IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

A Sequential MPC Approach to Reactive Planning for Bipedal Robots using Safe Corridors in Highly Cluttered Environments

Kunal S. Narkhede, Abhijeet M. Kulkarni, Dhruv A. Thanki and Ioannis Poulakakis

***Abstract*—This letter presents a sequential Model Predictive Control (MPC) approach to reactive motion planning for bipedal robots in highly cluttered environments with moving obstacles. The approach relies on a directed convex decomposition of the free space, which provides a safe corridor in the form of an ordered collection of mutually intersecting obstacle-free polytopes and waypoints. These are subsequently used to define a corresponding sequence of MPC programs that drive the system to a goal location avoiding static and moving obstacles. This way, the planner focuses on the free space in the vicinity of the robot, thus alleviating the need to consider all the obstacles simultaneously and reducing computational time. We verify the efficacy of our approach in high-fidelity simulations with the bipedal robot Digit, demonstrating robust reactive planning and trajectory realization amidst static and moving obstacles.**

***Index Terms*—Humanoid and bipedal locomotion, motion planning, safe corridors, sequential model predictive control.**

I. Introduction

NAVIGATING in the presence of obstacles is a prerequisite for bringing robots into human-populated spaces. Owing to their morphological and functional traits, humanoids and bipedal robots like Digit[1] (cf. Fig. 1) are ideally suited for such spaces. To bring such robots a step closer to navigating highly cluttered environments, this letter proposes a sequential Model Predictive Control (MPC) approach, which enables *reactive* trajectory planning and realization so that static and moving obstacles in the robot's vicinity can be avoided.

At its core, the problem of bipedal robot navigation involves finding suitable footstep sequences and consistent desired trajectories that transfer the robot to a desired location, avoiding obstacles on the way. Prior work in the area is extensive, and addresses several important facets of the general problem. For example, [1], [2] deal with footstep planning amidst static obstacles, [3], [4] focus on translating footstep plans to desired Center of Mass (COM) trajectories, [5], [6] emphasize tracking user-specified velocity or orientation commands online, [7] examines robustness in a similar setting, while [8], [9] focus on locomotion stability guarantees [10]–[12] for bipedal walkers navigating cluttered spaces by switching among limit-cycle gait primitives. Yet, autonomous planning of reactive walking motions that enable bipedal robots to respond in real time to changes in their surroundings cannot be addressed simply by combining available methods. Indeed, [1] computes desired footsteps for obstacle avoidance, but the resulting plan cannot be altered online to account for changing environments. Similarly, [2] relies on human input and is not suitable for online implementation. On the other hand, [3]–[7] either require prespecified footsteps or input from a kinematic planner or an operator, while [8], [9] also do not address changes in the workspace. These observations set the stage for our work, which aims at reactive motion planning for bipedal robots in highly cluttered environments.

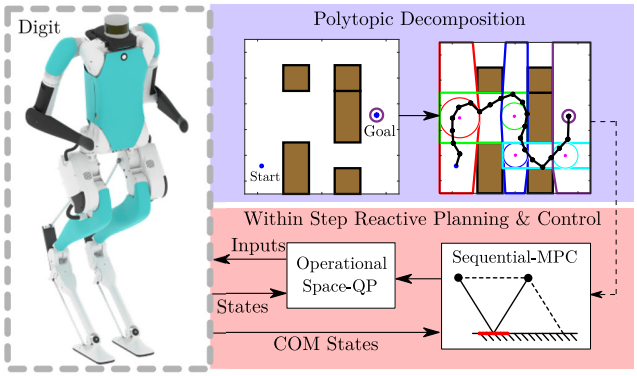


Figure 1. Overview of the proposed framework. An RRT-guided sequential decomposition of the workspace to obstacle-free polytopes is used to formulate a sequence of MPC programs that drive the LIP to the goal and avoid collisions with static and moving obstacles. The suggested LIP-based plan is then realized on Digit using a QP-based operational space controller.

Several related methods adopt reduced-order models such as the Linear Inverted Pendulum (LIP) [13] as predictive models in the context of MPC schemes that incorporate obstacles via distance-based constraints [14], [15]. However, such constraints become active in the optimization only when the robot comes close to an obstacle [16]. An alternative approach for taking obstacles into account relies on the notion of a discrete-time Control Barrier Function (CBF) [17], which was recently used within MPC to expand the nodes of a Rapidly-exploring Random Tree (RRT) to plan paths for the LIP [18]. However, these methods employ *one* constraint per static obstacle, which may result in unnecessarily large optimization programs, hindering the system's ability to react in real time to changes in its vicinity.

Manuscript received: April, 27, 2022; Revised July, 25, 2022; Accepted August, 17, 2022. This letter was recommended for publication by Editor Dr. A. Kheddar upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by NSF MRI-2018905.

Kunal S. Narkhede, Abhijeet M. Kulkarni, Dhruv A. Thanki, and Ioannis Poulakakis are with the Department of Mechanical Engineering, University of Delaware, Newark, Delaware 19716, USA (e-mail: kunalnk@udel.edu; amkulk@udel.edu; thankid@udel.edu; poulakas@udel.edu).

Digital Object Identifier (DOI): see top of this page.

[1]Digit is designed and manufactured by Agility Robotics https://www.agilityrobotics.com/robots#digit

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

Treating each obstacle separately by (generally non-convex) distance or CBF constraints can be avoided by bounding the robot's motion inside pre-computed convex regions of obstacle-free space. Dividing the free space into convex regions can be done using Iterative Regional Inflation by Semidefinite (IRIS) programming [19]. This method requires an obstacle-free seed point and proceeds with greedily constructing a large free convex region. Using user-supplied seeding, IRIS programming has been employed in [2] to formulate a mixed-integer convex program that assigns footsteps to safe convex regions. IRIS programming was also combined with automatic seeding in [20] to cover the free space by a small number of large safe polytopic regions; a mixed-integer program was solved to generate collision-free trajectories for a quadrotor. This method requires longer computational times to approximate the free space and solve the associated mixed-integer program, making it difficult to implement online.

To enable online computation of collision-free trajectories, several methods start by carving out a Safe Corridor (SC), i.e., a collection of overlapping obstacle-free convex regions connecting the initial and goal locations [21]–[24]. While, as [20], these methods have been developed for quadrotors, the underlying ideas are relevant to our approach and are discussed here. The methods in [21], [23] generate online a sequence of axis-aligned cubes in free space. However, this form of SCs may result in poor free-space utilization, particularly in highly cluttered environments. As a result, the solution space can be significantly limited, which is an important concern in our sequential MPC approach. To utilize the free space more efficiently, [22], [24] use polyhedron-shaped SCs. More relevant to our method is [22], which creates the SC by "inflating" polytopes around the chords of a seed path with the explicit requirement that consecutive polytopes mutually intersect. The overall process is faster than [20] and can be used for re-planning at $2-3$ Hz. However, since this method does not optimize over the maximum ellipsoid inscribed in each polytope (as IRIS does [19]) the resulting SCs are narrower than those generated by IRIS. These methods effectively plan open-loop desired trajectories inside SCs and rely on re-computing the SC and the trajectory to achieve reactivity to changes in the environment.

We adopt some ideas from these approaches and propose an efficient method that enables reactive collision-free navigation for bipedal robots like Digit. Akin to [22], our method uses a seed path to construct a safe walking corridor in the form of sequentially intersecting polytopes connecting the starting and goal locations. However, as our approach relies on IRIS [19] it captures more free space in the SCs than [22], albeit in a computationally more efficient way than [20]. The resulting SCs are used to define constraints (via discrete-time CBFs) and objectives (via suitable waypoints) to a sequence of LIP-based MPC programs that drives the robot to the goal. The method is implemented in high-fidelity simulations with Digit using Operational Space Control (OSC) formulated as a weighted Quadratic Program (QP) akin to [25], demonstrating robust realization of LIP-based plans in the presence of static and moving obstacles. The main contribution of our approach is that reactivity to changes in the environment is infused *within* the SC, allowing the robot to respond to moving obstacles and disturbances *without* re-planning the corridor. This is possible because of the significant computational benefits offered by the proposed method, even for highly-cluttered spaces comprising tens of obstacles, allowing us to solve the MPCs multiple times within each step Digit takes.

II. A 3D LIP Model for Bipedal Motion Planning

In its common configuration, the 3D LIP consists of a point mass atop a massless prismatic leg (cf. Fig. 2). Let $(x, y) \in \mathbb{R}^2$ be the location of the mass with respect to an inertia frame and $(\dot{x}, \dot{y}) \in \mathbb{R}^2$ the corresponding velocity. It is assumed that the mass is constrained to move in a horizontal plane located at constant height H. We use u^x and u^y to denote the distance between the Center of Pressure (COP) of the stance foot and the COM in the x and y directions.

The continuous-time evolution of the LIP along the x axis is governed by $\ddot{x} = -(g/H)u^x$, where g is the gravitational acceleration. This expression can be integrated to result in

$$\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{\omega}\sinh\omega t \\ 0 & \cosh\omega t \end{bmatrix} \begin{bmatrix} x(0) \\ \dot{x}(0) \end{bmatrix} + \begin{bmatrix} 1-\cosh\omega t \\ -\omega\sinh\omega t \end{bmatrix} u^x \quad (1)$$

where $x(0)$, $\dot{x}(0)$ are initial conditions and $\omega = \sqrt{g/H}$. The motion along the y axis is governed by identical dynamics.

To plan motions for Digit using the 3D LIP, we will approximate the reachable region of the swing foot with a rectangle as in [18] (cf. Fig. 2). The orientation of this region relative to the vertical axis of a body-fixed frame $\{x_c, y_c, z_c\}$ is denoted by θ. We assume that θ is updated instantaneously at the exchange of support and that $\dot{\theta} = 0$ during a step. Thus, if $\theta(0)$ is the angle of the rectangle of the leg providing support, the corresponding angle for the swing foot when it becomes the support foot in the forthcoming step is

$$\theta(t) = \theta(0) + u^\theta \quad (2)$$

where u^θ is a step change assumed to satisfy $lb^\theta \leq u^\theta \leq ub^\theta$ for suitable lower and upper bounds lb^θ and ub^θ.

Next, we assume that each step has duration T and that $[kT, (k+1)T)$ is the interval spanned by the k-th step. If

$$\mathrm{x}_k = \begin{bmatrix} x_k & \dot{x}_k & y_k & \dot{y}_k & \theta_k \end{bmatrix}^\mathsf{T} \quad \text{and} \quad \mathrm{u}_k = \begin{bmatrix} u_k^x & u_k^y & u_k^\theta \end{bmatrix}^\mathsf{T}$$

denote the state and input at the beginning of the k-th step, the state of the system at the $(k+1)$ step is given by

$$\mathrm{x}_{k+1} = A\mathrm{x}_k + B\mathrm{u}_k \quad (3)$$

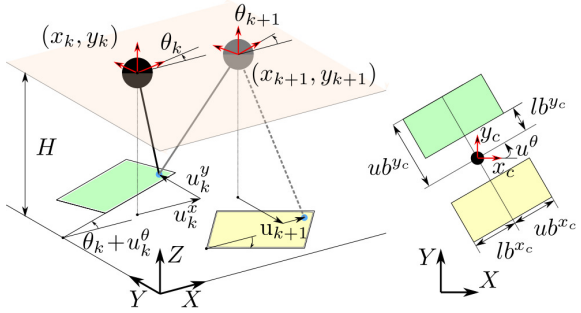


Figure 2. Reachable spaces for the left (green) and right (yellow) feet, defined with bounds relative to the frame $\{x_c, y_c, z_c\}$ attached at the COM.

where $A = \text{diag}\{\hat{A},\ \hat{A},\ 1\}$, $B = \text{diag}\{\hat{B},\ \hat{B},\ 1\}$ and $\hat{A}, \hat{B}$ are block diagonal matrices obtained by (1) by setting $t = T$. To plan motions in the (x, y)-plane we define the output

$$\mathrm{y}_k = C\mathrm{x}_k \tag{4}$$

where C is the 2×5 matrix so that $C\mathrm{x} = \begin{bmatrix} x & y \end{bmatrix}^{\mathsf{T}}$.

To avoid LIP motions that challenge the robot's capabilities, we introduce the following time-dependent constraint set for the state and inputs

$$\mathcal{XU}_k = \big\{(\mathrm{x}_k, \mathrm{u}_k) \mid \mathrm{lb}_k \le R(\theta_k + u_k^\theta)^{\mathsf{T}}\mathrm{u}_k \le \mathrm{ub}_k, \text{ and } \delta_{\min} \le \sqrt{\Delta x_k^2 + \Delta y_k^2} \le \delta_{\max}\big\} \tag{5}$$

where $\mathrm{lb}_k = \begin{bmatrix} lb_k^{x_c} & lb_k^{y_c} & lb_k^{\theta} \end{bmatrix}^{\mathsf{T}}$ (resp. ub_k) includes lower (resp. upper) bounds of the reachability constraints and $R(\theta_k + u_k^\theta)$ is the corresponding 3D rotation matrix (cf. Fig. 2), $\delta_{\min}$ and $\delta_{\max}$ are lower and upper bounds on the COM travel distance between steps, and $\Delta x_k = x_{k+1} - x_k$, $\Delta y_k = y_{k+1} - y_k$. The first part of (5) corresponds to constraints on the reachability rectangles and their orientation, while the second part to constraints on the COM travel distance; the latter is included to avoid infeasible motions, effectively bounding the average speed.

III. LIP Reactive Planning via MPC

This section presents our method for planning LIP paths to a goal position, avoiding obstacles in the workspace $\mathcal{W}$.

A. Safe Corridors via Path-Guided Free Space Decomposition

Consider first the static obstacles $\mathcal{W}^{\mathrm{s}}$ in $\mathcal{W}$. We "inflate" the obstacles to account for the nontrivial dimensions of Digit, which is assumed to be enclosed in a disc of radius 0.5 m. We also assume that all the obstacles forming $\mathcal{W}^{\mathrm{s}}$ are convex. While this assumption does not entail a significant loss of generality [19], it will allow us to compute efficiently a sequence of free polytopes $\mathcal{H}_i$, $i = 0, 1, ..., M-1$, along a pre-computed obstacle-free path; these polytopes constitute a SC within which the LIP can be steered to the goal via a chain of MPC programs.

We begin with generating a sequence of points in $\mathcal{W} \setminus \mathcal{W}^{\mathrm{s}}$ that connect an initial location with a final desired one. This can be done using any sampling-based planning algorithm; here, we use RRT* to find a sequence of such points $\Pi = \{\mathrm{p}_1, ..., \mathrm{p}_K\}$ in $\mathcal{W} \setminus \mathcal{W}^{\mathrm{s}}$. While generating Π, we make sure that the line segment joining any two consecutive points in Π does not intersect any static obstacle. With the availability of Π, we implement Algorithm 1 to construct a chain of free polytopes; Algorithm 1 calls the following functions.

- Generate_Polytopes: Given $\mathcal{W}^{\mathrm{s}}$ and a "seed" point p in the free space $\mathcal{W} \setminus \mathcal{W}^{\mathrm{s}}$, this function creates a polytope $\mathcal{H} \subset \mathcal{W} \setminus \mathcal{W}^{\mathrm{s}}$. This is done using IRIS [19]; here, the algorithm is terminated so that the seed point p is included in the interior of the returned polytope $\mathcal{H}$.
- Chebyshev_Center: This function takes a pair of polytopes $\mathcal{H}_{i_1}$ and $\mathcal{H}_{i_2}$ and returns the Chebyshev center[2] w of their intersection $\mathcal{H}_{i_1} \bigcap \mathcal{H}_{i_2}$, or an empty list if they are disjoint. Determining whether two polytopes intersect and computing the corresponding Chebyshev center is done via a linear program.
- Intersecting_Polytopes: This function takes two seed points p_{j_1} and p_{j_2} from Π together with the corresponding polytopes $\mathcal{H}_{i_1}$ and $\mathcal{H}_{i_2}$ generated by Generate_Polytopes, and returns an ordered sequence

$$\bar{\mathcal{G}} = \{(\mathcal{H}^\ell, \mathrm{w}^\ell) \mid \ell = 1, ..., L+1\}$$

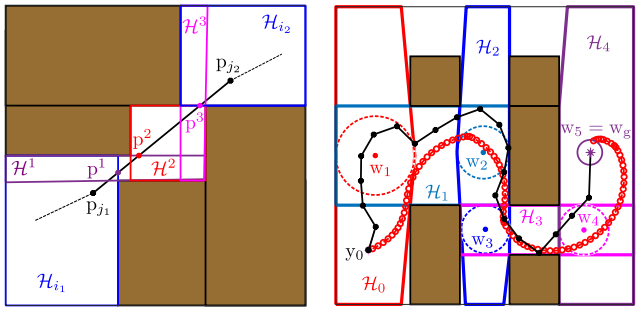
where $\mathcal{H}^{L+1} = \mathcal{H}_{i_2}$. The chain $\{\mathcal{H}_{i_1}, \mathcal{H}^1, ..., \mathcal{H}^L, \mathcal{H}_{i_2}\}$ is constructed as in Fig. 3 so that it includes sequentially pair-wise intersecting polytopes connecting $\mathcal{H}_{i_1}$ and $\mathcal{H}_{i_2}$. The points $\mathrm{w}^1 \in \mathcal{H}_{i_1} \bigcap \mathcal{H}^1$ and $\mathrm{w}^\ell \in \mathcal{H}^{\ell-1} \bigcap \mathcal{H}^\ell$ for $\ell = 2, ..., L+1$ are the corresponding Chebyshev centers. When $\mathcal{H}_{i_1}$ and $\mathcal{H}_{i_2}$ intersect, $L = 0$ and the function returns $(\mathcal{H}^1, \mathrm{w}^1)$ where $\mathcal{H}^1 = \mathcal{H}_{i_2}$.
- Poly_Line_Intersect: This function takes a polytope $\mathcal{H}$, a point $\mathrm{p}_1 \in \mathcal{H}$ and a point $\mathrm{p}_2 \notin \mathcal{H}$ and returns the point at

[2]Terminology: The Chebyshev center of two intersecting polytopes is the center of the largest ball inscribed in their intersection.

Algorithm 1 PolyFsGen algorithm

Given: y_0, $\Pi = \{\mathrm{p}_1, ..., \mathrm{p}_K\}$, w_{g}, $\mathcal{W}^{\mathrm{s}}$
$\mathcal{H}_0$ = Generate_Polytopes($\mathcal{W}^{\mathrm{s}}, \mathrm{y}_0$)
Initialize list: $i = 0$, $j = 0$, $\mathcal{G} = \{(\mathcal{H}_0, \mathrm{y}_0)\}$
$\mathrm{p}_0 = \mathrm{y}_0$, $\mathrm{p}_{K+1} = \mathrm{w}_{\mathrm{g}}$
while $\mathrm{w}_{\mathrm{g}} \notin \mathcal{H}_i$ OR $j < K+1$ **do**
  $j \leftarrow j + 1$
  if $\mathrm{p}_j \in \mathcal{H}_i$ **then**
    continue
  else
    $\mathcal{H}_{\mathrm{new}} \leftarrow$ Generate_Polytopes($\mathcal{W}^{\mathrm{s}}, \mathrm{p}_j$)
    $\bar{\mathcal{G}} =$ Intersecting_Polytopes($\mathcal{H}_i, \mathrm{p}_{j-1}, \mathcal{H}_{\mathrm{new}}, \mathrm{p}_j$)
    $\mathcal{G} \leftarrow \mathcal{G} \cup \bar{\mathcal{G}}$
    $i \leftarrow$ length($\mathcal{G}$)
    $\mathcal{H}_i \leftarrow \mathcal{H}_{\mathrm{new}}$
  end if
end while
return $\mathcal{G}$

function Intersecting_Polytopes($\mathcal{H}_{i_1}, \mathrm{p}_{j_1}, \mathcal{H}_{i_2}, \mathrm{p}_{j_2}$)
  $\mathrm{w}_{\mathrm{new}} =$ Chebyshev_Center($\mathcal{H}_{i_1}, \mathcal{H}_{i_2}$)
  if $\mathrm{w}_{\mathrm{new}}$ is empty **then**
    $\ell = 0$, $\mathrm{p}^0 = \mathrm{p}_{j_1}$, $\mathcal{H}^0 = \mathcal{H}_{i_1}$
    while $\mathrm{w}_{\mathrm{new}}$ is empty **do**
      $\mathrm{p}^{\ell+1} =$ Poly_Line_Intersect($\mathcal{H}^\ell, \mathrm{p}^\ell, \mathrm{p}_{j_2}$)
      $\mathcal{H}^{\ell+1} =$ Generate_Polytopes($\mathcal{W}^{\mathrm{s}}, \mathrm{p}^{\ell+1}$)
      $\mathrm{w}^{\ell+1} =$ Chebyshev_Center($\mathcal{H}^\ell, \mathcal{H}^{\ell+1}$)
      $\mathrm{w}_{\mathrm{new}} =$ Chebyshev_Center($\mathcal{H}^{\ell+1}, \mathcal{H}_{i_2}$)
      $\ell \leftarrow \ell + 1$
    end while
    $\bar{\mathcal{G}} \leftarrow \{(\mathcal{H}^1, \mathrm{w}^1), ..., (\mathcal{H}^\ell, \mathrm{w}^\ell), (\mathcal{H}_{i_2}, \mathrm{w}_{\mathrm{new}})\}$
  else
    $\bar{\mathcal{G}} \leftarrow \{(\mathcal{H}_{i_2}, \mathrm{w}_{\mathrm{new}})\}$
  end if
  return $\bar{\mathcal{G}}$
end function



(a) (b)

Figure 3. (a) The workings of function Intersecting_Polytopes. The function takes a pair of non-intersecting polytopes $\mathcal{H}_{i_1}$ and $\mathcal{H}_{i_2}$ generated by the seed points p_{j_1} and p_{j_2}, respectively, and computes the point p^1 at which the line segment from p_{j_1} to p_{j_2} intersects $\mathcal{H}_{i_1}$. This point is then used as a new seed point to obtain a new polytope $\mathcal{H}^1$. Then, the point p^2 at which the segment from p^1 to p_{j_2} intersects the new polytope $\mathcal{H}^1$ is computed and used as a new seed to obtain $\mathcal{H}^2$. The process continues until $\mathcal{H}^\ell$ intersects $\mathcal{H}_{i_2}$. (b) Given $(\mathcal{H}_i, \mathrm{w}_{i+1})$, $i = 0, ..., 4$ obtained by Algorithm 1 based on an RRT plan Π (black), a sequence of MPCs is defined that outputs a collision-free path (red) connecting the initial y_0 and goal w_g locations.

which the line segment connecting p_1 and p_2 intersects the boundary of $\mathcal{H}$.

Algorithm 1 begins with calling Generate_Polytopes to create a polytope $\mathcal{H}_0$ in $\mathcal{W} \setminus \mathcal{W}^\mathrm{s}$ that contains the initial location $\mathrm{y}_0 = C\mathrm{x}_0$. The algorithm then checks if the goal $\mathrm{w}_\mathrm{g} \in \mathcal{H}_0$, and, if not, it runs through the points p_j, $j = 1, ..., K$, of the plan Π to find the first point p_{j_1} that is not contained in $\mathcal{H}_0$. The pairs $(\mathcal{H}_0, \mathrm{y}_0)$ and $(\mathcal{H}_\mathrm{new}, \mathrm{p}_{j_1})$ are then passed to Intersecting_Polytopes. If $\mathcal{H}_0 \bigcap \mathcal{H}_\mathrm{new} \neq \emptyset$, the function returns $(\mathcal{H}^1, \mathrm{w}^1)$ where $\mathcal{H}^1 = \mathcal{H}_\mathrm{new}$ and w^1 is the Chebyshev center of the intersection $\mathcal{H}_0 \bigcap \mathcal{H}_\mathrm{new}$. If, on the other hand, $\mathcal{H}_0 \bigcap \mathcal{H}_\mathrm{new} = \emptyset$, the function produces a sequence of pairs of polytopes and Chebyshev centers $\bar{\mathcal{G}}$ connecting $\mathcal{H}_0$ and $\mathcal{H}_\mathrm{new}$. The end result is a SC consisting of a sequence of pairwise intersecting polytopes $\{\mathcal{H}_0, \mathcal{H}_1, ..., \mathcal{H}_{M-1}\}$ with $\mathrm{y}_0 \in \mathcal{H}_0$ and waypoints $\{\mathrm{w}_1, ..., \mathrm{w}_M\}$ such that

$$\mathrm{w}_i \in \mathcal{H}_{i-1} \bigcap \mathcal{H}_i \quad \text{for} \quad i = 1, ..., M-1 \tag{6}$$

and $\mathrm{w}_M = \mathrm{w}_\mathrm{g} \in \mathcal{H}_{M-1}$.

Remark 1: In general, if the initial y_0 and/or goal w_g locations change, Algorithm 1 must be executed anew. This is not necessary, however, when y_0, w_g are in the union of an available sequence of polytopes. For example, if a SC $\{\mathcal{H}_0, ..., \mathcal{H}_{M-1}\}$ is available and y_0, w_g change so that $\mathrm{y}_0 \in \mathcal{H}_m$ and $\mathrm{w}_\mathrm{g} \in \mathcal{H}_n$ for some m and n, then the given sequence can be "rewired" to a new sequence $\{\mathcal{H}_m, \mathcal{H}_{m+1}, ..., \mathcal{H}_n\}$ if $m < n$ or $\{\mathcal{H}_m, \mathcal{H}_{m-1}, ..., \mathcal{H}_n\}$ if $m > n$ without the need to execute Algorithm 1 again.

B. Sequential Model Predictive Control

In what follows, we assume that an ordered collection of pairs $(\mathcal{H}_i, \mathrm{w}_{i+1})$, $i = 0, ..., M-1$, satisfying (6) is available.

1) Static obstacles: Each polytope $\mathcal{H}_i$, $i = 0, ..., M-1$ is free from collisions with the static obstacles, and can be represented as a finite collection of l_i closed half-spaces $\mathcal{H}_i = \{\mathrm{y} \in \mathbb{R}^2 \mid P_i\mathrm{y} \leq b_i\}$, where P_i is an $l_i \times 2$ matrix and $b_i \in \mathbb{R}^{l_i}$. As usual, $P_i\mathrm{y} \leq b_i$ is the shorthand notation for the system of inequalities $(r_i)_j^\mathsf{T}\mathrm{y} \leq (b_i)_j$, $j = 1, ..., l_i$, where $(r_i)_j^\mathsf{T}$ is j-th row of P_i and $(b_i)_j$ the j-th element of b_i. To each $\mathcal{H}_i$ we can then associate l_i smooth scalar-valued functions $h_{ij} : \mathbb{R}^5 \to \mathbb{R}$,

$$h_{ij}(\mathrm{x}) = (b_i)_j - (r_i)_j^\mathsf{T} C\mathrm{x}, \quad j = 1, ..., l_i \tag{7}$$

so that the intersection of the corresponding 0-superlevel sets

$$\mathcal{C}_{ij} = \{\mathrm{x} \in \mathbb{R}^5 \mid h_{ij}(\mathrm{x}) \geq 0\}, \quad j = 1, ..., l_i \tag{8}$$

contains the states of the 3D-LIP (4) that are mapped via the output (4) onto the free polytopes $\mathcal{H}_i$. Let $\mathcal{C}_i = \bigcap_{j=1}^{l_i} \mathcal{C}_{ij}$ be the intersection of the (unbounded) polyhedra (8).

With these definitions, collision-free evolution is achieved by selecting the input of the 3D-LIP so that its state is constrained to evolve in the corresponding $\mathcal{C}_i$ when it starts in $\mathcal{C}_i$. If for some state x_0 we have $h_{ij}(\mathrm{x}_0) \geq 0$ for all $j = 1, ..., l_i$ (that is, $\mathrm{x}_0 \in \mathcal{C}_i$), future evolution in $\mathcal{C}_i$ can be ensured by choosing u_k so that the following inequalities are simultaneously (over j) satisfied

$$h_{ij}\left(A\mathrm{x}_k + B\mathrm{u}_k\right) \geq (1-\gamma) h_{ij}\left(\mathrm{x}_k\right), \quad j = 1, ..., l_i \tag{9}$$

where $0 < \gamma \leq 1$. The existence of an input value u_k that satisfies (9) implies that h_{ij} is a discrete-time exponential CBF [17] for the 3D-LIP dynamics (3). Note that for each $i = 0, ..., M-1$ and each $j = 1, ..., l_i$, the constraint (9) is affine in u_k and will be incorporated in an MPC program defined for each $i = 0, ..., M-1$ to ensure the system is safe with respect to collisions with static obstacles.

2) Moving obstacles: We consider n^d moving obstacles $\nu = 1, ..., n^\mathrm{d}$ of elliptical shape, suitably inflated as in [26] to account for the robot's dimensions. At time k, each moving obstacle can be represented by a pair $(\mathrm{p}^\mathrm{d}_{\nu,k}, P^\mathrm{d}_\nu(\varphi_{\nu,k}))$, where $\mathrm{p}^\mathrm{d}_{\nu,k}$ is the center of the ν-th ellipse, $\varphi_{\nu,k}$ its orientation, and the matrix $P^\mathrm{d}_\nu(\varphi_{\nu,k})$ captures its shape [26]. Assuming that $\mathrm{p}^\mathrm{d}_{\nu,k}$ and $\varphi_{\nu,k}$ are known at each k, collision with the ν-th obstacle is avoided by imposing the barrier constraint

$$h^\mathrm{d}_{\nu,k+1}(A\mathrm{x}_k + B\mathrm{u}_k) \geq (1-\gamma_\nu) h^\mathrm{d}_{\nu,k}(\mathrm{x}_k), \ \nu = 1, ..., n^\mathrm{d} \tag{10}$$

where $0 < \gamma_\nu \leq 1$ and

$$h^\mathrm{d}_{\nu,k}(\mathrm{x}_k) = \left(C\mathrm{x}_k - \mathrm{p}^\mathrm{d}_{\nu,k}\right)^\mathsf{T} P^\mathrm{d}_\nu(\varphi_{\nu,k}) \left(C\mathrm{x}_k - \mathrm{p}^\mathrm{d}_{\nu,k}\right) - 1 \ . \tag{11}$$

3) Sequential MPC: Given a SC represented as an ordered collection $(\mathcal{H}_i, \mathrm{w}_{i+1})$, $i = 0, ..., M-1$, of free polytopes and waypoints satisfying (6), we define here a corresponding sequence MPC(i), $i = 0, ..., M-1$, of MPC programs. In this sequence, the objective of the i-th MPC is to drive the output (4) of the 3D-LIP towards the $(i+1)$ waypoint w_{i+1} while keeping it within $\mathcal{H}_i$, avoiding moving obstacles, and satisfying the relevant state and input constraints (5). Then, composing the MPCs according to the sequence

$$\cdots \longrightarrow \mathrm{MPC}(i) \xrightarrow{C\mathrm{x}_k \in \mathcal{H}_{i+1}} \mathrm{MPC}(i+1) \longrightarrow \cdots$$

for $i = 0, ..., M-1$ results in a sequence of control values that drives the 3D-LIP to the goal location $\mathrm{w}_\mathrm{g} = \mathrm{w}_M \in \mathcal{H}_M$. Switching from MPC($i$) to MPC($i+1$) is triggered when the output $C\mathrm{x}_k$ enters the polytope $\mathcal{H}_{i+1}$.

The objective of MPC(i) is captured by the desired state

$$S\mathrm{x}^\mathrm{des}_{i+1} = \begin{bmatrix} \mathrm{w}_{i+1}^\mathsf{T} & \theta_{i+1} & 0_{1\times 2} \end{bmatrix}^\mathsf{T} \tag{12}$$

where S is the 5×5 selection matrix that re-organizes the state of the 3D-LIP so that $S\mathrm{x} = \begin{bmatrix} x & y & \theta & \dot{x} & \dot{y} \end{bmatrix}^\mathsf{T}$, and w_{i+1} is the target waypoint. The desired orientation θ_{i+1} is set to be the angle of the vector that connects the position $C\mathrm{x}_k$ of the 3D-LIP at step k with w_{i+1}.

The terminal cost of MPC(i) can now be defined by

$$J_{i,N}(\mathrm{x}_N) = \|S\mathrm{x}_N - S\mathrm{x}_{i+1}^{\mathrm{des}}\|_{W_1}^2 \tag{13}$$

where W_1 is a 5×5 diagonal weighting matrix. The running cost is constructed similarly to (13) with the modification of including an additional term that penalizes the magnitude of the 3D-LIP input at each step; that is,

$$J_{i,k}(\mathrm{x}_k, \mathrm{u}_k) = \|S\mathrm{x}_k - S\mathrm{x}_{i+1}^{\mathrm{des}}\|_{W_2}^2 + \|\mathrm{u}_k\|_{W_3}^2 \tag{14}$$

where W_2 and W_3 are 5×5 and 3×3 diagonal weighting matrices, respectively. Note that, by (12), minimizing (13)-(14) also penalizes the velocity of the LIP, effectively introducing a damping effect which is necessary since the barrier polyhedra $\mathcal{C}_i$ do not bound the LIP velocity. This effect facilitates the execution of the plan by Digit.

To summarize, the i-th MPC is defined as:

$$\begin{aligned}
\underset{X,U}{\text{minimize}} \quad & \sum_{\kappa=k}^{k+N-1} J_{i,\kappa}(\mathrm{x}_\kappa, \mathrm{u}_\kappa) + J_{i,k+N}(\mathrm{x}_{k+N}) \\
\text{subject to} \quad & \mathrm{x}_{\kappa+1} = A\mathrm{x}_\kappa + B\mathrm{u}_\kappa \\
& \mathrm{x}_k = \mathrm{x}_{\mathrm{init}},\ (\mathrm{x}_\kappa, \mathrm{u}_\kappa) \in \mathcal{XU}_\kappa \\
& \Delta h_{ij}(\mathrm{x}_\kappa, \mathrm{u}_\kappa) \geq -\gamma h_{ij}(\mathrm{x}_\kappa),\ j = 1, ..., l_i \\
& \Delta h_{\nu,\kappa}^{\mathrm{d}}(\mathrm{x}_\kappa, \mathrm{u}_\kappa) \geq -\gamma_\nu h_{\nu,\kappa}^{\mathrm{d}}(\mathrm{x}_\kappa),\ \nu = 1, ..., n^{\mathrm{d}}
\end{aligned}$$

where $\mathcal{XU}_\kappa$ is the constraint set (5), h_{ij} and $h_{\nu,\kappa}^{\mathrm{d}}$ are the barrier functions (7) and (11), and $\Delta h_{ij}(\mathrm{x}_\kappa, \mathrm{u}_\kappa) = h_{ij}(\mathrm{x}_{\kappa+1}) - h_{ij}(\mathrm{x}_\kappa)$ and $\Delta h_{\nu,\kappa}^{\mathrm{d}}(\mathrm{x}_\kappa, \mathrm{u}_\kappa) = h_{\nu,\kappa+1}^{\mathrm{d}}(\mathrm{x}_{\kappa+1}) - h_{\nu,\kappa}^{\mathrm{d}}(\mathrm{x}_\kappa)$. Given $\mathrm{x}_{\mathrm{init}}$, the solution of MPC$(i)$ returns the sequences $X = \{\mathrm{x}_{k+1}, ..., \mathrm{x}_{k+N}\}$ and $U = \{\mathrm{u}_k, ..., \mathrm{u}_{k+N-1}\}$. We then apply the first element u_k in U and proceed with solving MPC(i) at the next step based on the new initial condition $\mathrm{x}_{\mathrm{init}}$. The process is repeated until $C\mathrm{x}_k \in \mathcal{H}_{i+1}$ for some k, where we switch to MPC$(i+1)$ and continue until the goal.

Before proceeding with numerical experiments to evaluate the performance of the method, a few remarks are in order.

Remark 2: The MPC programs described above effectively "drive" the evolution of the LIP away from the boundaries of the SC in a *feedback* fashion. This is different from methods such as [21]–[23], which plan open-loop trajectories inside the SC and achieve reactivity to changes in the environment by re-planning the SC and updating the desired trajectory. While, as detailed below, online re-planning of the SCs is certainly possible using our method, an additional layer of reactivity is infused *within* a given SC, thus allowing the system to respond to disturbances and moving obstacles without the need to compute a new SC.

Remark 3: For the MPC programs to be well defined, the intersection between two successive polytopes in the SC must be nonempty. Depending on the size of the Chebyshev discs that can be inscribed in the nonempty intersections, the choice of the parameters γ in the definition (9) of the CBFs and $\delta_{\min}$ in the constraint set (5) can affect the feasibility of the associated MPCs. First, note that choosing γ close to zero "pushes" the system away from the polytope boundaries, resulting in safer but more conservative evolution [16]. Thus, choosing a small γ makes it more difficult for the system to reach the intersection, particularly when the size of the Chebyshev balls is small. In practice, however, we found this rarely to be a problem, and can always be resolved by choosing larger values for γ, even for very small Chebyshev balls. Regarding $\delta_{\min}$, positive values force the system to move forward by a minimum distance and may interfere with the satisfaction of the CBF constraints for narrow intersections. This can always be avoided by setting $\delta_{\min} = 0$.

C. Numerical Experiments and Performance Evaluation

In this section, we numerically investigate the effectiveness of our approach. The quality of realizing the plans on Digit is discussed in the following section. We consider three sets of randomly generated environments containing (i) rectangular obstacles with different sizes, (ii) rectangular obstacles with different sizes and orientations, (iii) general polytopic obstacles; see [27, Fig. 4]. In each case, we generate 50 random environments with $30, 40, 50$ and 60 obstacles in a confined $50\mathrm{m} \times 50\mathrm{m}$ space; hence, 150 environments are considered for each obstacle population, giving a total of 600 maps. The obstacle coverage is roughly 40% in all maps, and the starting and goal locations are kept the same. All computations in this section were performed on an Intel PC with i7-9750H processor (2.60 GHz) and 16GB RAM.

1) Safe Corridors: We employ Algorithm 1 to generate SCs and computes the Chebyshev centers of the associated polytopes. For IRIS, we use the implementation on GitHub [28], which relies on Mosek to perform the requisite semidefinite optimizations. We compare the results in terms of free-space utilization and runtimes with the method in [22], which is also used for comparisons with other methods in the relevant literature; see [24] for example. For the method in [22] we use the implementation on GitHub [29]. Note here that this method requires a point-cloud representation of the obstacles. To do this, we discretize the randomly generated maps with a resolution of $0.25\mathrm{m}$ and use a bounding box set at $(5\mathrm{m}\times5\mathrm{m})$ to limit collision checking as in [22]. Both methods are seeded with the same skeleton paths Π computed using RRT*. We note first that, given the seed path, both algorithms were able to compute SCs in all the 600 environments considered.

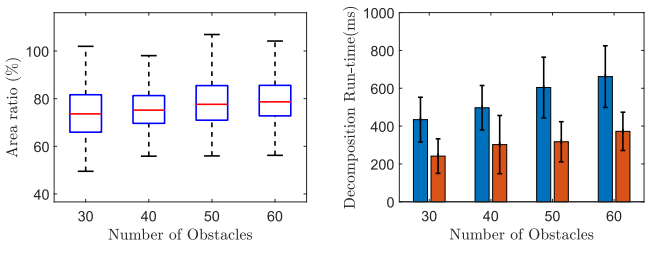


Figure 4. Size of SCs computed by [22] as a percentage of those by Algorithm 1. (b) Average runtimes for [22] (blue) and Algorithm 1 (orange).

Figure 4 summarizes our findings. It can be seen from Fig. 4(a) that the proposed method utilizes the relevant free space more efficiently, generating larger SCs than [22]. This is not surprising, given the fact that IRIS optimizes over the volume of the ellipsoids that can be inscribed in the returned polytopes. This is important in the context of our sequential MPC approach since larger polytopes correspond to larger solution spaces and provide more flexibility to the robot in the presence of moving obstacles or disturbances. Furthermore, as Fig. 4(b) shows, the average computation time required using our method is almost less than half than the one needed by [22], requiring times that are on average less than 400 ms. This is an indication that the proposed polytopic decomposition can be used online. Although we do not explore this feature here, as we focus on reactivity *without* recomputing the SC, it is still reassuring to know that re-planning the SC is still possible every 2-3 steps Digit takes.

2) MPC Computations: Here, we use the SCs obtained by Algorithm 1 to define the sequential MPC programs, which are then solved using the interior point solver shipped with CasADi [30]. The LIP parameters are selected to (approximately) match Digit and are given in the accompanying video. Figure 5(a) shows the average time required to solve the MPC for horizons $N = 2, 3, 4$ while Fig. 5(b) presents the percentage of the 600 environments for which a safe path is computed using our method. For a given horizon, the runtime does not vary significantly with the number of static obstacles. This is because the MPC does not consider the full environment at once; rather, it focuses on the free space in the vicinity of the robot, capturing it in a polytope described by a small number of constraints. This is beneficial, particularly in highly cluttered spaces containing large numbers of obstacles. Finally, as expected, Fig. 5(a) shows that the time required to solve the MPCs increases with N while the success rates for $N = 3$ and $N = 4$ are comparable, as indicated in Fig. 5(b). Thus, in what follows, we use $N = 3$ as a reasonable trade-off between success rates and computational time.

IV. Realizing 3D-LIP Plans on Digit

We apply the proposed approach on Digit (cf. Fig. 6) walking amidst static and moving obstacles. Digit weighs 47.9kg and it is approximately 1.58m tall when standing up. Each leg features 8 Degrees of Freedom (DOF), two of which correspond to passive leaf springs. Furthermore, Digit incorporates a pair of 4 DOF arms. Thus, the robot has a total of 30 DOFs: 20 actuated, 4 passive and 6 corresponding to the position and orientation of a body-fixed frame.

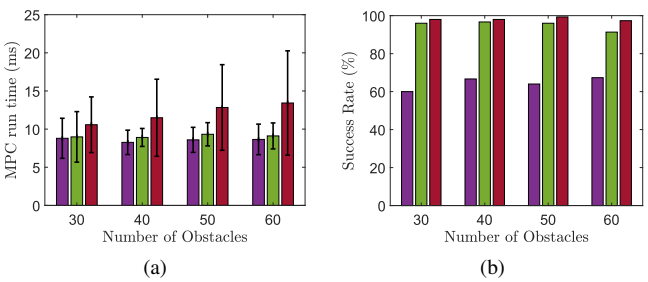


Figure 5. (a) Average MPC runtimes per iteration for horizons $N = 2$ (purple), $N = 3$ (green), $N = 4$ (red). (b) Corresponding success rates.

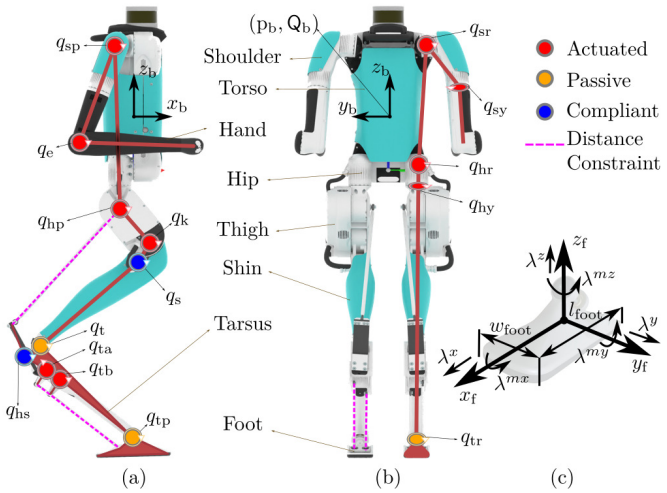


Figure 6. The bipedal robot Digit and relevant coordinates. (a) Side view; the kinematic loops actuating the tarsus and foot are highlighted with the dotted magenta lines. (b) Back view. (c) Ground contact constraints.

A. Digit Dynamics: Notation and Assumptions

Digit's configuration can be described with respect to an inertia frame by the position $\mathrm{p_b} \in \mathbb{R}^3$ and the unit quaternion $\mathsf{Q_b} \in \mathbb{H}$ associated with the orientation of a body-fixed frame $\{x_\mathrm{b}, y_\mathrm{b}, z_\mathrm{b}\}$ attached at Digit's torso (cf. Fig. 6). The configuration of the free-floating model is $q = \left(\mathrm{p_b}, \mathsf{Q_b}, q^\mathrm{L}, q^\mathrm{R}\right) \in \mathbb{R}^3 \times \mathbb{H} \times \mathcal{Q}_\mathrm{s}^\mathrm{L} \times \mathcal{Q}_\mathrm{s}^\mathrm{R}$, where $q^{\iota\in\{\mathrm{L,R}\}} \in \mathcal{Q}^{\iota\in\{\mathrm{L,R}\}}$ include the variables for the left and right leg/arm pairs of the robot. For our controller design, we will focus on the single support phase and assume that the deflections of the leaf springs are negligible. Under these assumptions, the single-support dynamics of Digit are

$$\begin{bmatrix} D(q) & J(q)^\mathsf{T} \\ J(q) & 0 \end{bmatrix} \begin{bmatrix} \dot{v} \\ -\lambda \end{bmatrix} = \begin{bmatrix} -c(q,v) + S_\tau^\mathsf{T}\tau \\ -\dot{J}(q,v)v \end{bmatrix} \tag{16}$$

where $v = \left(\dot{\mathrm{p}}_\mathrm{b}, \Omega_\mathrm{b}, \dot{q}^\mathrm{L}, \dot{q}^\mathrm{R}\right)$, Ω_b is the angular velocity of the body-fixed frame, $D(q)$ is the inertia matrix, $c(q,v)$ contains the velocity-dependent and gravitational forces, S_τ is the input selection matrix and $\tau \in \mathbb{R}^{20}$ are the motor torques. Due to space limitations, we do not provide a detailed account of how the kinematics loops are treated. We only mention here that the Jacobian $J(q)$ incorporates all the holonomic constraints associated with the single-support phase and $\lambda \in \mathbb{R}^{16}$ are the corresponding constraint forces. Finally, contact limitations are satisfied by requiring the contact wrench $(\lambda^x, \lambda^y, \lambda^z, \lambda^{mx}, \lambda^{my}, \lambda^{mz})$ to lie in a linearized Contact Wrench Cone (CWC) $\mathcal{K}$, as in [31].

B. Digit Control: Weighted Operational Space QP

A weighted QP controller akin to [25] is used to command suitable inputs to Digit's actuators so that the following locomotion objectives are achieved; see [27] for more details.

1) 3D-LIP following: The first objective is twofold; first, to realize the suggested foot placement u_k obtained by the most recent solution of the MPC, and second, to impose the trajectory of the LIP on Digit's COM. The MPC is solved within the $k-1$ step, $t \in [t_{k-1}, t_k)$, by using feedback from

Digit to predict the state $\mathrm{x}_{\mathrm{init}}$ of the LIP at the beginning of the next step. Given the result $\mathrm{u}_k = \begin{bmatrix} u_k^x & u_k^y & u_k^\theta \end{bmatrix}^\mathsf{T}$ of the MPC, we define the desired trajectory for the position of the swing foot as in [32]. For the orientation of the swing foot, the desired trajectory is defined in quaternion coordinates by interpolating between the initial and final orientations of the foot using spherical linear interpolation as in [27].

Next, we define the desired trajectory for Digit's torso based on updated predictions of the continuous-time trajectory of the LIP. At time t_{k-1}, we use robot data and (1) to predict the LIP's trajectory within the step, which defines the desired trajectory for the torso's COM. Note that the LIP trajectory prediction can be updated within the current step $[t_{k-1}, t_k)$, e.g., each time the MPC is solved; this way, more recent information from Digit is incorporated, resulting in improved robustness. Finally, to maintain Digit's torso upright and pointing towards the direction of the motion, we command zero pitch and roll angles, and the desired yaw angle is given by $(\phi^z_{\mathrm{st}} + \phi^z_{\mathrm{sw}})/2$, where ϕ^z_{st} and ϕ^z_{sw} are the yaw angles of the support and swing feet, respectively.

Then, if $\eta_1 \in \mathbb{R}^{12}$ is the error from the desired trajectory [27], we define the performance index associated with following the LIP-based trajectory as $\Psi_1 = \|\ddot{\eta}_1 + K_{\mathrm{D}}\dot{\eta}_1 + K_{\mathrm{P}}\eta_1\|^2$, where K_{P} and K_{D} are gains.

2) Angular momentum: Due to its point mass, the LIP cannot capture the effect of the angular momentum on Digit. To mitigate this effect, we compute the angular momentum about Digit's COM $\eta_2 = A_{\mathrm{G}}^{\mathrm{ang}} v$, where $A_{\mathrm{G}}^{\mathrm{ang}}$ is the angular part of the centroidal momentum matrix, and minimize $\Psi_2 = \|\dot{\eta}_2 + K_{\mathrm{G}}\eta_2\|^2$, where K_{G} is a gain matrix.

3) Arms desired configuration: Given that Ψ_1 does not impose any restriction on the motion of the robot's arms, minimizing the angular momentum according to Ψ_2 may result in undesirable arm movements. To avoid this, we define the cost $\Psi_3 = \left\|\ddot{\eta}_3 + K_{\mathrm{D,a}}\dot{\eta}_3 + K_{\mathrm{P,a}}(\eta_3 - \eta_3^{\mathrm{des}})\right\|^2$, where η_3 is the configuration of the left and right arms, η_3^{des} its desired value, and $K_{\mathrm{P,a}}$, $K_{\mathrm{D,a}}$ are suitable gains.

4) Input effort: To attenuate occasional spikes in the control signal, we minimize $\Psi_4 = \|\tau\|^2$.

5) Weighted OSC-QP: As in [25], [33], the aforementioned performance indices are combined in a single cost function with weights that reflect their respective level of importance, resulting in the following QP:

$$
\begin{aligned}
\underset{\dot{v},\tau,\lambda}{\text{minimize}} \quad & \psi_1\Psi_1 + \psi_2\Psi_2 + \psi_3\Psi_3 + \psi_4\Psi_4 \\
\text{subject to} \quad & (16) && \text{(Dynamics)} \\
& lb_\tau \le \tau \le ub_\tau && \text{(Torque Limits)} \\
& (\lambda^x, \lambda^y, \lambda^z, \lambda^{mx}, \lambda^{my}, \lambda^{mz}) \in \mathcal{K} && \text{(CWC)}
\end{aligned}
$$

In the cost, higher priorities are assigned larger weights. This QP is solved given (q, v) at every step of the control loop.

V. Results

We use MuJoCo as our simulation environment with simulation loop running at 2kHz and test our method in various scenarios. Here, due to space limitations, we report results for the $(10\mathrm{m} \times 10\mathrm{m})$ map with 10 static obstacles shown in Fig. 7; the accompanying video contains more examples. We also consider a moving circular obstacle of radius $0.5\mathrm{m}$, the motion of which is assumed to be known (cf. Fig. 7(a)). We begin by obtaining an ordered collection $(\mathcal{H}_i, \mathrm{w}_{i+1})$, $i = 1, ..., M-1$ of free polytopes and waypoints using Algorithm 1. Then, we proceed with solving the corresponding sequence of MPCs and the low-level QP. As in Section III-C, we use the interior point solver shipped with CasADi [30] with the same parameters as those used to obtain Fig. 5 and horizon $N = 3$. The average runtimes are of the order of 10ms (cf. Fig. 5). Thus, u_k can be updated frequently *within* each step in the light of more recent information from Digit, greatly improving robustness to external forces and model mismatch, and drastically reducing the tracking error. We limit the frequency of MPC to 15Hz owing to the good tracking performance of the QP, which is solved using a pre-shipped solver with CVXPY [34] at 400Hz.

Figure 7(a) shows Digit successfully avoiding static and moving obstacles in the environment considered. Note that the moving obstacle constraint is taken into account in the corresponding MPC when the obstacle is less than $5\mathrm{m}$ away from the robot. This slightly increases the average time for solving the MPC to $\approx$ 12ms. We also observe that the tracking error remains small (cf. Fig. 7(b)); the error increases when turning is required to avoid the moving obstacle, but never exceeds $3.5\mathrm{cm}$. Additionally, we tested robustness to unexpected external forces applied at Digit's torso. The x, y and z components of the force were sampled uniformly over $[-50, 50]$ N and the force was applied for 100ms at random time instants separated at most 2s. We conducted 30 trials, one out of which resulted in a failure; Fig. 7(c) shows the remaining 29 successful trials. Owing to the ability to solve the MPCs at sufficiently high frequency, the robot was able to maintain balance and complete the objective of reaching the goal without collisions.

Figure 7(d) considers the same scenario as Fig. 7(a) but with a SC computed using the method in [22]. The SC is narrower than the one obtained using Algorithm 1, causing the MPC to become infeasible. This illustrates the importance of effective free-space utilization, as larger SCs provide more free space for the robot to maneuver while avoiding moving obstacles.

VI. Conclusion

We presented a sequential MPC framework for bipedal robots navigating through complex environments with moving obstacles. The main idea was to decompose the relevant free space as a sequence of mutually intersecting polytopes. This way a safe walking corridor was created and used to define a sequence of MPC programs, where collision avoidance is certified via discrete-time CBFs. We tested our framework in high-fidelity simulations with the bipedal robot Digit, demonstrating robust reactive obstacle avoidance in highly cluttered environments with moving obstacles. Finally, disadvantages of the proposed approach in its current form are that, first, our formulation of the MPC programs is nonconvex due to reachability and moving obstacles constraints, second, the

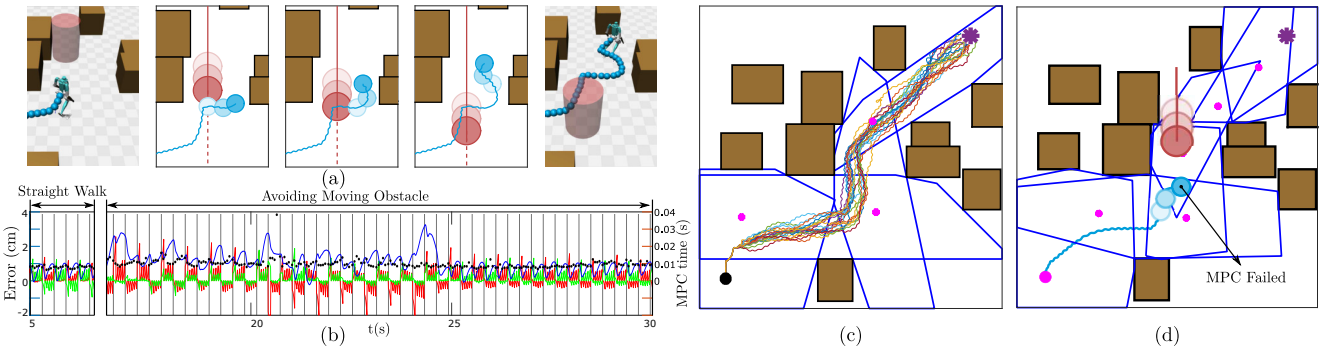


Figure 7. (a) Snapshots of Digit (blue circle) traversing an obstacle course while avoiding an obstacle moving at $0.3\mathrm{m/s}$ (red circle). For clarity, the middle tiles present the top view of only part of the environment. The full environment and the SC obtained by Algorithm 1 are shown in (c). (b) Tracking error between the MPC path and Digit's COM; errors in the x, y and z axes are in red, green, and blue, respectively. Black dots show the times at which an MPC solution becomes available. (c) Digit's COM paths under randomly applied 3D forces for the same environment and SC as in (a) but without the moving obstacle. (d) Same scenario as in (a) but with a narrower SC computed using [22] showing failure of the robot to complete the task.

resulting paths are suboptimal, and, third, collision avoidance does not extend to the full continuous motion. Future work will focus on addressing these issues.

REFERENCES

[1] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *Proc. IEEE Int. Conf. Humanoid Robots*, 2012, pp. 674–679.

[2] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Proc. IEEE Int. Conf. Humanoid Robots*, 2014, pp. 279–286.

[3] 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 *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, 2003, pp. 1620–1626 vol.2.

[4] R. Tedrake, S. Kuindersma, R. Deits, and K. Miura, "A closed-form solution for real-time ZMP gait generation and feedback stabilization," in *Proc. IEEE Int. Conf. Humanoid Robots*, 2015, pp. 936–940.

[5] S. Faraji, S. Pouya, and A. Ijspeert, "Robust and agile 3D biped walking with steering capability using a footstep predictive approach," in *Proc. Robot. Sci. Syst.*, 2014.

[6] S. Xin, R. Orsolino, and N. Tsagarakis, "Online relative footstep optimization for legged robots dynamic walking using discrete-time model predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 513–520.

[7] A. Tanguy, D. De Simone, A. I. Comport, G. Oriolo, and A. Kheddar, "Closed-loop MPC with dense visual SLAM - stability through reactive stepping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 1397–1403.

[8] M. S. Motahar, S. Veer, and I. Poulakakis, "Composing limit cycles for motion planning of 3D bipedal walkers," in *Proc. IEEE Conf. Decis. Control*, 2016, pp. 6368–6374.

[9] S. Veer, M. S. Motahar, and I. Poulakakis, "Almost driftless navigation of 3D limit-cycle walking bipeds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5025–5030.

[10] S. Veer and I. Poulakakis, "Safe adaptive switching among dynamical movement primitives: Application to 3D limit-cycle walkers," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 3719–3725.

[11] S. Veer, Rakesh, and I. Poulakakis, "Input-to-state stability of periodic orbits of systems with impulse effects via Poincaré analysis," *IEEE Trans. Autom. Control*, vol. 64, no. 11, pp. 4583–4598, 2019.

[12] S. Veer and I. Poulakakis, "Switched systems with multiple equilibria under disturbances: Boundedness and practical stability," *IEEE Trans. Autom. Control*, vol. 65, no. 6, pp. 2371–2386, 2020.

[13] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2001, pp. 239–246.

[14] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A reactive walking pattern generator based on nonlinear model predictive control," *IEEE Robot. Automat. Lett.*, vol. 2, no. 1, pp. 10–17, 2017.

[15] X. Xiong, J. Reher, and A. D. Ames, "Global position control on underactuated bipedal robots: Step-to-step dynamics approximation for step planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 2825–2831.

[16] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *Proc. Amer. Control Conf.*, 2021, pp. 3882–3889.

[17] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Proc. Robot. Sci. Syst.*, 2017.

[18] S. Teng, Y. Gong, J. W. Grizzle, and M. Ghaffari, "Toward safety-aware informative motion planning for legged robots," *arXiv:2103.14252*, 2021.

[19] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI*, H. L. Akin, N. M. Amato, V. Isler, and A. F. van der Stappen, Eds. Springer, 2015, pp. 109–124.

[20] ——, "Efficient mixed-integer planning for UAVs in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 42–49.

[21] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1476–1483.

[22] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1688–1695, 2017.

[23] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 344–351.

[24] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-Repeat-Replan: A complete and robust system for aggressive flight in complex environments," *IEEE Trans. Robot.*, vol. 36, no. 5, pp. 1526–1545, 2020.

[25] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," *ACM Trans. Graph.*, vol. 29, no. 4, p. 71, 2010.

[26] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4459–4466, 2019.

[27] K. S. Narkhede, A. M. Kulkarni, D. A. Thanki, and I. Poulakakis, "A sequential MPC approach to reactive planning for bipedal robots," *arXiv:2205.00156*, 2022.

[28] R. Deits, "iris-distro," https://github.com/rdeits/iris-distro, 2015.

[29] S. Liu, "Decomputil," https://github.com/sikang/DecompUtil, 2017.

[30] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi-A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[31] S. Caron, Q.-C. Pham, and Y. Nakamura, "Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5107–5112.

[32] Y. Gong and J. W. Grizzle, "Zero dynamics, pendulum models, and angular momentum in feedback control of bipedal locomotion," *arXiv:2103.14252*, 2021.

[33] T. Apgar, P. Clary, K. R. Green, A. Fern, and J. W. Hurst, "Fast online trajectory optimization for the bipedal robot Cassie," in *Proc. Robot., Sci. Syst.*, 2018.

[34] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.