_0 is the ALIP state at the end of the current stance phase, and x_1 is the ALIP state at the end of the next stance phase. The initial stance foot position, p_0, is unconstrained, and the remaining footstep locations are constrained to lie in either $\mathcal{P}_j$ or $\mathcal{P}_q$ using integer variables.

$$\delta p_n^* = \begin{bmatrix} v_{des,x}(T_{ss} + T_{ds}) \\ v_{des,y}(T_{ss} + T_{ds}) + \sigma_n l \\ 0 \end{bmatrix}. \tag{17}$$

where $\sigma_n = -1$ for left-stance phases and $+1$ for right stance. We add quadratic regularization costs on T and u, weighted by the positive scalars w_T and w_u:

$$J_{reg} = w_T \|T - T^*\|^2 + w_u \|u\|^2. \tag{18}$$

B. Dynamics Constraints

The initial state constraint (15b) evaluates (12) to relate the current ALIP state to the initial step-to-step ALIP state via ankle torque and step timing adaptation. The dynamics constraints (15c) are the step-to-step ALIP dynamics (10).

C. Foothold Constraints

Each convex polygonal foothold is defined by a plane $f_i^T p = b_i$ and a set of linear constraints $F_i p \leq c_i$. The logical constraint (15d) is enforced with the big-M formulation

$$\begin{aligned}
F_i p_n &\leq c_i + M(1 - \mu_{n,i}) && (19a)\\
f_i^T p_n &\leq b_i + M(1 - \mu_{n,i}) && (19b)\\
-f_i^T p_n &\leq -b_i + M(1 - \mu_{n,i}). && (19c)
\end{aligned}$$

With appropriately normalized F_i and f_i, (19) corresponds to relaxing each foothold constraint by M meters when $\mu_i = 0$. Since our problem scale is on the order of 2 m, we choose M = 10 for simplicity[3]. The binary constraint (15f) and the summation constraint (15e) imply that exactly one foothold must be chosen per single-stance phase.

[3] M must be large enough for every relaxed foothold to contain every unrelaxed foothold, but should otherwise be small for numerical stability

D. CoM, Timing, Input, and Footstep Limits

We add the following constraints to reflect the physical limitations of the robot:

- We add a soft-constraint on the CoM position of ± 35 cm. in each direction.
- We limit the total single stance duration to the range [0.27, 0.33] seconds.
- We add a crossover constraint to prevent the feet from crossing the $x - z$ plane.
- We limit the ankle torque to 22 Nm to keep the center of pressure within the blade foot.
- With $T_{min} = 0.27$ seconds left in the nominal single stance time, we add a time-varying trust region constraint that p_1 cannot deviate more than T cm from the previous solution.

V. Output Synthesis For Operational Space Control

To realize the planned walking motion on the physical robot, MPFC outputs are tracked with an inverse-dynamics based OSC (Fig. 2C). This section describes the construction of the outputs tracked by the OSC.

A. Center of Mass Reference

Given a footstep plan from MPFC, we construct a CoM trajectory which enforces the local planarity assumption of the ALIP model by constructing the least-inclined plane passing through the current and imminent stance foot positions (Fig. 5). Letting $p = p_{n+1} - p_n$, the plane parameters are the solution to

$$\begin{bmatrix} p_x & p_y \\ -p_y & p_x \end{bmatrix} \begin{bmatrix} k_x \\ k_y \end{bmatrix} = \begin{bmatrix} p_z \\ 0 \end{bmatrix}. \tag{20}$$

After solving for k_x and k_y, we define the reference trajectory for the CoM height in the stance frame as

$$z_c(t) = H + k_x x_c(t) + k_y y_c(t). \tag{21}$$

B. Swing Foot Reference

We continuously adapt the swing foot trajectory $p_{sw}(t)$ to the updated swing-phase duration and planned next footstep position with a planning QP similar to [46]. First we generate an additional waypoint above the line connecting the initial and final foot location, following an adaptive clearance scheme, then we find a single-segment polynomial swing-foot trajectory through this waypoint.

1) Adaptive Swing Foot Clearance: Our adaptive swing foot clearance scheme (Fig. 6) picks a waypoint to be the midpoint of the swing-foot trajectory by adapting both the direction and clearance of the waypoint according to the displacement of the swing foot. This gives sufficient clearance when stepping up and down steps without unnecessarily high steps on flat ground.

Let the swing foot position at the beginning of the swing phase be $p_{sw,0}$, the target foot position for the end of swing be $p_{sw,des}$, and define $\Delta p = p_{sw,des} - p_{sw,0}$. We construct a unit

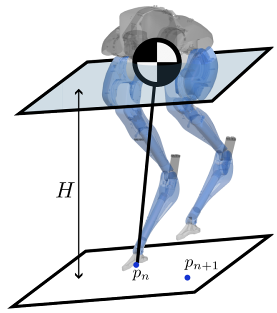


Fig. 5: To enforce the planarity assumption of the ALIP, we use OSC to drive Cassie's CoM to a virtual plane defined by current and upcoming stance foot positions.

vector $\hat{n}_p$ which is perpendicular to Δp and lies in the plane spanned by Δp and the world z axis. When Δp is small, for example when the robot is stepping in place, small variations in height estimates can lead to $\hat{n}_p$ pointing in inconsistent directions, therefore we blend $\hat{n}_p$ with the unit z-vector, $\hat{e}_z$ to get a blended direction, $\hat{n}_b$:

$$\hat{n}_b = (1 - s)\hat{e}_z + s\hat{n}_p$$

where

$$s = \text{clamp}\left(\frac{\|\Delta p\| - 0.1}{0.1}, 0, 1\right).$$

The final waypoint location is then defined as

$$p_{mid} = p_{sw,0} + \frac{1}{2}\Delta p + c_{clear}\frac{\hat{n}_b}{\|\hat{n}_b\|}$$

where $c_{clear} = c + \min(c, \Delta p_z)$ is the final swing foot clearance, and c is a tuneable parameter representing the swing foot clearance on flat ground, which we set to 15 cm in our experiments.

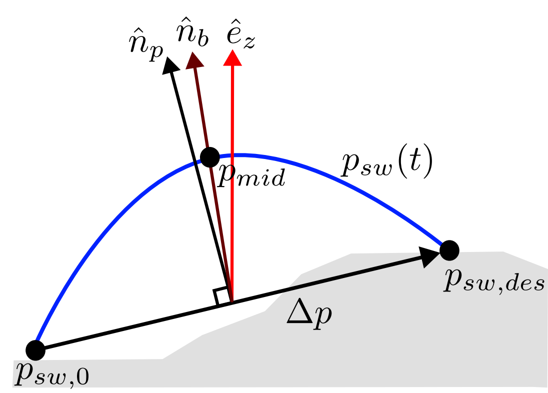


Fig. 6: Trajectory from the swing foot position at the beginning of the swing phase, $p_{sw,0}$ to the next footstep solution from MPFC, $p_{sw,des}$. We adapt the direction and clearance of the trajectory's midpoint, p_{mid}, based on the relative positions of $p_{sw,0}$ and $p_{sw,des}$ to ensure sufficient ground clearance.

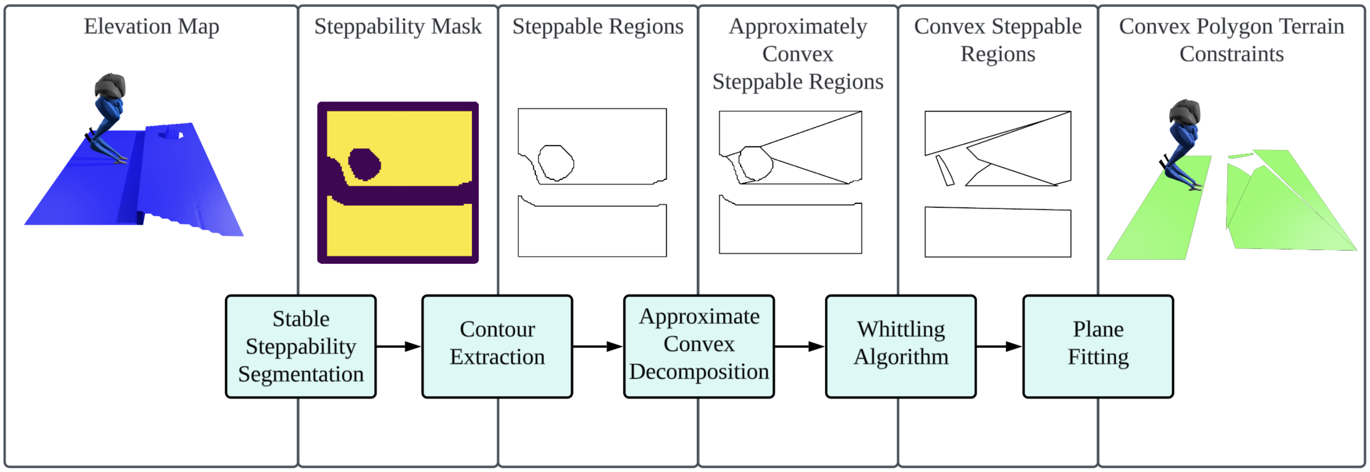


Fig. 7: Pipeline for converting an elevation map of the terrain into a set of convex polygons which can be used to plan safe footsteps. Our Stable Steppability Segmentation produces a temporally-consistent steppability mask representing a 2D overhead view of the safe terrain. We then extract the boundaries of the safe terrain as non-convex polygons, which we decompose into convex polygons using an algorithm based on approximate convex decomposition.

2) Swing foot Planning QP: After finding the desired mid-spline waypoint p_{mid}, we solve (22) to update the swing foot trajectory to the new footstep target $p_{sw,des}$ and swing phase duration T. In addition to passing through the desired midpoint and ending at the target location, we constrain the swing foot trajectory to be continuous up to acceleration with the previously planned swing foot trajectory:

$$\begin{aligned} &\text{minimize} \quad \int_0^T \ddot{p}_{sw}(t)^2 dt \\ &\text{subject to } p_{sw,k}(t_{k-1}) = p_{sw,k-1}(t_{k-1}) \quad p_{sw}(T) = p_{sw,des} \\ &\qquad \dot{p}_{sw,k}(t_{k-1}) = \dot{p}_{sw,k-1}(t_{k-1}) \quad \dot{p}_{sw}(T) = 0 \\ &\qquad \ddot{p}_{sw,k}(t_{k-1}) = \ddot{p}_{sw,k-1}(t_{k-1}) \quad \ddot{p}_{sw}(T) = 0 \\ &\qquad p_{sw}(T/2) = p_{mid} \end{aligned} \tag{22}$$

where k indexes each OSC control cycle. We transcribe (22) as a QP which optimizes over the coefficients of a polynomial representing the swing foot trajectory. By using the initial swing foot position as $p_{sw,0}$ at the beginning of the swing phase, we ensure that the trajectory starts at the initial swing foot position without needing to explicitly enforce that constraint for every control cycle.

C. Constant References

We track a constant pelvis roll and pitch of zero, and a constant swing-leg hip yaw (abduction) angle of zero. We track a commanded pelvis yaw rate from the remote control, and a swing toe angle so that Cassie's foot makes an angle of $\arctan k_x$ with the ground.

D. Ankle Torque

We add a quadratic cost to the OSC QP which penalizes the difference between MPFC ankle torque solution and the ankle torque commanded by OSC.

VI. Stable Steppability Segmentation and Convex Decomposition

Now that we have established a control framework for walking over convex polygons, we must devlop an effective pipeline for approximating the safe terrain as convex polygons online (Fig. 2A). This section introduces our proposed solution, "Stable Steppability Segmentation" (S3) and a complementary convex decomposition procedure similar to that used in [9] (Fig. 7).

S3 uses local information to classify the safety of each pixel in an elevation map, yielding a binary steppability mask of the terrain. We then perform contour extraction on this mask, and a 2D convex decomposition on the resulting steppable regions. Finally, we fit plane parameters to the resulting convex polygons using the elevation map.

Because safety criteria are local, we encourage temporal consistency by simply adding hysteresis to the classification of each pixel. We also use additional metrics beyond local planarity to determine steppability, as discussed in Section VI-A. The simplicity of our approach allows the entire pipeline from elevation mapping to publishing convex polygons to run in real time on a single CPU thread. The remainder of this section explains S3 and our accompanying convex decomposition procedure in detail.

A. Stable Steppability Segmentation

The goal of steppability segmentation is to determine where on the elevation map is safe to step. Because the segmentation determines the foothold constraints for MPFC, it is important that the segmentation algorithm is

- Robust to noise and sensor fusion artifacts
- Temporally consistent
- Computed in real time
- Appropriately conservative.

To accomplish this, we consider a setup where various safety criteria can influence whether an elevation map pixel

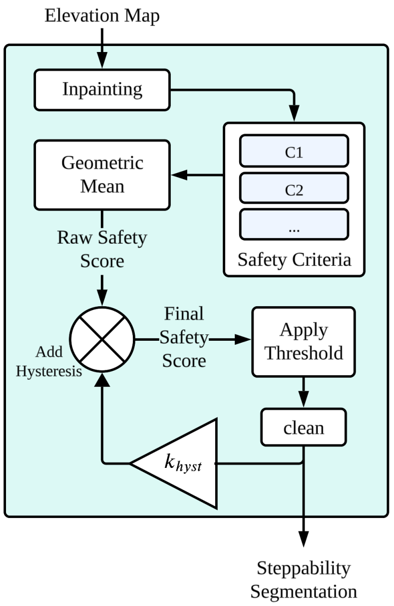


Fig. 8: Block diagram of S3, our proposed terrain segmentation approach. Safety criteria are combined into an overall safety score for each elevation map pixel, before applying hysteresis to enhance temporal consistency.

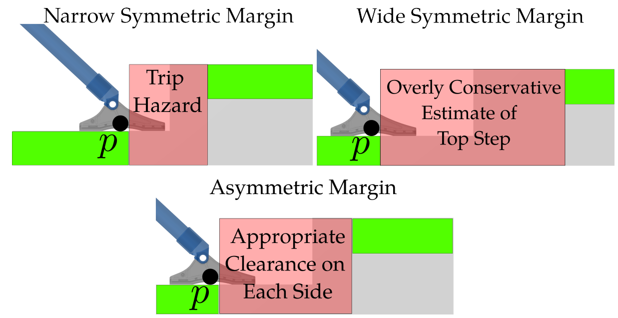


Fig. 9: Different levels of safety margin added to the terrain segmentation lead to different constraints on the footstep position, p. Plane segmentation does not handle asymmetric treatment of edges when adding safety margin, leading to undue conservatism. The desire to specifically avoid stepping below an edge motivated our curvature based safety criterion.

is safe to step on (Fig. 8). A safety criteria is a function which maps the elevation map to a pixel-wise "safety score" in the range $[0, 1]$, where 1 is completely safe, and 0 is unsafe. These safety criteria are combined via their geometric mean to yield an overall safety score. Robustness to noise and artifacts is accomplished via inpainting and filtering in each safety criteria computation, for example median filtering the elevation map to remove outliers. Temporal consistency is achieved through adding hysteresis to the overall safety score based on the previuous segmentation. Realtime computation is achieved through the simplicity of our algorithm, and the small size of our elevation map. Because the safety criteria are meant to be local to each pixel, they could also be parallelized on the GPU for large elevation maps. The next subsection outlines what we mean by "appropriately conservative" and introduces a curvature-based safety criterion which accomplishes this goal.

1) Curvature Safety Criterion: During our original experiments [9], we used a plane segmentation approach [32], and had difficulty picking a safety margin which would result in Cassie not taking huge steps over ledges, and also not tripping over curbs (Fig. 9).

This experience taught us that we want to keep more distance from a ledge when we are stepping near the bottom of the ledge, than the top of the ledge. An intuitive metric for rejecting terrain which is not "reasonably flat" which captures this effect is measuring whether terrain is lower than its surroundings. Treating the elevation map as an image, this looks like applying a kernel that compares the height of each pixel to the average of the pixels around it:

$$\begin{pmatrix} 1/8 & 1/8 & 1/8 \\ 1/8 & -1 & 1/8 \\ 1/8 & 1/8 & 1/8 \end{pmatrix}. \tag{23}$$

This particular kernel is, in fact, a Laplacian kernel, used to compute the curvature of an image, meaning we can use standard image processing tools to efficiently calculate this component of the safety score. Letting E be the elevation map, the curvature criterion is computed via Eq. (24),

$$c_{curve} = \min(1, \exp(-\alpha_c \text{LoG}(E))) \tag{24}$$

where LoG is the Laplacian of Gaussian filter, which convolves the elevation map first with a Gaussian filter, then takes the Laplacian. The pixel-wise exponential $\exp(-\alpha_c \text{LoG}(E))$ maps regions of positive curvature to the interval $(0, 1]$, with the score exponentially approaching 0 as the curvature increases. The scale factor α_c can be used to tune how aggressively positive curvature is punished. Finally, we take the min of the criterion with 1 to ensure that no bonus points are awarded for negative curvature.

2) Inclination Safety Criterion: The inclination safety criterion treats steep terrain as unsafe, by considering the magnitude of the z component of the surface normal at each elevation mapping pixel. We estimate the normal using the covariance matrix of the positions around each pixel [17], then square the z component to yield the inclination safety criterion,

$$c_{inc} = n_z(E)^2. \tag{25}$$

To give context for how c_{inc} classifies terrain in practice, we pick 0.7 as the final safety threshold for S3. For a pixel which is otherwise considered safe, this means a slope greater than about 33° is unsafe, since $\cos^2(33°) = 0.7$.

3) Combining Safety Criteria: The final safety score is the geometric mean of the score for each criteria, plus a hysteresis value for all pixels classified safe in the previous frame. Pixels with a final score above some threshold are considered safe. The result is a binary image is postprocessed to give a 2D view of the safe terrain around the robot. Letting k index the time series of segmentations, the S3 output is given by

$$S_k = \text{clean}\left(\left[\left(\prod_{i=1}^{M} c_i(E_k)\right)^{1/M} + k_{hyst} S_{k-1}\right] > k_{safe}\right)$$

where M is the total number of safety criteria, and $\text{clean}(S) = \text{open}(\text{close}(\text{erode}(S)))$. This adds a safety margin and removes any thin holes or protrusions.

B. Convex Planar Decomposition

Finally, we convert the binary steppability mask into a set of convex planar polygons. We identify connected components of steppable terrain from the mask, and extract their outlines as 2D polygons. In general, these are non-convex polygons with holes (caused, for example, by small obstacles or other unsteppable areas), but we require convex foothold constraints for the MPFC. We use a two stage process to find a set of convex polygons whose union is an inner approximation these non-convex polygons. This avoids creating many small triangles like an exact convex decomposition would, leading to fewer mixed integer constraints in the MPFC.

First, we perform approximate convex decomposition (ACD)[18] on each polygon. ACD returns a decomposition of the original region into polygons which are d-approximately convex, where d the depth of the largest concave feature.

After filtering out polygons with area less than 0.05 m^2, we find a convex inner-approximation of these nearly convex polygons with a greedy approach we name the whittling algorithm (Algorithm 1), after the way it makes incremental cuts to the polygon. We initialize the output polygon, $\mathcal{P}$ as the convex hull of the original polygon, then take $\mathcal{P}$ to be the intersection of itself with greedily chosen half-spaces until no vertices of the original polygon are contained in the interior $\mathcal{P}$. To reduce the number of cuts we make, we initially sort the vertices by their distance to the boundary of $\mathcal{P}$.

Algorithm 1 Whittling Algorithm

Require: Input polygon vertices $V = \{v_0 \dots v_n\}$
  procedure WHITTLE(V)
    $\mathcal{P} \leftarrow \text{ConvexHull}(V)$
    Sort v_i by distance to $\partial\mathcal{P}$
    for all v_i **do**
      if $v_i \in \text{Interior}(\mathcal{P})$ **then**
        $H = \text{MakeCut}(v_i, V)$
        $\mathcal{P} \leftarrow \mathcal{P} \cap H$
    return $\mathcal{P}$

$\text{MakeCut}(V, v_i)$ is a nonlinear program inspired by maximum margin classification [48] which finds a such that the half-space $H = \{x \mid a^T(x - v_i) \leq 0\}$ contains as much of V as possible:

$$a = \arg\min_a \sum_{j \neq i} \max(a^T(v_i - v_j), 0)^2$$
$$\text{subject to } \|a\|_2^2 = 1 \tag{26}$$

We solve (26) using a custom gradient-based solver, which we detail in Appendix C. Using the normal of the closest face of $\mathcal{P}$ to v_i provides a high-quality initial guess for the solver.

To fit these polygons to the terrain, we project the 2D vertices onto the elevation map to recover the 3D position of each vertex. We then use least-squares to find the best fit plane to these vertices, yielding our final polygon representation.

VII. Experimental Setup

This section explains the practical implementation of MPFC and our perception stack. The full set of parameters used for S3, MPFC, and their supporting algorithms is given in Appendix D. The entire perception and control system consists of six processes communicating between three separate computers. Cassie's target PC runs a Simulink Real-Time application which publishes joint positions and velocities and IMU data, at 2kHz, and subscribes to torque commands. Communication between the target PC and Cassie's onboard Intel NUC occurs over UDP. The NUC runs the state estimator and a torque publisher to communicate with the target PC, and the operational space control process, which also includes the swing foot planner.

The perception stack and MPFC are run on an off-board ThinkPad p15 Laptop with an 8-core, 2.3 GHz Intel 1180H processor and 24 GB of RAM. The perception stack has a dedicated polling thread for the Intel RealSense, and a second thread for elevation mapping, terrain segmentation, and convex decomposition. We establish communication between the state estimator, operational space controller, torque publisher, perception stack, and MPFC using LCM [49] for low latency.

Except for the low-level target PC, all processes use the Drake systems framework [50] to drive their operation. We solve the MPFC problem using Gurobi, and the OSC QP using FCCQP [51]. Open source code for all of our contributed components will be provided in `dairlib`[4].

A. State Estimation

We use the contact-aided invariant extended Kalman filter developed by Hartley et al. [52] to estimate the pose and velocity of the floating base.

B. RealSense D455 Depth Camera

The RealSense is mounted to Cassie's pelvis, pointed downward toward the terrain in front of the robot. We use the `librealsense2` C++ to subscribe to RealSense frames via a dedicated polling thread, with the perception stack thread accessing these frames through a shared buffer. We apply a decimation filter to reduce the density of the point cloud.

C. Robot-Centric Elevation Mapping

We use the probabilistic elevation mapping framework developed by Fankhauser et al. in [31] to construct a robot-centric elevation map of the terrain. This framework represents the terrain as a grid around the robot, with the height of each cell updated by point cloud measurements through a Kalman filter. Because Cassie's legs are visible in the camera frame, we crop out any points inside bounding boxes around Cassie's leg links. To correct for state estimate z-drift, before each point cloud update, we adjust the height of the elevation map by adding the height difference between the elevation map and the current stance foot.

[4]https://github.com/DAIRLab/dairlib

VIII. Results

The perception and control architecture presented in this paper enables Cassie to walk over previously unseen terrain by identifying safe terrain and planning stabilizing footsteps subject to non-convex terrain constraints in real time. This section presents hardware experiments to show these capabilities and support our key claims. First, we showcase underactuated walking over discontinuous terrain on hardware with the Cassie biped. We report consistent sub-10-millisecond solve times for MPFC. Next, we show the improved temporal consistency and faster run time of S3 compared to explicit plane segmentation. Finally we summarize the capabilities of MPFC and S3 as a complete system, highlighting the performance improvements as a result of the contributions in this paper.

A. Walking on Discontinuous Terrains

We showcase Cassie walking over discontinuous and unstructured terrains using our perception and control stack. A single trial traversing steps, a curb, and a grass hill is shown in Fig. 12. Additional trials showcasing the versatility of S3 + MPFC are shown in the supplemental video.

B. Controller Solve Times

To support our claims of faster than 100 Hz MPFC solve times, we compile solve times across 11:17 minutes of walking data from three experiments on the brick steps shown in Fig. 12a. We show the distribution of solve times from these logs in Fig. 10. This data uses a planning horizon of $N = 3$ footsteps, including the initial stance phase. We give summary statistics of MPFC solve times in Table I. The maximum solve time observed was 12.6 ms, with 99.9% of solves taking less than 7.7 ms.

TABLE I: MPFC Solve-Time Statistics (134,654 Solves)

Mean	Median	99.9th Percentile	Maximum
0.0022	0.0020	0.0077	0.0126

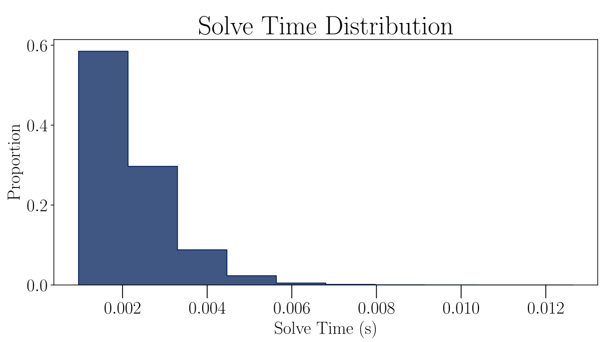


Fig. 10: Solve-time distribution of MPFC.

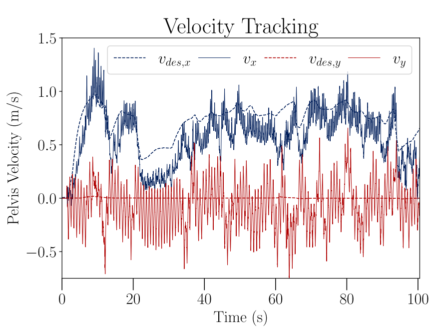
Fig. 11: The environments used to collect data for benchmarking the perception stack's performance. From top to bottom the terrains are **Lab**, **Brick Steps**, and **Grass**

C. Perception Stack Evaluation

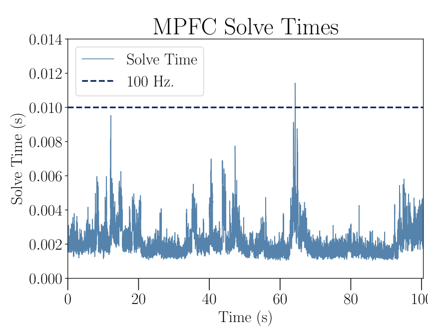
This section supports our claims of improved temporal consistency and faster run times compared to explicit plane segmentation. We use elevation mapping data from three terrains to evaluate the performance of our segmentation approach (Fig. 11). The **Lab** terrain establishes a baseline performance for each method in an ideal environment, where state estimate z-drift is the only potential challenge. The **Brick Steps** terrain features a set of brick steps where the bricks have settled over time, making the steps uneven, and unlikely to be segmented into a single plane by plane-segmentation methods. Similarly, the **Grass** terrain is challenging for plane segmentation approaches because our stance foot drift-correction conflicts with the height of the point cloud, introducing artifacts into the elevation map.

We use the plane segmentation module developed by Miki at al. in the `elevation_mapping_cupy` software package [32] with default parameters (hereafter labeled EM_cupy) as a plane segmentation baseline. This algorithm has a similar structure to S3, starting by filtering the elevation map, classifying each cell as steppable or not, and then (unlike S3), trying to segment the steppable cells into planes. Each connected component of steppable terrain is checked for planarity, and if it fails, RANSAC [53] is used to find smaller planes within that connected component. The authors of [32] also provide the option to disable RANSAC plane refinement, instead accepting or rejecting each connected component of steppable terrain

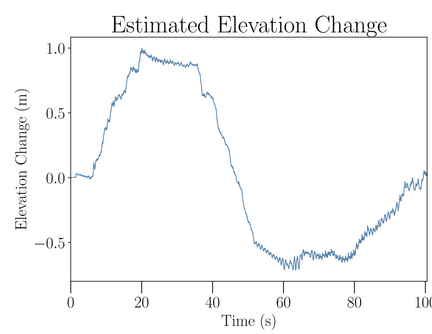
(a) Motion tiles showing Cassie ascending and descending steps, stepping over a curb onto the grass, and walking up a grassy slope in one continuous walking trial.



(b) Plot of the velocity tracking performance of the robot using our control stack.



(c) MPFC Solve times during the above trial.



(d) Plot of the elevation change over the trial, estimated from the onboard state estimator.

Fig. 12: Cassie Walks on unstructured terrain using our proposed perception and control stack, climbing and descending a set of steps, stepping over a curb, and walking up a grassy hill. Our perception stack identifies safe terrain and decomposes it into convex polygons online while the robot is walking at over 0.5 m/s. Footage can be viewed in the supplemental video.

TABLE II: Comparison of S3 and Plane Segmentation Baselines

	S3 (Ours)	**EM_cupy**	**EM_cupy_NR**
Steppability Criteria	Curvature, Inclination	Roughness, Inclination	Roughness, Inclination
Incorporates History	Yes	No	No
Plane Refinement	None	RANSAC [53]	Reject regions with slope $\geq 30°$
Inpainting Method	Navier-Stokes [54]	Least Neighboring Value	Least Neighboring Value

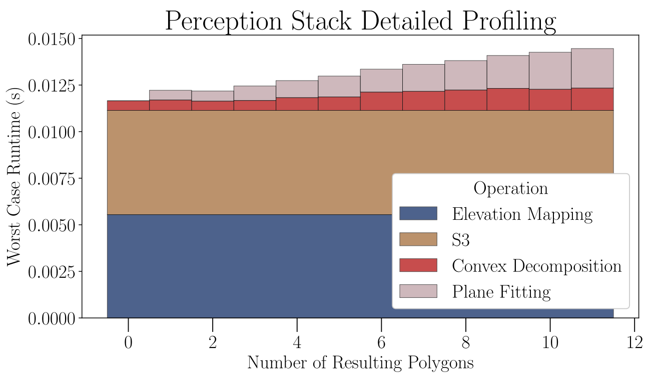


Fig. 13: Detailed profiling of our perception stack, showing the worst-case runtime of each component. "Convex Decomposition" includes all steps necessary to convert the steppability mask from S3 into 2D convex polygons. S3 and Plane Fitting use unoptimized python implementations, providing an avenue for further run time improvements. Profiling is performed on the ThinkPad p15 laptop used for hardware experiments.

in its entirety based on the estimated surface normal. We also test this variant, henceforth labeled EM_cupy_NR, where NR denotes "No RANSAC" or "No Refinement." Because EM_cupy_NR is identical to the default EM_cupy algorithm except for lacking a global planarity requirement on steppable regions, these results will support our argument that explicit plane segmentation is particularly brittle. Table II summarizes the differences between S3 and the baselines.

1) Computation Time: We support the claim that our perception stack is real-time with detailed profiling. To show that the entire pipeline is real-time, we profile the pipeline on 90 seconds of walking data from **Brick Steps**. We report the worst-case observed computation time for each step of the perception stack in Fig. 13, breaking the convex decomposition steps out by the number of resulting polygons. We note that the worst-case cumulative compute times stay below the 33 ms required for keeping up with the RealSense frame rate.

As a benchmark against other segmentation approaches, we compare the run times of S3, EM_cupy, and EM_cupy_NR for each test environment in Fig. 14, and find S3 to be the most consistent, with the lowest worst-case computation time.

2) Temporal Consistency: We measure the temporal consistency of each segmentation approach via the intersection over union (IoU) of consecutive segmentations. IoU measures the ratio of pixels labeled as safe in both segmentation frames to the number of pixels labeled safe in either frame. Because data is lost when the elevation map moves relative to the world, we restrict the IoU computation to pixels which are present in both frames. A frame-to-frame IoU of 1 represents perfect temporal consistency, and 0 represents no overlapping safe terrain between segmentations.

The distributions of frame-to-frame IoU for one minute of walking data in each environment are shown in Fig. 15. Our approach consistently achieves an IoU close to 1 across environments, representing excellent temporal consistency, while EM_cupy struggles with temporal consistency even in the lab setting, due to imperfect depth estimation and artifacts from sensor fusion.

EM_cupy_NR is more temporally consistent than EM_cupy, but using the estimated normal as a final check on the steppability of a region makes the algorithm susceptible to outlier frames where large regions of terrain drop out. This highlights that requiring regions to be globally planar is mostly responsible for the poor temporal consistency of EM_cupy. The segmentation output from each algorithm at 1 second intervals is shown in Fig. 16, and animations of the segmentation state are shown in the supplemental video.

D. Summary of Capabilities

We briefly summarize the capabilities of our perception and control architecture, and the performance improvements made possible by MPFC and S3. Compared to our deadbeat ALIP footstep planner based on [28], MPFC is more robust, even on hard, flat surfaces, due to the inclusion of workspace constraints, ankle torque, and step timing adaptation. MPFC and S3 also enable Cassie to walk up and down steps and curbs up to 16 cm tall when each step is deep enough to use foot placement for stabilization. In contrast to our original perception implementation in [9], using S3 for terrain segmentation allows the robot to walk continuously with perception in the loop due to its temporal consistency and robustness to state-estimate drift artifacts. This is the case even on grass, where proprioceptive and exteroceptive ground height estimates conflict. We discuss limitations of our stack in Section IX-C.

IX. Discussion

Here we discuss implementation details and limitations which we hope will be useful to those implementing similar pipelines in the future.

A. S3 Hysteresis Behavior

We chose a hysteresis value of 0.6, with a safety threshold of 0.7. This high level of hysteresis enhanced the robustness of the perception stack without any tradeoffs for the terrains we tested on. Because we apply a safety margin to the segmentation *after* the hysteresis step using an erosion filter, pixels adjecent to unsafe terrain will have a lower effective threshold for being rejected in the following segmentation.

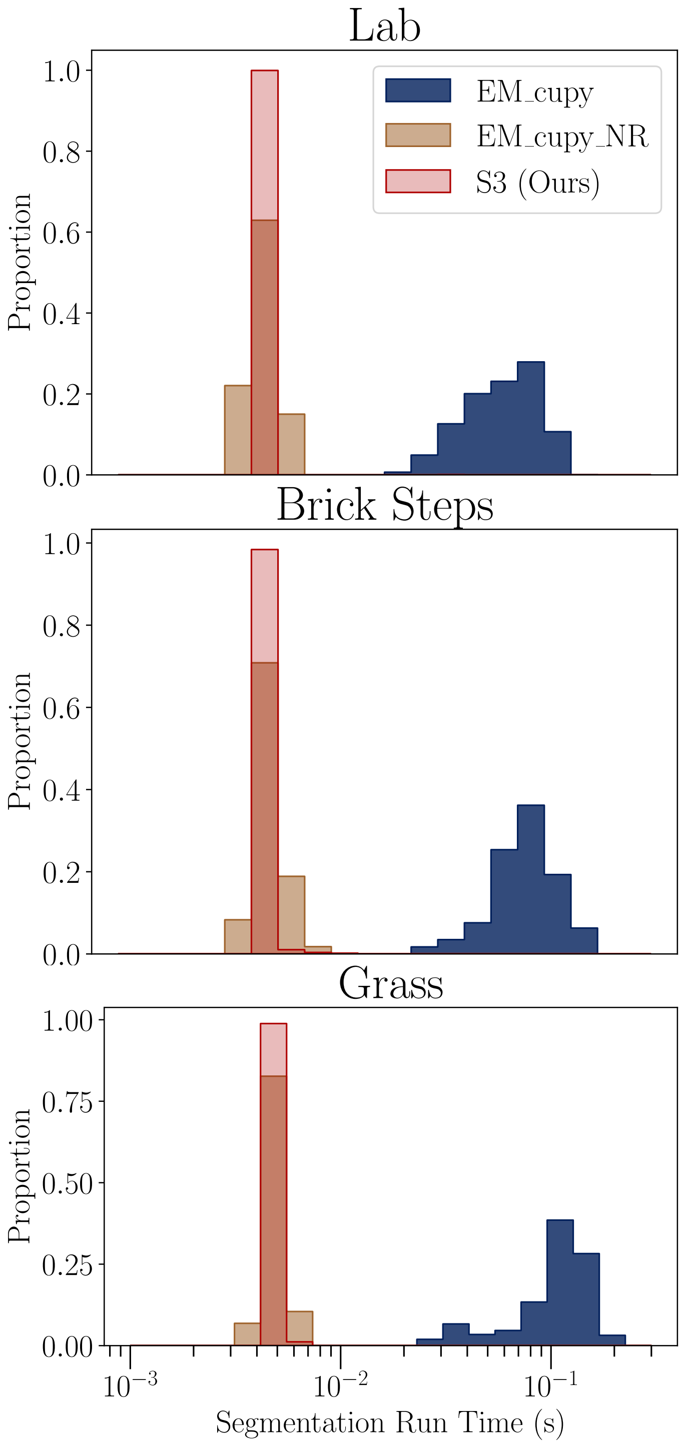


Fig. 14: Histogram of the run time of each segmentation algorithm over 60 seconds of elevation mapping data from each test environment. S3, is the most consistent and has a similar median run time to EM_cupy_NR. EM_cupy is the slowest, with a highly variable run time, due to the repeated use of RANSAC to refine the plane segmentation.

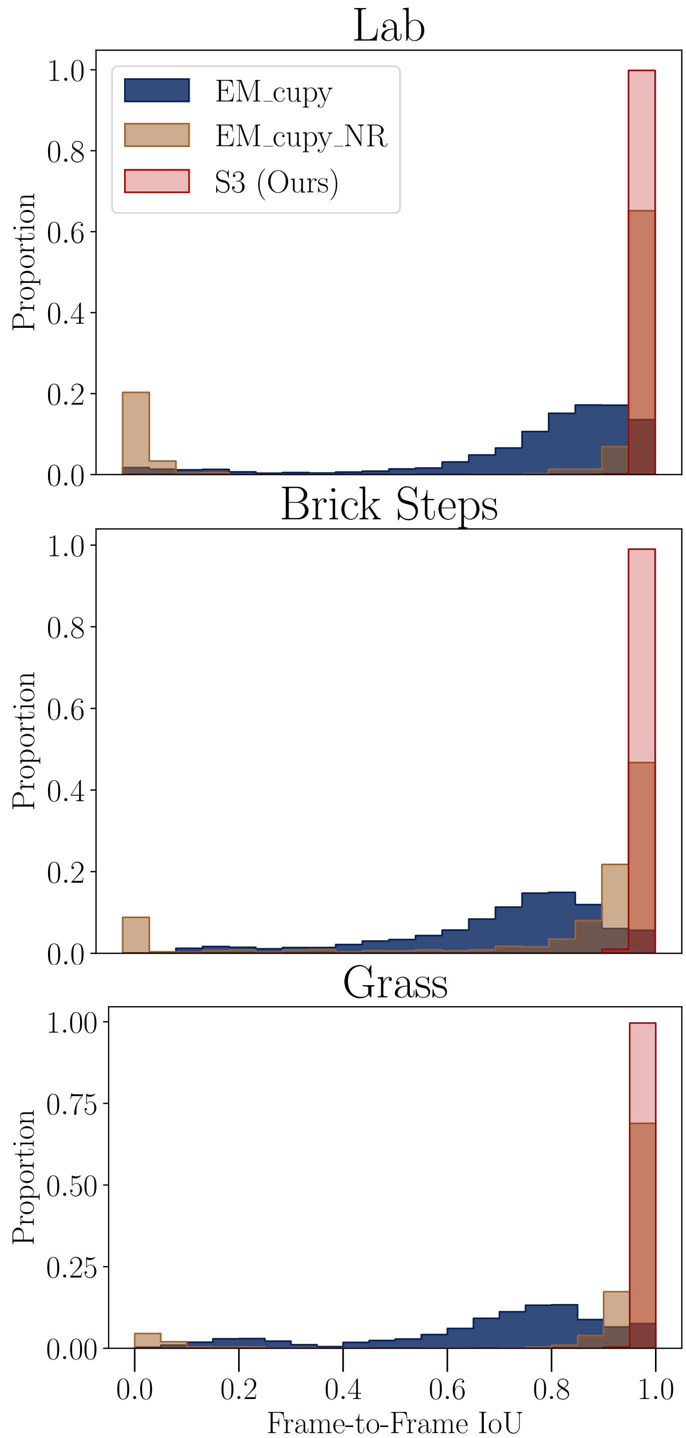


Fig. 15: Histogram of the frame-to-frame IoU of the safe terrain segmentation over 60 seconds of elevation mapping data from each test environment. Our method reliably achieves a frame-to-frame IoU close to 1 across environments, representing excellent temporal consistency.

This results in a behavior where unsafe regions are "seeded" by their least safe pixels, and grow over time until they reach sufficiently safe terrain. When walking more dynamically or on more challenging terrains, it may be desirable to decrease the hysteresis so that this growth process happens faster. For completeness, to show the effect of hysteresis on temporal consistency, we show the distribution of frame-to-frame IoU values for varying levels of hysteresis on the **Brick Steps** in Fig. 17.

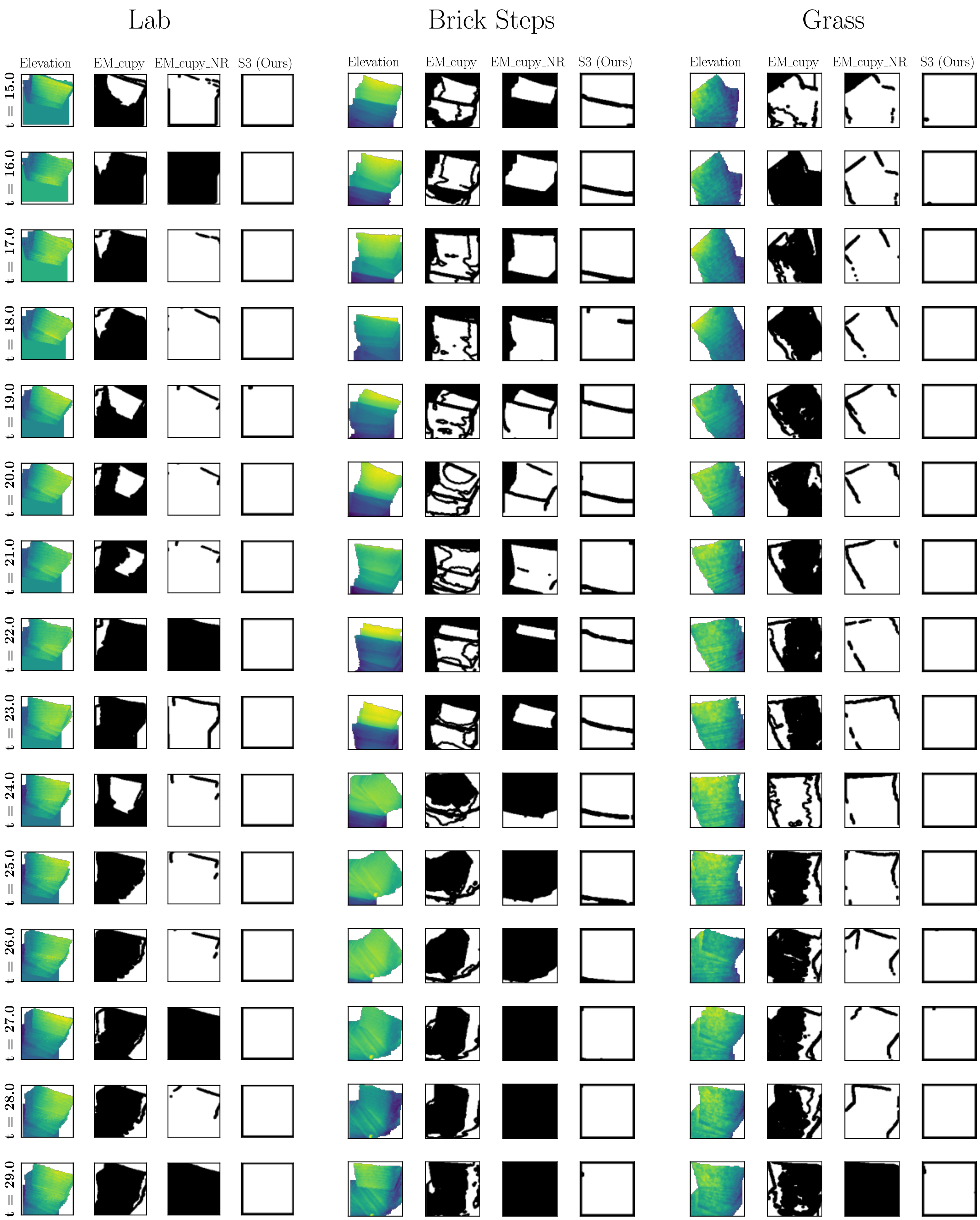


Fig. 16: Tiles showing the output of each segmentation method for each evaluation environment at 1 second intervals. In the **Lab** and **Grass** environments, the use of Navier-Stokes based inpainting allows S3 to correctly identify the entire elevation map as steppable. Unlike the baselines, the S3 segmentation does not experience "flickering" of the steppable terrain. Animations of these segmentation results can be seen in the supplemental video.

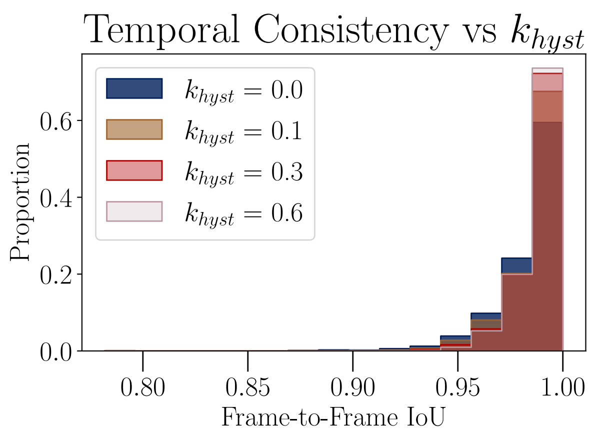


Fig. 17: Frame-to-Frame IoU of the S3 terrain segmentation results with varying levels of hysteresis, evaluated on the **Brick Steps** data.

B. Implementation Details

This section discusses design choices which were motivated by the need to handle edge cases and increase the practical robustness of our implementation.

1) Inpainting: Because we only use a single depth camera, whose field of view does not span the entire diagonal of the elevation map, we sometimes lack elevation data for terrain near robot, especially when walking diagonally, leaving the question of how S3 should classify these cells. Classifying these cells as unsafe can cause the robot to fall when the operator does not have a real-time view of the map, and drives the robot toward the unmapped regions. We solve this by inpainting the missing portions of the elevation map using the Navier-Stokes based method implemented in OpenCV [54], before inputting the elevation map to S3. This method matches the value and gradient of the image at the boundary of the missing terrain. While adding additional depth sensors would be ideal, our solution highlights a general theme: due to Cassie's underactuation, we found it most robust to resolve ambiguous design decisions by favoring steppability.

2) ALIP State Estimation: Impacts during touchdown and compliance in Cassie's hip-roll joints can cause undesirable spikes and oscillations in the lateral floating base velocity estimate, and therefore the angular momentum estimate. We increase our controller's robustness to these issues by using a Kalman filter with ALIP dynamics to smooth our estimate of the ALIP state during single support. We use (1) for the dynamics model, with full-state measurement, and assume a much higher measurement noise for the angular momentum than for the CoM position.

3) Foostep Height Lookup: Before sending a footstep command to the OSC, we refine the vertical footstep position by looking up the height of the planned footstep position on a smoothed, inpainted copy of the elevation map. Because the ALIP dynamics do not depend on the vertical footstep position, we do not need to propogate this adjustment back to MPFC.

C. Limitations and Failure Modes

The failure mode most insufficiently addressed by this work was slipping. If the robot did not fall immediately due to slipping, the slip could introduce large errors into the elevation map, making recovery difficult. The likelihood of slips could be reduced by adaptively constraining the workspace of the ALIP model (effectively the friction cone) in situations where slips are likely. Because slips were most common when walking up or down steps, another possibility would be to narrow the workspace constraint when stepping up or down. The fast swing foot motions and moderate CoM height changes experienced walking on steps also pushed the boundaries of what could be tracked with our OSC framework, and more challenging terrains will likely require considering more expressive dynamics at the MPC level.

X. Conclusion And Future Work

We present a complete perception and control stack for underactuated bipedal walking on rough terrain. We formulate Model Predictive Footstep Control as a single MIQP which can be solved at over 100 Hz. to stabilize walking over discontinuous terrain without a pre-specified foothold sequence. Motivated by the brittleness of plane segmentation for safe terrain classification, we develop Stable Steppability Segmentation, a simple algorithm for temporally consistent safe terrain segmentation, and a complementary convex polygon decomposition algorithm for generating foothold constraints online. We demonstrated our proposed perception and control stack on the underactuated Cassie biped through outdoor experiments. Future work will consider more expressive models than the ALIP, to increase the robustness of the controller and allow bipeds to walk on shallow footholds and execute large step-to-step height changes.

Acknowledgements

We thank the DAIR Lab, especially Minku Kim, Hien Bui, Leon Kim, Bruke Baraki, Bibit Bianchini and Grey Sarmiento, for the time they have dedicated to Cassie experiments.

References

[1] "Supplemental Video (short)." [Online]. Available: https://youtu.be/eCOD1bMi638

[2] "Supplemental Video (full)." [Online]. Available: https://youtu.be/tPiFMXeWW0o

[3] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3D dynamic walking on stepping stones with control barrier functions," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. Las Vegas, NV, USA: IEEE, Dec. 2016, pp. 827–834.

[4] Z. Xiang, V. Paredes, and A. Hereid, "Adaptive Step Duration for Precise Foot Placement: Achieving Robust Bipedal Locomotion on Terrains with Restricted Footholds," Mar. 2024.

[5] M. Dai, X. Xiong, and A. Ames, "Bipedal Walking on Constrained Footholds: Momentum Regulation via Vertical COM Control," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 10 435–10 441.

[6] L. Krishna, U. A. Mishra, G. A. Castillo, A. Hereid, and S. Kolathaya, "Learning Linear Policies for Robust Bipedal Locomotion on Terrains with Varying Slopes," in *2021 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), Sep. 2021, pp. 5159–5164.

[7] X. Xiong and A. Ames, "SLIP Walking Over Rough Terrain via H-LIP Stepping and Backstepping-Barrier Function Inspired Quadratic Program," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2122–2129, Apr. 2021.

[8] J. Shim, C. Mastalli, T. Corbères, S. Tonneau, V. Ivan, and S. Vijayakumar, "Topology-Based MPC for Automatic Footstep Placement and Contact Surface Selection," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. London, United Kingdom: IEEE, May 2023, pp. 12 226–12 232.

[9] B. Acosta and M. Posa, "Bipedal Walking on Constrained Footholds with MPC Footstep Control," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, Dec. 2023, pp. 1–8.

[10] Z. Gu, Y. Zhao, Y. Chen, R. Guo, J. K. Leestma, G. S. Sawicki, and Y. Zhao, "Robust-Locomotion-by-Logic: Perturbation-Resilient Bipedal Locomotion via Signal Temporal Logic Guided Model Predictive Control," Mar. 2024.

[11] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS International Conference on Humanoid Robots*, Nov. 2014, pp. 279–286.

[12] G. Gibson, O. Dosunmu-Ogunbi, Y. Gong, and J. Grizzle, "Terrain-Adaptive, ALIP-Based Bipedal Locomotion Controller via Model Predictive Control and Virtual Constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 6724–6731.

[13] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous Contact, Gait, and Motion Planning for Robust Multilegged Locomotion via Mixed-Integer Convex Optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, Jul. 2018.

[14] N. Fey, R. J. Frei, and P. M. Wensing, "3D Hopping in Discontinuous Terrain Using Impulse Planning with Mixed-Integer Strategies," *IEEE Robotics and Automation Letters*, pp. 1–8, 2024.

[15] Y. Ding, C. Li, and H.-W. Park, "Kinodynamic Motion Planning for Multi-Legged Robot Jumping via Mixed-Integer Convex Program," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 3998–4005.

[16] T. Corbères, C. Mastalli, W. Merkt, I. Havoutis, M. Fallon, N. Mansard, T. Flayols, S. Vijayakumar, and S. Tonneau, "Perceptive Locomotion through Whole-Body MPC and Optimal Region Selection," May 2023.

[17] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive Locomotion through Nonlinear Model Predictive Control," Aug. 2022.

[18] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polygons," *Computational Geometry*, vol. 35, no. 1, pp. 100–123, Aug. 2006.

[19] D. Calvert, B. Mishra, S. McCrory, S. Bertrand, R. Griffin, and J. Pratt, "A Fast, Autonomous, Bipedal Walking Behavior over Rapid Regions," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, Nov. 2022, pp. 24–31.

[20] R. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, *Footstep Planning for Autonomous Walking Over Rough Terrain*, Jul. 2019.

[21] 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, 2003, pp. 1620–1626 vol.2.

[22] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taix, and A. Del Prete, "SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 6604–6610.

[23] D. Song, P. Fernbach, T. Flayols, A. D. Prete, N. Mansard, S. Tonneau, and Y. J. Kim, "Solving Footstep Planning as a Feasibility Problem Using L1-Norm Minimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5961–5968, Jul. 2021.

[24] F. Risbourg, T. Corbères, P.-A. Léziart, T. Flayols, N. Mansard, and S. Tonneau, "Real-time Footstep Planning and Control of the Solo Quadruped Robot in 3D Environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 12 950–12 956.

[25] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, Oct. 2001, pp. 239–246 vol.1.

[26] X. Xiong and A. Ames, "3-D Underactuated Bipedal Walking via H-LIP Based Gait Synthesis and Stepping Stabilization," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2405–2425, Aug. 2022.

[27] R. Griffin, J. Foster, S. Pasano, B. Shrewsbury, and S. Bertrand, "Reachability Aware Capture Regions with Time Adjustment and Cross-Over for Step Recovery," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, Dec. 2023, pp. 1–8.

[28] Y. Gong and J. Grizzle, "One-Step Ahead Prediction of Angular Momentum about the Contact Point for Control of Bipedal Locomotion: Validation in a LIP-inspired Controller," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 2832–2838.

[29] O. Dosunmu-Ogunbi, A. Shrivastava, G. Gibson, and J. W. Grizzle, "Stair Climbing using the Angular Momentum Linear Inverted Pendulum Model and Model Predictive Control," Jul. 2023.

[30] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov. 2015, pp. 881–888.

[31] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic Terrain Mapping for Mobile Robots With Uncertain Localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, Oct. 2018.

[32] T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, and M. Hutter, "Elevation Mapping for Locomotion and Navigation using GPU," Apr. 2022.

[33] M. Fallon and M. Antone, "Plane Seg – Robustly and Efficiently Extracting Contact Regions from Depth Data," 2019. [Online]. Available: https://github.com/ori-drs/plane_seg

[34] D. Wisth, M. Camurri, and M. Fallon, "Vilens: Visual, inertial, lidar, and leg odometry for all-terrain legged robots," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 309–326, 2023.

[35] T. Bin, J. Yao, T. Lun Lam, and T. Zhang, "Real-Time Polygonal Semantic Mapping for Humanoid Robot Stair Climbing," in *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, Nov. 2024.

[36] S. McCrory, B. Mishra, R. Griffin, J. Pratt, and H. E. Sevil, "Bipedal navigation planning over rough terrain using traversability models," in *SoutheastCon 2023*, 2023, pp. 89–95.

[37] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, "TAMOLS: Terrain-Aware Motion Optimization for Legged Systems," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3395–3413, Dec. 2022.

[38] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning," in *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, Jul. 2021.

[39] H. Duan, A. Malik, J. Dao, A. Saxena, K. Green, J. Siekmann, A. Fern, and J. Hurst, "Sim-to-Real Learning of Footstep-

Constrained Bipedal Dynamic Walking," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 10 428–10 434.
[40] H. Duan, B. Pandit, M. S. Gadde, B. J. van Marum, J. Dao, C. Kim, and A. Fern, "Learning vision-based bipedal locomotion for challenging terrain," *arXiv preprint arXiv:2309.14594*, 2023.
[41] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control," *Science Robotics*, vol. 9, no. 86, p. eadh5401, 2024. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adh5401
[42] S. Yu, N. Perera, D. Marew, and D. Kim, "Learning generic and dynamic locomotion of humanoids across discrete terrains," *arXiv preprint arXiv:2405.17227*, 2024.
[43] Boston Dynamics, "Starting on the Right Foot with Reinforcement Learning." [Online]. Available: https://bostondynamics.com/blog/starting-on-the-right-foot-with-reinforcement-learning/
[44] K. Ogata, *Modern Control Engineering*, 4th ed. USA: Prentice Hall PTR, 2001.
[45] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control using divergent component of motion," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2600–2607.
[46] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Walking Control Based on Step Timing Adaptation," Mar. 2020.
[47] 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*, May 2013, pp. 3103–3109.
[48] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.
[49] 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*, Oct. 2010, pp. 4057–4062.
[50] Russ Tedrake and the Drake Development Team, "Drake: Model-Based Design and Verification for Robotics," 2019.
[51] B. Acosta, "FCCQP: A Whole Body Control QP Solver with Full Friction Cones." [Online]. Available: https://github.com/Brian-Acosta/fcc_qp/
[52] 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, Mar. 2020.
[53] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:12349413
[54] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

APPENDIX

A. Lateral Reset Map Adjustment

Because B_{ds} is decoupled in x and y, our MPFC implementation assumes $f(t) = 1$ for the lateral compenents of the ALIP state, which corresponds to instantaneous weight transfer at the beginning of double-stance. In this case, (6) evaluates to

$$B_{ds} = A^{-1}(A_r - I)B_{CoP}. \tag{27}$$

This helps the robot more closely track the desired step width by compensating for systematic error in swing foot tracking. The robot consistently steps with a wider step width than the commanded footstep position. Due to compliance and backlash in Cassie's hip roll joints, we are unable to raise the PD gains on the lateral swing foot position beyond the values in Table IV. Adjusting the reset map to assume instantaneous weight transfer acts as a feedforward correction by increasing the model's estimate of how much momentum will be absorbed by a larger lateral footstep size. To calculate the final value of B_{ds}, we take the inner 2×2 submatrix, which corresponds to the coronal plane, from (27), and the 4 corner values from (7), which correspond to the sagittal plane.

B. Constructing the Desired-Velocity Subspace

Here we derive the affine subspace of P2 orbits which achieve a given desired velocity v_{des}. We will define the projection matrices Π_0 and Π_1, and the offsets d_0 and d_1, and then for the general case, we have that

$$\Pi_{n+2} = \Pi_n$$
$$d_{n+2} = d_n$$

We start by unrolling the s2s dynamics over two footsteps, and applying the P2 orbit constraint $x_2 = x_0$:

$$x_0 = A_{s2s}^2 x_0 + A_{s2s}B_{s2s}\delta p_0 + B_{s2s}\delta p_1. \tag{28}$$

We substitute the velocity constraint $(\delta p_0 + \delta p_1) = 2T_{s2s}v_{des}$ into (28) and solve for x_0:

$$x_0 = G(A_{s2s}B_{s2s} - B_{s2s})\delta p_0 + 2T_{s2s}GB_{s2s}v_{des} \tag{29}$$

where $G = (I - A_{s2s}^2)^{-1}$. From (29) we have a definition of the desired velocity subspace as an offset based on v_{des} and the span of $L_0 = G(A_{s2s} - I)B_{s2s} \in \mathbb{R}^{4\times 2}$. We convert this to the desired form (16) by left-multiplying with Π_0, a projection matrix to the orthogonal complement of the range of L_0. Because Π_0 maps $L_0\delta p_0$ to zero for any δp_0 by construction, this leaves us with

$$\Pi_0 x_0 = 2\Pi_0 T_{s2s}GB_{s2s}v_{des} \tag{30}$$

so $d_0(v_{des}) = 2T_{s2s}GB_{s2s}v_{des}$. To find Π_1, we use

$$\begin{aligned} x_1 &= A_{s2s}x_0 + B_{s2s}\delta p_0 \\ \therefore\ x_1 &= A_{s2s}(L_0\delta p_0 + d_0) + B_{s2s}\delta p_0 \\ \therefore\ x_1 &= (A_{s2s}L_0 + B)\delta p_0 + A_{s2s}d_0 \\ \therefore\ L_1 &= A_{s2s}L_0 + B, d_1 = A_{s2s}d_0 \end{aligned} \tag{31}$$

And Π_1 is similarly constructed as a projection to span$(L_1)^\perp$.

C. Whittling Algorithm Cut Solver

This section presents a gradient based solver for optimization problems on S^1, which we use to solve (26) quickly online. Given an optimization problem

$$\begin{aligned} &\underset{x\in\mathbb{R}^2}{\text{minimize}}\ f(x) \\ &\text{subject to}\ \|x\|_2^2 = 1, \end{aligned}$$

the associated first-order optimality conditions are

$$\|x\|_2^2 = 1 \tag{32}$$
$$\nabla f(x) + \nu x = 0 \tag{33}$$

where $\nu \in \mathbb{R}$ is a Lagrange multiplier for the unit norm constraint. Because we are optimizing over the unit circle, the main idea of our solution approach is to rotate x in the direction which decreases the cost until $\nabla f(x)$ is parallel to x, meaning that we have satisfied the optimality conditions. Our solver is summarized in Algorithm 2.

Algorithm 2 MakeCut Solver

Require: Cost function f, Initial guess $x \in S^1$, Optimality Tolerance ϵ, Line search parameters $\alpha > 0, \beta \in (0, 1)$
procedure SOLVE(f, x, ϵ)
  $\theta \leftarrow \infty$
  while $|\theta| > \epsilon$ **do**
    $\theta \leftarrow (\nabla f(x) - x\langle\nabla f(x), x\rangle) \times x$
    $t \leftarrow \alpha/|\theta|$
    while $f(\text{Rotate}(\theta t, x)) > f(x)$ **do** $t \leftarrow \beta t$
    $x \leftarrow \text{Rotate}(\theta t, x)$
  return x

where

$$\text{Rotate}(\theta, x) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} x.$$

The θ update finds the direction to rotate x by considering the component of ∇f orthogonal to x, using the cross product with x to convert this direction into a scalar rotation angle. We then perform a line search, starting with a fixed initial step size α for improved convergence speed. As an implementation note, we re-normalize x at each iteration to avoid drift in the unit-norm constraint.

D. Controller and Perception Stack Parameters

The following tables give the parameters used for each component of our stack. For conciseness of notation, diagonal matrices are represented as d$[\cdots]$, where the vector argument to d represents the entries on the diagonal of the matrix.

TABLE III: MPFC Parameters

Symbol	**Meaning**	**Value**
N	MPFC Horizon	3 steps
t_{min}	Minimum single-stance duration	0.27 s
t_{max}	Maximum single-stance duration	0.33 s
H	ALIP height	0.85 m
T_{ss}	Nominal single-stance duration	0.3 s
T_{ds}	Double-stance duration	0.1 s
w_T	Time regularization weight	100
l	Step width	0.2 m
w_u	Ankle torque regularization	0.01
u_{max}	Maximum ankle torque	22 Nm
Q_N	Terminal state cost hessian	d$[100, 100, 1, 1]$
Q	Runnning state cost hessian	d$[0.001, 0.1, 0.01, 0.001]$
R	Running step size cost hessian	d$[25, 25, 4]$
–	CoM soft position limits	± [0.35, 0.35] m
–	CoM soft velocity imits	± [2.5, 1.5] m/s
–	CoM soft constraint cost weight	1000

TABLE IV: OSC Gains

OSC Objective	**W**	**Kp**	**Kd**
Toe joint angle	1	1500	10
Hip yaw angle	2	40	2
CoM [x, y, z]	[0, 0, 10]	[0, 0, 100]	[0, 0, 6]
Pelvis [roll, pitch, yaw]	[2, 4, 0.02]	[200, 200, 0]	[10, 10, 4]
Swing Foot [x, y, z]	[4, 4, 2]	[220, 180, 180]	[6, 5.5, 5.5]
Ankle Torque	10	–	–

TABLE V: Perception Stack Parameters

Symbol	**Meaning**	**Value**
	Elevation Mapping	
–	Map Size	3×3 m
–	Map Resolution	0.03 m
	S3	
k_{hyst}	Safety Hysteresis	0.6
k_{safe}	Safety Threshold	0.7
σ_{LoG}	LoG Standard Deviation for c_{curve}	2 pixels (6 cm)
α_c	c_{curve} scaling parameter	5
–	c_{inc} normal calculation kernel size	5 pixels (15 cm)
	Convex Decomposition	
d	ACD Concavity Limit	0.25 m