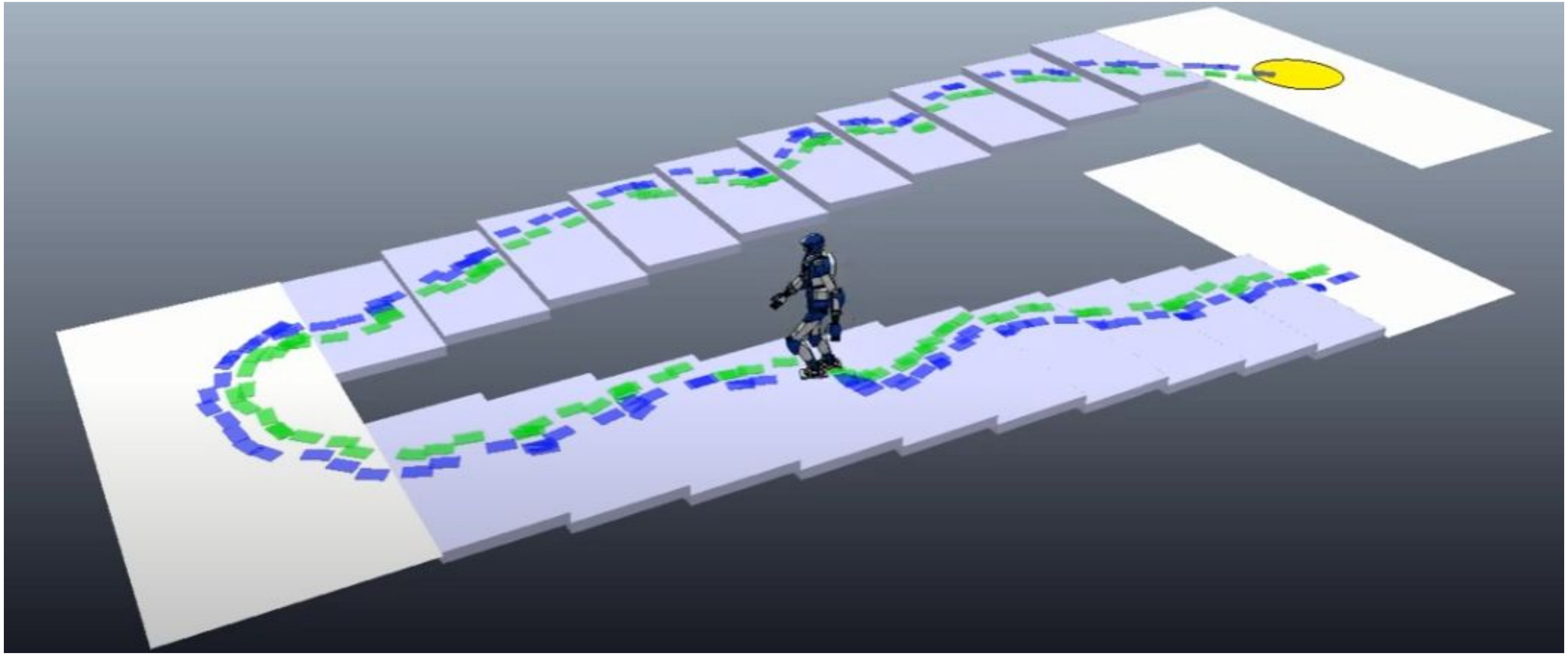
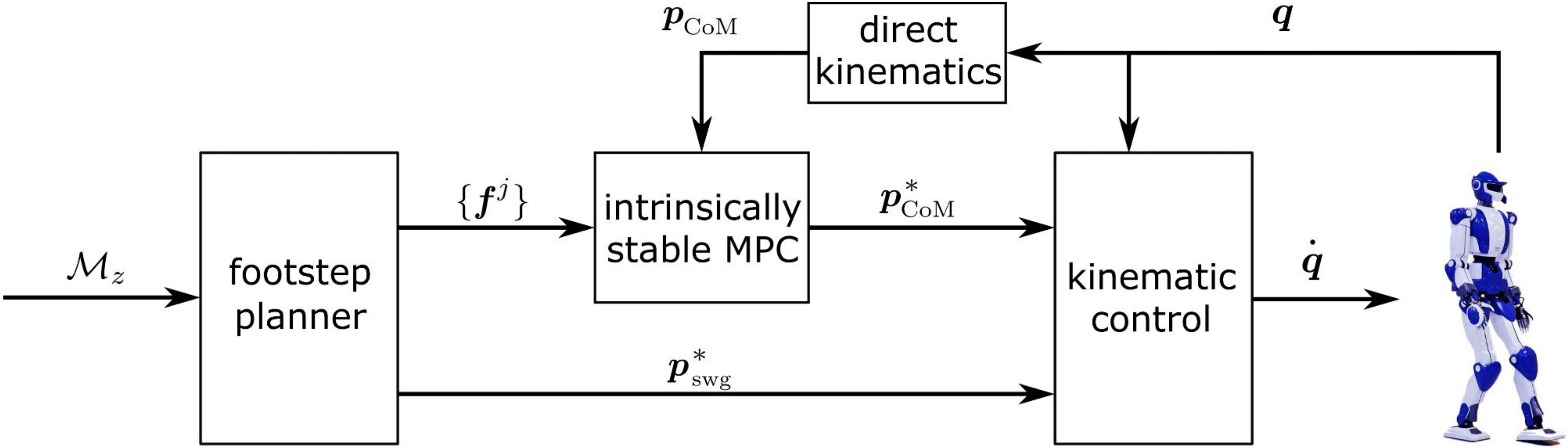
Autonomous humanoid navigation in multi-floor environments

Colonna 1697124 - Manganaro 1817504 - Zaramella 2025806
Autonomous and Mobile Robotics
Sapienza University of Rome

The Main Task



General Approach



Footstep Feasibility

R1

$\boldsymbol{f}^{j}$ is fully in contact within the same patch.

R2
stance feasibility

$\boldsymbol{f}^{j}$ is kinematically admissible form the previous foot-step $\boldsymbol{f}^{j-1}$.

R3
step feasibility

$\boldsymbol{f}^{j}$ s reachable from $\boldsymbol{f}^{j-2}$ through a collision-free trajectory $\boldsymbol{p}_{swg}{}^{j}$.

$\boldsymbol{f}^{j} = (x_f^j , y_f^j , z_f^j , \theta_f^j)$

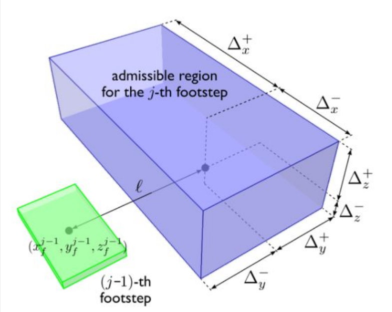
$\boldsymbol{p}_{swg}{}^{j} = (h^{j})$

Footstep Feasibility

R1

$\boldsymbol{f}^{j}$ is fully in contact within the same patch.

R2
stance feasibility



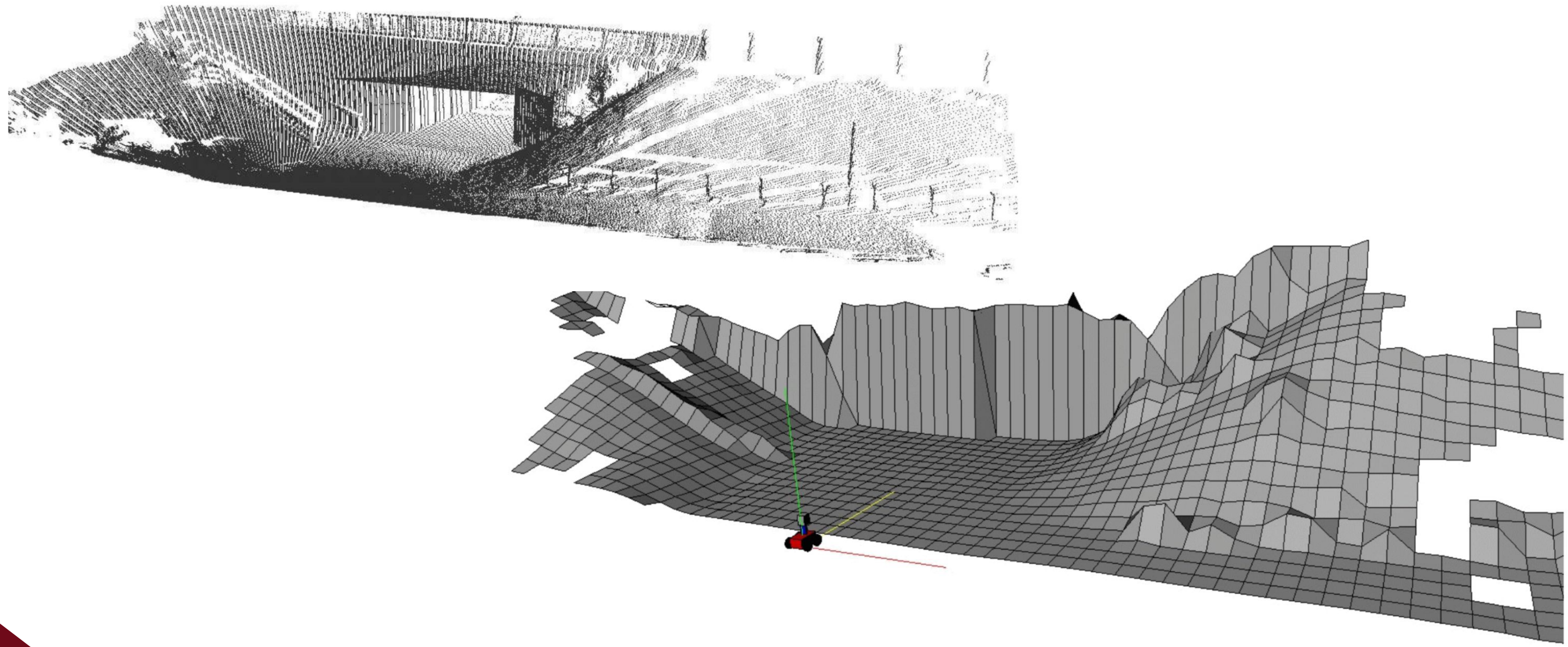
R3
step feasibility

$\boldsymbol{f}^{j}$ s reachable from $\boldsymbol{f}^{j-2}$ through a collision-free trajectory $\boldsymbol{p}_{swg}{}^{j}$.

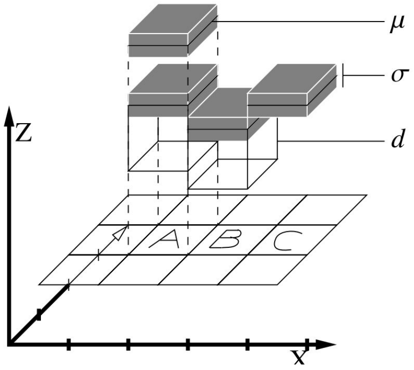
$$\boldsymbol{f}^{j} = (x_f^j, y_f^j, z_f^j, \theta_f^j)$$

$$\boldsymbol{p}_{swg}{}^{j} = (h^j)$$

Limitations of the Elevation Map



Multi-Level Surface Map

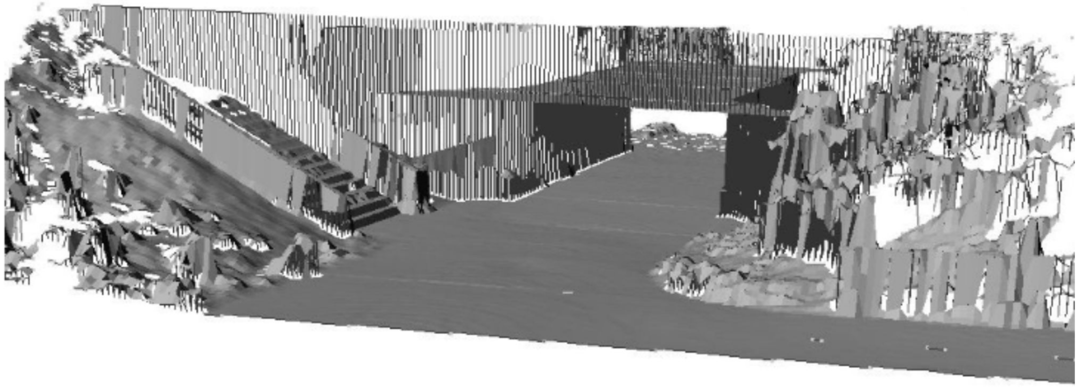


Extensions to Elevation Map:

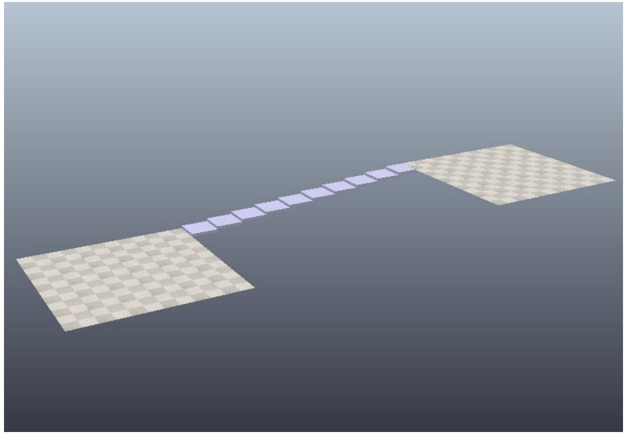
- multiple surfaces
- depth representation

Cell variables:

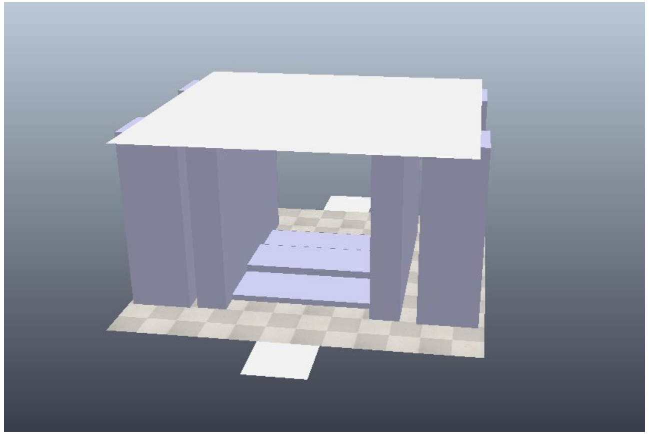
- μ : height mean
- σ : height variance
- d : depth value



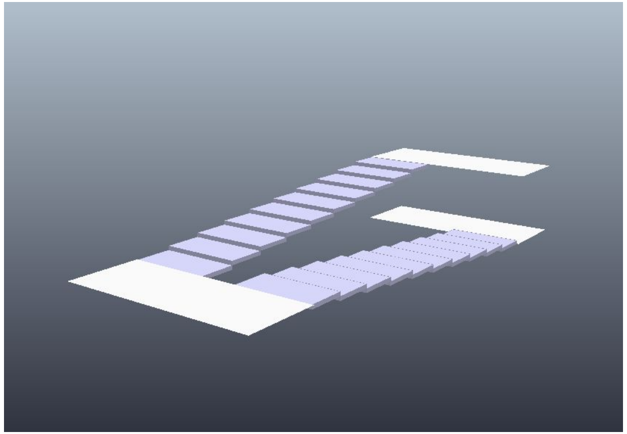
Environments



Stairway



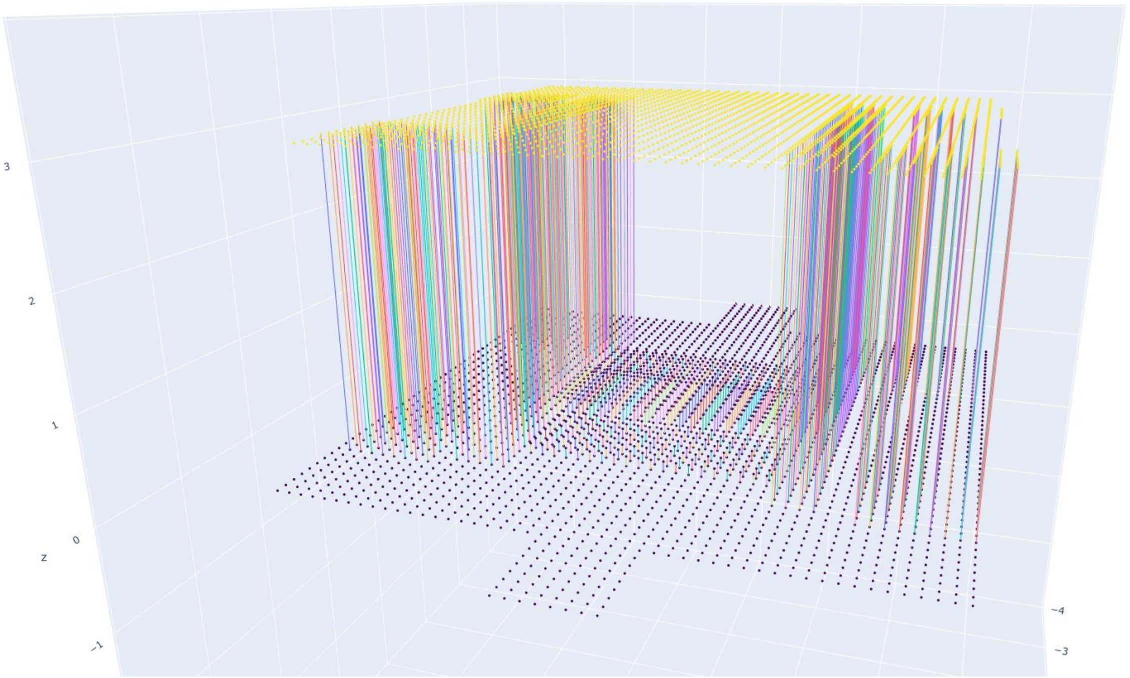
Tunnel



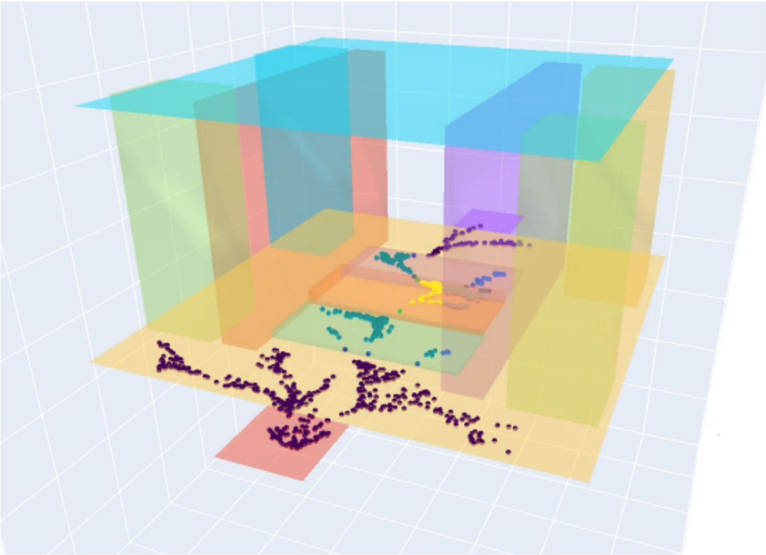
Floors

MLS Map Visualization

Tunnel environment



Footstep planning



To find a path to go from a starting point ‘s’ to a goal point ‘g’ we have to explore the space.

To do so, we use the RRT algorithm. It is a random algorithm that allows to explore the space in a randomic way.

Our algorithm’s structure

We need as input:

INPUT →

- start stance ‘f’
- goal region $\mathcal{G}$
- multi level surface map
- max number of iterations

The tree $\mathcal{T}$ that we create, has for each node:

NODE →

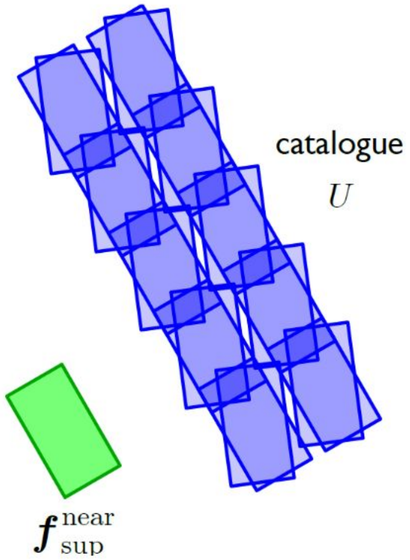
- stance ‘f’
- node parent
- list of children
- id (of swg foot)
- trajectory

Pseudocode

Algorithm 1: Footstep Planner

root the tree $\mathcal{T}$ at $v_{\text{ini}} \leftarrow (\boldsymbol{f}_L, \boldsymbol{f}_R)$;
$i \leftarrow 0$;
repeat
    $i \leftarrow i + 1$;
    generate a random point $\boldsymbol{p}_{\text{rand}}$ on the ground;
    select the closest vertex v_{near} in $\mathcal{T}$ to $\boldsymbol{p}_{\text{rand}}$ according to $\gamma(\cdot, \boldsymbol{p}_{\text{rand}})$;
    randomly select from the primitive catalogue U a candidate footstep $\boldsymbol{f}^{\text{cand}}$;
    if $\boldsymbol{f}^{\text{cand}}$ is feasible w.r.t. R1–R2 **then**
        $h \leftarrow h_{\text{min}}$;
        $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
        while $h \leq h_{\text{max}}$ **and** $\text{Collision}(\boldsymbol{p}^{\text{cand}}_{\text{swg}})$ **do**
            $h \leftarrow h + \Delta h$;
            $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
        end
        if $h \leq h_{\text{max}}$ **then**
            $v_{\text{new}} \leftarrow (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}^{\text{near}}_{\text{sup}})$;
            add vertex v_{new} to $\mathcal{T}$ as a child of v_{near};
            compute midpoint $\boldsymbol{m}$ between the feet at v_{new};
        end
    end
until $\boldsymbol{m} \in \mathcal{G}$ **or** $i = i_{max}$;

Primitives catalogue



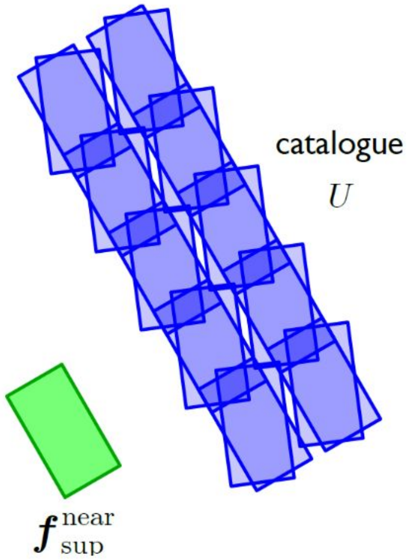
Pseudocode

Algorithm 1: Footstep Planner

root the tree $\mathcal{T}$ at $v_{\text{ini}} \leftarrow (\boldsymbol{f}_L, \boldsymbol{f}_R)$;
$i \leftarrow 0$;
repeat
    $i \leftarrow i + 1$;
    generate a random point $\boldsymbol{p}_{\text{rand}}$ on the ground;
    select the closest vertex v_{near} in $\mathcal{T}$ to $\boldsymbol{p}_{\text{rand}}$ according to $\gamma(\cdot, \boldsymbol{p}_{\text{rand}})$;
    randomly select from the primitive catalogue U a candidate footstep $\boldsymbol{f}^{\text{cand}}$;
    if $\boldsymbol{f}^{\text{cand}}$ is feasible w.r.t. R1–R2 **then**
        $h \leftarrow h_{\min}$;
        $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
        while $h \leq h_{\max}$ **and** Collision($\boldsymbol{p}^{\text{cand}}_{\text{swg}}$) **do**
            $h \leftarrow h + \Delta h$;
            $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
        end
        if $h \leq h_{\max}$ **then**
            $v_{\text{new}} \leftarrow (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}^{\text{near}}_{\text{sup}})$;
            add vertex v_{new} to $\mathcal{T}$ as a child of v_{near};
            compute midpoint $\boldsymbol{m}$ between the feet at v_{new};
        end
    end
until $\boldsymbol{m} \in \mathcal{G}$ **or** $i = i_{max}$;

Prand generation

Primitives catalogue



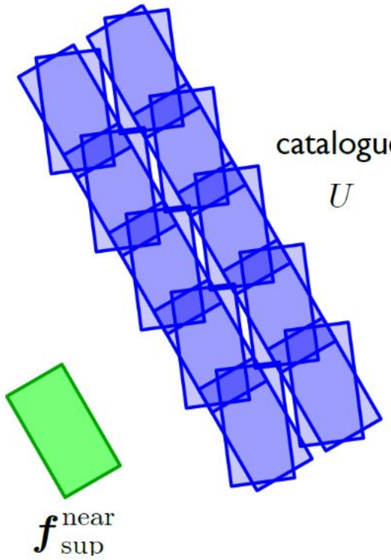
Pseudocode

Algorithm 1: Footstep Planner

root the tree $\mathcal{T}$ at $v_{\text{ini}} \leftarrow (\boldsymbol{f}_L, \boldsymbol{f}_R)$;
$i \leftarrow 0$;
repeat
  $i \leftarrow i + 1$;
  generate a random point $\boldsymbol{p}_{\text{rand}}$ on the ground;
  $\boxed{\text{select the closest vertex } v_{\text{near}} \text{ in } \mathcal{T} \text{ to } \boldsymbol{p}_{\text{rand}} \text{ according to } \gamma(\cdot, \boldsymbol{p}_{\text{rand}});}$
  randomly select from the primitive catalogue U a candidate footstep $\boldsymbol{f}^{\text{cand}}$;
  if $\boldsymbol{f}^{\text{cand}}$ is feasible w.r.t. R1–R2 **then**
    $h \leftarrow h_{\text{min}}$;
    $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
    while $h \leq h_{\text{max}}$ **and** Collision$(\boldsymbol{p}^{\text{cand}}_{\text{swg}})$ **do**
      $h \leftarrow h + \Delta h$;
      $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
    end
    if $h \leq h_{\text{max}}$ **then**
      $v_{\text{new}} \leftarrow (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}^{\text{near}}_{\text{sup}})$;
      add vertex v_{new} to $\mathcal{T}$ as a child of v_{near};
      compute midpoint $\boldsymbol{m}$ between the feet at v_{new};
    end
  end
until $\boldsymbol{m} \in \mathcal{G}$ **or** $i = i_{max}$;

Vnear selection

Primitives catalogue



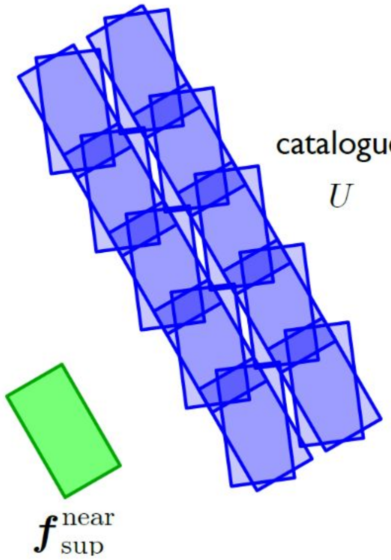
Pseudocode

Algorithm 1: Footstep Planner

root the tree $\mathcal{T}$ at $v_{\text{ini}} \leftarrow (\boldsymbol{f}_L, \boldsymbol{f}_R)$;
$i \leftarrow 0$;
repeat
  $i \leftarrow i + 1$;
  generate a random point $\boldsymbol{p}_{\text{rand}}$ on the ground;
  select the closest vertex v_{near} in $\mathcal{T}$ to $\boldsymbol{p}_{\text{rand}}$ according to $\gamma(\cdot, \boldsymbol{p}_{\text{rand}})$;
  randomly select from the primitive catalogue U a candidate footstep $\boldsymbol{f}^{\text{cand}}$;
  if $\boldsymbol{f}^{\text{cand}}$ is feasible w.r.t. R1–R2 **then**
    $h \leftarrow h_{\min}$;
    $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
    while $h \le h_{\max}$ **and** Collision($\boldsymbol{p}^{\text{cand}}_{\text{swg}}$) **do**
      $h \leftarrow h + \Delta h$;
      $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
    end
    if $h \le h_{\max}$ **then**
      $v_{\text{new}} \leftarrow (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}^{\text{near}}_{\text{sup}})$;
      add vertex v_{new} to $\mathcal{T}$ as a child of v_{near};
      compute midpoint $\boldsymbol{m}$ between the feet at v_{new};
    end
  end
until $\boldsymbol{m} \in \mathcal{G}$ **or** $i = i_{max}$;

Calculate Vcand

Primitives catalogue



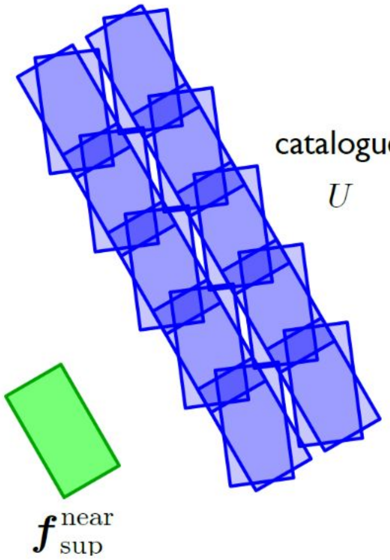
Pseudocode

Algorithm 1: Footstep Planner

root the tree $\mathcal{T}$ at $v_{\text{ini}} \leftarrow (\boldsymbol{f}_L, \boldsymbol{f}_R)$;
$i \leftarrow 0$;
repeat
$i \leftarrow i + 1$;
generate a random point $\boldsymbol{p}_{\text{rand}}$ on the ground;
select the closest vertex v_{near} in $\mathcal{T}$ to $\boldsymbol{p}_{\text{rand}}$ according to $\gamma(\cdot, \boldsymbol{p}_{\text{rand}})$;
randomly select from the primitive catalogue U a candidate footstep $\boldsymbol{f}^{\text{cand}}$;
if $\boldsymbol{f}^{\text{cand}}$ is feasible w.r.t. R1–R2 **then**
$h \leftarrow h_{\min}$;
$\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
while $h \leq h_{\max}$ **and** $\text{Collision}(\boldsymbol{p}^{\text{cand}}_{\text{swg}})$ **do**
$h \leftarrow h + \Delta h$;
$\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
end
if $h \leq h_{\max}$ **then**
$v_{\text{new}} \leftarrow (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}^{\text{near}}_{\text{sup}})$;
add vertex v_{new} to $\mathcal{T}$ as a child of v_{near};
compute midpoint $\boldsymbol{m}$ between the feet at v_{new};
end
end
until $\boldsymbol{m} \in \mathcal{G}$ **or** $i = i_{max}$;

Constraints check

Primitives catalogue



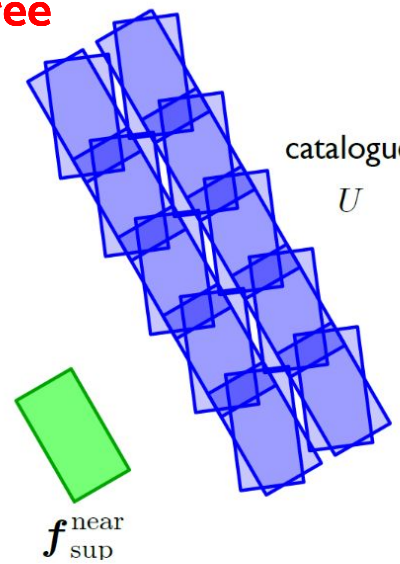
Pseudocode

Algorithm 1: Footstep Planner

root the tree $\mathcal{T}$ at $v_{\text{ini}} \leftarrow (\boldsymbol{f}_L, \boldsymbol{f}_R)$;
$i \leftarrow 0$;
repeat
    $i \leftarrow i + 1$;
    generate a random point $\boldsymbol{p}_{\text{rand}}$ on the ground;
    select the closest vertex v_{near} in $\mathcal{T}$ to $\boldsymbol{p}_{\text{rand}}$ according to $\gamma(\cdot, \boldsymbol{p}_{\text{rand}})$;
    randomly select from the primitive catalogue U a candidate footstep $\boldsymbol{f}^{\text{cand}}$;
    if $\boldsymbol{f}^{\text{cand}}$ is feasible w.r.t. R1–R2 **then**
        $h \leftarrow h_{\min}$;
        $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
        while $h \leq h_{\max}$ **and** $\text{Collision}(\boldsymbol{p}^{\text{cand}}_{\text{swg}})$ **do**
            $h \leftarrow h + \Delta h$;
            $\boldsymbol{p}^{\text{cand}}_{\text{swg}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}, h)$;
        end
        if $h \leq h_{\max}$ **then**
            $v_{\text{new}} \leftarrow (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}^{\text{near}}_{\text{sup}})$;
            add vertex v_{new} to $\mathcal{T}$ as a child of v_{near};
            compute midpoint $\boldsymbol{m}$ between the feet at v_{new};
        end
    end
until $\boldsymbol{m} \in \mathcal{G}$ **or** $i = i_{max}$;

Add Vcand to the tree

Primitives catalogue



Nearest neighbour and Candidate footstep

1) Generate Prand randomically (considering x, y and z of the map).

2) Find the nearest node of the three to Prand

$$\gamma(v, \boldsymbol{p}_{rand}) = ||\boldsymbol{m}(v) - \boldsymbol{p}_{rand}|| + k_\mu |\psi(v, \boldsymbol{p}_{rand})|$$

m(v) = midpoint of feet
Km = positive scalar
$\psi(v, \boldsymbol{p}_{rand})$ = angle between robot sagittal axis and the line join m to Prand

3) Select a random primitive, check R2 and create a new node Vcand

Vcand has: $\boldsymbol{f}_{\text{swg}}^{cand} = \boldsymbol{f}_{\text{sup}}^{near}$ and $\boldsymbol{f}_{\text{sup}}^{cand}$ = primitive selected

Vcand z = nearest height to Vnear allowed

Constraints check and goal verification

4) Check if Vcand respects all the constraints

Check R1, R3 and R2 (the second point) $\xrightarrow{\text{if respects everything}}$ add Vcand to the three as child of Vnear

5) Verify if Vcand (just added to the tree) reach the Goal region

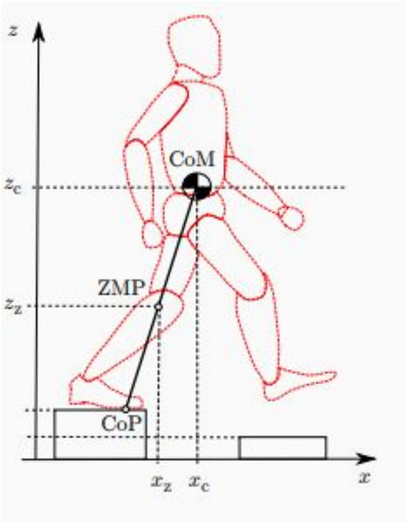
Yes → Finish! Path found

No → Do another iteration

Gait generation via MPC

Based on “Humanoid gait generation on uneven ground using intrinsically stable MPC” paper

Once a footstep plan has been computed, it is passed to a generation module which must provide a compatible trajectory for the humanoid CoM. It is based on a MPC scheme which enforces an explicit stability constraint to ensure that the resulting CoM trajectory is bounded w.r.t. Zero Moment Point (ZMP).



Linearization:

$$\frac{\ddot{z}_c + g}{z_c - z_z} = \omega^2$$

CoM vertical motion satisfy this

$$\begin{cases} \ddot{x}_c = \omega^2 (x_c - x_z) \\ \ddot{y}_c = \omega^2 (y_c - y_z). \\ \ddot{z}_c = \omega^2 (z_c - z_z) - g \end{cases}$$

Will satisfy also this (typical LIP equations)

With:

Pc = $(x_c, y_c, z_c)^T$ Position of CoM

Pz = $(x_z, y_z, z_z)^T$ Position of ZMP

ω = design parameter

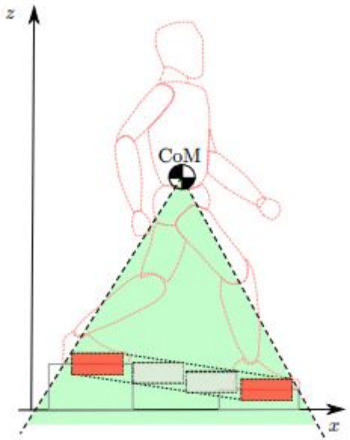
Motion model, constraints and stability

CoM stability: bound CoM wrt ZMP (LIP stability)

$$\frac{1}{\omega}\frac{1-e^{-\delta\omega}}{1-e^{-N\delta\omega}}\sum_{i=0}^{N-1} e^{-i\delta\omega}\dot{x}_z^{k+i} = x_c^k + \frac{\dot{x}_c^k}{\omega} - x_z^k$$

Balance constraint: to set ZMP inside the cone having vertex CoM and edges passing through vertices of support polygon (linearize consider a box)

$$-\frac{1}{2}\begin{pmatrix}\tilde{d}_x^z\\ \tilde{d}_y^z\\ d_z^z\end{pmatrix} \leq R_{k+i}^T\begin{pmatrix}x_z^{k+i}-x_f^{k+i}\\ y_z^{k+i}-y_f^{k+i}\\ z_z^{k+i}-z_f^{k+i}\end{pmatrix} \leq \frac{1}{2}\begin{pmatrix}\tilde{d}_x^z\\ \tilde{d}_y^z\\ d_z^z\end{pmatrix}$$



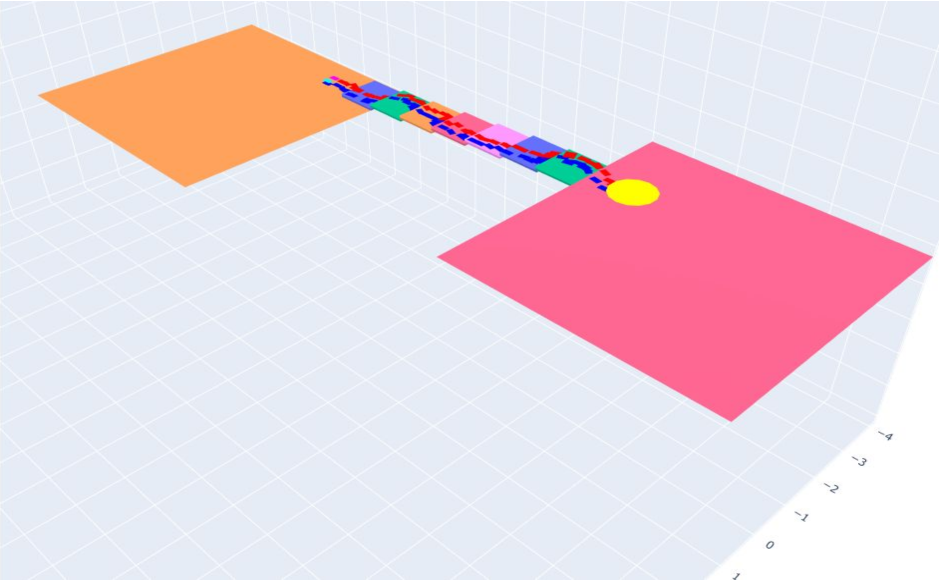
QP problem: try to bring ZMP closer to the center of moving box

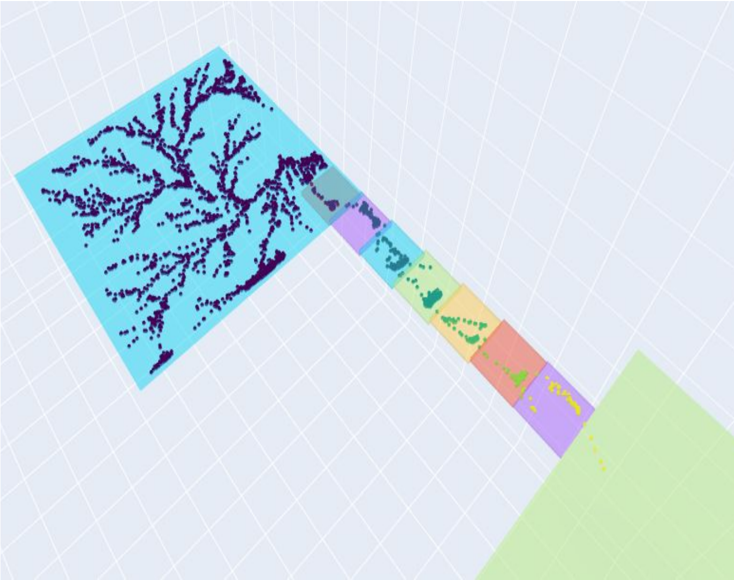
$$\min_{\dot{\boldsymbol{p}}_z^k,\ldots,\dot{\boldsymbol{p}}_z^{k+N-1}} \sum_{i=0}^{C-1} \left\|\dot{\boldsymbol{p}}_z^{k+i}\right\|^2 + \beta\left\|\boldsymbol{p}_z^{k+i} - \boldsymbol{p}_{\mathrm{mc}}^{k+i}\right\|^2$$

s.t. ZMP and stability constraints

Stairway

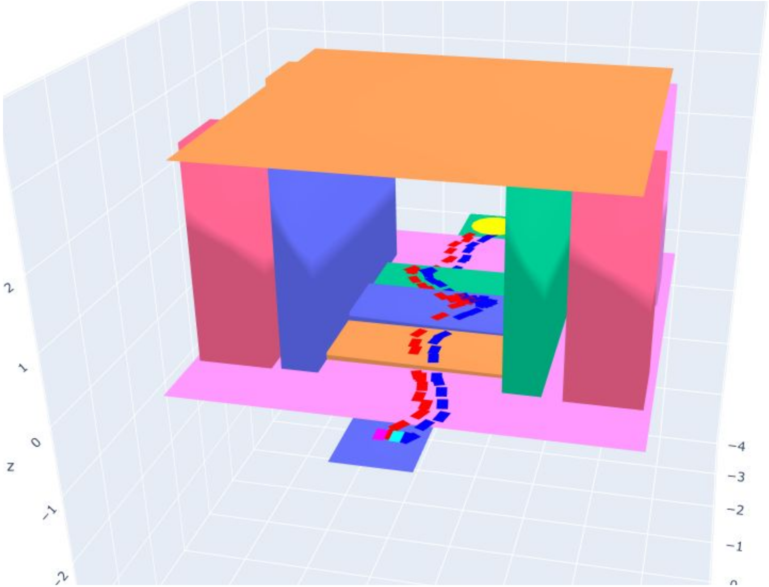
Plot of an example of tree and path generated

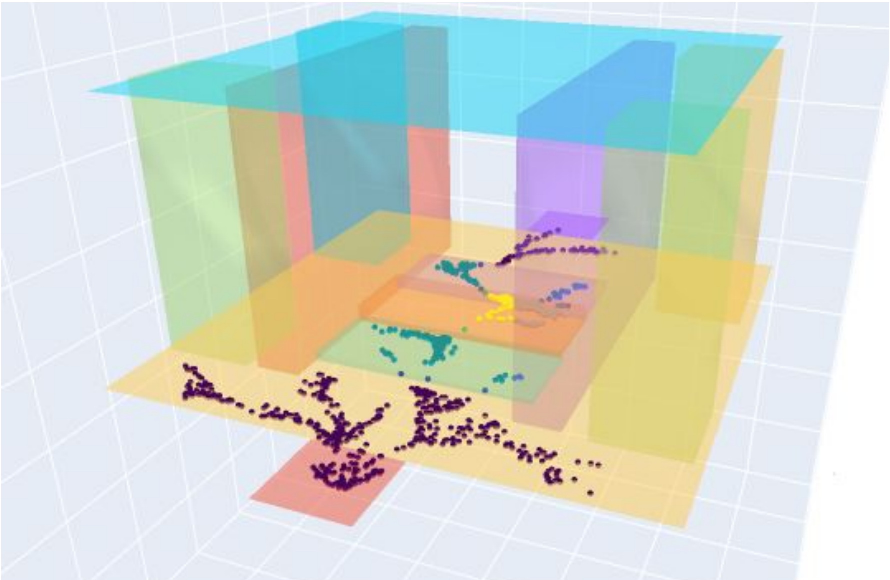




Tunnel

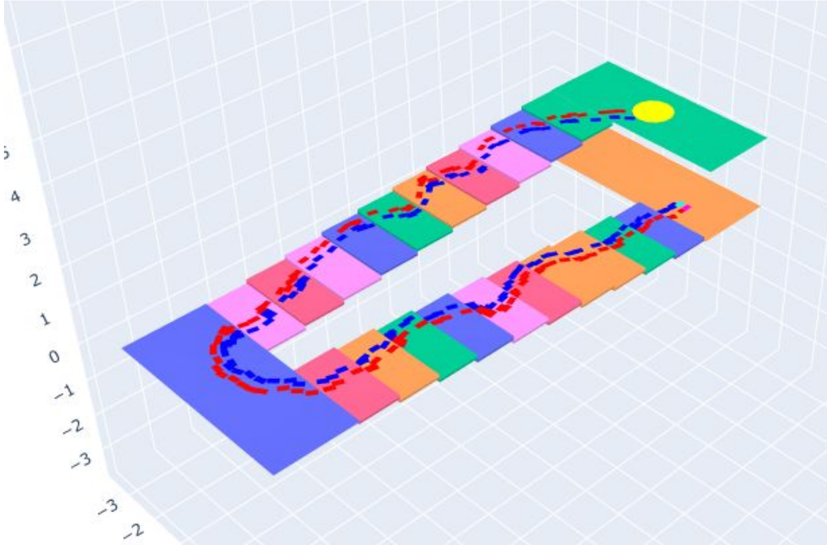
Plot of an example of tree and path generated

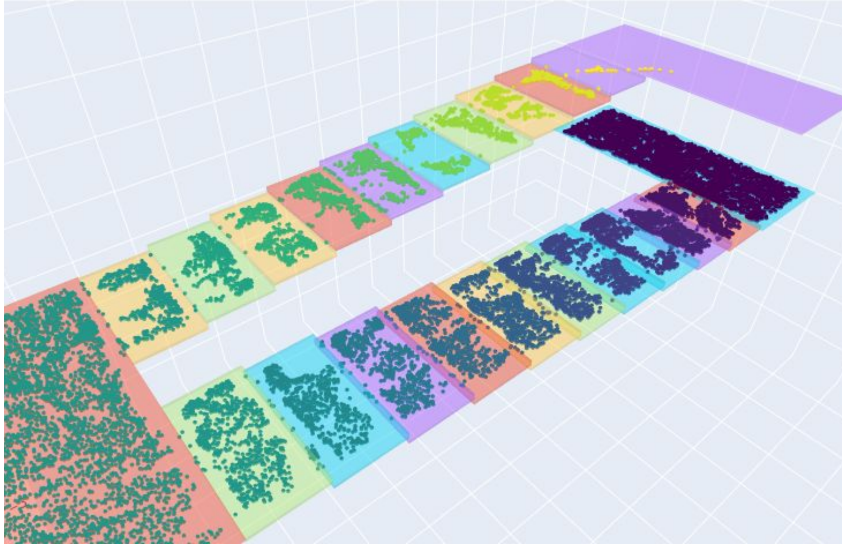




Floors

Plot of an example of tree and path generated



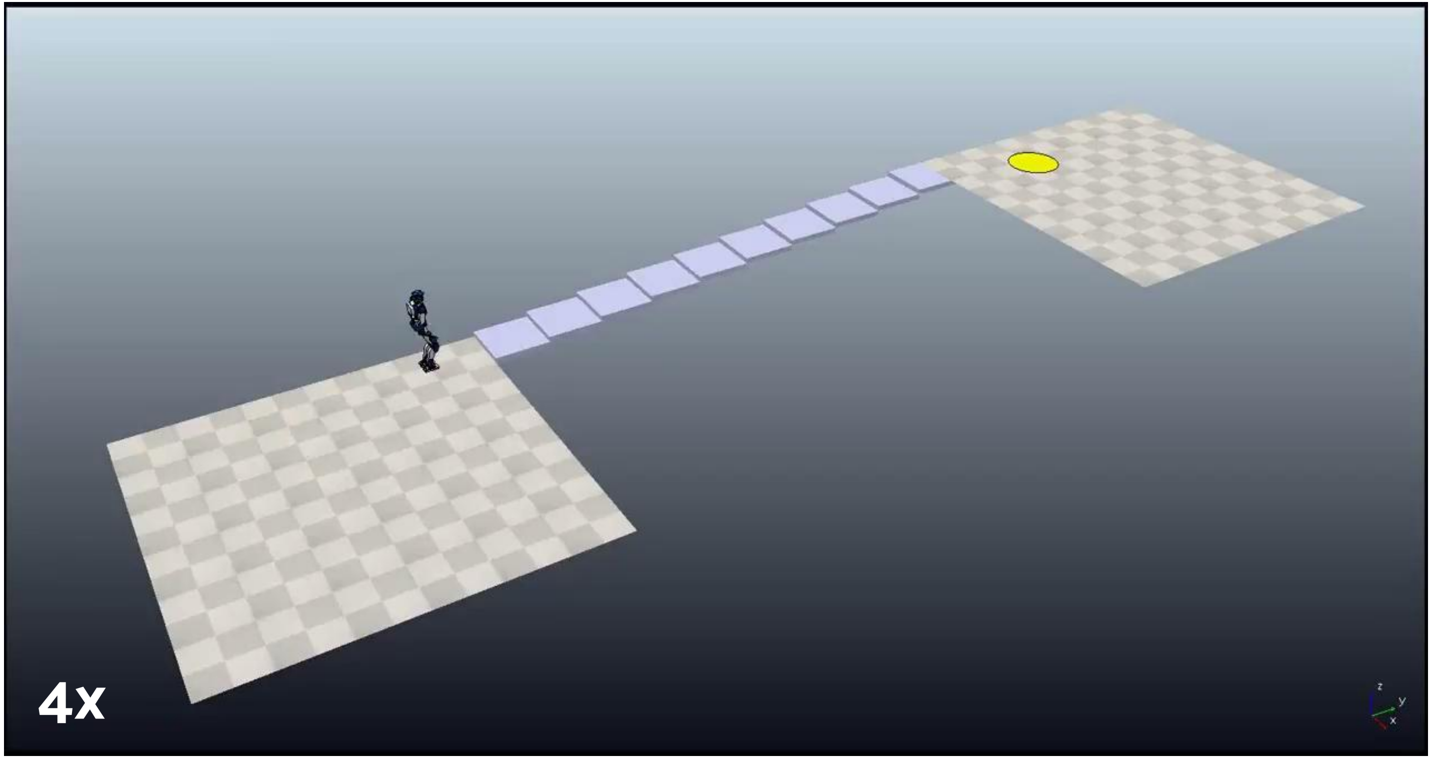


Planning Results

Map	tree size (avg)	goal steps (avg)	time (sec)	total iterations (avg)
Stairway	3844	145.3	243.7	8561.6
Tunnel	1469.3	71.1	45.11	3603.2
Floors	34216.3	269.2	11768.7	59304.1

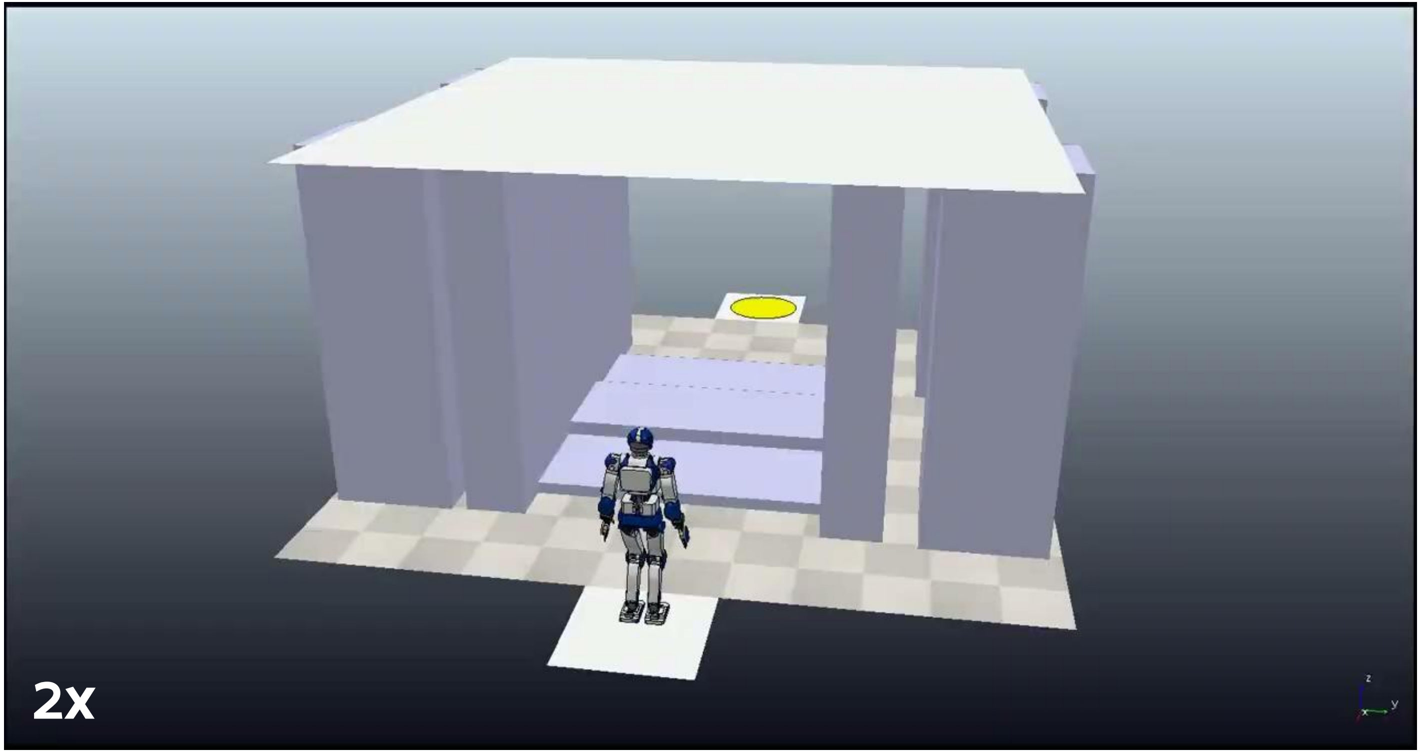
Tree size : the number of steps added to the tree
Goal steps : the number of steps needed to reach the goal
Time : the total time of a run of the algorithm
Total iterations: the number of iterations executed until a path was found

Simulations



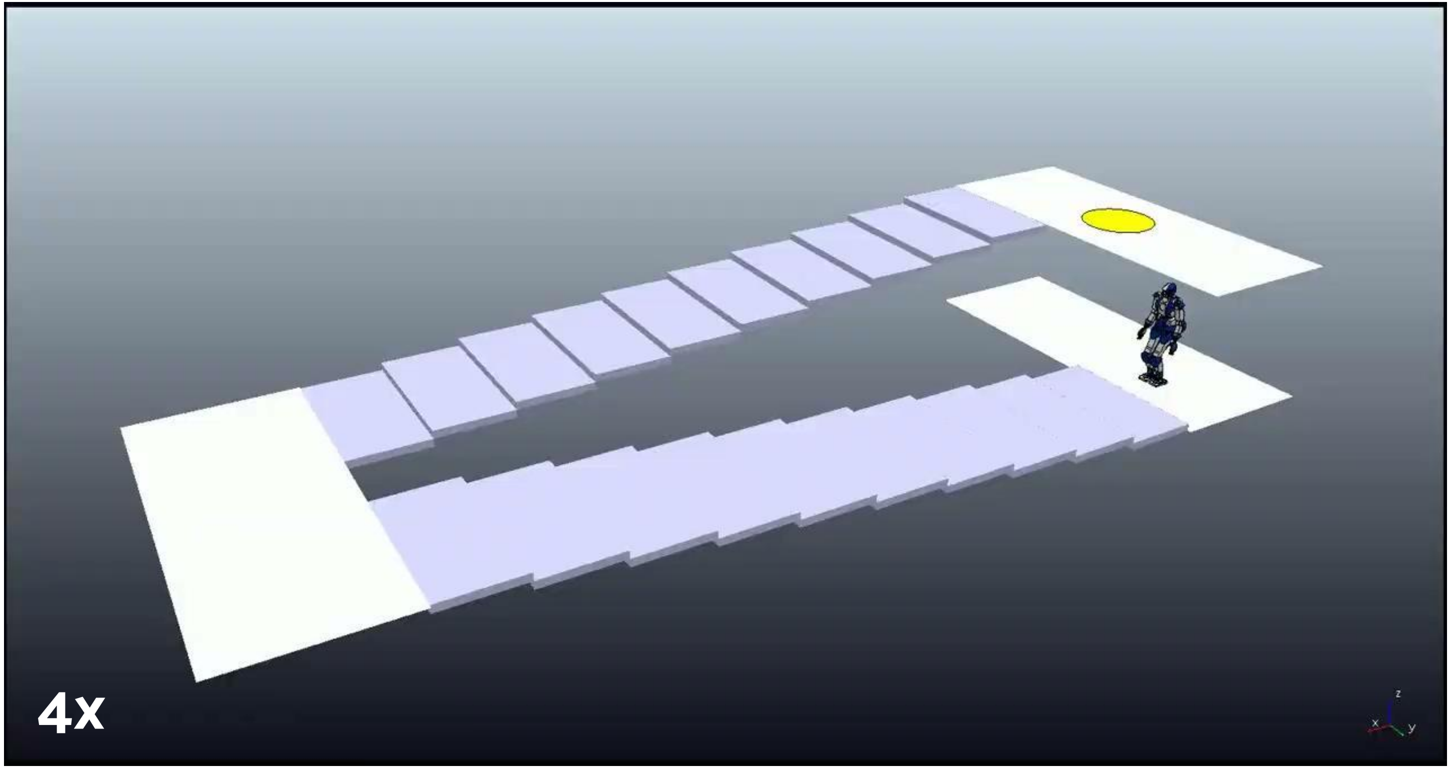
Simulation in ***Stairway*** environment (**4x**)

Simulations



Simulation in ***Tunnel*** environment (**2x**)

Simulations



Simulation in ***Floors*** environment (**4x**)

Conclusion and future works

We presented a planning and control framework for humanoid locomotion in a Multi-Level environment.

The **MLS map** has been shown to be able to correctly represent environments such as multi-floor areas or tunnels.

Some possible further steps could be:

- **RRT* implementation**
- **on-line footstep planner**
- **MLS map update with sensors information**
- **implementation with a real robot**

References

- P. Ferrari, N. Scianca, L. Lanari, and G. Oriolo, "**An Integrated Motion Planner/Controller for Humanoid Robots on Uneven Ground**," in 2019 18th European Control Conference (ECC). Naples, Italy: IEEE, June 2019, pp. 1598–1603. Available here.

- P. P. R. Triebel and W. Burgard, "**Multi-level surface maps for outdoor terrain mapping and loop closing**," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2006. Available here.

- **Visualization library:** Plotly

- **Slides template:** pietro-nardelli/sapienza-ppt-template

Thank you for the attention!