SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER, CONTROL AND MANAGEMENT ENGINEERING

Real-Time Safe Bipedal Robot Navigation using Linear Discrete Control Barrier Functions

AUTONOMOUS AND MOBILE ROBOTICS

Students:
Eugenio Bugli
Damiano Imola
Salvatore Michele Rago

Academic Year 2024/2025

Contents

1 Introduction

Eugenio:
Humanoid robots are inherently underactuated thanks to to unilateral ground contacts thus a strong coupling exists between path planning and gait control. A path is considered safe if the robot does not collide with any obstacle and its dynamics and physical limitations are respected. Due to the high complexity of humanoids, we cannot decouple path planning from motion control without taking into account the dynamics. We should solve gait optimization problems based on the robot's full order model or the reduced one. Due to computational complexity, reduced order models such the Linear Inverted Pendulum (LIP) are often employed. In our case, since Control Barrier Functions are employed to ensure safety in path planning, we will pre-compute heading angles ans use approximated linead DCBFs. This is needed since we may have problems at computation level due to the non linearity of kinematics and path constraints.

Salvatore:
The term *humanoid* refers to a robot with structure and kinematics similar to a human body. It is designed for locomanipulation and represent the best choice to navigate and interact in an environment that is structured for humans.
Real-time safe navigation is a crucial task for humanoid robots in real-world applications. A path is considered safe if it does not collide with any obstacle while fulfilling the robot's dynamics and physical constraints. In order to carry out such complex task in real-time, path planning is usually decoupled from gait control, resulting in a significant reduction of the computational load.
The aim of this work is to implement the solution proposed by Peng et al. in "Real-Time Safe Bipedal Robot Navigation using Linear Discrete Control Barrier Functions", which consists in a unified safe path and gait planning framework to be executed in real-time. It models the humanoid's walking dynamics by a Linear Inverted Pendulum, and leverages Model Predictive Control and Control Barrier Functions to deliver a collision-free path while satisfying specific constraints.
In the following chapters, we will delve into the details of this approach, discuss the results, and propose some improvements.

2 LIP

Eugenio:
This reduced model assumes that during the motion the Center of Mass (CoM) will have a constant height H.

$$\dot{v}_x = \frac{g}{H}(p_x - f_x) \qquad \dot{v}_y = \frac{g}{H}(p_y - f_y) \tag{1}$$

With (p_x, p_y) we denote the position of the CoM and with (v_x, v_y) its velocity with respect to the x-axis and y-axis. The stance foot position, which is the position in which both feets are in contact with the ground, is denoted with (f_x, f_y).

Given the position (p_{x_k}, p_{y_k}) and velocities (v_{x_k}, v_{y_k}) of the CoM at the k-th step, the closed-form solutions of the step-to-step discrete dynamics can be written as follows :

$$\begin{bmatrix} p_{x_{k+1}} \\ v_{x_{k+1}} \end{bmatrix} = A_d \begin{bmatrix} p_{x_k} \\ v_{x_k} \end{bmatrix} + B_d f_{x_k} \qquad \begin{bmatrix} p_{y_{k+1}} \\ v_{y_{k+1}} \end{bmatrix} = A_d \begin{bmatrix} p_{y_k} \\ v_{y_k} \end{bmatrix} + B_d f_{y_k} \tag{2}$$

Where $\beta = \sqrt{\frac{g}{H}}$ and the two matrices are:

$$A_d = \begin{bmatrix} cosh(\beta T) & \frac{sinh(\beta T)}{\beta} \\ \beta sinh(\beta T) & cosh(\beta T) \end{bmatrix} \qquad B_d = \begin{bmatrix} 1 - cosh(\beta T) & -\beta sinh(\beta T) \end{bmatrix} \tag{3}$$

By defining the state of our system as $x = [p_x, v_x, p_y, v_y, \theta]^T \in \mathbb{R}^5$ and the control input as $u = [f_x, f_y, \omega]^T \in \mathbb{R}^3$, where θ is the heading angle and ω is its turning rate, the step-to-step dynamics of the 3D-LIP model is written as follows:

$$x_{k+1} = A_L x_k + B_L u_k \tag{4}$$

Where the two matrices are defined as follows:

$$A_L = \begin{bmatrix} A_d & 0 & 0 \\ 0 & A_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B_d = \begin{bmatrix} B_d & 0 & 0 \\ 0 & B_d & 0 \\ 0 & 0 & T \end{bmatrix} \tag{5}$$

3 3D-LIP Model with Heading Angles

Salvatore:
If the full dynamic model of the humanoid is used to simulate its motion, it becomes computationally impossible to perform joint path and gait planning, due to its high dimensionality and non-linearity. Therefore, a simplifying model must be used. For this scope Peng et al. introduced the "3D-LIP Model with Heading Angle", which describes the discrete dynamics of the Center of Mass (CoM) similarly to the one of an inverted pendulum in three dimensions.

3.1 Local Robot Reference Frame

The state $\mathbf{x}$ and the input $\mathbf{u}$ of the dynamic model are defined as:

$$\mathbf{x} := \big(p_x,\ v_x,\ p_y,\ v_y,\ \theta\big)^T \in \mathcal{X} \subset \mathbb{R}^5,$$
$$\mathbf{u} := \big(f_x,\ f_y,\ \omega\big)^T \in \mathcal{U} \subset \mathbb{R}^3,$$

where (p_x, v_x) are the CoM position and translational velocity along the x-axis, f_x is the x-coordinate of the stance foot position, θ and ω are the humanoid's orientation and turning rate, respectively. $\mathcal{X}$ is the set of the allowed states, while $\mathcal{U}$ is the set of the admissible inputs.

Both the state and the input are expressed in the local coordinates of the robot, which are time-related. It means that (p_{x_k}, p_{y_k}) represents the position of the CoM at simulation time step k in the reference frame (RF_k) that originates from $(p_{x_{k-1}}, p_{y_{k-1}})$, and is rotated by an angle θ_k around the z-axis with respect to RF_{k-1}. The relation between the vectors in different reference frames is represented in Figure 1.

The reference frame at time step 0 is considered the "inertial" or "global" frame. A transformation between the inertial and moving frames is necessary to obtain the position of the humanoid in the global map and to deal with obstacles. Transformations are performed as follows:

$$\mathbf{T_k} = \begin{pmatrix} \cos\theta_{k,\,\mathrm{glob}} & -\sin\theta_{k,\,\mathrm{glob}} & p_{x,\,k-1,\,\mathrm{glob}} \\ \sin\theta_{k,\,\mathrm{glob}} & \cos\theta_{k,\,\mathrm{glob}} & p_{y,\,k-1,\,\mathrm{glob}} \\ 0 & 0 & 1 \end{pmatrix},$$

$$\begin{pmatrix} f_{x,\,k,\,\mathrm{glob}} \\ f_{y,\,k,\,\mathrm{glob}} \\ 1 \end{pmatrix} = \mathbf{T_k} \begin{pmatrix} f_{x,\,k,\,\mathrm{loc}} \\ f_{y,\,k,\,\mathrm{loc}} \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} p_{x,\,k,\,\mathrm{glob}} \\ p_{y,\,k,\,\mathrm{glob}} \\ 1 \end{pmatrix} = \mathbf{T_k} \begin{pmatrix} p_{x,\,k,\,\mathrm{loc}} \\ p_{y,\,k,\,\mathrm{loc}} \\ 1 \end{pmatrix},$$

$$\begin{pmatrix} v_{x,\,k,\,\mathrm{glob}} \\ v_{y,\,k,\,\mathrm{glob}} \end{pmatrix} = \begin{pmatrix} \cos\theta_{k,\,\mathrm{glob}} & -\sin\theta_{k,\,\mathrm{glob}} \\ \sin\theta_{k,\,\mathrm{glob}} & \cos\theta_{k,\,\mathrm{glob}} \end{pmatrix} \begin{pmatrix} v_{x,\,k,\,\mathrm{loc}} \\ v_{y,\,k,\,\mathrm{loc}} \end{pmatrix},$$

$$\theta_{k+1,\mathrm{glob}} = \theta_{k,\mathrm{glob}} + \theta_{k+1,\mathrm{loc}}, \qquad \omega_{\mathrm{glob}} = \omega_{\mathrm{loc}}.$$

The positional vectors are roto-translated using a 3×3 homogeneous matrix. The velocity vectors are only rotated around the z-axis (indeed, a translation would change their magnitude) using a rotation matrix. The global robot's orientation is obtained by summing the latest local variation to the previous global angle. The angular velocity does not need to be transformed because it is along the z-axis, which is fixed.

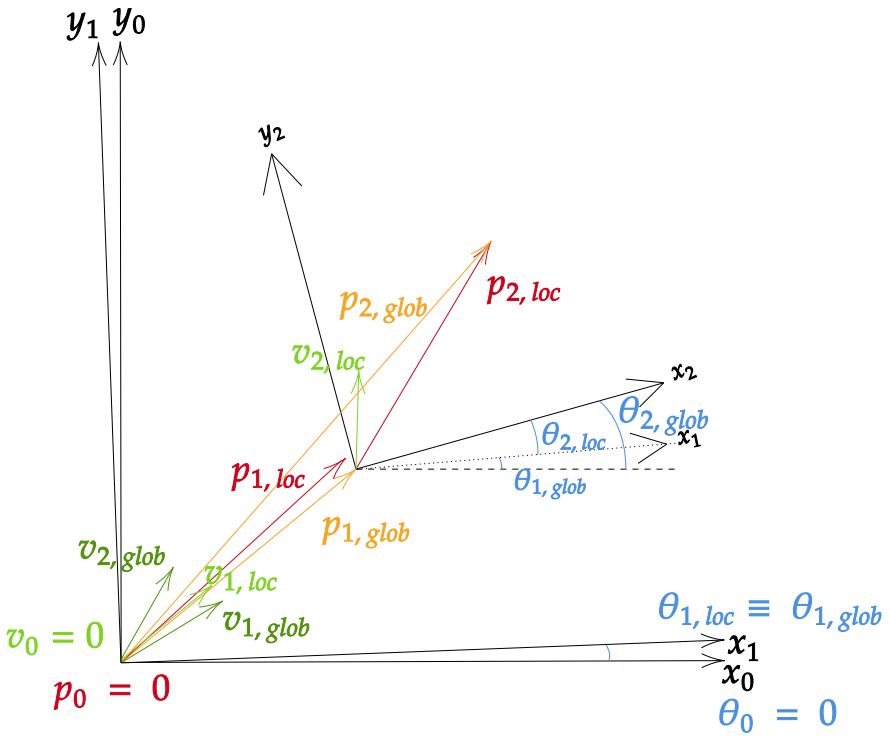

Figure 1: An example of the evolution of the 3D-LIP model's state, highlighting the relationship between local and global coordinates. The initial state is $\mathbf{0}$ and the RF $(x_0,\, y_0)$ is the inertial frame. The local RF translates and rotates with the simulation time k: the position of RF_2 $(x_2,\, y_2)$ has its origin in $p_{1,\text{glob}}$, while its orientation is given by $\theta_{2,glob}$. The pose of the successive frames is computed analogously.

3.2 Model Definition

Assuming that the height H of the CoM is constant during the motion, according to [1] we can express the CoM velocity along the x-axis as:

$$\dot{v}_x = \frac{g}{H}\left(p_x - f_x\right),$$

with g as the magnitude of the gravitational acceleration. Since by definition $\dot{\theta} \coloneqq \omega$, by Euler integration we obtain that:

$$\theta_{k+1} = \theta_k + \omega\, T,$$

where T is the duration of the sampling interval. By further assuming that the duration of a single humanoid's step is fixed and equal to T, we can derive the closed-form step-to-step discrete dynamics of the 3D-LIP model:

$$\mathbf{x}_{k+1} = \mathbf{A_L}\,\mathbf{x}_k + \mathbf{B_L}\,\mathbf{u}_k, \tag{6}$$

with:

$$\mathbf{A_L} \coloneqq \begin{pmatrix} \mathbf{A_d} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{A_d} & 0 \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix}, \qquad \mathbf{B_L} \coloneqq \begin{pmatrix} \mathbf{B_d} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{B_d} & 0 \\ 0 & 0 & T \end{pmatrix},$$

$$\mathbf{A_d} \coloneqq \begin{pmatrix} \cosh \beta T & \frac{\sinh \beta T}{\beta} \\ \beta \sinh \beta T & \cosh \beta T \end{pmatrix}, \qquad \mathbf{B_d} \coloneqq \begin{pmatrix} 1 - \cosh \beta T \\ -\beta \sinh \beta T \end{pmatrix}, \qquad \beta \coloneqq \sqrt{\frac{g}{H}}$$

In the following chapter, these dynamics will be used as the internal process representation of an MPC. Along with the constraints, they will grant that the found solution is meaningful to the humanoid motion.

4 MPC

Eugenio:
The dynamic model is used to define a MPC problem to compute the optimal stepping positions for save nagivation and locomotion. The LIP-MPC is defined as:

$$J^* = \min_{u_{0:N-1}} \sum_{k=1}^{N} [(p_{x_k} - g_x)^2 + (p_{y_k} - g_y)^2]$$
$$\text{s. t} \quad x_{k+1} = A_L x_k + B_L u_k \qquad c_l \leq c(x_k, u_k) \leq c_u$$

Minimizing the cost function means that the robot is moving towards the goal position. This problem is subject to the satisfaction of the robot's dynamics and the kinematics and path constraints. The kinematics and path constraints are captured in $c(x_k, u_k)$ and they are often nonlinear. This nonlinearity is related to presence of the heading angle θ_k. These constraints are linearized by pre-computing the turning rates $\bar{\omega}_k$ for all the horizon length, in order to have them fixed in the MPC calculation.

5 LIP-MPC: Gait planning with Model Predictive Control

Salvatore:
The LIP dynamics defined in (6) is used as a model of the process inside a Model Predictive Control (MPC) scheme. The MPC controller uses that model to predict the future output along a *prediction horizon* N, namely the predefined number of time steps to look out in the future. Based on those forecasts and the provided constraints, the MPC computes the sequence of control actions that optimize a given cost function during the prediction horizon. Then, only the first input of that sequence is taken and provided to the process. The real output will be used at the next time step to compute the new predictions and control actions.
In our case, the LIP-MPC is used to respond instantaneously to the changes in the humanoid's state, while providing optimal stepping positions for stable locomotion and safe navigation. It is formulated as follows:

$$
\begin{aligned}
J^* = \min_{\mathbf{u}_{0:N-1}} & \sum_{k=1}^{N} q(\mathbf{x}_k) \qquad (7)\\
\text{s.t.} \quad & \mathbf{x}_k \in X, \quad k \in [1, N] \\
& \mathbf{u}_k \in U, \quad k \in [0, N-1] \\
& \mathbf{x}_{k+1} = \mathbf{A_L}\mathbf{x}_k + \mathbf{B_L}\mathbf{u}_k, \quad k \in [0, N-1] \\
& \mathbf{c}_l \leq \mathbf{c}(\mathbf{x}_k, \mathbf{u}_k) \leq \mathbf{c}_u, \quad k \in [0, N-1],
\end{aligned}
$$

where $q(\mathbf{x}_k)$ is the cost function to minimize along the prediction horizon. It drives the humanoid toward the goal by minimizing the distance between its current position and the target position. It is defined as:

$$
q(\mathbf{x}_k) = \left(p_{x_k} - g_x\right)^2 + \left(p_{y_k} - g_y\right)^2 \qquad \forall k \in [1, N]\,,
$$

where the goal position (g_x, g_y) is expressed in the humanoid's local RF. The LIP dynamics is included in the MPC definition to specify how the future states are predicted. Whereas, all the constraints that the optimization problem is subject to are captured by $\mathbf{c}(\mathbf{x}_k, \mathbf{u}_k)$. In order to reduce the computational load, they are all expressed linearly, and they include the walking velocities, leg reachability, and maneuverability constraints, and the linear control barrier function, which will be described in details in the following sections.

5.1 Heading Angle Preprocessing

Many of the constraints imposed in the MPC make use of θ in a non-linear way: for example, the kinematic constraints often show sinusoidal terms having θ as argument. This hinders real-time computation. The solution proposed by Peng et al. in [1] consists in using precomputed values for θ and ω, and it is often sufficient to linearize some constraints.

The values of θ and ω are not determined by the optimization of the previously defined cost function performed by the MPC. Instead, their values throughout the prediction horizon are computed at the beginning of each time step with the following formulae:

$$
\begin{gathered}
\omega_k = \max\Big\{\min\left\{\text{atan2}(g_y - p_y,\, g_x - p_x) - \theta, \quad \omega_{max}\right\}, \quad \omega_{min}\Big\} \qquad \forall k \in [0, N-1]\,, \\
\theta_0 = \theta, \qquad \theta_k = \theta_{k-1} + \omega_k\, T \qquad \forall k \in [1, N]\,,
\end{gathered}
$$

where p_x, p_y and θ are the position and orientation of the humanoid at the start of the simulation time step, T is the sampling time, and the goal position is expressed in local coordinates. ω_{min} and ω_{max} are the bounds on the robot turning rate: they must be added due to the actuation limits and to avoid sharp turns, which would threaten

the stability of the humanoid. With the specified formulae, θ rotates with a velocity ω until the robot points to the goal.

5.2 Kinematic Constraints

The stepping positions provided by the MPC solution must comply with specific kinematic constraints in order to be physically feasible. In this work, Peng et al. decided to enforce the walking velocities, leg reachability, and maneuverability constraints defined as follows.

5.2.1 Walking Velocities Constraint

$$\begin{pmatrix} v_{x_{\min}} \\ v_{y_{\min}} \end{pmatrix} \leq \begin{pmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{pmatrix} \begin{pmatrix} v_{x_{k+1}} \\ s_v v_{y_{k+1}} \end{pmatrix} \leq \begin{pmatrix} v_{x_{\max}} \\ v_{y_{\max}} \end{pmatrix} \qquad \forall k \in [0,\, N-1]\,.$$

The pre-multiplying matrix rotates the velocities vector by an angle $-\theta_k$ around the z-axis: namely, it takes the velocity vector $\mathbf{v}_{k+1}$ back to the orientation of RF_k. In the vector resulting from the transformation, the first component is the forward velocity, while the second is the lateral velocity, both in RF_k. At this point, the constraint imposes that the velocity is within the lateral and longitudinal limits, $v_{y_{\{\min,\,\max\}}}$ and $v_{x_{\{\min,\,\max\}}}$ respectively.
s_v is 1 if stance foot is the right one, -1 otherwise. It is meant to limit the lateral velocity in such a way that, at the end of each step, the stance foot is on the opposite side but the humanoid does not lose balance. For instance, if the right foot is the stance, since the body tends to fall on the left side, we want the lateral velocity not to increase too much toward the direction of the positive y. Therefore, we set $s_v = 1$, pushing the MPC to find a solution which is closer to $v_{y_{\min}}$.

5.2.2 Leg Reachability Constraint

$$\begin{pmatrix} -l_{\max} \\ -l_{\max} \end{pmatrix} \leq \begin{pmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{pmatrix} \begin{pmatrix} p_{x_k} \\ p_{y_k} \end{pmatrix} \leq \begin{pmatrix} l_{\max} \\ l_{\max} \end{pmatrix} \qquad \forall k \in [0,\, N-1]\,,$$

As stated before, the matrix in this constraint transforms the orientation of $(p_{x_k},\, p_{y_k})$ to the one of RF_{k-1}. It prevents the inputs computed at time k from moving the CoM too far away from $(p_{x_{k-1}},\, p_{y_{k-1}})$. Indeed, the distance between the previous and the new position cannot be farther than $l_{\max}$, which is the maximum reachable distance of the swing foot in both directions on the ground.

5.2.3 Maneuverability Constraint

$$\begin{pmatrix} \cos\theta_k & \sin\theta_k \end{pmatrix} \begin{pmatrix} v_{x_k} \\ v_{y_k} \end{pmatrix} \leq v_{x_{\max}} - \frac{\alpha}{\pi}|\omega_k| \qquad \forall k \in [0,\, N-1]\,,$$

This constraint is meant to reduce the humanoid's walking speed while it is turning. For this reason, it combines the longitudinal and angular velocities. It ensures that the component of the CoM velocity in the robot's heading direction (given by the vectors projection on the left side) is lower than a quantity that depends on the turning rate and some constant values. α is a simulation parameter.

5.3 Control Barrier Functions

5.3.1 Definitions

Considering a continuous and differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, we can define the *safety set* S and its boundary ∂S as:

$$S \coloneqq \left\{ \mathbf{x} \in X \mid h(\mathbf{x}) \geq 0 \right\},$$
$$\partial S \coloneqq \left\{ \mathbf{x} \in X \mid h(\mathbf{x}) = 0 \right\}.$$

They are the region and its perimeter where the robot can freely move without colliding with obstacles. According to [2], assume that there exists a class $\mathcal{K}$ function γ such that

$$0 < \gamma(h(\mathbf{x})) \leq h(\mathbf{x}),$$

and the following holds:

$$\Delta h(\mathbf{x}_k,\, \mathbf{u}_k) \coloneqq h(\mathbf{x}_{k+1}) - h(\mathbf{x}_k),$$
$$\forall \mathbf{x}_k \in S \qquad \exists \mathbf{u}_k \text{ s.t. } \Delta h(\mathbf{x}_k,\, \mathbf{u}_k) \geq -\gamma(h(\mathbf{x}_k)). \tag{8}$$

Then, $h(\cdot)$ is a discrete control barrier function (DCBF). We can also take γ as a scalar such that $0 < \gamma \leq 1$, and (8) becomes:

$$\forall \mathbf{x}_k \in S \qquad \exists \mathbf{u}_k \text{ s.t. } \Delta h(\mathbf{x}_k,\, \mathbf{u}_k) \geq -\gamma * h(\mathbf{x}_k).$$

It means that, if the state starts from the safety set S, there exists an input that, when applied, produces a state which will sill be in S: namely, S is invariant.

5.3.2 Linear Discrete Control Barrier Functions (LDCBF)

Assume that for each obstacle there is a function of the robot's configuration such that $F(\mathbf{x}) = 0$ when the robot touches the boundary of the obstacle, $F(\mathbf{x}) > 0$ when it does not collide with the obstacle, and $F(\mathbf{x}) < 0$ if it is inside the obstacle. It is convenient to choose $h(\cdot) = F(\cdot)$. This is the case in our project, where the DCBF is a function of $\vec{x}$, the Cartesian position of the CoM: therefore, $h(\cdot) \colon \mathbb{R}^2 \to \mathbb{R}$. Moreover, we assume that either the obstacle associated with $F(\cdot)$ is convex or $F(\cdot)$ describes the

convex shape that encloses the non-convex obstacle. However, if $F(\cdot)$ is a non-linear function, we obtain a non-linear DCBF constraint, which should be avoided for the reasons exposed before.
Peng et al. in [1] linearized the DCBF by approximating the safe region. The plane is partitioned in two halves: one containing the obstacle and the other containing the robot (as shown in Fig. 2). The safe region is the latter, and it is defined as:

$$h\left(\vec{x}\right) = \eta^T \left(\vec{x} - c\right) \geq 0,$$

where $c \in \mathbb{R}^2$ is the point on the obstacle's edge that is closest to the CoM position $\vec{x}$, while $\eta \in \mathbb{R}^2$ is the normal vector that connects $\vec{x}$ to the boundary. Hence, η is also the vector orthogonal to the line that defines the half-planes, pointing toward the safe region. All of these components are shown in Figure (2).

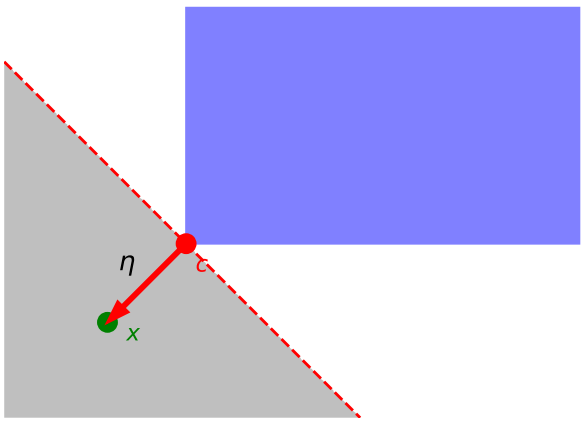

Figure 2: In this plot, the CoM is represented as a green point, and the obstacle is a blue rectangle. The point on the obstacle's boundary that is closest to $\vec{x}$ is the red one, which represents c. The red vector connecting c to $\vec{x}$ is η. The safe region defined by this LDCBF is the gray area. In that half-plane, the humanoid's CoM is free to move without hitting any obstacle.

If multiple obstacles are in the environment, the safe region is the intersection of the regions produced by each LDCBF.
By enforcing that the states $\mathbf{x}_{1\ldots N+1}$ produced by the optimal inputs $\mathbf{u}_{0\ldots N}$ computed by the MPC satisfy $h(\vec{x}) > 0$, we guarantee that, during the motion, the humanoid's CoM does never collide with the obstacles.

Vectors η and c are computed as follows. Indicating with A and B the endpoints of the edge of the obstacle that is closest to $\vec{x}$ we have:

$$\overrightarrow{AX} \coloneqq \vec{x} - A, \qquad \overrightarrow{AB} \coloneqq B - A,$$
$$t \coloneqq \frac{\overrightarrow{AX} \cdot \overrightarrow{AB}}{\left\|\overrightarrow{AB}\right\|^2}, \qquad \tilde{t} = \max\left\{0,\ \min\left\{1,\ t\right\}\right\},$$
$$c \coloneqq A + \tilde{t} * \overrightarrow{AB},$$
$$\eta \coloneqq \vec{x} - c.$$

Due to the presence of min and max, the specified expressions of c and η are non-linear. However, the constraint enforced on the MPC solution at simulation time k is based on the values of c and η computed at the beginning of the time step. Therefore, they are included in the constraint as constants, and the LDCBF becomes a linear combination of $\vec{x}$.

5.3.3 Our variation of the LDCBF

From Figure (2) it is evident that a such defined LDCBF constraint allows the humanoid to get very close to the obstacle's boundary. And, even though the CoM will not collide with it, the footstep positions provided by the MPC will likely overlap with the obstacle. Therefore, we developed a variant of the LDCBF proposed by Peng et al., defined as follows:

$$h\left(\vec{x}\right) = \eta^T\left(\vec{x} - c\right) - \delta \geq 0, \tag{9}$$

where c and η are the previously defined vectors, while δ is the distance that the CoM must keep from the c (i.e., from the obstacle's boundary) to satisfy the constraint. The result is shown in Figure 3
Forcing the control barrier function not to take values below a threshold allows us to take into account the footstep positions, and to generate trajectories that are not only feasible for the CoM, but for the motion of the whole humanoid.

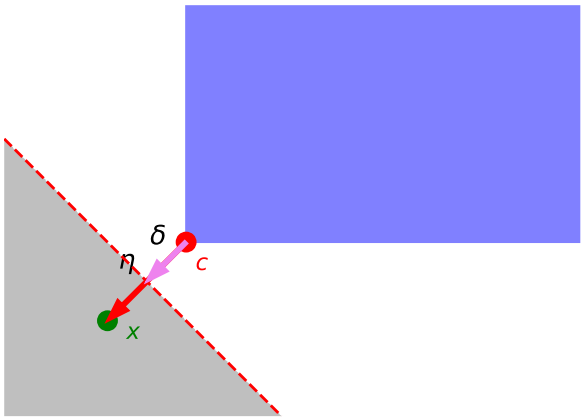

Figure 3: This plot shows the same environment of Figure 2, but here expression (9) with $\delta > 0$ was used as LDCBF. It is clear that the safe zone is not adjacent to the obstacle, and $\vec{x}$ maintains a distance δ from the obstacle.

6 Conclusion

This is the conclusion.

References

[1] Chengyang Peng, Victor Paredes, Guillermo A. Castillo, and Ayonga Hereid1. Real-time safe bipedal robot navigation using linear discrete control barrier functions. *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[2] Jun Zeng, Bike Zhang, and Koushil Sreenath. Safety-critical model predictive control with discrete-time control barrier function, 2021.

[3] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Inverse Differential Kinematics*, chapter 3.5, pages 123–128. Springer, 2009.