Real-time Dynamic Bipedal Avoidance

Tianze Wang, Jason White, and Christian Hubicki

***Abstract*— In real-world settings, bipedal robots must avoid collisions with people and their environment. Further, a biped can choose between modes of avoidance: (1) adjust its pose while standing or (2) step to gain maneuverability. We present a real-time motion planner and multibody control framework for dynamic bipedal robots that avoids multiple moving obstacles and automatically switches between standing and stepping modes as necessary. By leveraging a reduced-order model (i.e. Linear Inverted Pendulum Model) and a half-space relaxation of the safe region, the planner is formulated as a convex optimization problem (i.e. Quadratic Programming) that can be used for real-time application with Model-Predictive-Control (MPC). To facilitate mode switching, we introduce center-of-pressure related slack-variables to the convex planning optimization that both shapes the planning cost function and provides a mode switching criterion for dynamic locomotion. Finally, we implement the proposed algorithm on a 3D Cassie bipedal robot and present hardware experiments showing real-time bipedal standing avoidance, stepping avoidance, and automatic switching of avoidance modes.**

I. Introduction

As robots are increasingly deployed in real-world environments, efficient collision-free path-planning becomes essential for unstructured environments. Legged systems specifically have multiple locomotion modes (i.e. stepping or standing), which further complicates this problem. While standing, a robot's avoidance capabilities are limited due to its fixed feet positions. Therefore, intelligent switching between stance and stepping is necessary for autonomy in practical applications Fig. 1A. Further, as legged systems and environments become more dynamic, motion planning algorithms must be expedited. In this paper, we present a real-time model-predictive controller (MPC) for dynamic bipedal locomotion that avoids moving obstacles while standing, stepping, and automatically switching between those modes Fig. 1.

A. Related Work

Extensive research has been done to study collision-free motion planning algorithms on platforms including drones [1], [2], [3], mobile robots [4], and legged systems [5], [6]. In [3], the motion plan is formulated as a Mixed-Integer Programming Problem using a convex relaxation. In [2], a half-space relaxation is used to find collision-free spaces. Barrier Functions (CBFs) are a common control technique that ensures safety and have been shown to achieve collision-free navigation in [4], [7].

*Toyota Research Institute provided funds to support this work.

[1]All authors are with the Department of Mechanical Engineering, Florida State University, FAMU-FSU College of Engineering, Tallahassee, FL 32310, USA. Corresponding author: tw19j@fsu.edu

Supporting Video: https://youtu.be/mf_Ugy2cJQU

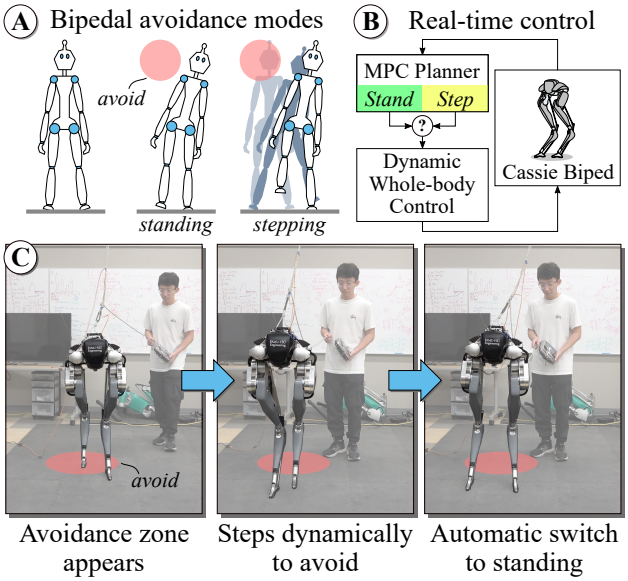

Fig. 1: (**A**) Bipeds have the option of avoiding by either standing or stepping. (**B**) This control and planning framework automatically switches between these modes as necessary (**C**) allowing bipeds to dynamically avoid obstacles.

For legged robots, much of the research addresses collision avoidance as a navigation and safety problem [8], [9]. Most approaches utilize a hierarchical control framework. High-level commands search for a feasible walking direction, then a mid-level planer finds feasible foot placements and CoM trajectory, and the low-level controller tracks the desired foot and body motion.

A high-level (i.e. global) planner develops a collision-free path using a map constructed by camera/LIDAR data [10]. A Control Lyapunov Function (CLF) with rapidly-exploring random tree (RRT) as in [5] is one approach to finding a path. Collision avoidance can be achieved by defining high-level safety behaviors and vision data as seen in [11]. In [9], the high-level task planner employs linear temporal logic (LTL) to guarantee navigation safety. However, the high-level planner has to run at low frequency due to its complexity and the resultant motion plan might not be dynamically or kinematically achievable.

Mid-level (i.e. local) planners serve to generate dynamically and kinematically feasible paths that are local to the high-level plan. In [8], a nonlinear trajectory optimization with collision avoidance is included to guarantee safety in a static environment. Collision avoidance can also be incorporated as kinematics constraints [12] when planning using centroidal dynamics and full-body kinematics. In [13], an MPC with collision avoidance based on distance constraints

was implemented in simulation. In [14], a Discrete-CBF-MPC based framework was implemented to ensure the robot stays in the safe set in simulation.

If formulated with tractable models, mid-level planners can solve at faster rates than high-level planners. Specifically, reduced-order models (ROMs) are a popular approach to fast planning due to their simplicity and comparatively few system states. Commonly used ROMs include the linear inverted pendulum[15], spring inverted pendulum model [16], and their variations. In[17] generated walking behaviors using a 3D actuated Spring Loaded Inverted Pendulum (3D-aSLIP) via the Hybrid Linear Inverted Pendulum (H-LIP) based stepping controller. In [18], the authors implemented a terrain-adaptive MPC based on the angular-momentum-based LIP (ALIP). ROMs also motivate bipedal research like dynamic balancing [19] and push-recovery analysis using capture-point based concepts [20], [21], [22].

While plenty ROM-based methods have been proposed to achieve successful real-time bipedal avoidance, doing so for both standing and stepping with avoidance mode switching remains to be explored. Many real-world applications require a robot to stand in place while performing a task. It is important for the robot to decide between: (1) remain standing and adjust its pose and (2) begin stepping to achieve agile maneuvers when an impending collision is predicted. In this paper, we aim to augment ROM-based motion planning algorithms with the ability to avoid moving obstacles and choose appropriate avoidance modes. Further, we seek to demonstrate that this motion plan is practical for full-body bipedal controllers, such as Operational Space Control [23].

B. Contributions

The contributions of this paper are:

- Bipedal avoidance of multiple dynamic obstacles.
- Novel bipedal MPC formulation with center-of-pressure slack variables used for discouraging/signalling mode switches as needed.
- Procedure for autonomous mode switching (i.e. standing and walking) to avoid obstacles.
- Implementation of MPC and whole-body control on a Cassie series bipedal robot demonstrating dynamic bipedal avoidance in real-time.

II. Methods

Here we present our method of generating dynamic bipedal motion plans that avoid moving obstacles using a convex model predictive controller (MPC). This section is structured as follows: A basic ROM-based planning framework for bipedal locomotion is introduced in Sec. II-B.1. We then augmented the framework with the ability to: (1) generate collision-free paths with multiple moving obstacles in Sec. II-B.2 and (2) switch between standing avoidance and stepping avoidance as needed in Sec. II-C.

A. Conventions

The following notation is used throughout the paper: $\mathbb{R}, \mathbb{R}^n, \mathbb{R}^{n\times m}$ represent the space of real numbers, vectors with length n, and matrices with n rows and m columns. Scalar values are lowercase and italicized letters (e.g. x). Bold, lowercase letters represent vectors (e.g. $\mathbf{q} \in \mathbb{R}^n$) while bold, uppercase letters represent matrices (e.g. $\mathbf{A} \in \mathbb{R}^{n\times m}$). Variables ($\mathbf{p}$, $\mathbf{v}$, $\mathbf{u}$) are positions, velocities, and control inputs respectively. Finally, we define the robot state vector as $\mathbf{q} = [\mathbf{p}; \mathbf{v}]$.

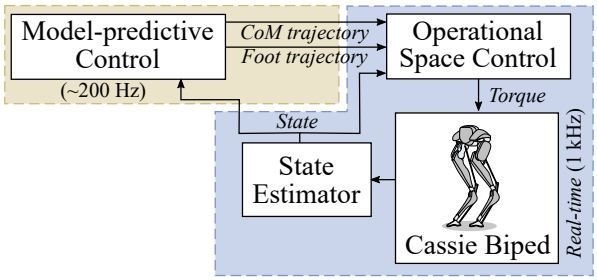

Fig. 2: Control block diagram for the Cassie Biped: 1. MPC computes a collision-free CoM and Feet trajectory at 200Hz. 2. The Operational Space Controller computes the torques necessary to track the MPC reference at 1kHz. 3. An EKF is implemented to estimate pelvis states.

B. Model Predictive Control

Model Predictive Control (MPC) is an online optimization-based controller that generates trajectories of system states and inputs that minimize a cost while satisfying constraints. The quadratic MPC formulation presented here includes additional terms for obstacle avoidance and mode switching, similar to [2] and takes the general form:

$$\min_{\mathbf{q}_k, \mathbf{u}_k, \rho_k, \mathbf{s}_k} \quad \underbrace{J(\mathbf{q}, \mathbf{u}, \mathbf{s})}_{\text{Tracking Cost}} + \sum_{k=1}^{N} \underbrace{||\rho_k||^2_{\mathbf{W}_{\text{slack}}}}_{\text{Slack Cost}}$$

$$\begin{aligned}
\text{s.t.} \quad & \dot{\mathbf{q}}_k = \mathbf{A}\mathbf{q}_k + \mathbf{B}\mathbf{u}_k && \text{(Dynamics)} \\
& \mathbf{q}_{k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_k \Delta T && \text{(Model Prediction)} \\
& f(\mathbf{q}_k, \rho_k, \mathbf{u}_k) \leq 0 && \text{(Slack Constraints)} \\
& h(\mathbf{q}_k, \mathbf{u}_k) \leq 0 && \text{(Task Constraints)}
\end{aligned}$$

where the robot states $\mathbf{q}$, control input $\mathbf{u}$, step length $\mathbf{s}$, and all slack variables ρ are the decision variables in the optimization. This formulation only includes quadratic costs and linear constraints to allow for fast convergence and real-time application.

1) MPC for Bipedal Locomotion:

We first present a general bipedal MPC formulation for stepping control.

a) Bipedal Model:

The Linear Inverted Pendulum Model (LIPM) is a reduced-order model which describes the essential dynamics of bipedal walking [15]. It models bipedal walking in a linear form by only modeling Center of Mass (CoM) states with massless legs and assuming constant walking height. Due to its linear formulation, it has been widely used for real-time trajectory planning of legged locomotion. Thus, we use

LIPM as our bipedal model throughout the paper. We define the motion of the CoM with respect to the contact point in Cartesian space (x,y) as shown in Fig. 3. The dynamics of the LIPM are:

$$\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{M} & \mathbf{0} \end{bmatrix} \mathbf{q} - \begin{bmatrix} \mathbf{0} \\ \mathbf{M} \end{bmatrix} \mathbf{u}; \quad \mathbf{M} = \begin{bmatrix} \frac{g}{z_0} & 0 \\ 0 & \frac{g}{z_0} \end{bmatrix} \tag{1}$$

where (g) is the gravity term, (z_0) is the reference walking height, ($\mathbf{q}$) and ($\mathbf{u}$) are predicted CoM states and control inputs. The control input $\mathbf{u}$ represents the center of pressure which can be treated as the robot's foot position in the LIPM.

b) Stepping Constraints and Costs:

Additional provisions are taken to ensure the model steps where we expect while maintaining stability. We include a cost related to the robot foot position change that drives the foot to the desired location. The reference stride length is calculated based on a commanded input. We denote s_x as the step length in the sagittal plane and s_y as the step length in the frontal plane. Note the stride length cost formulations in the x and y directions are different and we will discuss them separately. The step length cost function in the x direction is defined as:

$$J^x_{\text{step}}(\mathbf{s}) = \sum_{k=1}^{N_s} ||s^x_k - s^x_{\text{ref},k}||^2_{\mathbf{W}_{\text{step}}} \tag{2}$$

where the reference step length is the product of the desired walking speed ($\dot{s}^x_{\text{ref}}$) and stepping time (T_{step}). N_s is the number of total steps taken and the relationship between the number of steps N_s with the total number of MPC nodes is $N = N_\mathrm{s}\frac{T_{\text{step}}}{\Delta T}$.

For the frontal plane, the controller freely chooses a step length provided the step does not result in a cross-step or self-collision. To enforce this, a slack variable ρ penalizes this behavior and is only non-zero only when $s^y_\mathrm{k} \leq s^y_{\text{ref}}$ where the reference step length s^y_{ref} in the y direction depends on the desired velocity and the nominal distance between the foot and the robot CoM.

$$J^y_{\text{step}}(\rho) = \sum_{k=1}^{N} ||\rho_{\text{step},k}||^2_{\mathbf{W}_{\text{step}}} \tag{3}$$

$$s^y_k \geq \rho_{\text{step},k} + s^y_{\text{ref}} \tag{4}$$

Note, Eq. 4 only holds when the left foot is the swing foot and should be replaced with $s^y_k \leq \rho_{\text{step},k} - s^y_{\text{ref}}$ for a right step.

We then compute the desired sagittal plane center of pressure reference trajectory $\mathbf{u}_{\text{ref,x}} \in \mathbb{R}^N$ using the foot length variable $\mathbf{s}$:

$$\mathbf{u}_{\text{ref,x}} = \begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \dots \\ \dots \\ \mathbf{1} \end{bmatrix} p_{\text{foot,x}} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots \\ \mathbf{1} & \mathbf{1} & \dots & \mathbf{1} \end{bmatrix} \begin{bmatrix} s_{\text{x},1} \\ s_{\text{x},2} \\ s_{\text{x},3} \\ \dots \\ s_{x,N_s} \end{bmatrix} \tag{5}$$

where $p_{\text{foot,x}}$ is the current foot position and $\mathbf{0}$, $\mathbf{1}$ are column vectors with length $N_r = \frac{T_{\text{step}}}{\Delta T}$. Note the frontal plane center of pressure reference trajectory $\mathbf{u}_{\text{ref,y}} \in \mathbb{R}^N$ has the identical formulation.

c) Tracking Cost: State and input tracking are also included in the cost function. The state cost function penalizes deviation from the target states:

$$J_{\text{target}}(\mathbf{q}) = \sum_{k=1}^{N} ||\mathbf{q}_k - \mathbf{q}_{\text{ref},k}||^2_{\mathbf{W}_{\text{target}}} \tag{6}$$

where the target state ($\mathbf{q}_{\text{ref}}$) has a weight matrix ($\mathbf{W}_{\text{target}}$). The target state is generated from a user (e.g. standing in place, stepping in place, or walking with a desired velocity). The control input cost is formulated in the same manner:

$$J_{\text{effort}}(\mathbf{q}) = \sum_{k=1}^{N} ||\mathbf{u}_k - \mathbf{u}_{\text{ref},k}||^2_{\mathbf{W}_{\text{effort}}} \tag{7}$$

where the foot position reference $\mathbf{u}_{\text{ref},k}$ encodes a preferred step length as seen in Sec. II-B.1.b.

d) Reachability Constraints:

Finally, we include a reachability constraint to ensure the planned foot locations are within the kinematic limits of the leg and is formulated in the following linear form,

$$-\mathbf{r}_{\max} \leq h(\mathbf{q}_k, \mathbf{u}_k) := (\mathbf{u}_k - \mathbf{q}_k) \leq \mathbf{r}_{\max} \tag{8}$$

where $\mathbf{r}_{\max}$ is the maximum distance the feet can be from the robot body.

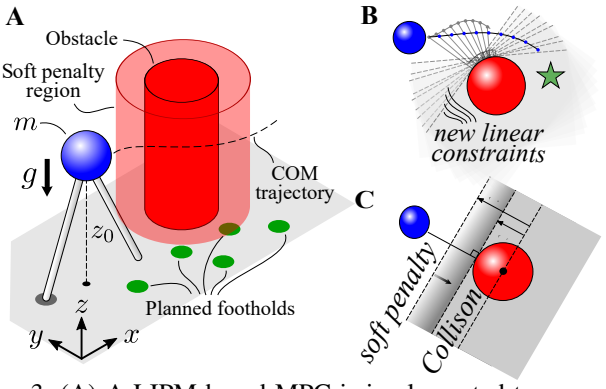

Fig. 3: (A) A LIPM-based MPC is implemented to generate collision-free trajectories. (B) The approximation is improved by linearizing constraints around a trajectory from the previous MPC solution. (C) We used a half-space relaxation with slack variables to model the collision-free regions.

2) Obstacle Avoidance Constraints and Costs:

In this section, we introduce our obstacle avoidance formulation. Choosing a collision-free path is complicated by the non-convexity introduced by the Euclidean distance constraint. In [2], the nonlinear Euclidean distance formula is relaxed by using a linear approximation. As stated in the paper, this formulation can be easily adapted for multi-object avoidance in both 2D and 3D space. Here we introduce the method briefly.

Given the robot initial states ($\mathbf{p}_0$), obstacle states ($\mathbf{p}_{\text{obs}}$), and safe distance (r), we define the collision region as a ball with center $\mathbf{p}_{\text{obs}}$ and radius r. We then find the intersection

point $\mathbf{p}_{\text{inter,k}}$ on the boundary of the collision region that is closest to the agent position $\mathbf{p}_0$. Finally, we compute the tangent plane at the intersection point which separates the space into safe and unsafe regions. The safe region should satisfy the following constraint:

$$\hat{\mathbf{d}}_k \cdot (\mathbf{p}_{\text{inter},k} - \mathbf{p}_k) \leq 0. \tag{9}$$

where $\mathbf{p}_k$ is the robot position prediction and $\hat{\mathbf{d}}_k$ is the normal vector of the tangent plane which can be computed by normalizing the vector $(\mathbf{p}_0 - \mathbf{p}_{\text{obs}})$. This hard constraint is only a rough estimate of the collision-free region, and commonly leads to feasibility issues. To resolve this, a slack variable $\rho_{\text{avoid},k}$ is introduced to make avoidance as a soft constraint Fig. 3C. A piecewise quadratic cost term penalizes $\rho_{\text{avoid},k}$ and drives the agent away from the obstacle only when it is inside the danger region. The constraint and cost are defined as:

$$J_{\text{avoid}}(\rho) = \sum_{k=1}^{N} ||\rho_{\text{avoid},k}||^2_{\mathbf{W}_{\text{avoid}}} \tag{10}$$

$$\hat{\mathbf{d}}_k \cdot (\mathbf{p}_{\text{inter},k} - \mathbf{p}_k) - \rho_{\text{avoid},k} \leq 0. \tag{11}$$

The estimation accuracy is further improved by linearizing the constraint around a trajectory instead of the agent's initial position Fig. 3B. This approach is implemented using the robot's trajectory from the previous MPC solution. For moving obstacles, we develop a rough estimate the obstacle's trajectory based on its current position and velocity when such data is available.

C. Double-Support and Single-Support Switching

Here we discuss the unified formulation which allows for autonomous switching between stance and stepping. The cost function discussed in Sec. II-B.1.c is expanded to include a slack variable used in both standing (Double-Support Phase) and stepping (Single-Support Phase):

$$J_{\text{effort}}(\mathbf{u}) = \sum_{k=1}^{N} ||\rho_{\text{input},k}||^2_{\mathbf{W}_{\text{effort}}} \tag{12}$$

$$||\mathbf{u}_k - \mathbf{u}_{\text{ref},k}|| \leq \rho_{\text{input},k} + \mathbf{u}_{\text{off}}. \tag{13}$$

The $\mathbf{u}_{\text{off}}$ variable is an offset parameter that represents the center of pressure boundary. For the LIPM, we can assume $\mathbf{u}_{\text{off}} = 0$ since the LIPM assumes point foot contact. However, for physical bipedal robots, $\mathbf{u}_{\text{off}}$ depends on the support region defined by its feet in stance. This formulation penalizes the objective function when the center of pressure is outside of the support polygon.

While the overall framework remains the same between standing and stepping, the MPC parameters are slightly different:

- During stance, the step length variable ($\mathbf{s}$) defined in Sec. II-B.1.b is 0 since foot positions do not change.
- $\mathbf{u}_{\text{off,step}} < \mathbf{u}_{\text{off,stand}}$ since bipeds have a larger support polygon during standing.

The value of ρ_{input} can be used to determine if stepping is necessary for avoiding an obstacle. We solve the standing MPC to obtain the reference trajectory during stance. Given a non-zero $\mathbf{u}_{\text{off}}$, the standing MPC slack variable should be small when the motion necessary to avoid an obstacle is minimal. As the robot body moves further to the side for avoidance and reaches the physical limit, the center of pressure $\mathbf{u}$ will finally cross the boundary $\mathbf{u}_{\text{off}}$ and ρ_{input} will start to increase accordingly. Once ρ_{input} crosses an upper threshold θ_{ub}, the command center of pressure $\mathbf{u}$ will not be feasible with the current foot placement, indicating making a step is necessary for avoidance. During the stepping phase, we solve the standing and stepping MPCs simultaneously. The stepping MPC is solved to balance the robot by providing the reference signal, while the standing MPC evaluates whether it is safe to switch back to standing with the current foot placement. As the obstacle moves away, the standing MPC slack variable ρ_{input} will decrease and the robot can switch back to standing when ρ_{input} is below a lower threshold θ_{lb}. Note, this method also works with unexpected disturbances caused by external forces (e.g. push-recovery analysis). By using this slack variable formulation instead of enforcing hard constraints on the control inputs $\mathbf{u}$, we are able to avoid feasibility issues that might result from large disturbances or obstacles.

D. Operational Space Controller

Trajectory tracking is achieved with an Operational Space Controller (OSC). Generally, the OSC computes a set of torques that satisfies constraints like dynamics and friction cone constraints while minimizing the tracking error in the task space. For legged robots, the tasks are usually defined as CoM tracking and foot position tracking. Given the CoM and foot reference from the MPC, we use a simple PD control law to compute the desired task space acceleration:

$$\ddot{\mathbf{x}}_{\text{cmd}} = \ddot{\mathbf{x}}_{\text{des}} + \mathbf{K}_{\text{p}}(\mathbf{x}_{\text{des}} - \mathbf{x}_{\text{cur}}) + \mathbf{K}_{\text{d}}(\dot{\mathbf{x}}_{\text{des}} - \dot{\mathbf{x}}_{\text{cur}}) \tag{14}$$

where $\ddot{\mathbf{x}}_{\text{des}}, \dot{\mathbf{x}}_{\text{des}}, \mathbf{x}_{\text{des}}$ are desired task space acceleration, velocity, and position respectively.

The optimal control problem is formulated as:

$$\begin{aligned}
\min_{\ddot{\mathbf{q}},\mathbf{u},\mathbf{f},\lambda} \quad & ||\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_{\text{cmd}}||^2_{\mathbf{W}} \\
\text{s.t.} \quad & \mathbf{M}\ddot{\mathbf{q}} + \mathbf{c} = \mathbf{B}\mathbf{u} + \mathbf{J}_{\text{f}}^T\mathbf{f} + \mathbf{J}_{\text{s}}^T\boldsymbol{\tau} && \text{(15a)} \\
& \ddot{\mathbf{x}} = \mathbf{A}\ddot{\mathbf{q}} + \dot{\mathbf{A}}\dot{\mathbf{q}} && \text{(15b)} \\
& \mathbf{u}_{\text{l}} \leq \mathbf{u} \leq \mathbf{u}_{\text{u}} && \text{(15c)} \\
& ||f^i_{\text{x}}|| + ||f^i_{\text{y}}|| \leq \mu||f^i_{\text{z}}|| \ , \ f^i_{\text{z}} \geq 0 && \text{(15d)} \\
& \mathbf{J}_{\text{s}}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{\text{s}}\dot{\mathbf{q}} = 0 && \text{(15e)}
\end{aligned}$$

where $(||\cdot||^2_{\mathbf{W}})$ is the weighted 2-norm cost, (15a) is the Rigid-Body Dynamics Constraint, (15b) computes the predicted task space acceleration where $\mathbf{A}, \dot{\mathbf{A}}$ represent the task space Jacobian matrix and its time derivative, (15c) limits the torques applied, (15d) is the approximated friction cone constraint for stance leg where i is the contact point index, and (15e) represents the robots kinematics constraints from closed-loop linkages in the robot morphology.

III. Simulation Results

A. Cassie

Tallahassee Cassie is a Cassie series bipedal robot designed and built by Agility Robotics. The floating-based model of the robot has 20 DoF. There are seven joints in each leg and each leg contains 5 actuated joints and 2 passive joint realized via a four-bar linkage with a mechanical spring. For our simulations, we used the Matlab Simulink model provided by Agility Robotics.

B. Controller Implementation

This section details the Cassie controller framework which consists of three components: a state estimator, a trajectory generator, and a tracking controller (shown in Fig. 2).

1) State Estimation:

A contact-aided Kalman filter based on [24] was developed in Matlab to estimate the pelvis translational and rotation states. The spring deflections are used to determine which foot is in contact.

2) QP Solver:

We implemented the proposed MPC and OSC in Matlab. The MPC and OSC QP are solved using OSQP, an operator splitting solver for quadratic programs [25]. We did not tune any OSQP parameters in the simulation.

3) Rigid Body Dynamics:

In OSC, we implemented the composite-rigid-body algorithm by Roy Featherstone [26] to compute the Rigid-Body Dynamics of Cassie shown in Eq. 15a. The algorithm was coded in Matlab and transferred to C++ using Matlab Coder for faster compilation and computation purposes.

4) MPC Setup:

We used the following MPC parameters through all simulation experiments where the MPC planning horizon is set to four steps ($N_s = 4$) and the step period is 0.4 seconds. Currently, we set the MPC discretization time to be only 0.1 seconds so that the MPC can be solved quickly (around 1 ms) while still providing good results. The safe distance r is currently set based on the size and velocity of the obstacle.

C. Simulation Results

1) Result 1: Stepping and Walking with MPC:

We first showed that our MPC framework can be used in general bipedal locomotion. In this case, we did not include any obstacles and set the MPC obstacle avoidance cost to 0. We showed Cassie can achieve $1.5m/s$ stable forward-walking and $0.5m/s$ side-walking with the proposed framework Fig. 4.

2) Result 2: Stepping/Walking with Avoidance:

We used two simulation experiments to demonstrate Cassie's dynamic obstacle avoidance with our MPC framework. In the first test, Cassie tried to avoid stationary obstacles during forward walking. As shown in Fig. 5, Cassie reduced its walking speed and walked to the side to avoid the obstacle even though the user command is to walk straight forward at $1.0m/s$.

In the second test, Cassie tried to step at the target position while avoiding multiple obstacles. The obstacles moved toward Cassie from different directions. In Fig. 6, we observe that MPC found collision-free paths with the presence of multiple obstacles. After both obstacles moved away, Cassie stepped back to the target position. In both simulation experiments, we set the obstacle radius to $0.3m$ and the danger region radius to $0.9m$.

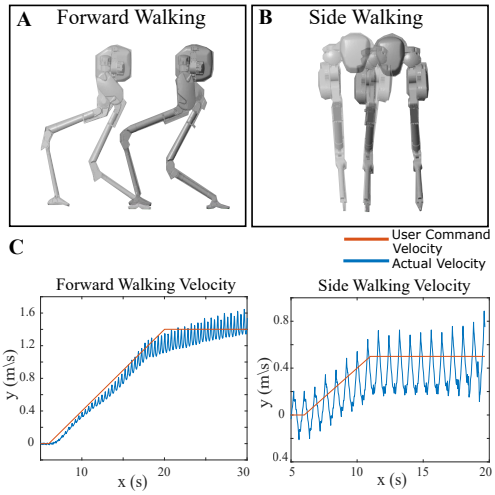

Fig. 4: (A) A stroboscopic sequence of Cassie in simulation walking forward at $1.5m/s$. (B) A stroboscopic sequence of Cassie in simulation walking sideways at $0.5m/s$. (C) Velocity plots show the high-level velocity command sent to MPC and Cassie's actual walking velocity in simulation.

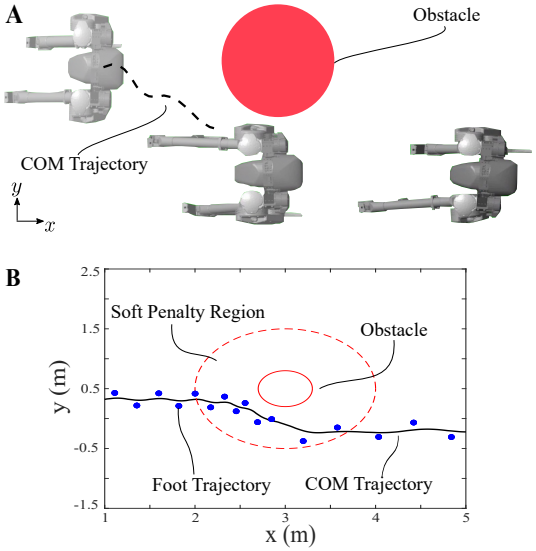

Fig. 5: (A) Cassie avoids stationary obstacles during forward walking. (B) Cassie's feet and COM trajectories from the simulation experiment.

3) Result 3: Automatic Mode Switching:

Then, we present our automatic mode-switching results Fig. 7. In this simulation, the standing and stepping MPC's target is to stabilize Cassie at the origin and we used the same obstacle setup as Sec. III-C.2. Cassie began by standing in place while an obstacle moved toward Cassie. The obstacle is $0.25m$ away from Cassie in the x direction and moves in the negative y direction. As the obstacle moved closer,

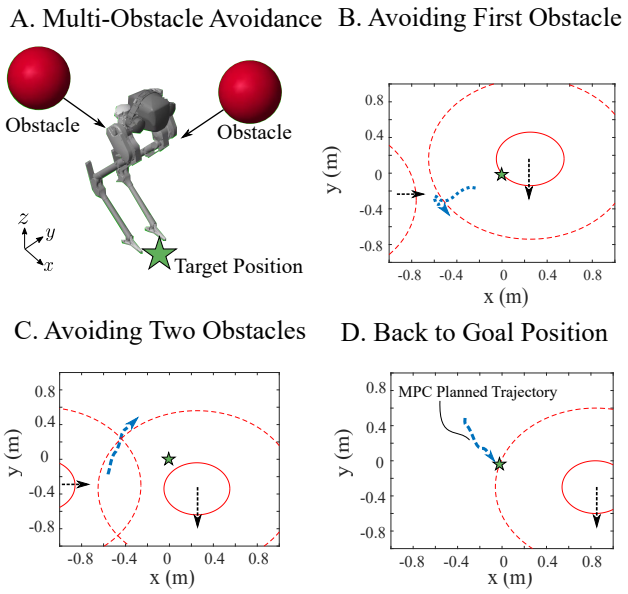

Fig. 6: (A) Cassie avoids multiple moving obstacles while trying to step toward a target position. (B) Cassie stepping away from the target position to avoid the first obstacle. (C) Cassie avoiding two obstacles simultaneously. (D) Cassie stepping back to the goal position.

the standing MPC slack variable increased and crossed the set threshold, θ_{ub}. Note, we only use the MPC predicted ρ_{input} of the current step. As a result, the controller switched to stepping mode. As shown in Fig. 7B, Cassie stepped backward to avoid the obstacle. As the obstacle moved away, Cassie began walking back to the origin and stopped stepping when the standing slack variable dropped below θ_{lb}. In the experiment, we set the control input offset $\mathbf{u}_{\text{off}}$ based on Cassie's standing feet positions and determined the slack variable threshold based on experimental data. In our case, we set $\mathbf{u}_{\text{off}} = \begin{bmatrix}0.05 & 0.1\end{bmatrix}$ and the threshold to be $\theta_{\text{ub}} = 0.04$, $\theta_{\text{lb}} = 0.001$. We want to point out that the sum of $\mathbf{u}_{\text{off}}$ and θ_{ub} is $\begin{bmatrix}0.09 & 0.14\end{bmatrix}$ which matches the actual contact region defined by Cassie's standing feet, denoting the feasible center of pressure regions.

4) Result 4: Weight Analysis:

Finally, we discuss the effect of each term in the cost function and provide a sensitivity analysis regarding the avoidance-related cost. The effects of J_{target}, J_{step}, J_{effort} are straightforward. J_{target} determines the state tracking performance while J_{step} affects the actual step width. For LIPM, the weight of J_{effort} should be infinity given LIPM assumes passive point-foot. For physical robots, this weight depends on the robot's capability of controlling the CoM position during the single-support phase.

Based on the mode switch experiment done in Sec. III-C.3, we further vary the avoidance cost J_{avoid} and switching threshold θ_{ub} without changing other terms and show the result as follows.

In general, larger $\mathbf{W}_{\text{avoid}}$ results in larger deviations from the desired states for avoidance, and larger θ_{ub} indicates more

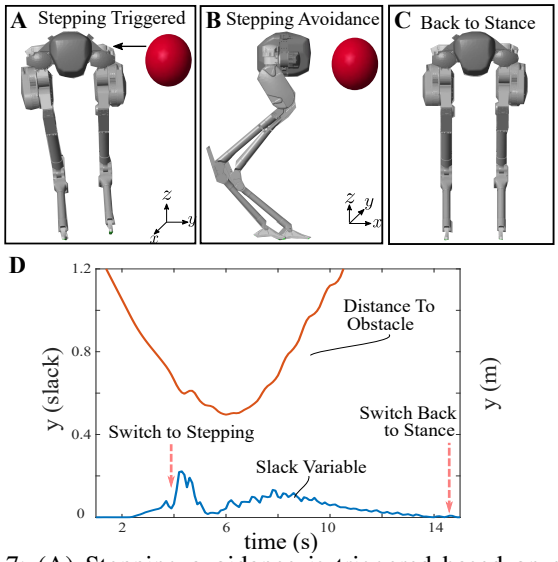

Fig. 7: (A) Stepping avoidance is triggered based on slack variable. (B) Cassie steps away from the obstacle. (C) Cassie switches back to standing based on the slack variable value. (D) The slack variable increases as the obstacle approaches and decreases back to 0 after the obstacle moves away and Cassie steps back to its original position.

θ_{ub} \ $\mathbf{W}_{\text{avoid}}$	500	1500	2500	3500	4500
0.01	F1	S2	S2	S2	S2
0.05	F1	S1	S2	S2	S2
0.1	F1	S1	S2	S2	S2
0.2	F1	S1	F2	F2	F2

TABLE I: (F1) Failure Case 1: Collision during the stance phase. (F2) Failure Case 2: Cassie loses balance during the stance phase. (S1) Success Case 1: Avoidance in the stance mode. (S2) Success Case 2: Avoidance in the stepping mode. There are 5 different weights $\mathbf{W}_{\text{avoid}}$ chosen based on the nominal state tracking cost $\mathbf{W}_{\text{target}} = 2500$.

capability of motion during the stance phase in the MPC.

Accordingly, we can find 4 different sets of behaviors in Table. I. First, when $\mathbf{W}_{\text{avoid}} = 500$, the MPC underestimates the necessity of avoidance and collision always occurs (F1). Then the MPC starts to realize the necessity of avoiding the obstacles as we increase $\mathbf{W}_{\text{avoid}}$. Cassie avoids the obstacle during the stance mode when the need for avoidance is relatively small (i.e. $\mathbf{W}_{\text{avoid}} = 1500$) (S1) and switches to stepping avoidance as the need ($\mathbf{W}_{\text{avoid}}$) further increases (S2). Finally, we found another failure case where the MPC drives Cassie to break the physical limit for avoidance during stance ($\mathbf{W}_{\text{avoid}} > 1500$ and $\theta_{\text{ub}} = 0.2$) (F2) because the MPC is overly optimistic about the robot's maneuverability during stance.

IV. Hardware Experiments

A. Experimental Setup

1) Software Setup:

We implemented the proposed algorithm on a physical Cassie bipedal robot. The primary computer (Target PC) on Cassie, run in Matlab, sensed the joint data and used the EKF

to estimate the pelvis states based on IMU measurements. The MPC and OSC were coded in C++ and run on a secondary computer with a Linux Environment. The two computers used UDP communication. The low-level tracking OSC is updated at 1kHz while the high-level MPC is currently updated at 200 Hz. We interpolate the trajectory from the high-level MPC to compute the reference for OSC.

2) Hardware Controller Tuning:

We used the same control framework in both simulation and hardware experiments. However, we noticed shaking issues with the swing foot and stance foot because of the spring attached to each leg and we solved the issue with the following changes:

- We reduce the task space derivative gains on the swing foot when computing the desired foot acceleration. However, other gains and weights in the QP remained the same between simulation and hardware experiments.
- We found that soft switching also reduced shaking (i.e. we limited the contact forces and foot torques that can be applied on the stance foot during the first $0.1s$ window of the stance phase). Currently, the foot torque limits were set to 0 and the contact forces increased linearly from 0N to 450N (1.5 times the robot's body weight). These control features are present in both simulation and hardware experiments.

3) Physical Setup:

The hardware experiments were carried on Tallahassee Cassie in a flat laboratory environment with a catch rope only for safety purposes. The robot can start at its nominal standing height or at its crouch pose. It can also start directly from its stepping mode. Currently, the obstacle positions are coded in the controller and can be changed by the radio controller during experiments.

B. Result 1: Standing and Stepping

We first showed MPC results without avoidance. During stepping, a foot position offset is added to the MPC reference to account for drifting caused by noises and model inaccuracies. We found that using the LIPM predicted acceleration as a feed-forward acceleration is helpful for stability during stance and applied a moving-average filter to obtain a smooth acceleration command.

C. Result 2: Stepping Avoidance

We then demonstrated stepping with avoidance. We used the radio controller to generate moving obstacles from different directions (i.e. left, right, front, and back). Cassie started stepping in place and then tried to avoid obstacles. The obstacle was set to be $0.5m$ away from the robot and the obstacle radius and soft penalty radius are set to be $0.3m$ and $0.6m$, respectively.

D. Result 3: Automatic Mode Switching

Finally, we showed the automatic mode switching on hardware Fig. 8 which also includes all the behaviors shown in Sec. IV-B and Sec. IV-C. The obstacle was set to be $0.5m$ away from the robot and approaching Cassie from its left behind. The obstacle radius is $0.3m$ and we changed the soft penalty radius to get two behaviors. When the soft penalty radius is small $0.6m$, Cassie avoids the obstacle during stance without triggering stepping. Then we increased the soft penalty radius to $0.8m$ and Cassie switched to stepping for avoidance. Finally, Cassie switched back to standing after the obstacle moved away.

V. Discussion

We demonstrated the effectiveness of the motion-planning framework for bipedal avoidance. Given there is no visual sensing on Cassie, we did not use physical obstacles in the hardware experiments. We will consider including physical obstacles in the future and use either Vicon motion capture or on-board vision to localize them. We are also interested in extending this framework to non-spherical obstacles.

While LIPM renders fast solving and online computation, we can not guarantee the feasibility of the trajectory due to its reduced-order assumption. This issue is mitigated by increasing the MPC update rate, adding reachability constraints, and smoothing the reference signal with a moving average filter. In the future, we aim to resolve this issue by incorporating centroidal dynamics and robot kinematic constraints into the MPC formulation.

Currently, the proposed MPC is able to stabilize Cassie with avoidance in two locomotion modes (i.e. standing and stepping) and switch between them. In the future, we want to extend this framework to multiple modes (e.g. standing avoidance, one-step avoidance, and multiple-step avoidance). Additionally, the framework can also be implemented for push-recovery analysis.

While the controller weights and gains require manual tuning, we noticed two parameters can be chosen easily based on physical intuition – which is promising for future extensions. The safe-region radius r can be set based on the agent's and obstacle's geometry and the MPC control offset u_{off} can be determined by the robot stance feet position and geometry. In the future, we want to utilize learning methods to learn such parameters instead of manually tuning them to achieve safe locomotion in unknown environments.

VI. Conclusions

We presented a motion planning method that allows for real-time dynamic bipedal avoidance with mode switching. By using a novel slack-variable formulation, the robot can automatically switch between standing mode and stepping modes to avoid moving obstacles. We verified our algorithm in simulation and on a 3D bipedal robot, Cassie, successfully achieving standing, stepping, and mode switching to enable real-time dynamic bipedal avoidance.

VII. ACKNOWLEDGMENT

This work was supported by the Toyota Research Institute. We also want to thank Ford Motor Company for their support. The authors want to thank Jeffrey Scott Welling for his assistance with the Cassie hardware experiment setup.

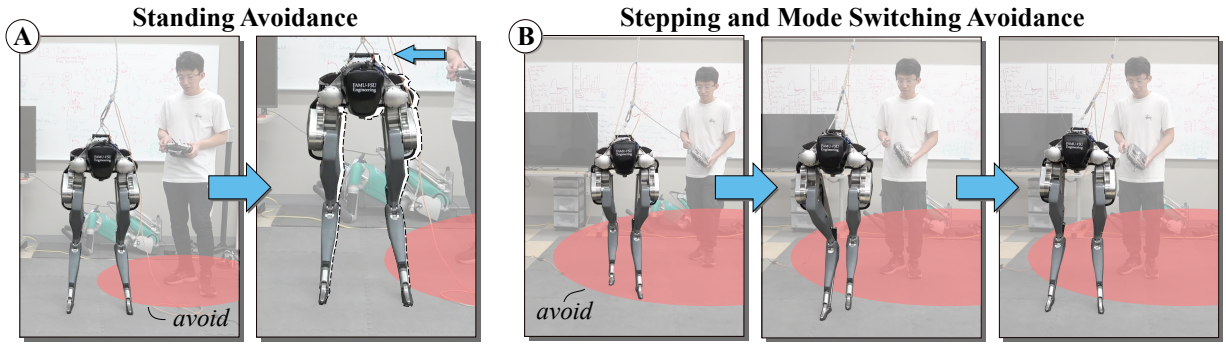

Fig. 8: Cassie hardware experiments: By changing the radius of the danger zone, Cassie achieved two different avoidance motions: (A) With a smaller collision radius, Cassie tried to avoid the obstacle during standing, and (B) With a larger collision radius, stepping is triggered to gain maneuverability for avoidance.

REFERENCES

[1] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using panoc," in *2018 European Control Conference (ECC)*, 2018, pp. 1523–1528.

[2] J. White, D. Jay, T. Wang, and C. Hubicki, "Avoiding dynamic obstacles with real-time motion planning using quadratic programming for varied locomotion modes," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 626–13 633.

[3] T. Marcucci, M. E. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *ArXiv*, vol. abs/2205.04422, 2022.

[4] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 363–374, 2020.

[5] J.-K. Huang and J. W. Grizzle, "Efficient anytime clf reactive planning system for a bipedal robot on undulating terrain," *IEEE Transactions on Robotics*, pp. 1–18, 2023.

[6] R. Grandia, A. Taylor, A. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," 05 2021, pp. 8352–8358.

[7] A. Thirugnanam, J. Zeng, and K. Sreenath, "Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 286–292.

[8] Z. Li, J. Zeng, S. Chen, and K. Sreenath, "Vision-aided autonomous navigation of bipedal robots in height-constrained environments," *ArXiv*, vol. abs/2109.05714, 2021.

[9] A. Shamsah, J. Warnke, Z. Gu, and Y. Zhao, "Integrated task and motion planning for safe legged navigation in partially observable environments," *ArXiv*, vol. abs/2110.12097, 2021.

[10] S. Gilroy, D. Lau, L. Yang, E. Izaguirre, K. Biermayer, A. Xiao, M. Sun, A. Agrawal, J. Zeng, Z. Li, and K. Sreenath, "Autonomous navigation for quadrupedal robots with optimized jumping through constrained obstacles," 07 2021.

[11] N. Scianca, P. Ferrari, D. D. Simone, L. Lanari, and G. Oriolo, "A behavior-based framework for safe deployment of humanoid robots," *Autonomous Robots*, vol. 45, pp. 435 – 456, 2021.

[12] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 295–302.

[13] N. Bohorquez, A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Safe navigation strategies for a biped robot walking in a crowd," *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 379–386, 2016.

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

[15] 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*, vol. 1, 2001, pp. 239–246.

[16] R. Blickhan, "The spring-mass model for running and hopping," *Journal of Biomechanics*, 1989.

[17] X. Xiong and A. Ames, "Dynamic and versatile humanoid walking via embedding 3d actuated slip model with hybrid lip based stepping," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6286–6293, 2020.

[18] 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)*. IEEE, 2022, pp. 6724–6731.

[19] J. White, D. Swart, and C. Hubicki, "Force-based control of bipedal balancing on dynamic terrain with the "tallahassee cassie" robotic platform," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6618–6624.

[20] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 200–207.

[21] M. A. Posa, T. Koolen, and R. L. Tedrake, "Balancing and step recovery capturability via sums-of-squares optimization," 2017.

[22] P. Zaytsev, S. J. Hasaneini, and A. Ruina, "Two steps is enough: No need to plan far ahead for walking balance," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6295–6300.

[23] T. Apgar, P. Clary, K. R. Green, A. Fern, and J. W. Hurst, "Fast online trajectory optimization for the bipedal robot cassie," *Robotics: Science and Systems XIV*, 2018.

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

[25] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[26] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.