



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

Contents

1	Introduction	2
2	3D-LIP Model with Heading Angles	3
2.1	Local Robot Reference Frame	3
2.2	Model Definition	4
3	3D-LIP Model with Heading Angles	5
4	MPC	7
5	LIP-MPC: Gait planning with Model Predictive Control	7
5.1	Heading Angle Preprocessing	8
5.2	Kinematic Constraints	9
5.2.1	Walking Velocities Constraint	9
5.2.2	Leg Reachability Constraint	9
5.2.3	Maneuverability Constraint	10
5.3	Control Barrier Functions	10
5.3.1	Definitions	10
5.3.2	Linear Discrete Control Barrier Functions (LDCBF)	11
5.3.3	Our variation of the LDCBF	12
6	Simulations	14
6.1	Simple Environment	14
7	Our Improvements	17
7.1	Subgoals and RRT*	17
7.2	Unknown Environments	18
8	Conclusion	19
	References	20

1 Introduction

The term *humanoid*, stands for a robot characterized by a structure and kinematics similar to the human body. These type of robots, which are inherently underactuated thanks to the unilateral ground contacts, are designed to navigate and interact in environments structured for humans. One of the most important task for humanoids is Real-time navigation, which represents how these robots should navigate an environment by planning a safe path. With this term, we refer to paths where the dynamics and physical limitation of the robot are respected and that do not involve any collision between the robot and the obstacles. Due to the high complexity of this task, path planning is usually decoupled from gait control, resulting in a significant reduction of the computational load.

The aim of this work is to re-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 that uses Linear Control Barrier Functions (LCBF) to avoid collision with obstacles. In the following chapters, we will delve into the details of this approach, discuss the results, and propose some improvements.

2 3D-LIP Model with Heading Angles

2.1 Local Robot Reference Frame

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

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

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:

$$\begin{aligned}\mathbf{T}_k &= \begin{pmatrix} \cos \theta_{k, \text{glob}} & -\sin \theta_{k, \text{glob}} & p_{x, k-1, \text{glob}} \\ \sin \theta_{k, \text{glob}} & \cos \theta_{k, \text{glob}} & p_{y, k-1, \text{glob}} \\ 0 & 0 & 1 \end{pmatrix}, \\ \begin{pmatrix} f_{x, k, \text{glob}} \\ f_{y, k, \text{glob}} \\ 1 \end{pmatrix} &= \mathbf{T}_k \begin{pmatrix} f_{x, k, \text{loc}} \\ f_{y, k, \text{loc}} \\ 1 \end{pmatrix}, \\ \begin{pmatrix} p_{x, k, \text{glob}} \\ p_{y, k, \text{glob}} \\ 1 \end{pmatrix} &= \mathbf{T}_k \begin{pmatrix} p_{x, k, \text{loc}} \\ p_{y, k, \text{loc}} \\ 1 \end{pmatrix}, \\ \begin{pmatrix} v_{x, k, \text{glob}} \\ v_{y, k, \text{glob}} \end{pmatrix} &= \begin{pmatrix} \cos \theta_{k, \text{glob}} & -\sin \theta_{k, \text{glob}} \\ \sin \theta_{k, \text{glob}} & \cos \theta_{k, \text{glob}} \end{pmatrix} \begin{pmatrix} v_{x, k, \text{loc}} \\ v_{y, k, \text{loc}} \end{pmatrix}, \\ \theta_{k+1, \text{glob}} &= \theta_{k, \text{glob}} + \theta_{k+1, \text{loc}}, \quad \omega_{\text{glob}} = \omega_{\text{loc}}.\end{aligned}$$

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.

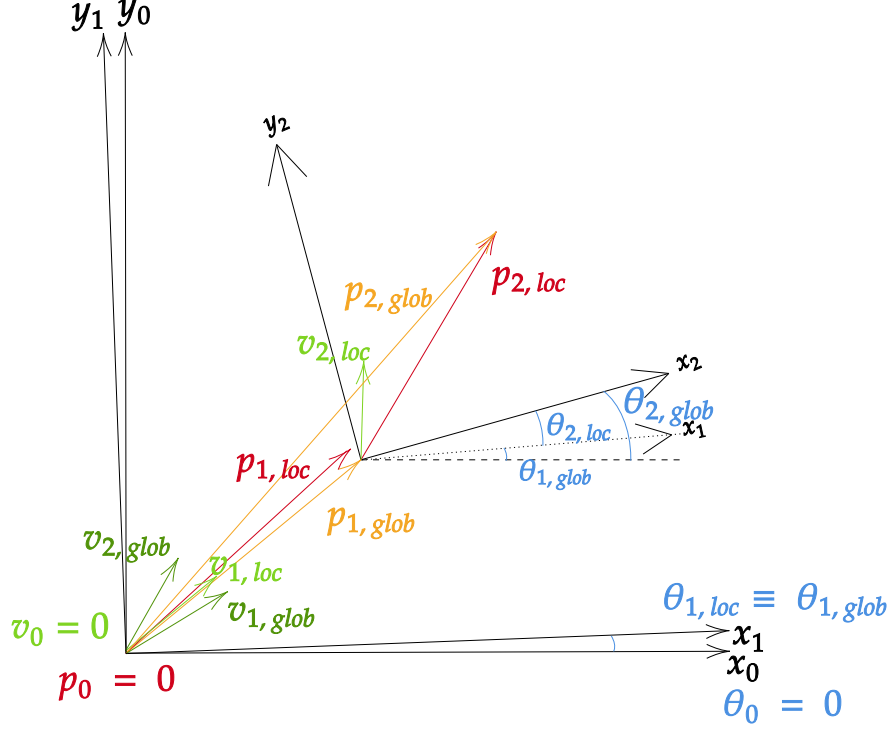


Figure 1: An example of the evolution of the 3D-LIP model's state, highlighting the relationship between local and global coordinates. The initial state is $\mathbf{0}$ and the RF (x_0, y_0) is the inertial frame. The local RF translates and rotates with the simulation time k : the position of $RF_2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

2.2 Model Definition

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

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

with g as the magnitude of the gravitational acceleration. Since by definition $\dot{\theta} := \omega$, 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:

$$\mathbf{x}_{k+1} = \mathbf{A}_L \mathbf{x}_k + \mathbf{B}_L \mathbf{u}_k, \quad (1)$$

with:

$$\mathbf{A}_L := \begin{pmatrix} \mathbf{A}_d & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{A}_d & 0 \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix}, \quad \mathbf{B}_L := \begin{pmatrix} \mathbf{B}_d & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{B}_d & 0 \\ 0 & 0 & T \end{pmatrix},$$

$$\mathbf{A}_d := \begin{pmatrix} \cosh \beta T & \frac{\sinh \beta T}{\beta} \\ \beta \sinh \beta T & \cosh \beta T \end{pmatrix}, \quad \mathbf{B}_d := \begin{pmatrix} 1 - \cosh \beta T \\ -\beta \sinh \beta T \end{pmatrix}, \quad \beta := \sqrt{\frac{g}{H}}$$

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.

3 3D-LIP Model with Heading Angles

Due to the high dimensionality and non-linearity of the full dynamic model of a humanoid, we have used a simplified model base on the standard Linear Inverted Pendulum formulation. This reduced model assumes that during the motion, the Center of Mass (CoM) will have a constant height during the motion. According to (citation), we can express the acceleration of the CoM position as follows:

$$\begin{cases} \dot{v}_x = \frac{g}{H}(p_x - f_x) \\ \dot{v}_y = \frac{g}{H}(p_y - f_y) \end{cases} \quad (2)$$

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 feet are in contact with the ground, is denoted with (f_x, f_y) .

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 dynamics of the model can be written as follows:

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (3)$$

This equation can be expanded as:

$$\dot{x} = \begin{bmatrix} \dot{p}_x \\ \dot{v}_x \\ \dot{p}_y \\ \dot{v}_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{g}{H} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{g}{H} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ v_x \\ p_y \\ v_y \\ \theta \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ -\frac{g}{H} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -\frac{g}{H} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ \omega \end{bmatrix} = \quad (4)$$

$$= \begin{bmatrix} A_c & 0 & 0 \\ 0 & A_c & 0 \\ 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} B_c & 0 & 0 \\ 0 & B_c & 0 \\ 0 & 0 & 0 \end{bmatrix} u(t) \quad (5)$$

where $A_c \in \mathbb{R}^{2 \times 2}$, $B_c \in \mathbb{R}^{2 \times 1}$ and $\beta = \sqrt{\frac{g}{H}}$.

From this continuous model, we can obtain the following discretized system:

$$x_{k+1} = A_L x_k + B_L u_k \quad (6)$$

where the two matrices are defined as:

$$A_L = \begin{bmatrix} A_d & 0 & 0 \\ 0 & A_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad B_L = \begin{bmatrix} B_d & 0 & 0 \\ 0 & B_d & 0 \\ 0 & 0 & T \end{bmatrix} \quad (7)$$

with:

$$A_d = \begin{bmatrix} \cosh(\beta T) & \frac{\sinh(\beta T)}{\beta} \\ \beta \sinh(\beta T) & \cosh(\beta T) \end{bmatrix} \quad B_d = \begin{bmatrix} 1 - \cosh(\beta T) \\ -\beta \sinh(\beta T) \end{bmatrix} \quad (8)$$

improve the discretization part

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:

$$J^* = \min_{u_{0:N-1}} \sum_{k=1}^N [(p_{x_k} - g_x)^2 + (p_{y_k} - g_y)^2]$$

$$\text{s. t } x_{k+1} = A_L x_k + B_L u_k \quad c_l \leq c(x_k, u_k) \leq c_u$$

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 (1) 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:

$$\begin{aligned}
J^* &= \min_{\mathbf{u}_{0:N-1}} \sum_{k=1}^N q(\mathbf{x}_k) \\
\text{s.t. } \quad &\mathbf{x}_k \in X, \quad k \in [1, N] \\
&\mathbf{u}_k \in U, \quad k \in [0, N-1] \\
&\mathbf{x}_{k+1} = \mathbf{A}_L \mathbf{x}_k + \mathbf{B}_L \mathbf{u}_k, \quad k \in [0, N-1] \\
&\mathbf{c}_l \leq \mathbf{c}(\mathbf{x}_k, \mathbf{u}_k) \leq \mathbf{c}_u, \quad k \in [0, N-1],
\end{aligned} \tag{9}$$

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:

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

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:

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

change theta precomputation and explain the new one

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.

Better explain all the constraints

5.2.1 Walking Velocities Constraint

Change this constraint to adhere to Michele&Andrew's suggestions.

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

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 $v_{y_{min}}$.

decide whether this text should be removed

5.2.2 Leg Reachability Constraint

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

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

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

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 $h : \mathbb{R}^n \rightarrow \mathbb{R}$, we can define the *safety set* S and its boundary ∂S as:

$$\begin{aligned} S &:= \{\mathbf{x} \in X \mid h(\mathbf{x}) \geq 0\}, \\ \partial S &:= \{\mathbf{x} \in X \mid h(\mathbf{x}) = 0\}. \end{aligned}$$

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:

$$\begin{aligned} \Delta h(\mathbf{x}_k, \mathbf{u}_k) &:= h(\mathbf{x}_{k+1}) - h(\mathbf{x}_k), \\ \forall \mathbf{x}_k \in S \quad \exists \mathbf{u}_k \text{ s.t. } \Delta h(\mathbf{x}_k, \mathbf{u}_k) &\geq -\gamma(h(\mathbf{x}_k)). \end{aligned} \tag{10}$$

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 (10) becomes:

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

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

Not
sure
about
it

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, $h(\cdot): \mathbb{R}^2 \rightarrow \mathbb{R}$. 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:

$$h(\vec{x}) = \eta^T (\vec{x} - c) \geq 0, \quad (11)$$

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).

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 $\mathbf{x}_{1...N+1}$ produced by the optimal inputs $\mathbf{u}_{0...N}$ 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:

$$\begin{aligned} \overrightarrow{AX} &:= \vec{x} - A, & \overrightarrow{AB} &:= B - A, \\ t &:= \frac{\overrightarrow{AX} \cdot \overrightarrow{AB}}{\|\overrightarrow{AB}\|^2}, & \tilde{t} &= \max\{0, \min\{1, t\}\}, \\ c &:= A + \tilde{t} * \overrightarrow{AB}, \\ \eta &:= (-1)^\gamma * (\vec{x} - c), \end{aligned}$$

where $\gamma = 1$ if \vec{x} is inside the obstacle, otherwise it is 0. 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} .

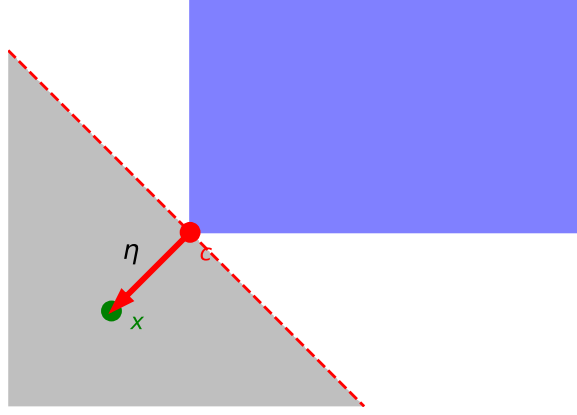


Figure 2: In this plot, the CoM is represented as a green point, and the obstacle is a blue rectangle. The point on the obstacle’s boundary that is closest to \vec{x} is the red one, which represents c . The red vector connecting c to \vec{x} is η . The safe region defined by this LDCBF is the gray area.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

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:

$$h(\vec{x}) = \eta^T (\vec{x} - c) - \delta \geq 0, \quad (12)$$

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.

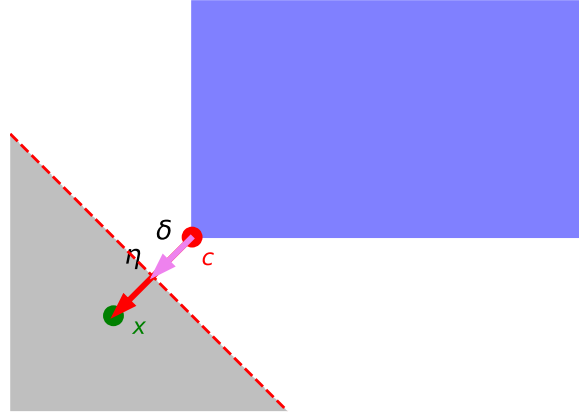


Figure 3: This plot shows the same environment of Figure 2, but here expression (12) with $\delta > 0$ was used as LDCBF. It is clear that the safe zone is not adjacent to the obstacle, and \vec{x} maintains a distance δ from the obstacle.

6 Simulations

In this chapter, we will test the previously described MPC in a virtual environment. Throughout the simulations, the whole complexity of the humanoid will be condensed in a single point, representing its CoM.

The objective of this set of tests is to prove that, using the inputs provided by the MPC proposed by Peng et al. in [1] and developed by us, a robot is able to reach a user-defined goal, while avoiding collisions with all the obstacles in the environment. The hyper-parameters used in the following simulations are summarized in Table 1.

Simulation Parameters	
Parameter	Value
α	1.44
CoM height	1 m
Step duration (T)	0.4 s
l_{\max}	$0.1\sqrt{3}$ m
$(v_{x_{\max}}, v_{y_{\max}})$	$(-0.1, 0.1)$ m s ⁻¹
$(v_{y_{\min}}, v_{y_{\min}})$	$(0.8, 0.4)$

Table 1: The MPC and LIP parameters used for all the simulations in this chapter.

6.1 Simple Environment

The first simulation takes place in a basic environment: the humanoid is positioned at $(0, 0)$ with orientation $\theta = 0$, and the goal position is $(5, 5)$. In between, we placed 3 obstacles with a quasi-circular shape. The choice of approximating circles with polygons was made to comply with the CBF computation described in 5.3.2. The results are shown in Figures 4 and 5.

By inspecting the plots, we see that, at the beginning of the simulation, a high turning velocity ω is commanded in order to make the robot point toward the goal. In the meantime, the MPC provides new footstep positions but, due to the maneuverability constraint, the high turning rate limits the velocity along the sagittal axis. Therefore, the robot is encouraged to move laterally, and the first inputs generate a large v_y that induces a *side-walking* motion.

Then, there is an overshoot in the plots of θ and ω . This behavior derives from the large sampling time of the robot. To lighten the computational load, in our simulations, we set equal sampling times for the MPC and the humanoid. This means that every T seconds a new footstep position and angular velocity are computed, and they are provided as input to the robot until new inputs are available, even though it could have a lower sampling time and, then, several different inputs could be commanded during T . Therefore, the ω needed to point to the goal is provided as input and executed for T

seconds. But, during that interval, the robot misses the right orientation and continues rotating. Hence, a new turning velocity is commanded in the opposite direction, and the sagittal axis is successfully aligned with the goal.

At this point, the robot moves forward to the goal, the orientation is kept constant, and the positional error decreases smoothly. Nevertheless, there is some chattering in the v_y plot. This is a regular behavior: at the end of each step, the humanoid changes the stance foot by *falling* on the opposite side, and it results in the lateral velocity changing sign.

Around the second 35 of the simulation (4th and 5th frames of Figure 5), the humanoid approaches one of the obstacles. It is unable to go forward, as this would lead the CoM into the circle. To keep the robot in the safe area defined by the LDCBF, the new MPC solution makes the humanoid move laterally. Thus, it can successfully avoid the obstacle. In this case, *side walking* occurs because the precomputed value of ω only aims at aligning the sagittal axis of the robot with the goal, and, once it is achieved, the turning rate is kept constant. Consequently, the humanoid can only escape the obstacle while facing the goal. This movement explains the minor variations in the position error and translational velocity plots around the second 35.

The humanoid can, then, realign itself (producing a minimal oscillation in the θ and ω plots) and continue navigating to the goal, which it ultimately reaches successfully.

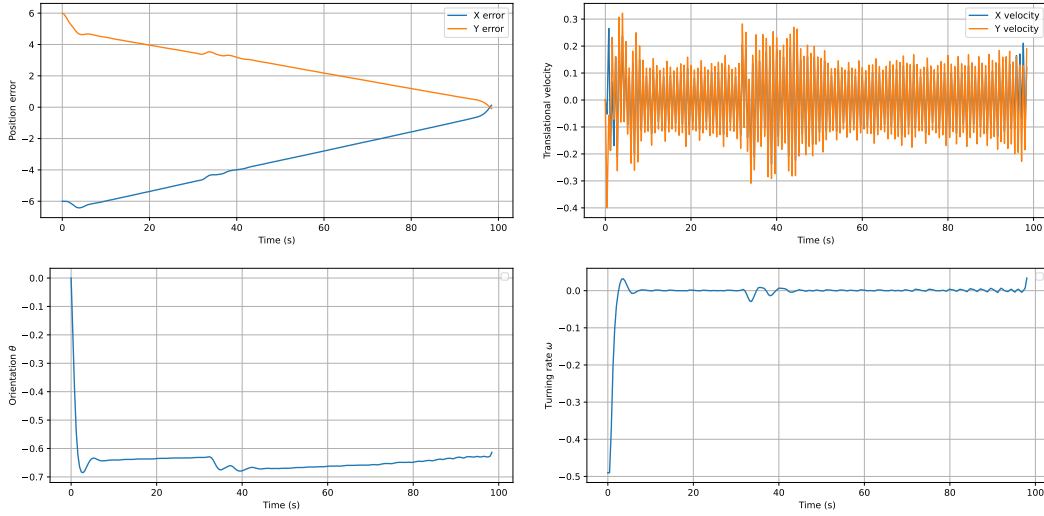


Figure 4: The figure depicts the evolution of the humanoid’s state and the input throughout the simulation. The top-left plot shows the error between the CoM and the goal position, while the top-right represents the translational velocities along the x- and y-axis. The bottom left and right plots show the theta and omega evolution, respectively. All the quantities are expressed in the global RF.

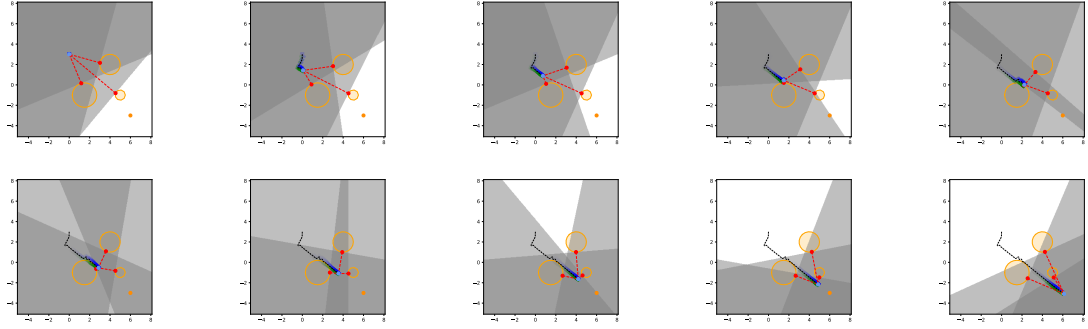


Figure 5: This sequence of frames illustrates the robot's trajectory to the goal. Obstacles are represented with orange shapes. The humanoid is represented by an isosceles triangle whose sagittal axis is aligned with the robot's. Each frame shows the vectors η and c along with the safe area associated with each obstacle. The darkest region represents the intersection of these semi-planes, which is the overall safe area.

7 Our Improvements

In this chapter, we introduce some modifications or additions to the solution proposed by Peng et al. in [1], aiming to improve the robot's behavior in more general and complex environments.

7.1 Subgoals and RRT*

TO IMPROVE

As described in Section 5, the solution provided by the MPC is the input that minimizes the cost function while satisfying all the constraints. When an obstacle lies between the start and the goal positions (as illustrated in Figure), the robot must get around it in order to approach the goal. However, it would require the robot to deliberately increase its distance from the goal before reducing it. It implies that the MPC should return a sub-optimal solution, though one that minimizes the cost function could exist. Consequently, the previous approach cannot guarantee goal attainment in such cases.

To address this, we compute a path from the start to the goal position using RRT*. Then, we request the humanoid to reach sequentially all the *subgoals*, namely the nodes of the tree in the path from the start to the goal. In this way, we ensure that the MPC produces feasible control inputs while adhering to the original cost-minimization framework, and the humanoid can successfully reach the goal.

The RRT* algorithm is used to compute fast a collision-free path, while minimizing the cost of the path, i.e. the sum of the edges' cost connecting the start to the goal node. Whenever a new node j is added to the tree, the cost of the each $e_{i,j}$ connecting it to node i is defined as:

$$cost(e_{i,j}) := cost(path_{i,start}) + dist(i,j) + e^{-clearance(j)}.$$

It is the sum of the cost of the path from j 's parent to the initial node, the Euclidean distance from the position represented by i to the one represented by j , and the exponential of the negative clearance (namely, the distance of j from the closest obstacle) of the new node.

FIGURE CAPTION: In this Figure, in order to reach the goal, the robot must first overcome the obstacle by reaching the red star. Indeed, the RRT* solution contains 3 nodes: the start, the star, and the goal positions. The first goal provided to the robot is the star, which represents a subgoal. Once it has been reached with the standard framework, the original goal is provided to the MPC. With this approach it is possible for the humanoid to travel along a path that would be unfeasible by just using the MPC.

Add
Figure
num

7.2 Unknown Environments

In unknown environments the robot does not rely on a pre-loaded map but rather builds its understanding on the fly. To achieve this, a 2D LiDAR range finder is employed to capture real-time measurements of the surroundings. These LiDAR readings are getting perturbed with zero-mean Gaussian noise to better simulate the uncertainty found in real-world sensor data. The noisy measurements are then processed using the DBSCAN clustering algorithm (with an epsilon of 0.3) to group nearby points into clusters that represent inferred obstacles.

This leads to a further tuning of the system, in order to achieve desired performances in real world scenarios; in fact provided that every real obstacle is convex, and assuming the obstacles are sufficiently distant, the LiDAR resolution is not too high (here 360°), and the DBSCAN epsilon remains small enough (here 0.3); these inferred obstacles will also be convex. By considering only these clustered obstacles—rather than every single detected point—the navigation system mirrors the behavior of the base simulation while accounting for realistic environmental complexity. In this way, the humanoid robot is able to navigate safely despite not knowing the environment a priori, relying solely on the dynamically inferred obstacles from its sensor data.

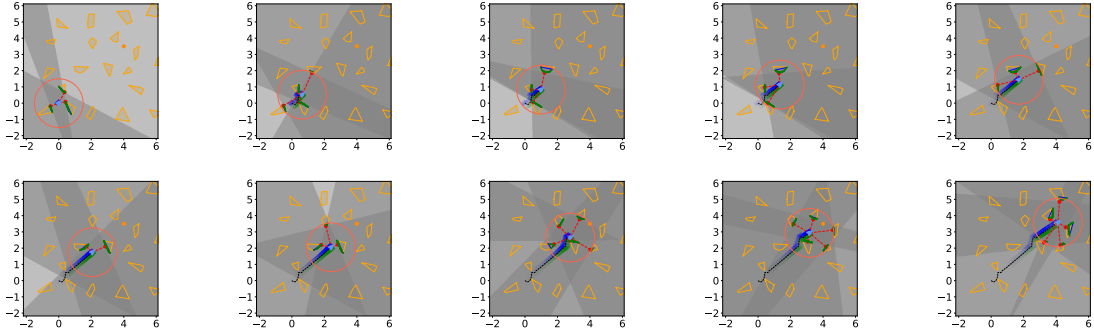


Figure 6: This is a description.

8 Conclusion

In this project, we successfully reproduced the methodology presented in Peng et al. [1], implementing a real-time safe navigation framework for bipedal robots using Linear Discrete Control Barrier Functions (LDCBFs). Our implementation validated the feasibility of using a Linear Inverted Pendulum (LIP) model combined with Model Predictive Control (MPC) to achieve stable and dynamically feasible locomotion in cluttered environments.

Minding about social robotics, beyond reproducing the original work, we introduced an additional *safety margin* in the computation of LDCBF constraints. This modification aimed to enhance obstacle avoidance robustness, ensuring a more conservative approach to collision prevention while maintaining real-time performance. Our simulations demonstrated that this adjustment effectively increased the reliability of navigation without introducing significant computational overhead; anyway this novelty results in additional system tweaking, so to obtain the best safety margin over the environment.

Further refinement of the safety margin parameter could provide an optimal trade-off between conservatism and maneuverability; an environment-based dynamic safety margin will be the best suited for that task. For example, someone can tweak it based on the number of obstacles inside the FOV, or based on the width of the available walking space.

Moreover, for safe gait planning with limited FOV, we leveraged LiDAR scans of the surrounding scene and applied DBSCAN algorithm so to obtain a convex obstacle dynamically computed. In this way, due to the flexibility of our system, this can be deployed in dynamic environments with walking humans, enhancing its usage in social robotics environments.

Despite these improvements, certain limitations remain. The precomputed turning rates, while simplifying real-time computation, may reduce flexibility in highly constrained environments. Future work could explore adaptive strategies for obstacle avoidance, integrating learning-based methods to enhance decision-making in dynamic environments.

Overall, this project provided valuable insights into safe gait planning for bipedal robots and reinforced the effectiveness of LDCBF-based navigation. Our contributions offer a promising direction for improving safety-critical real-time locomotion, with potential applications in both simulation and real-world robotic deployments.

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.