2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

A Collision-Free MPC for Whole-Body Dynamic Locomotion and Manipulation

Jia-Ruei Chiu, Jean-Pierre Sleiman, Mayank Mittal, Farbod Farshidian, Marco Hutter

***Abstract*— In this paper, we present a real-time whole-body planner for collision-free legged mobile manipulation. We enforce both self-collision and environment-collision avoidance as soft constraints within a Model Predictive Control (MPC) scheme that solves a multi-contact optimal control problem. By penalizing the signed distances among a set of representative primitive collision bodies, the robot is able to safely execute a variety of dynamic maneuvers while preventing any self-collisions. Moreover, collision-free navigation and manipulation in both static and dynamic environments are made viable through efficient queries of distances and their gradients via a euclidean signed distance field. We demonstrate through a comparative study that our approach only slightly increases the computational complexity of the MPC planning. Finally, we validate the effectiveness of our framework through a set of hardware experiments involving dynamic mobile manipulation tasks with potential collisions, such as locomotion balancing with the swinging arm, weight throwing, and autonomous door opening.**

I. INTRODUCTION

The resemblance of legged robots to their biological counterparts has made them an ideal option for various settings such as industrial inspection, environment exploration, and search and rescue. Their unrivaled agility allows traversal over a range of rough terrains – from a regular staircase to an unstructured subterrain. Compared to a wheel-based mobile manipulator, a legged manipulator can further exploit its six degrees-of-freedom (DoFs) floating base to increase the reachability of the arm [1]. However, due to the added complexity, motion planning and control remain highly challenging for these systems.

A common approach to the whole-body planning problem for mobile manipulation treats the base and manipulator as decoupled subsystems [2]–[4]. While such a decomposition yields simpler and computationally tractable problems, heuristics combining the separate plans may not suffice for large or dynamic object interaction where tight coordination of the base and arm are necessary [5]. In contrast, unifying locomotion and manipulation in a single framework can generate coordinated base and arm motions. Optimal control is a tool able to manage multiple objectives and constraints jointly for the base and the manipulator [5]–[7].

Sleiman *et al*. [6] propose a framework that unifies whole-body dynamic locomotion and manipulation planning into a single Model Predictive Control (MPC) problem. This formulation is applied to a legged mobile manipulator performing various free-motion and object-manipulation tasks. However, the planner does not account for any potential collisions, which restricts deployment on the robot and requires hand-crafted heuristics to prevent any damages. In this work, we build upon their formulation and extend it for considering robot self-collisions as well as environment-collisions in static and dynamic scenes.

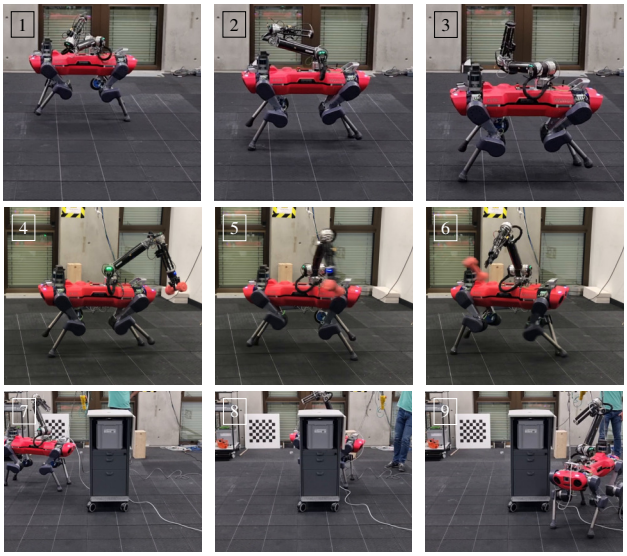

Fig. 1: (1)-(3) Balancing with arm while trotting sideways and switching trotting direction without self-collision between the elbow and LiDAR cage in (2). (4)-(6) Collision-free dynamic backward weight throwing while trotting. (7)-(9) Navigation towards an end-effector position target behind a static obstacle.

Our main contributions are listed as follows:

- We extend the whole-body MPC planner to avoid self- and environment-collisions during coordinated locomotion and manipulation without any added heuristics.
- We present detailed comparisons and benchmarks for different collision-avoidance techniques, thereby providing a useful reference for the implementation of efficient collision-free motion planning.
- We enable a legged mobile manipulator to safely execute autonomous tasks on hardware. Most importantly, we show that the slight increase in computational cost does not compromise the MPC frequency, thus ensuring collision-free motions in real-time.

To the best of our knowledge, this is the first work exhibiting dynamic and collision-free whole-body legged locomotion and manipulation on a real platform without offline planning.

This research was supported in part by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics (NCCR Robotics), and in part by TenneT.

All authors are with the Robotic Systems Lab, ETH Zurich, Zurich 8092, Switzerland. M. Mittal is also with NVIDIA. (Email: jichiu, @student.ethz.ch)

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9812280

II. Related Work

A. Self-Collision Avoidance

In recent years, learning-based methods have been explored in the context of self-collision avoidance. Support vector machines (SVM) have been widely applied to learn a continuously differentiable self-collision boundary function in joint-space [8]–[10]. This boundary is then used to formulate the collision constraints within a quadratic programming (QP) formulation. Nöel *et al*. [11] apply a multi-layer perceptron to approximate the joint-space distance field to generate repulsive torques. However, these methods usually require expensive offline computation. In contrast, online approaches such as [12]–[14] use the signed distances between pairs of collision primitives to provide efficient collision avoidance, but rely on reactive controllers. In this work, we adopt a similar strategy for self-collision avoidance; however, we rely on a receding-horizon planner, which provides us with both reactive as well as look-ahead capabilities.

B. Environment-Collision Avoidance

When considering motion planning problems involving collision-avoidance requirements, sampling-based algorithms have been widely studied and used for collision-free path planning [15], [16]. However, these planners often require post-processing steps to smoothen the computed trajectories and do not scale well to high-dimensional spaces.

Another prominent approach is encoding the problem as a constrained-optimization program. For instance, trajectory optimization has shown to yield smooth, locally optimal, and collision-free trajectories in cluttered environments. A more known approach, Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [17], [18], maximizes a cost function composed of the integrals of trajectory smoothness and obstacle proximity along the path. It approximates the robot with a set of spheres to efficiently query the robot-to-obstacle distances in a pre-computed Signed Distance Field (SDF). Over the past few years, CHOMP has been the foundation of many motion planners. Oleynikova *et al*. [19], [20] explore the collision-free trajectory in a similar fashion but with a Euclidean SDF (ESDF) and a smooth potential function for continuous gradients. For legged locomotion, Fankhauser *et al*. [21] build an SDF from a 2.5D elevation map and fit each robot foot with a sphere for collision-free swing leg trajectory planning. Similarly, for posture control in confined spaces, Buchanan *et al*. [22], [23] use CHOMP while employing two 2.5D elevation maps for the ceiling and the floor and place multiple spheres along the edges of the robot torso.

Compared to 2.5D elevation maps, 3D volumetric maps capture the environment better for mobile manipulation. *Voxblox* [24] and *FIESTA* [25] are novel tools for volumetric ESDF mapping. Together with the use of spherical approximations of collision bodies, Pankert *et al*. [7], and Gärtner *et al*. [26] have applied *Voxblox* to avoid static obstacles for a wheel-based mobile manipulator and a legged robot respectively. By leveraging a faster update rate from *FIESTA*, Mittal *et al*. [5] have shown dynamic obstacle-avoidance during object interaction with a wheel-based mobile manipulator. All of these methods enforce collision avoidance as soft constraints within an optimal control scheme.

C. Sphere Decomposition of Robot

In all the previous works, the collision spheres approximation of the robot is performed manually. In this work, we seek to automate this procedure by leveraging ideas from computer graphics. Hubbard *et al*. [27] construct a sphere-tree with different levels of accuracy based on the approximated medial axis surface. The hierarchy of spheres is useful for broad-phase collision checking. Later works in [28], [29] iteratively update the medial axis for a tighter sphere-tree and complete coverage. Given a user-specified number of spheres, the approach in [30] approximates the mesh representation by minimizing the summed volume outside the object's surface. A more recent work by Voelz *et al*. [31] approximates primitive collision bodies for efficient distance and gradient queries in the ESDF. Our work implements this algorithm for decomposing the robot's collision bodies for collision avoidance.

III. PROBLEM FORMULATION

A. Whole-Body MPC Planner

As mentioned in Sec. I, our work extends the whole-body MPC planner developed in [6]. Therefore, we start by providing a brief description of the underlying framework, while also highlighting the main developments.

1) Solver: The planner is based on the optimal control solver introduced in [32], [33] which employs the Sequential Linear Quadratic (SLQ) technique, a continuous-time variant of Differential Dynamic Programming (DDP). Extensions to the algorithm were made in [34] and [35] to further augment the formulation with inequality constraints through a relaxed-barrier method and generic constraints through an augmented-Lagrangian approach, respectively.

2) System Dynamics: We apply our method to a quadrupedal mobile manipulator performing dynamic tasks. As previously shown in [6], a suitable model for such a poly-articulated system would be a full centroidal dynamic description where the limbs of the robot are not assumed to be massless. Moreover, to properly capture the robot-object dynamic coupling, the object dynamics are augmented to the overall system flow map. Therefore, the Equations of Motion (EoM) are given by:

$$\begin{cases} \dot{\boldsymbol{p}}_{com} = \sum_{i=1}^{n_c} \boldsymbol{f}_{c_i} + m\boldsymbol{g} \\ \dot{\boldsymbol{l}}_{com} = \sum_{i=1}^{n_c} \boldsymbol{r}_{com,c_i} \times \boldsymbol{f}_{c_i} \\ \dot{\boldsymbol{q}}_b = \boldsymbol{A}_b^{-1} \left(\boldsymbol{h}_{com} - \boldsymbol{A}_j \dot{\boldsymbol{q}}_j \right) \\ \dot{\boldsymbol{q}}_j = \boldsymbol{v}_j \\ \dot{\boldsymbol{q}}_o = \boldsymbol{v}_o \\ \dot{\boldsymbol{v}}_o = \boldsymbol{M}_o^{-1} \left(\boldsymbol{J}_{c_o}^T \boldsymbol{f}_{c_o} - \boldsymbol{b}_o \right) \end{cases} . \tag{1}$$

The robot state $\boldsymbol{x}_r = (\boldsymbol{h}_{com}, \boldsymbol{q}_b, \boldsymbol{q}_j) \in \mathbb{R}^{12+n_a}$ collects the centroidal momentum, base pose, and joint positions. The

centroidal momentum $\boldsymbol{h}_{com} = (\boldsymbol{p}_{com}, \boldsymbol{l}_{com}) \in \mathbb{R}^6$ is composed of the linear and angular momentum. The input vector $\boldsymbol{u} = (\boldsymbol{f}_{c_1}, \ \ldots, \boldsymbol{f}_{c_{n_c}}, \boldsymbol{v}_j) \in \mathbb{R}^{3n_c+n_a}$ consists of contact forces at n_c contact points and joint velocities. The object state $\boldsymbol{x}_o = (\boldsymbol{q}_o, \boldsymbol{v}_o) \in \mathbb{R}^{2n_o}$ captures the object generalized positions and velocities. Furthermore, $\boldsymbol{r}_{com,c_i}$ is the position of the i-th contact point w.r.t. the center of mass, while $\boldsymbol{A}(\boldsymbol{q}) = [\boldsymbol{A}_b(\boldsymbol{q}) \ \ \boldsymbol{A}_j(\boldsymbol{q})] \in \mathbb{R}^{6\times(6+n_a)}$ is the centroidal momentum matrix which maps generalized velocities to centroidal momenta. The robot-object interaction happens through the contact force $\boldsymbol{f}_{c_o}$, which is mapped to generalized torques through the contact Jacobian $\boldsymbol{J}_{c_o}$. The term $\boldsymbol{M}_o$ denotes the generalized mass matrix, whereas $\boldsymbol{b}_o$ encapsulates the remaining generalized forces.

3) Cost Function: We encode all robot- and object-centric tasks in a single cost function. This would also include any collision-avoidance constraints that would be added as soft-constraints in the form of a penalty function as follows

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{u}, t) = &\alpha_1 \, ||\boldsymbol{r}_{IE} - \boldsymbol{r}_{IE}^{ref}||^2_{\boldsymbol{Q}_{ee}} + \alpha_2 \, ||\boldsymbol{x}_r - \boldsymbol{x}_r^{ref}||^2_{\boldsymbol{Q}_r} \\ &+ \alpha_3 \, ||\boldsymbol{x}_o - \boldsymbol{x}_o^{ref}||^2_{\boldsymbol{Q}_o} + ||\boldsymbol{u} - \boldsymbol{u}^{ref}||^2_{\boldsymbol{R}} \\ &+ L_c(\boldsymbol{x}_r, t), \end{aligned} \tag{2}$$

where $\boldsymbol{r}_{IE} \in \mathbb{R}^3$ denotes the end-effector position in the inertial frame. The weighting matrices $\boldsymbol{Q}_{ee}, \boldsymbol{Q}_r$ and $\boldsymbol{Q}_o$ are positive semi-definite, and $\boldsymbol{R}$ is positive definite. The cost term $L_c(\boldsymbol{x}_r, t)$ represents the summation of all penalties corresponding to the self-collision and environment-collision constraints. The parameters $\alpha_1, \alpha_2, \alpha_3 \in \{0, 1\}$ are used to determine the combination of active cost terms according to the task description.

4) Constraints: A common mathematical way to describe collision avoidance is by an inequality distance constraint

$$h_i(\boldsymbol{x}_r, t) = d_i(\boldsymbol{x}_r, t) - \epsilon_i \geq 0, \tag{3}$$

where $d_i(\boldsymbol{x}, t)$ computes the distance between the i-th collision pair and ϵ_i denotes the minimum allowed distance threshold for each collision pair. As explained in Sec. III-B and Sec. III-C, the distance function and threshold follow different definitions for self-collision and environment-collision avoidance. We recall that the SLQ-MPC solver can handle inequality path constraints as soft constraints by absorbing them into the cost function through penalty functions. In this work, we choose to penalize the collision distance constraints through relaxed barrier functions (RBF) [34]:

$$B(h) = \begin{cases} -\mu \ln(h), & h \geq \delta \\ \mu\beta(h; \delta), & h < \delta \end{cases}, \tag{4}$$

where $\beta(\cdot; \delta)$ is a quadratic function that yields a continuous and twice-differentiable barrier function with bounded curvature [34]. As a result, Eq. 2 includes the collision avoidance cost term, $L_c(\boldsymbol{x}_r, t)$ in the following form

$$L_c(\boldsymbol{x}_r, t) = \sum_{i=1}^{n_p} B(h_i(\boldsymbol{x}_r, t)), \tag{5}$$

where n_p is the total number of the self-collision and environment-collision pairs.

Additional time-dependent switched constraints are included in the MPC formulation, such as contact complementarity constraints at the arm's end-effector and the feet, friction cone constraints, and input limits. The reader is referred to [6] for more details.

B. Self-Collision Avoidance

Instead of triangular meshes, we represent the robot with primitive collision bodies and use GJK-based distance queries in *FCL* [36]. For each i-th collision body pair, we acquire the shortest distance d_i between the nearest two points. Given the distance result and the nearest points ${}_{\mathcal{I}}\boldsymbol{p}_{i_A}$ and ${}_{\mathcal{I}}\boldsymbol{p}_{i_B}$ in the inertial frame, each self-collision distance d_i in Eq. 3 can be written as the signed distance between the two nearest points as follows

$$d_i(\boldsymbol{x}_r) = \pm||{}_{\mathcal{I}}\boldsymbol{p}_{i_A}(\boldsymbol{x}_r) - {}_{\mathcal{I}}\boldsymbol{p}_{i_B}(\boldsymbol{x}_r)||_2 \tag{6}$$

with a negative sign for colliding bodies and a positive sign for the non-overlapping case. For self-collision avoidance, we set the distance threshold $\epsilon_i = 0.1 \ m$ for all collision pairs. The gradient of the distance function is also necessary since the SLQ solver requires a Gauss-Newton Hessian approximation of the soft constraints penalties. Following the derivation in [37], [38], we compute the gradient of the distance function w.r.t the robot state through

$$\frac{\partial d_i}{\partial \boldsymbol{x}_r} = \pm \hat{\boldsymbol{n}}_i^T ({}_{\mathcal{I}}\boldsymbol{J}_{i_A} - {}_{\mathcal{I}}\boldsymbol{J}_{i_B}), \tag{7}$$

where ${}_{\mathcal{I}}\boldsymbol{J}_{i,A}$ and ${}_{\mathcal{I}}\boldsymbol{J}_{i,A}$ are the Jacobian of ${}_{\mathcal{I}}\boldsymbol{p}_{i_A}$ and ${}_{\mathcal{I}}\boldsymbol{p}_{i_B}$ and $\hat{\boldsymbol{n}}_i$ is the normal vector

$$\hat{\boldsymbol{n}}_i = ({}_{\mathcal{I}}\boldsymbol{p}_{i_A} - {}_{\mathcal{I}}\boldsymbol{p}_{i_B})/||{}_{\mathcal{I}}\boldsymbol{p}_{i_A} - {}_{\mathcal{I}}\boldsymbol{p}_{i_B}||_2. \tag{8}$$

1) Narrow and Broad Phase Distance Query: Given n collision bodies, the naïve approach to avoid self-collision is by performing the narrow-phase or direct distance queries between each collision pair. This purely narrow-phase method would result in the worst-case $O(n^2)$ complexity.

To reduce the complexity, we can apply the broad-phase distance query [36]. A broad-phase manager builds a hierarchy of bounding boxes for a set of collision bodies. Thus, the distance query between an object and a manager is performed by recursive traversal and continues in-depth only if it reduces the shortest distance currently cached. In this manner, we avoid narrow-phase queries for faraway collision objects in the manager. Furthermore, each broad-phase query would only yield the distance result to the nearest managed collision body. Fewer distance results also mean less time for computing gradients. Overall, using the broad-phase strategy can alleviate the computation of proximity queries. Through thorough comparisons, we later show in Sec. V-C whether it is also advantageous in solving the MPC problem.

2) Robot Collision Modeling: We only consider potential collisions between the arm and the torso, since the arm rarely collides with the legs due to the mechanical joint limits. Therefore, no collision primitives are defined on robot's legs. Ideally, one would use collision bodies of various shapes and sizes to capture as many details as possible. In this way,

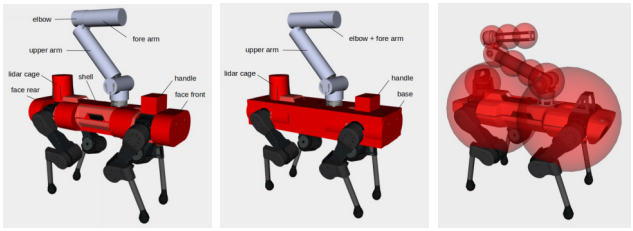

Fig. 2: Robot detailed (left) and simplified (middle) collision model. In the detailed model, the torso comprises the following three links: face front, shell, and face rear, with 11 collision bodies in total. In the simplified model, one box encloses the torso and one cylinder combines the elbow and forearm. (right) Sphere approximations for the simplified model.

more precise collision distances are available for collision avoidance. We describe the base in detail with thirteen collision bodies, consisting of eleven for the torso, one for the front handle, and one for the LiDAR cage as depicted in Fig. 2. The collision model of the arm consists of three collision bodies, each for the upper arm, elbow, and forearm.

In comparison, simplified modeling is beneficial to easing computation. We trim off trivial details by replacing the eleven torso collision bodies with one big box and grouping the elbow and forearm with one cylinder. This simplified modeling requires only three and two collision bodies for the base and the arm, respectively. The middle figure in Fig. 2 visualizes the simplified collision model.

3) Strategies: Given the two query approaches and the two modeling fashions, we propose the following strategies: For detailed modeling, query the distances between

a. 39 collision pairs using the naïve approach.
b. 3 arm collision bodies and 1 broad-phase manager for 13 base collision bodies.

For simplified modeling, query the distances between

c. 6 collision pairs using the naïve approach.
d. 2 arm collision bodies and 1 broad-phase manager for 3 base collision bodies.

In Sec. V-C.1, we examine how each modeling and query fashion affects the MPC performance.

C. Environment-Collision Avoidance

Similar to [5], [7], [18], [26], we enclose the robot with a set of collision spheres and query the signed distances at the sphere centers in the SDF. Each sphere contributes one collision constraint (3) with the distance function d_i and threshold ϵ_i defined as:

$$d_i(\boldsymbol{x}_r, t) = SDF({}_{\mathcal{I}}\boldsymbol{p}_i(\boldsymbol{x}_r, t)) \tag{9a}$$

$$\epsilon_i = r_i, \tag{9b}$$

where ${}_{\mathcal{I}}\boldsymbol{p}_i$ denotes the sphere center, a function of the robot configuration, and r_i is the corresponding sphere radius.

1) Automatic Sphere Approximation: Unlike prior works manually approximating the robot's collision bodies into spheres, we implement an automatic sphere approximation algorithm [31] fitting a box or a cylinder with enclosing spheres of the same radius. The approximation respects a user-defined maximal distance δ_{max} by which a sphere's surface can exceed from the collision body surface. It allows us to reuse the collision primitives defined for self-collision avoidance, leading to straightforward generalizations of this collision-free motion planner over various robots. In this work, we avoid the obstacles by applying the sphere approximation algorithm to the simplified model in Sec. III-B.2. The visualization of the approximation result is shown in Fig. 2.

2) ESDF Mapping: We use the *FIESTA* mapping algorithm to create an SDF map of the scene [25]. Using point-cloud information and robot's odometry, the algorithm first creates an occupancy map and uses it to compute the SDF map. Similar to [5], we use tri-linear interpolation to query distances and compute gradients which are then cached to save computation time during MPC planning. We show in Sec. V-C.2 that this methodology adds little computation cost compared to the blind case.

IV. SYSTEM DESCRIPTION

The proposed MPC framework and the underlying whole-body controller (WBC) are robot agnostic. The rigid body dynamics of the robot is handled by the *Pinocchio* C++ library [39], [40]. The distance queries between collision primitives are supported by the *Flexible Collision Library* (*FCL*) [36] based on the Gilbert-Johnson-Keerthi (GJK) algorithm [41]. The SLQ algorithm for the MPC planner is based on the *OCS2* toolbox [42], which implements efficient and numerically stable optimal control for switched systems.

We perform several hardware tests with the ANYmal C platform equipped with DynaArm, a torque-controllable 4-DoF robotic arm. Together with powerful actuators, DynaArm is capable of highly dynamic manipulation with a payload capability of up to 7 kg. The robot's onboard computer (Intel Core i7-8850H CPU@4GHz hexacore processor) takes care of both the MPC-based planner and the WBC. The main control loop along with the state estimator are executed at 400 Hz. The MPC loop plans with a time horizon of 1 s, at an update rate of 70 Hz.

V. RESULTS

We perform various experiments to show the ability of the MPC formulation for legged mobile manipulation. In Sec. V-A, we show balancing with the arm and throwing a weight while respecting self-collision avoidance. In Sec. V-B, we showcase environment-collision avoidance via SDF for tasks such as dynamic obstacle avoidance and door passing.

A. Self-Collision Avoidance

In all the experiments, the RBF parameters are $\mu = 10^{-2}$ and $\delta = 10^{-3}$ for all collision pairs. We carry out several dynamic motions, which were only simulated in the previous work [6] due to the lack of self-collision avoidance.

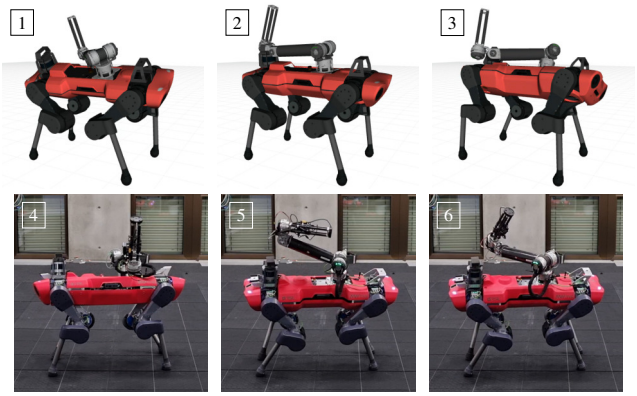

Fig. 3: (1)-(3) Blind simulated and (4)-(6) collision-free real-world balancing with the arm when the base roll angle changes between $\pm 20^o$. Self-collision may occur to the blind robot between its arm and LiDAR in (2), whereas the arm swiftly avoids collision in (5).

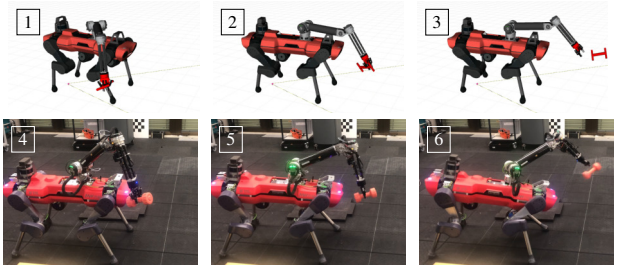

Fig. 4: (1)-(3) Simulation of blind forward weight throwing while in stance with potential collision between the arm and handle in (2). (4)-(6) Hardware collision-free demonstration.

Balancing with Arm: For free-motion modes, we use the manipulator as a tail to balance dynamic base motions with $\alpha_2 = 1$ and $\alpha_1 = \alpha_3 = 0$. By reducing the penalties on the arm joint positions and velocities, we grant the MPC more freedom to exploit the wide range of motion and the high speed of the Dynaarm for balancing. For the first example, we command the robot to instantly change the base roll angle from $+20^o$ to -20^o while standing. As shown in Fig. 3, quick changes in the base orientation may lead to a collision between the arm and the LiDAR in the simulated blind robot. On the contrary, with self-collision avoidance, the manipulator does not only maintain its high speed but rapidly avoids crashing into the LiDAR on the hardware. In the second scenario, the robot trots sideways at a relatively high speed and instantly switches directions. The arm also quickly steers away from the LiDAR while swinging in the opposite direction Fig. 1. Both scenarios demonstrate the efficacy of self-collision avoidance in static and dynamic maneuvers.

Weight Throwing: To fully exploit the capabilities of the MPC planner, we also showcase self-collision avoidance during object manipulation with $\alpha_3 = 1$ and $\alpha_1 = \alpha_2 = 0$. The robot is commanded to throw a 1.5 kg dumbbell to the target position. The MPC planner optimizes the throwing motions according to the given switching time. In the first mission, the target position is 2 m to the front and 2 m to the left of the base center. As shown in Fig. 4, the upper

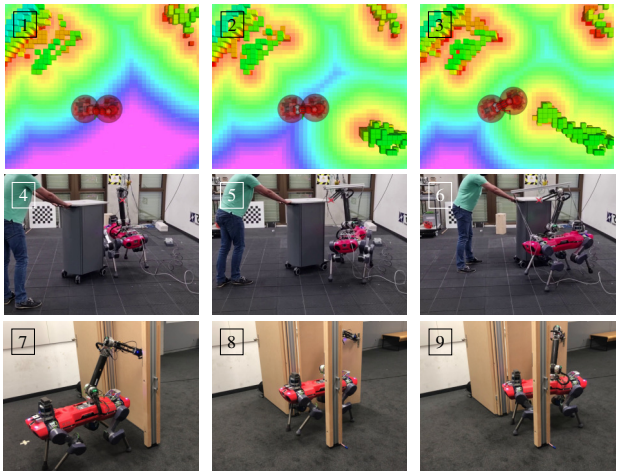

Fig. 5: (1)-(3) Visualization of the sphere approximation, ESDF slice, and occupancy grip map while avoiding an incoming obstacle. (4)-(6) Dynamic obstacle avoidance with an approaching cart while keeping track of an end-effector target position. (7)-(9) Autonomous door opening with collision-avoiding base yaw adaptation.

arm of the blind robot may collide with the front handle of the base in the simulation. However, with self-collision avoidance, the manipulator swiftly negotiates with the handle and prevents potential damage to the hardware. Finally, in the second scenario, the robot is tasked with throwing 2 m to the back and 1 m to the right while trotting, as shown in Fig. 1. In this case, the throwing maneuver can be further enhanced by having the robot trot during the task execution, thus generating more dynamic movements through a wider range of base motions.

B. Environment-Collision Avoidance

For all environment-collision avoidance tasks, we set $\mu = 0.5$ and $\delta = 0.02$. The map using *FIESTA* is built with the resolution of 10 cm using the LiDAR sensor on the robot. As shown in Fig. 2, the robot is enclosed by spheres with $\delta_{max} = 40$ cm, 10 cm, 5 cm, 5 cm, and 10 cm for the base, shoulder, upper arm, elbow plus forearm, and the LiDAR cage respectively.

Dynamic Obstacles: To showcase the efficiency of the algorithm, we present whole-body planning for dynamic obstacle avoidance. An end-effector target position is imposed while a cart is pushed around the robot. Presented in Fig. 5, the legged manipulator is capable of keeping track of the target while moving away from the approaching cart. Additionally, the base roll is exploited for better arm extension.

Door Opening: Ultimately, we test our framework in an autonomous door-opening scenario. To avoid collision with the door frame during interaction in [6], we imposed a high penalty on the lateral motion of the base while passing through the door. Enhanced with environment-collision avoidance through SDF, we can now automate the procedure without additional heuristics. We command the robot to push the door open to an angle of 100^o. As we can see in Fig. 5, the robot starts in front of the handle and is close to one side

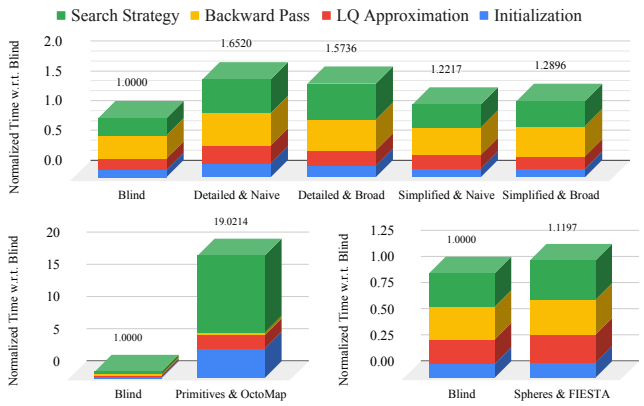

Fig. 6: Average MPC computation per iteration for: (top) self-collision avoidance, (bottom left) environment-collision avoidance using collision primitives with *OctoMap*, and (bottom right) using sphere approximations with *FIESTA*.

of the door frame. By adapting its torso yaw, the robot can navigate through the passage safely without any collisions. It can also avoid the approaching person during the execution.[1]

C. Computation Benchmark

For both self-collision and obstacle avoidance, we collect five runs of reaching specific end-effector targets to benchmark the whole-body collision avoidance behaviors. Each run lasts around 1000 MPC iterations, and the average MPC computation per iteration over all five runs is analyzed. All the results are normalized w.r.t. the blind case, which means no self-collision and no environment-collision avoidance.

1) Self-Collision Avoidance: We compare the four strategies against the blind case as shown in Fig. 6. As expected, using the detailed model with the naïve approach is the most expensive. However, compared to the naïve strategy, applying the broad-phase technique saves little time with the detailed model but is more expensive than the simplified model. For both models, broad-phase managing indeed speeds up the Linear Quadratic (LQ) Approximation step, which means less time for distance queries and gradient computation. Nevertheless, it takes longer to perform the line search step of the algorithm. The problem is that, at certain configurations, a manager may have multiple objects simultaneously closest to the queried object as illustrated in Fig. 7. This would result in a discontinuity of gradient directions, which is unfavorable for gradient-based optimization. Besides, as the broad-phase manager with the simplified model only saves about two narrow-phase queries, it reduces about 2% in the LQ Approximation compared to the naïve approach. As a consequence, we opt for the naïve strategy with the simplified robot collision model.

2) Environment-Collision Avoidance: To benchmark our method which relies on the primitive collision bodies, we also compare it to the proximity query introduced in [43] for distances between shape primitives and an *OctoMap* [44]. We build both the *FIESTA* map and *OctoMap* from the "Cow

[1]https://www.youtube.com/watch?v=m3rJWJVzYuY

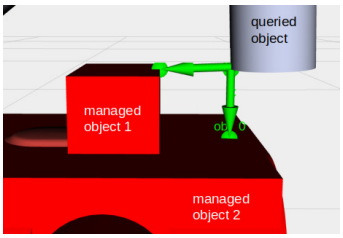

Fig. 7: Direction discontinuity in gradients. The broad-phase technique may experience non-smooth jumps of distance gradient directions at certain configurations.

and Lady dataset" in [24] with 5 cm resolution. Fig. 6 shows that using shape primitives with *OctoMap* is 19 times as expensive as the blind case. This method essentially regards an *OctoMap* as a broad-phase manager with each map cell as a collision box. Therefore, it would suffer from the gradient issue as well. The large amount of cells in the map is also the culprit of the extremely expensive computation. On the contrary, collision avoidance by reading the cached distances and gradients from the *FIESTA* map given the sphere centers leads to only about 11.97% increase in the computation time.

VI. DISCUSSION & CONCLUSION

In this paper, we endow the unified whole-body MPC framework [6] with self-collision and environment-collision avoidance capabilities, which enables dynamic maneuvers involving potential collisions to be safely executed on hardware. The robot collision model is represented with collision primitives. The naïve self-collision distance query with the simplified model has proven its efficiency in real-time planning. By further enclosing the collision primitives with a set of collision spheres, fast updates of the *FIESTA* map and computationally negligible queries of the pre-computed distances and gradients allow both static and dynamic obstacle avoidance. We conduct weight throwing and locomotion balancing with the swinging arm to demonstrate rapid whole-body self-collision avoidance during both free-motions and object manipulation scenarios. The autonomous door opening task in which the robot avoids the static door frame and the approaching human further validates the obstacle avoidance capabilities of the MPC framework.

It is worth noting that although soft constraints cannot guarantee strict constraint satisfaction, allowing small violations facilitates the solver's convergence and prevents it from failing. Thus, the minimum allowed distance threshold for both collision constraints is used to provide room for constraint violations before the actual collision occurs. Moreover, environment-collision avoidance actually suffers from limitations of the LiDAR. For instance, the robot cannot see points in close proximity to the LiDAR. The arm in the LiDAR's field of view also creates a blind spot in the front. Furthermore, since the map update rate is dictated by the LiDAR update rate at 15 Hz, the speed of the dynamic obstacles was limited during our experiments so that they don't suddenly appear within the distance threshold, thus blowing up the cost. Nonetheless, all these shortcomings can be alleviated with additional sensors.

REFERENCES

[1] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter, "Alma-articulated locomotion and manipulation for a torque-controllable robot," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8477–8483.

[2] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, *et al.*, "Autonomous door opening and plugging in with a personal robot," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 729–736.

[3] M. P. Murphy, B. Stephens, Y. Abe, and A. A. Rizzi, "High degree-of-freedom dynamic manipulation," in *Unmanned Systems Technology XIV*, vol. 8387. International Society for Optics and Photonics, 2012, p. 83870V.

[4] B. U. Rehman, M. Focchi, J. Lee, H. Dallali, D. G. Caldwell, and C. Semini, "Towards a multi-legged mobile manipulator," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3618–3624.

[5] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated object interaction in unknown scenes with whole-body mobile manipulation," *arXiv preprint arXiv:2103.10534*, 2021.

[6] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[7] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6177–6184, 2020.

[8] N. B. Figueroa Fernandez, S. S. Mirrazavi Salehian, A. Billard, N. B. Figueroa Fernandez, S. S. Mirrazavi Salehian, A. Billard, N. B. Figueroa Fernandez, S. S. Mirrazavi Salehian, and A. Billard, "Multi-arm self-collision avoidance: A sparse solution for a big data problem." *Proceedings of the Third Machine Learning in Planning and Control of Robot Motion (MLPC) Workshop.*, p. 6, 2018.

[9] S. S. M. Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1205–1232, 2018.

[10] M. Koptev, N. Figueroa, and A. Billard, "Real-time self-collision avoidance in joint space for humanoid robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1240–1247, 2021.

[11] T. Noël, T. Flayols, J. Mirabel, J. Carpentier, and N. Mansard, "A hybrid collision model for safety collision control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1722–1728.

[12] M. Schwienbacher, T. Buschmann, S. Lohmeier, V. Favot, and H. Ulbrich, "Self-collision avoidance and angular momentum compensation for a biped humanoid robot," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 581–586.

[13] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar, "A strictly convex hull for computing proximity distances with continuous gradients," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 666–678, 2014.

[14] J. J. Quiroz-Omaña and B. V. Adorno, "Whole-body control with (self) collision avoidance using vector field inequalities," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4048–4053, 2019.

[15] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[16] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, pp. 995–1001 vol.2.

[17] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.

[18] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[19] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 5332–5339.

[20] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1474–1481, 2018.

[21] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5761–5768.

[22] R. Buchanan, T. Bandyopadhyay, M. Bjelonic, L. Wellhausen, M. Hutter, and N. Kottege, "Walking posture adaptation for legged robot navigation in confined spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2148–2155, 2019.

[23] R. Buchanan, L. Wellhausen, M. Bjelonic, T. Bandyopadhyay, N. Kottege, and M. Hutter, "Perceptive whole-body planning for multilegged robots in confined spaces," *Journal of Field Robotics*, vol. 38, no. 1, pp. 68–84, 2021.

[24] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.

[25] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," *arXiv preprint arXiv:1903.02144*, 2019.

[26] M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter, "Collision-free mpc for legged robots in static and dynamic scenes," *arXiv preprint arXiv:2103.13987*, 2021.

[27] P. M. Hubbard, "Approximating polyhedra with spheres for time-critical collision detection," *ACM Transactions on Graphics (TOG)*, vol. 15, no. 3, pp. 179–210, 1996.

[28] G. Bradshaw and C. O'Sullivan, "Sphere-tree construction using dynamic medial axis approximation," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002, pp. 33–40.

[29] ——, "Adaptive medial-axis approximation for sphere-tree construction," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 1, pp. 1–26, 2004.

[30] R. Wang, K. Zhou, J. Snyder, X. Liu, H. Bao, Q. Peng, and B. Guo, "Variational sphere set approximation for solid objects," *The Visual Computer*, vol. 22, no. 9, pp. 612–621, 2006.

[31] A. Voelz and K. Graichen, "Computation of collision distance and gradient using an automatic sphere approximation of the robot model with bounded error," in *ISR 2018; 50th International Symposium on Robotics*. VDE, 2018, pp. 1–8.

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

[33] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *17th IEEE-RAS International Conference on Humanoid Robotics, Humanoids, UK, 2017*, pp. 577–584.

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

[35] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Constraint handling in continuous-time ddp-based model predictive control," 2021.

[36] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3859–3866.

[37] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Citeseer, 2013, pp. 1–10.

[38] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[39] J. Carpentier, F. Valenza, N. Mansard, *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," https://stack-of-tasks.github.io/pinocchio, 2015–2019.

[40] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.
[41] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.
[42] "OCS2: An open source library for optimal control of switched systems," [Online]. Available: https://github.com/leggedrobotics/ocs2.
[43] J. Pan, I. A. Şucan, S. Chitta, and D. Manocha, "Real-time collision detection and distance computation on point cloud sensor data," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3593–3599.
[44] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.