A Real-Time Approach for Humanoid Robot Walking including Dynamic Obstacles Avoidance

Luca Rossini, Enrico Mingo Hoffman, Seung Hyeon Bang, Luis Sentis, Nikos G Tsagarakis

A Real-Time Approach for Humanoid Robot Walking including Dynamic Obstacles Avoidance

Luca Rossini[1], Enrico Mingo Hoffman[1,2], Seung Hyeon Bang[3], Luis Sentis[3,4], and Nikos G. Tsagarakis[1]

***Abstract*— This paper proposes a novel approach to online replan the walking trajectory of a biped humanoid robot to avoid unexpected interactions and impacts with dynamic obstacles that may compromise the balance of the humanoid robot. The proposed method adjusts the position of the contacts of a preplanned global trajectory according to the position of moving obstacles and the robot's dynamic properties. The methodology includes a graph-based footstep planner to generate a footstep sequence aware of possible changes in a dynamic environment, a Model Predictive Controller based on the Single-Rigid Body Dynamics to track the computed footsteps, and a final Whole-Body Control layer to compute proper joint torque commands. Preliminary results using the proposed approach are presented to demonstrate the effectiveness of the proposed framework in simulated scenarios with the DRACO3 humanoid bipedal platform.**

I. Introduction

A primary goal of humanoid robotics research is to deploy complex machines in real working and domestic environments. This challenging goal is characterized by continuously varying conditions, such as moving obstacles, that can lead to unexpected interactions and impacts that may compromise the walking stability of humanoids and result from falling incidents. To address these challenges, humanoid robots must be able to adapt to dynamic environments and avoid risks caused by unexpected changes in the environment, particularly while walking.

From a broader perspective, the contact sequence and the stable whole-body trajectory can be adjusted by implementing an architecture composed of three layers [1]: the *footstep planner*, the *simplified model controller*, and the *whole-body controller*. The *footstep planner* layer provides a discrete sequence of footsteps and timings [2], [3] that will be used as a reference by the downstream layers. The *simplified model controller* generates trajectories for the Center of Mass (CoM) and contacts, i.e., feet, including forces, using the footsteps as a reference for a *Model Predictive Control* (MPC) problem. This system can react to external disturbances and guarantee stable and reliable locomotion.

[1] Humanoid and Human Centered Mechatronics (HHCM) Lab, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163 Genova, Italy. E-mail: {luca.rossini, enrico.mingo, nikos.tsagarakis}@iit.it

[2] Inria Nancy Grand Est, 615 rue du Jardin Botanique, 54600 Villers les Nancy, France. E-mail: enrico.mingo-hoffman@inria.fr

[3] Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, TX 78712. E-mail: bangsh0718@utexas.edu, lsentis@austin.utexas.edu

[4] Co-Founder of Apptronik, Inc., 110701 Stonehollow Dr STE 150, Austin, TX, 78758, USA

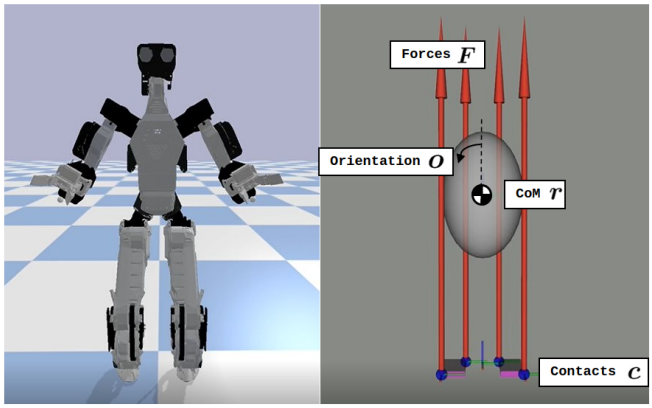


Fig. 1: The humanoid robot Draco3 simulated in PyBullet (left), and the SRBD model (right).

Finally, the *Whole-Body Controller* (WBC) maps the Cartesian motions and contact force references into the robot's joint torques.

In this work, we extend a Single Rigid Body Dynamics (SRBD) non-linear MPC, which has been proven to be robust to external disturbances [4], to consider also the avoidance of moving obstacles that can interfere with the planned trajectory. This is done by adding a contact planner on top of the MPC, which refines the contact positions depending on the position of moving obstacles and the robot's dynamic properties. The contact sequence is then used as a reference by the MPC, warm-starting the optimization problem and further improving the computational efficiency. In this way, it is possible to achieve robust locomotion and be aware of possible collisions with moving obstacles populating the environment. This represents a solution to a problem that is difficult to be resolved online directly inside the MPCs due to the high non-convexity of this kind of constraint.

II. Previous Work

Neglecting the complexity associated with the dynamics of walking motions, planning only the position of the contacts allows the implementation of particular constraints, such as collision avoidance, which are notoriously hard to model [5]. The most straightforward footstep planner provides a sequence of footsteps that ignore collisions or kinematic constraints of the upper body in the free-collision space. This space contains all the footstep locations that do not overlap with a region occupied by an obstacle. Considering only the footsteps' position, a collision avoidance constraint is straightforward and does not influence the planner's performance.

A classic approach is to define a set of transition motions and plan a sequence of footsteps connecting the start and goal state with search algorithms on discrete graphs, like the A* algorithm [6]. However, the size of the transition set must compromise computational cost and planner efficiency since a smaller set reduces the number of solvable maps [7]. An extension of the A* search algorithm has been presented in [8] with the D* Lite that continuously computes the shortest path in the environment. The current starting state progresses with the execution of the planned trajectory, with the robot being able to adapt to changes in the environment but still not accounting for the whole-body motion.

Another possibility is to find a feasible footstep sequence incorporating the planning problem in the WBC. This way, whole-body motions will not be separated from the planning process, with steps automatically triggered by one of the tasks. Indeed, the CoM motion is accomplished by contact forces, and the relation between these quantities should be considered to guarantee the overall motion's feasibility. The main disadvantages of this solution are the computational complexity and the tendency to suffer from local minima, especially when considering highly non-linear and non-convex constraints, such as collision avoidance. For instance, *Trajectory Optimization* (TO) with the whole-body dynamics, obtains beautiful and realistic motions [9], [10], [11], [12] but with high computation time that prevents their online application at high rates. Recently, [13] presented some promising results on an MPC based on the whole-body dynamics of a biped robot, obtained by reducing the time horizon through a Hybrid Zero-Dynamics, which provides a good guess of the robot state at the final time in a reasonable amount of time. However, the approach has been validated on a 4 DoFs biped robot, which simplifies a complete humanoid's complex dynamics.

On the other hand, there exists a variety of robot dynamic simplifications that enhance the performance of the trajectory planning algorithm by relying on specific assumptions. For instance, one can simplify the robot motion with the CoM dynamics, using equations that link the variation of the linear and angular momentum with the joint position and force constraints, in the so-called *Centroidal Dynamics* (CD) [14], [15]. Usually, this approach assumes a constant zero angular momentum variation [4], tracked by a specific upper-body task that can result in strange and possibly dangerous motions.

Despite the impressive advancement of online trajectory planning for biped locomotion, none of the previous works, capable of online implementation, considers dynamic collision avoidance and are limited to changes in the walking surface [16]. Collision avoidance is a highly non-linear and non-convex constraint whose inclusion in the optimization problem drastically under-performs the solver, which may even fail to find a solution due to local minima. The idea presented in this work is to reduce the high computational cost of the problem mentioned above by planning a sequence of footsteps aware of possible collision with moving obstacles and then computing a whole-body trajectory + WBC solving a non-linear MPC using the output of the footstep planner as a reference. Compared to [17], which reduced the computational complexity of moving through narrow environments separating the contact planner from the whole-body planner, we successfully perform robust online locomotion planning from external disturbances and moving obstacles. Furthermore, besides collision avoidance, the contact planner improves the performance of the MPC by suggesting a good set of initial footsteps that makes the solver converge to an optimal solution more quickly. Figure 2 shows a graphical representation of the presented algorithm.

This paper is structured as follows: Sec. III describes the procedure used to adjust the nominal contact sequence depending on the measured position of the moving obstacles (i.e., the contact space local planner). The SRBD, compliant with the optimized contact sequence, is computed following the MPC formulation introduced in Sec. IV and Sec. V. Finally, the proposed approach is validated in a set of simulations described in Sec. VI. A discussion about the results obtained and some possible future works are finally reported in Sec. VII.

III. Contact Space Local Planner

This section introduces the optimization-based planner used to adjust the contact positions while considering the position of the moving obstacles populating the environment. The robotic system moves in a 3-dimensional workspace $\mathscr{W}$ continuously establishing and breaking contacts, driving its body through contact forces $\boldsymbol{W}_{c,i}$ exerted in specific contact locations $\boldsymbol{p}_i \in SE(3)$. Each end-effector allowed to interact with the environment is associated with a frame $\mathscr{F}_i$ oriented so that the z-axis coincides with the normal to the contact surface. The contact space local planner takes in input a contact sequence $C = \langle \boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_N \rangle$ computed offline by a generic method (i.e., a sample or graph-based planner). A contact occurs at a specific time $t \in [t_i, t_{i+1}]$ with t_i and t_{i+1} being the starting and finishing times correspondent to the contact $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$. Two contacts $\boldsymbol{c}_i$, $\boldsymbol{c}_j$ are adjacent if they are referred to the same end-effector $\mathscr{F}_i = \mathscr{F}_j$ and if they occur in correspondence of two next instant of time, meaning that they are the start and goal swing pose of the contact $\mathscr{F}_i$ starting at t_i. The workspace is populated by relatively slow-moving obstacles that can be considered stationary at the current time in a pose $\boldsymbol{p}_{\mathrm{O},j} \in SE(3)$, occupying a volume $\mathscr{V}_{\mathrm{O},j} \in \mathscr{W}$.

The considered contact sequence planner consists of repetitively adjusting the offline planned contact sequence generating a new optimal trajectory C^* when a moving obstacle interferes with the nominal feasible trajectory. The optimized trajectory C^* has the same dimension as C, meaning that the planner cannot change the number of contacts. Furthermore, the gait sequence and timing adaptation are, so far, out of the scope of this planner, which acts by adjusting the contact positions only. At each iteration, the planner solves a contact-space graph-optimization problem in the form of a Non-Linear Least Squared Problem (NLLSP), similarly to our

previous work [18]:

$$f(C) = \sum_{k=0}^{m} f_k(C) \tag{1}$$

$$C^* = \underset{C}{\operatorname{argmin}}\ f(C) \tag{2}$$

with

$$f_k(C) = f_k(C_k) = \boldsymbol{e}_k(C_k)^T \boldsymbol{\Omega}_k \boldsymbol{e}_k(C_k) \tag{3}$$

$f(C)$ is the cumulative objective function of each k-th cost $f_k(C)$. In a graph-optimization each variable $\boldsymbol{c}_i \in C$ defines a vertex, while each edge is associated with an error function $\boldsymbol{e}_k(C_k)$, which depends on a small subset of neighbors vertices $C_k = \{\boldsymbol{c}_{k,0}, \ldots, \boldsymbol{c}_{k,M}\} \in C$ connected by the edge itself. Each edge is weighted depending on the value of the diagonal information matrix $\boldsymbol{\Omega}_k$.

The rest of this section describes the objective functions used to define and solve the graph-based contact space local planner. As stated in our previous work [18], in a graph-based optimization, we can locally activate and deactivate states and constraints depending on the robot's state and environment, encoding the sparsity of the problem more intuitively, thus reducing the computational cost. In this way, we can also store an arbitrarily sizeable global path without affecting the planner's performance, thus avoiding dead-ends of the solution.

A. Relative Distance Constraint

The relative distance constraint is one of the most critical constraints for the feasibility of the adjusted contact sequence C^*. Indeed, reducing the optimization state space to consider only the contact positions and timing results in a loss of information regarding the robot state, e.g., the joint and link positions. Thus, binding the contacts' relative distance is essential to avoid long, kinematically unfeasible contact transitions and close contacts that can lead to a self-collision [19].

The NLLSO problem requires the formulation of constraints as piece-wise continuous objective functions whose violation is penalized by a rapid increase of the cost function. Thus, the relative distance constraint is formulated as follows:

$$\boldsymbol{e}_{k,\mathrm{RD}}(\boldsymbol{c}_i, \boldsymbol{c}_{i+1}) = \frac{1}{\exp(\boldsymbol{d}_{\mathrm{rel},i} - \boldsymbol{d}_{\min})^S} + \exp(\boldsymbol{d}_{\mathrm{rel},i} - \boldsymbol{d}_{\max})^S \tag{4}$$

where $\boldsymbol{e}_{k,RD} \in \mathbb{R}^3$ is the relative distance edge associated to the i-th vertex, $\boldsymbol{d}_{\mathrm{rel},i}$ is the relative distance between $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$ expressed in the local frame $\mathscr{F}_i$:

$$\boldsymbol{d}_{\mathrm{ref},i} = \boldsymbol{R}_i \cdot (\boldsymbol{c}_{i+1} - \boldsymbol{c}_i) \tag{5}$$

With $\boldsymbol{R}_i \in SO(3)$ being the rotation matrix from the inertial frame to the local frame $\mathscr{F}_i$. The relative distance is bounded between $\boldsymbol{d}_{\min}$ and $\boldsymbol{d}_{\max}$, and $S = 10$ is a constant hand-tuned exponential value.

B. Collision Avoidance

The collision avoidance constraint computes the distance between the associated i-th vertex and the j-th obstacle, which is assumed to be stationary at the current planner iteration, and compares it with a threshold value:

$$e^j_{k,\mathrm{coll}}(\boldsymbol{c}_i, \boldsymbol{p}_{\mathrm{O},j}) = \begin{cases} \left(\frac{-d(\boldsymbol{c}_i, \boldsymbol{p}_{\mathrm{O},j}) + d_{\mathrm{th}}}{S}\right)^n & d < d_{\mathrm{th}} \\ 0 & d > d_{\mathrm{th}} \end{cases} \tag{6}$$

with $e^j_{k,\mathrm{coll}}(\boldsymbol{c}_i, \boldsymbol{p}_{\mathrm{O},j})$ being the j-th element of the $j \times 1$ collision edge $\boldsymbol{e}_{k,\mathrm{coll}}$ related with the absolute distance $d(\boldsymbol{c}_i, \boldsymbol{p}_{\mathrm{O},j})$ between the i-th vertex and the j-th obstacle. The edge is activated every time the distance goes below a user-defined threshold $d_{\mathrm{th}} = 0.2$m, avoiding the computation of useless distances between the contacts and far obstacles.

C. Contact Height Constraint

An important aspect when dealing with the optimization of the contact poses is guaranteeing that they occur in correspondence with a contact surface. Indeed, the planner can move the contacts anyway in the 3D workspace if not constrained differently. Contacts must be placed close to surfaces to exert the required contact forces that move the robot in the environment. Thus, a contact height constraint is formulated as a simple parabolic function whose minimum corresponds to the contact point. Assuming that the normal-to-the-surface of each element in the nominal contact sequence does not change, we can project the end-effector to the plane perpendicular to the normal, whose origin is the nominal contact position. This is done by minimizing the distance between the end-effector and its projection on the plane mentioned above:

$$e_{k,c_z}(\boldsymbol{c}_i) = ((\boldsymbol{c}_i - \boldsymbol{p}_{S,i}) \cdot \overrightarrow{n})^2 \tag{7}$$

with $\boldsymbol{p}_{S,i}$ being the origin of the normal, and $\overrightarrow{n}$ the normal to the plane. The assumption of unchanged normal-to-the-surface is valid when the robot moves on large surfaces such as walls or ground characterized by a constant normal on each surface point. However, this assumption is unsuited if the robot establishes contact with smaller objects or convex shapes. Extending this constraint to more generic surfaces is part of future works.

IV. Simplified Model Controller

The contact space local planner provides a contact trajectory aware of the most recent position of static and moving obstacles that may interfere with the robot's motion. Then, the simplified model controller implements a control law that generates feasible contact and CoM trajectories compliant with the planned nominal contact sequence. The control problem is formulated as a non-linear MPC, considering the SRBD as a model simplification to match the computational cost requirements for an online implementation. The SRBD reduces the robot's whole-body motion to the Centroidal Dynamics assuming constant centroidal inertia $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ computed with the robot in a nominal standing configuration.

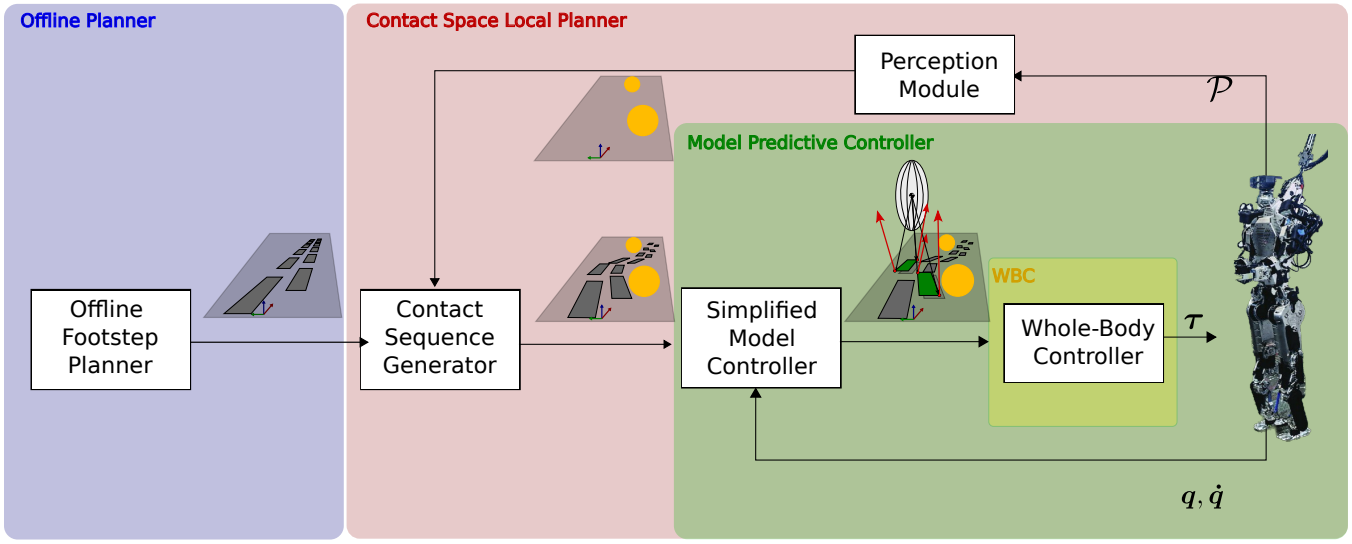


Fig. 2: Block diagram of the proposed approach. A contact sequence computed offline (blue) is computed as a nominal trajectory once in a static environment. The contact space local planner (red) adjusts the initial sequence depending on the obstacles (yellow spheres) perceived by a perception module. The adjusted solution is provided to the MPC (green) that computes a CoM and contact trajectory. A whole-body controller (yellow) maps the solution to the robot's joint torques.

The single rigid body orientation is assumed to coincide with the base frame orientation $\boldsymbol{o}(t) \in SO(3)$ expressed as quaternions, with the base frame being rigidly attached to the torso of the robot. This assumption simplifies the mapping from the single rigid body orientation to the complete robot orientation, assuming that most of this rotation is given by the torso motion. For the sake of brevity, all the time-dependent quantities will be written without the independent variable t and are assumed to be continuous functions unless differently specified. The SRBD equations of motion can be written as follows:

$$\begin{aligned} \dot{\boldsymbol{P}} &= m(\ddot{\boldsymbol{r}} + \boldsymbol{g}) = \sum_{i=0}^{N_c} \sum_{j=0}^{N_v} \boldsymbol{W}^{(i,j)} \\ \dot{\boldsymbol{L}} &= \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \sum_{i=0}^{N_c} \sum_{j=0}^{N_v} [(\boldsymbol{c}^{(i,j)} - \boldsymbol{r}) \times \boldsymbol{W}^{(i,j)}]. \end{aligned} \tag{8}$$

In (8), surface contacts are modeled using N_v contact points $\boldsymbol{c}^{(i,j)} \in \mathbb{R}^3$ around the contact link, hence the force contact vector $\boldsymbol{W}^{(i,j)} \in \mathbb{R}^3$ is a vector of pure forces referred to the i-th contact and j-th vertex expressed in the inertial frame. $\boldsymbol{P} \in \mathbb{R}^3$ and $\boldsymbol{L} \in \mathbb{R}^3$ are the linear and angular momentum respectively. Usually, a rectangular-shaped contact is modeled using four points, one on each vertex, and twenty-four variables define the two surface contacts. However, to further simplify the SRBD, the feet have been modeled with two non-parallel line contacts (Fig. 1). This simplification prevents the generation of the momentum along the axis passing through the two points, with a consequent loss of controllability in a single stance. Placing the contact point onto two non-parallel lines mitigates this effect, which becomes negligible in the double support phases.

The position, velocity, and acceleration terms can be collected by defining the vectors of the generalized coordinates:

$$\boldsymbol{\pi} = \begin{bmatrix} \boldsymbol{r}^T & \boldsymbol{o}^T & \boldsymbol{c}^T \end{bmatrix}^T \tag{9}$$

$$\boldsymbol{\nu} = \begin{bmatrix} \dot{\boldsymbol{r}}^T & \boldsymbol{\omega}^T & \dot{\boldsymbol{c}}^T \end{bmatrix}^T \tag{10}$$

$$\boldsymbol{\alpha} = \begin{bmatrix} \ddot{\boldsymbol{r}}^T & \dot{\boldsymbol{\omega}}^T & \ddot{\boldsymbol{c}}^T \end{bmatrix}^T \tag{11}$$

where the vector $\boldsymbol{c}$, $\dot{\boldsymbol{c}}$, and $\ddot{\boldsymbol{c}}$ concatenate all the $N_c \cdot N_v$ contact positions, velocities, and accelerations, respectively. The angular velocity vector $\boldsymbol{\omega}$ is calculated from the quaternion $\boldsymbol{o}$ using the *quaternion propagation* described in [20]. The optimal control problem requires the definition of cost terms and constraints equation, listed in the following paragraphs, as well as the choice of the states $\boldsymbol{x}$ and controls $\boldsymbol{u}$:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\pi}^T & \boldsymbol{\nu}^T \end{bmatrix}^T \tag{12}$$

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{\alpha}^T & \boldsymbol{W}^T \end{bmatrix}^T \tag{13}$$

with $\boldsymbol{W}$ being the vector of all the $N_c \cdot N_v$ contact forces.

A. Cost Terms

The cost function is made by several components whose minimization makes the robot behavior converge to the desired one. Each cost term is weighted to give more importance to the specific attitudes of the robot, and fine yet intuitive tuning is required to approach different locomotion problems.

1) CoM Height: If not otherwise advised, a naive solution for the CoM and contact forces that respect the equations of motion in (8) is the free-falling single rigid body. For this reason, tracking a reference CoM height above the contacts is essential to guarantee the robot's standing position. The CoM height task is formulated as follows:

$$\psi_{\mathrm{CoM},z} = \left\| r_{z,\mathrm{ref}} - r_z \right\|^2_{\Lambda_{\mathrm{CoM},z}} \tag{14}$$

where $\Lambda_{\mathrm{CoM},z}$ is a scalar weight value to track the reference value $r_{z,\mathrm{ref}}$.

2) Contact Tracking: The robot locomotion naturally emerges from the MPC solution by tracking the contact sequence from the contact planner. Indeed, the optimal control problem generates a CoM trajectory and contact forces to track the nominal contact trajectory while keeping stability, even if a reference CoM trajectory is not injected in the optimization problem. The contact tracking term is split into two terms to track the references on the x-y and z-axis differently:

$$\psi_{\boldsymbol{c}} = \sum_{i=0}^{N_c} \sum_{j=0}^{N_v} \left[\left\| \boldsymbol{c}_{x,y,\text{ref}}^{(i,j)} - \boldsymbol{c}_{x,y}^{(i,j)} \right\|_{\Lambda_{\boldsymbol{c}_{x,y}}}^2 + \left\| c_{z,\text{ref}}^{(i,j)} - c_z^{(i,j)} \right\|_{\Lambda_{c_z}}^2 \right] \tag{15}$$

With $\boldsymbol{c}_{x,y,\text{ref}}$ and $c_{z,\text{ref}}$ being the reference values on the x-y and z axis, respectively, and $\Lambda_{\boldsymbol{c}_{x,y}} < \Lambda_{c_z}$ being the weights of the two cost terms. By weighing less the x-y components of the tracking terms, the MPC can autonomously adjust the contact location in the neighborhood of the nominal ones while following the swing trajectory on the z-axis. This strategy is beneficial to react to an external disturbance and avoid the fall, but also to re-arrange the contact location optimally depending on the robot's dynamic state. Indeed, the contact sequence coming from the contact space planner is aware of the surrounding moving obstacles, but it is based on heuristics, which do not consider the robot's whole-body motion. Thus, the nominal contact sequence can be intended as a high-level initial guess that draws a feasible path avoiding any unexpected and undesirable interaction with the environment. The MPC acts on a lower level, optimizing the step position and analyzing the robot's state during the execution of the locomotion task. Although this approach does not guarantee collision safety, fine-tuning and a proper selection of the heuristics can mitigate the possibility of a collision.

Given two adjacent contacts of the same end-effector $\mathscr{F}_i$, the reference trajectory on the x-y axis is computed by tracing a line that starts from the initial pose and ends at the goal pose, while the reference on the z-axis is defined as a half-period sine function).

3) Regularization Terms: To reduce the rate of change of the CoM velocity $\ddot{\boldsymbol{r}}$, and to minimize the exerted contact force $\boldsymbol{W}^{(i,j)}$ of each contact, as well as the angular velocity of the base $\boldsymbol{\omega}$ that can generate undesirable differences in the robot inertia, three regularization terms are added to the cost function in the form:

$$\psi_{\text{reg}} = \|\ddot{\boldsymbol{r}}\|_{\Lambda_{\ddot{\boldsymbol{r}}}}^2 + \sum_{i=0}^{i=N_c} \sum_{j=0}^{j=N_v} \left\| \boldsymbol{W}^{(i,j)} \right\|_{\Lambda_{\boldsymbol{W}^{(i,j)}}}^2 + \|\boldsymbol{\omega}\|_{\Lambda_{\boldsymbol{\omega}}}^2 \tag{16}$$

B. Constraints

A list of inequality constraints bounds the contact forces and relative positions during the contact and fly phases. Equality constraints are used to impose the SRBD in (8).

1) Reaction Force Constraints: Contact forces are bounded to match the surface friction characteristics using the non-linear friction cone. The force should lie inside a linearized friction cone to guarantee safe contact. The cone's height depends on the normal component exerted, and the angle depends on the static friction coefficient μ_s. Assuming a uniform static friction coefficient, the constraint is formalized as follows:

$$\begin{bmatrix} -1 & 0 & -\mu_s \\ 1 & 0 & \mu_s \\ 0 & -1 & -\mu_s \\ 0 & 1 & \mu_s \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \hat{W}_x^{(i,j)} \\ \hat{W}_y^{(i,j)} \\ \hat{W}_z^{(i,j)} \end{bmatrix} = \boldsymbol{f}_c \hat{\boldsymbol{W}}^{(i,j)} \leq \boldsymbol{0}_{5\times 1} \tag{17}$$

with $\hat{\boldsymbol{W}}^{(i,j)} = \boldsymbol{R}^{(i,j)T} \boldsymbol{W}^{(i,j)}$, and $\boldsymbol{R}^{(i,j)T}$ the rotation matrix expressing the orientation of the (i,j) contact w.r.t. the inertial frame.

Further, the force must be bound to zero when the contact is non-active since the contact cannot exert any force. This is done by bounding the contact force between zero values during the flying phases.

$$\boldsymbol{W}^{(i,j)} = \boldsymbol{0} \quad \text{if } \boldsymbol{c}^{(i,j)} \text{ is non-active} \tag{18}$$

2) Contact Constraints: Modeling a surface contact with two or more point contacts requires a constraint that fixes the relative distance for this point since they belong to the same rigid body. This is done by setting the relative velocity between the vertices belonging to the same contact to zero:

$$\dot{\boldsymbol{c}}^{(i,j)} - \dot{\boldsymbol{c}}^{(i,k)} = \boldsymbol{0} \tag{19}$$

with $i = (0, \ \dots \ , N_c)$, $j = (0, \ \dots \ , N_v)$, and $k = (j, \ \dots \ , N_v)$, thus resulting in a set of $N_c \cdot (N_v - 1)!$ constraints. Additionally, in a dual way w.r.t. the force constraints, the contact velocity must be zero during the contact phases.

$$\dot{\boldsymbol{c}}^{(i,j)} = \boldsymbol{0} \quad \text{if } \boldsymbol{c}^{(i,j)} \text{ is active}$$

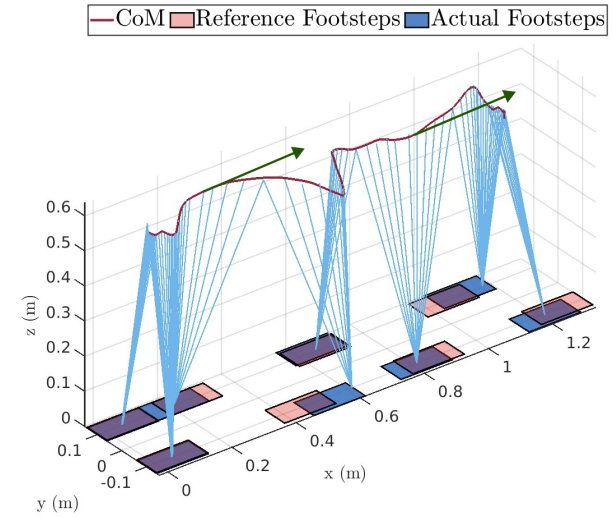
V. MPC Formulation

By combining the cost terms in Sec. IV-A and the constraints in Sec. IV-B, this Section introduces the non-linear optimization problem whose solution gives the CoM and reaction force trajectories, i.e., the decision variables for the Optimal Control Problem (OCP), which are then fed as references into the low-level WBC. The OCP has been transcripted using a Direct Multiple Shooting (DMS) method [21] using the Receding Horizon Principle [22] and assuming a fixed size prediction time window $T = 1.5$s discretized on $N = 20$ time intervals, equally spaced by a constant sample time $dt = T/N = 0.075$ms. The DMS method discretizes the continuous OCP into N *shooting intervals* to solve the resulting Non-Linear Program (NLP) efficiently. The shooting intervals discretize the original problem into a grid of $N+1$ states $\boldsymbol{x}_k$, and N controls $\boldsymbol{u}_k$:

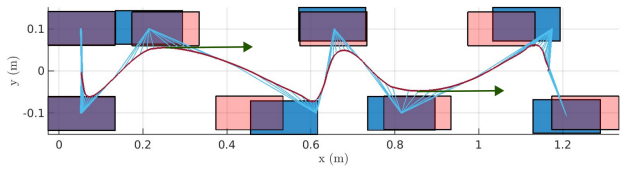
$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_0^T & \boldsymbol{x}_1^T & \dots & \boldsymbol{x}_N^T \end{bmatrix}^T \tag{20}$$

and the N control are collected in the control vector $\boldsymbol{U}$:

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_0^T & \boldsymbol{u}_1^T & \dots & \boldsymbol{u}_{N-1}^T \end{bmatrix}^T \tag{21}$$



(a)



(b)

Fig. 3: MPC solution in the forward walking scenario rejecting a force of magnitude $2.5 \cdot m_r$ (green arrows). The red line is the planned CoM trajectory, the blue line connecting the CoM and the footholds indicates when contact is active, the blue and red tiles are the reference footsteps from the contact planner, and the actual ones are computed by the MPC reacting to the external disturbances. Side 3a and top view 3b

Thus, the optimization problem becomes:

$$\min_{\boldsymbol{X},\boldsymbol{U}} \sum_{k=0}^{N} \psi_k = \min_{\boldsymbol{X},\boldsymbol{U}} \sum_{k=0}^{N} \left(\psi_{k,\text{CoM}} + \psi_{k,\boldsymbol{c}} + \psi_{k,\text{reg}}\right) \tag{22a}$$

s.t.

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{\phi}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)dt \tag{22b}$$

$$\boldsymbol{x}_{k+1} - \tilde{\boldsymbol{x}}_k = 0 \tag{22c}$$

$$\boldsymbol{f}_c \boldsymbol{R}_k^{(i,j)T} \boldsymbol{W}_k^{(i,j)} \leq 0 \tag{22d}$$

$$\dot{\boldsymbol{c}}_k^{(i,j)} - \dot{\boldsymbol{c}}_k^{(i,k)} = 0 \tag{22e}$$

$$\begin{cases} \dot{\boldsymbol{c}}_k^{(i,j)} = 0 & \text{if } \boldsymbol{c}_k^{(i,j)} \text{ is active} \\ \boldsymbol{W}_k^{(i,j)} = 0 & \text{if } \boldsymbol{c}_k^{(i,j)} \text{ is non-active} \end{cases} \tag{22f}$$

$$\tag{22g}$$

The discrete DMS requires the extra condition in (22c), called *continuity condition*. Indeed, the OCP solves contemporary for all the N time segments. This constraint guarantees that the propagation (integration) of the state $\boldsymbol{x}_k$ from t_k to $t_{k+1} = t_k + dt$, evaluated at the final time t_{k+1}, coincides with the initial state value $\boldsymbol{x}_{k+1}$ of the adjacent segment from t_{k+1} to t_{k+2}, ensuring that the two segments join at the boundaries.

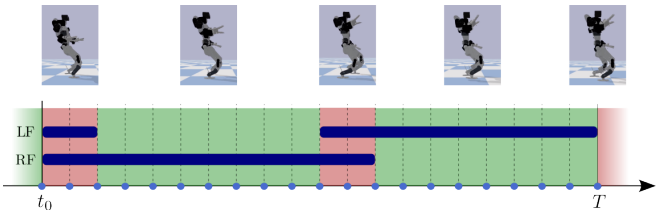


Fig. 4: Graphical representation of the gait sequence and timing for the walking simulations carried out on the DRACO 3 humanoid robot. The blue dots represent the receding horizon nodes, and the blue areas show the contact timing for the left (LF) and right (RF) feet. The red and green areas represent double and single stance zones, respectively

When the control loop is closed, the initial state of the optimization $\boldsymbol{x}_0$ is set to be equal to the robot's measured state, allowing the MPC to sense any external disturbance and react to that.

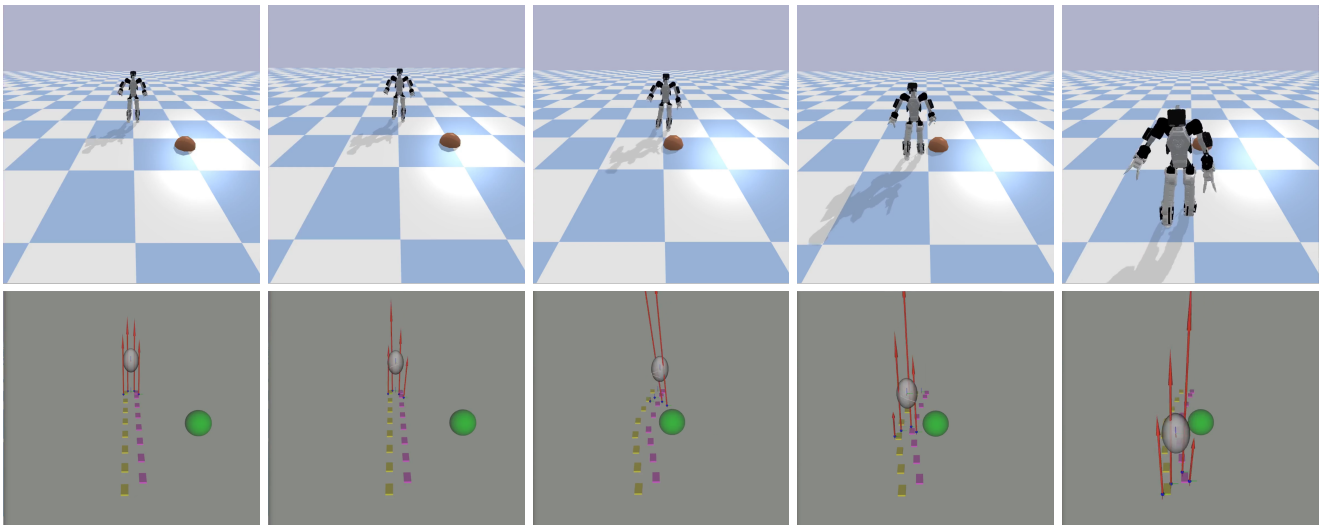
VI. Results

The OCP is created and managed using the Horizon framework presented in [11], and the MA57 [23] and IPOPT [24] libraries are used to solve the non-linear problem in (22). The simulations carried out on the DRACO 3 robot custom-built by Apptronik and upgraded for extended usage by HCRL at the University of Texas at Austin [25] are collected in the attached video.[1] DRACO 3 is a biped humanoid robot $h_r = 1.35$ m tall and $m_r = 39$ kg heavy, with 25 DoF, six for each limb, and one neck pitch joint to actuate the Multisense S7 mounted on its head. The robot has been designed with the legs' proximal actuation achieved bearing a cable-based drive system, which reduces the robot leg mass, making the SRBD a suitable model simplification. Simulations use the PyBullet dynamic engine on a desktop PC mounting an Intel® Core™ i7-9700K Octa-core CPU @ 3.60GHz, an NVIDIA GeForce GTX 1650, and a memory of 16 GB.

First, the disturbance rejection has been tested by pushing the robot twice while walking forward with a relatively strong force of magnitude $2.5 \cdot m_r$. Fig. 3 shows the contacts and CoM trajectories planned by the MPC. Specifically, when the robot is pushed forward at $x_{CoM} \simeq 0.25$ m, and $x_{CoM} \simeq 0.85$ m the robot takes a longer step with the right foot to recover from falling caused by the sudden acceleration along the x-axis. Then, the robot decelerates to match the contact references with the left foot again and restore the nominal planned walking trajectory. The same happens later with the right foot, with the MPC able to reject both disturbances.

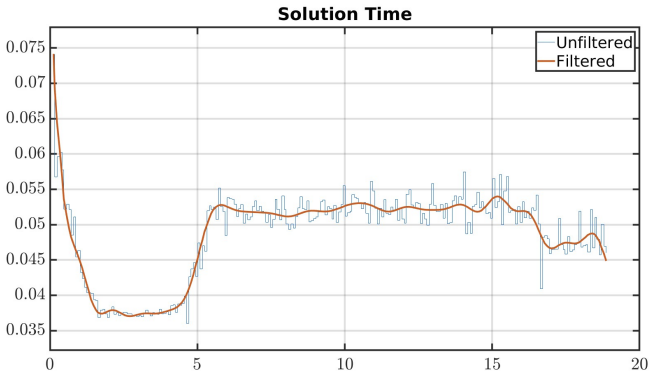
Further, the contact space local planner and the MPC have been tested to plan a safe and robust walk forward while avoiding a moving spherical obstacle that suddenly enters the scene and interferes with the nominal trajectory. The MPC references are linearly interpolated to match the control layer frequency, which runs at 800 Hz. The Cartesian and force references from the MPC solution are converted in robot joint torques using the WBC presented in [10].

[1] https://youtu.be/RqJ6GH32LWI

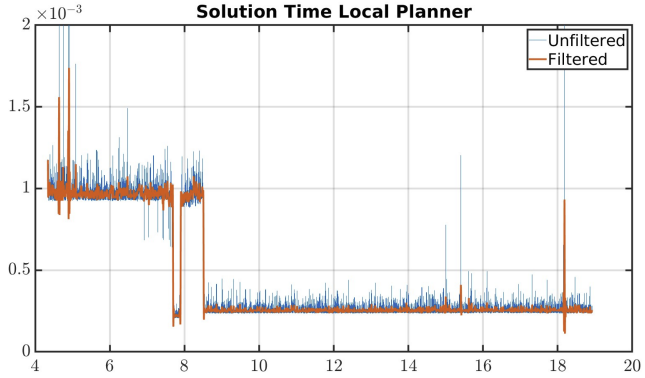


(a) $t = 0$ (b) $t = 2$ (c) $t = 4$ (d) $t = 7$ (e) $t = 12$

Fig. 5: Screenshots of the simulations carried out in the dynamic obstacle avoidance scenario



(a)



(b)

Fig. 6: Solution time required by the MPC (6a) and the contact space local planner (6b)

At this stage, a naive forward and equally spaced walking sequence is used as a nominal global solution, and the position of the moving obstacle is known to the local planner. A perception algorithm for a realistic environment can be easily implemented following our previous work in [18]. The gait timing is also defined as alternating double and single stances distributed along the $N+1$ nodes to guarantee a more stable walk. Specifically, the robot has to take two steps in the given time horizon length T by:

- Standing on two feet for the first two nodes.
- Taking a step with the left foot for the next eight nodes.
- Transiting between the left and right step, passing through a two-nodes double stance.
- Taking a step with the right foot for the last eight nodes.

A graphical representation of the gait sequence and timing is given in Fig. 4. The presented simulation shows results carried out in a dynamic environment with one spherical obstacle with a radius of 30 cm entering the scene and perturbing the nominal trajectory during its execution. At $t = 2$ s (Fig. 5b), during the execution of the second step, the obstacle starts moving and after two seconds stops in the middle of the nominal contact sequence (Fig. 5c). The contact space local planner instantaneously reacts by correctly adjusting the feet' position to walk around the obstacle. The optimized contacts are sent to the MPC, which generates a feasible trajectory to accomplish the task, even in the presence of the moving obstacle (Figs. 5d, 5e). A computational time of approximately 0.46 ms, Fig. 6, for the contact space local planner permits its implementation in the inner control loop, which usually runs around 1 kHz. Furthermore, the MPC average solution time (55 ms) is well below the threshold of the 75 ms set by the optimal control problem parameters and does not show evident fluctuation during the motion of the obstacle, occurring around $t = 7$ s.

VII. Conclusions

This paper proposes an online planning strategy to cope with dynamic moving obstacles during locomotion. This is done by exploiting a nominal footstep sequence that is continuously adjusted according to the perceived position of the moving obstacles. A fast SRBD-based MPC module generates CoM and contacts motions that avoid obstacles tracking the footsteps, which also provides a good initial guess for the OCP, improving the convergence of the solution. Finally, a WBC maps the Cartesian and force references to the joint torque references sent to the robot. Preliminary results were obtained in simulations on the humanoid robot DRACO 3, fostering a future implementation on the real hardware. The reported simulation tested the MPC strategy alone and coupled it with the contact space local planner to avoid a moving obstacle interfering with the nominal trajectory during the execution of the task.

Future work will focus on implementing the planning pipeline on the real hardware in increasing complexity scenarios to exploit all the capabilities of the presented planning and control algorithm on non-flat terrains.

Currently, the contact space local planner is acting on the contact poses only, while the gait timing is unchanged from the nominal offline computed trajectory. However, augmenting the hyper-graph vertices to include the time of occurrence of each contact, depending on the dynamic state of the robot and environment, will enable the planner to change the gait sequence as well. Additionally, considering time as a variable will allow the definition of new objective functions, such as minimizing the trajectory execution time, which is particularly appealing in modern applications. However, this involves vertex augmentation, requiring a deeper investigation of the constraints the footstep planner uses to guarantee a safe transition even when the local planner adjusts the gait sequence.

VIII. Acknowledgment

The Draco 3 humanoid robot has been supported by the US Navy ONR Award Nr. N000142212204. This project has received funding from the European Union's Horizon 2020 under grant agreement No. 101016007 CONCERT.

References

[1] T. Koolen, S. Bertrand, G. Thomas, T. De Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a momentum-based control framework and application to the humanoid robot atlas," *International Journal of Humanoid Robotics*, vol. 13, pp. 1 650 007–1, 03 2016.

[2] D. Kanoulas, A. Stumpf, V. S. Raghavan, C. Zhou, A. Toumpa, O. Von Stryk, D. G. Caldwell, and N. G. Tsagarakis, "Footstep planning in rough terrain for bipedal robots using curved contact patches," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4662–4669.

[3] S. Caron and Q.-C. Pham, "When to make a step? tackling the timing problem in multi-contact locomotion by topp-mpc," 2016.

[4] G. Romualdi, S. Dafarra, G. L'Erario, I. Sorrentino, S. Traversaro, and D. Pucci, "Online non-linear centroidal mpc for humanoid robot locomotion with step adjustment," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 412–10 419.

[5] N. Perrin, *Biped Footstep Planning*. Dordrecht: Springer Netherlands, 2019, pp. 1697–1717.

[6] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 629–634.

[7] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," *IEEE-RAS International Conference on Humanoid Robots*, 04 2009.

[8] J. Garimort, A. Hornung, and M. Bennewitz, "Humanoid navigation with dynamic footstep plans," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3982–3987.

[9] M. Diehl, H. Bock, H. Diedam, and P.-B. Wieber, *Fast Direct Multiple Shooting Algorithms for Optimal Robot Control*. Springer Berlin Heidelberg, 2006, pp. 65–93.

[10] J. Ahn, S. J. Jorgensen, S. H. Bang, and L. Sentis, "Versatile locomotion planning and control for humanoid robots," *Frontiers in Robotics and AI*, vol. 8, 2021.

[11] F. Ruscelli, A. Laurenzi, N. G. Tsagarakis, and E. Mingo Hoffman, "Horizon: A trajectory optimization framework for robotic systems," *Frontiers in Robotics and AI*, vol. 9, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2022.899025

[12] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[13] M. Y. Galliker, N. Csomay-Shanklin, R. Grandia, A. J. Taylor, F. Farshidian, M. Hutter, and A. D. Ames, "Planar bipedal locomotion with nonlinear model predictive control: Online gait generation using whole-body dynamics," 2022.

[14] D. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, 10 2013.

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

[16] D. Calvert, B. Mishra, S. McCrory, S. Bertrand, R. Griffin, and J. Pratt, "A fast, autonomous, bipedal walking behavior over rapid regions," in *IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 24–31.

[17] I. Kumagai, M. Morisawa, S. Nakaoka, and F. Kanehiro, "Efficient locomotion planning for a humanoid robot with whole-body collision avoidance guided by footsteps and centroidal sway motion," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 251–256.

[18] L. Rossini and N. G. Tsagarakis, "From offline to online: A perception-based local planner for dynamic obstacle avoidance," in *IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 163–170.

[19] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[20] B. Graf, "Quaternions and dynamics," 2008.

[21] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*, 2nd ed. Society for Industrial and Applied Mathematics, 2010.

[22] D. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 7, pp. 814–824, 1990.

[23] I. S. Duff, "Ma57—a code for the solution of sparse symmetric definite and indefinite systems," *ACM Trans. Math. Softw.*, vol. 30, no. 2, p. 118–144, 2004.

[24] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 03 2006.

[25] S. H. Bang, C. Gonzalez, J. Ahn, N. Paine, and L. Sentis, "Control and evaluation of a humanoid robot with rolling contact knees," 2022.