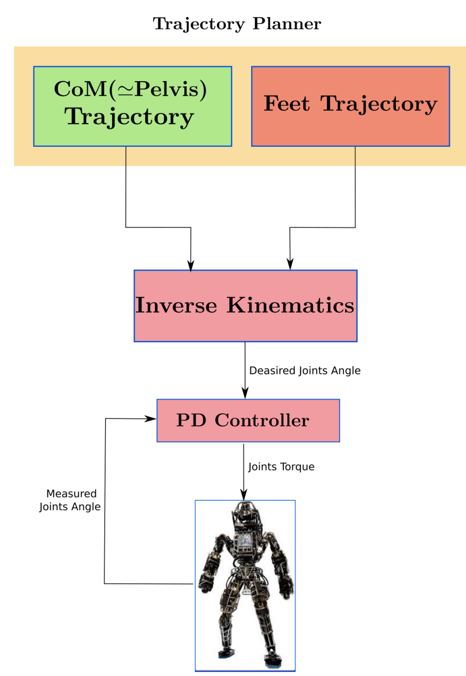
Locomotion Planning Based on Divergent Component of Motion (DCM)

Introduction

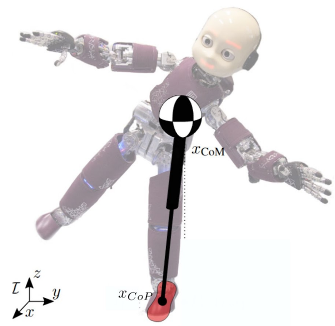
In this project you will plan the Center of Mass (CoM) trajectory for a biped by using the Divergent Component of Motion (DCM) concept for locomotion on flat terrain. The code structure will include three blocks: the DCM Planner, Foot Trajectory Planner, and Inverse Kinematics. The only block that you need to implement is the DCM Planner. In other words, you only need to open and edit the `DCMTrajectoryGenerator.py` class and follow the comments that have been written in this class. In the following figure, the different blocks of locomotion planning and control has been illustrated. The `DCMTrajectoryGenerator.py` is responsible for CoM motion generation that has been indicated by a green rectangle. Note that we consider a practical assumption that CoM is located on a fixed point on the pelvis. You can find the starting point of the project in the following: `https://github.com/MiladShafiee/LR-Biped-First-Project.git`



DCM and CoM Motion Planning

In this section we will plan the DCM and CoM trajectory following the method presented in: Englsberger, Johannes, Christian Ott, and Alin Albu-Schäffer. "Three-dimensional bipedal walking control based on divergent component of motion." IEEE transactions on Robotics 31.2 (2015): 355-368.(254).

Here we elaborate the equations in detail. For the inverted pendulum equation we have:



$$\ddot{x}_c = \omega^2(x_c - r_{CoP}) \tag{1}$$

This equation has been derived by finding the momentum around the Center of Pressure (COP, also ZMP) that is equal to zero with the dynamic balancing condition. Then we define the DCM dynamics as follows:

$$\xi = x_c + \frac{\dot{x}_c}{\omega} \tag{2}$$

where ξ is the DCM, x_c is the CoM position, and $\omega = \sqrt{\frac{g}{z_c}}$ is the natural frequency of the DCM dynamics. By reordering (2), we can derive the CoM dynamics:

$$\dot{x}_c = \omega(\xi - x_c) \tag{3}$$

This shows that the CoM has stable first-order dynamics (i.e. it follows the DCM). By differentiating (2) and inserting (3) and (1), we can write the DCM dynamics:

$$\dot{\xi} = \omega(\xi - r_{CoP}) \tag{4}$$

The DCM has unstable first-order dynamics (it is "pushed" by the CoP), whereas the CoM follows the DCM with stable first-order dynamics.

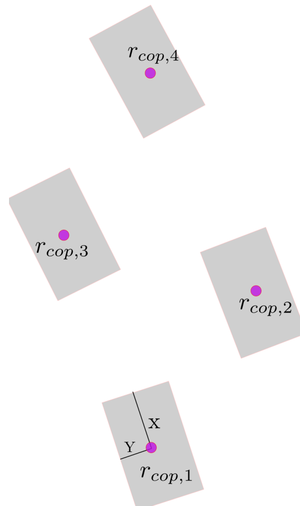
To find the desired DCM trajectory from given constant CoPs, the solution for the DCM Dynamics is:

$$\xi(t) = r_{cop} + (\xi_0 - r_{cop})e^{\omega t} \tag{5}$$

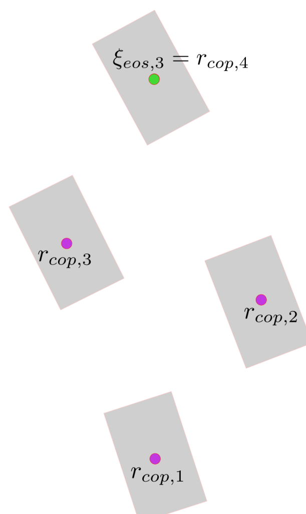
where $\xi_0 = \xi(0)$. The "internal" timestep t is reset at the beginning of each step, i.e., $t \in [0, T]$ (T is the duration of the step). In the following, we will present a step by step method for DCM and CoM motion planning based on (5).

1. First, we select the foot step position and step duration based on the desired velocity and considering the kinematic and dynamic constraint of the robot:

2. Place the desired CoP in a fixed location inside of the foot print. This condition guarantees dynamic balance during locomotion:



3. We place the last DCM position on the last CoP (Capturability constraint). For planning, we assume that the DCM will come to a stop over the final previewed foot position, i.e., $\xi_{eos,N-1} = r_{cop,N}$ (where *eos* is end-of-step):



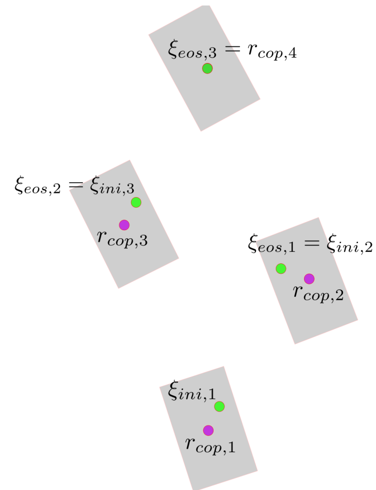
4. By having the constant desired *CoP* positions for each step and the last DCM Position (located on the CoP), we find the desired DCM locations at the end of each step via recursion using equation (5):

$$\xi_{eos,i} = r_{cop,i} + (\xi_{ini,i} - r_{cop,i})e^{\omega T} \quad (6)$$

$$\xi_{ini,i} = r_{cop,i} + (\xi_{eos,i} - r_{cop,i})e^{-\omega T} \quad (7)$$

$$\xi_{eos,i-1} = \xi_{ini,i} \quad (8)$$

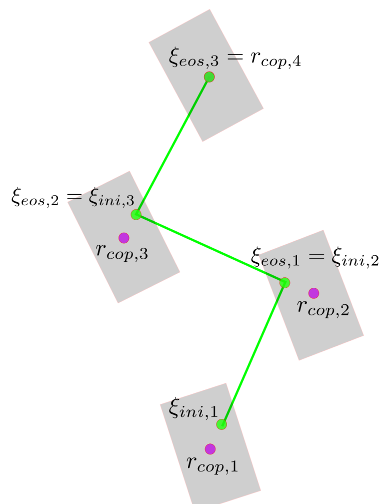
where $\xi_{ini,i}$ is the ith initial desired DCM.



5. Now based on (7) and (5), the reference trajectories for the DCM position of the ith step can be computed as:

$$\xi_i(t) = r_{cop,i} + (\xi_{eos,i} - r_{cop,i})e^{\omega(t-T)} \tag{9}$$

The "internal" step time t is reset at the beginning of each step, i.e., $t \in [0, T]$ (T is the duration of the step).



After having the full DCM trajectory, we can find the CoM trajectory by substituting the current DCM and CoM into equation (3) and then calculating a numerical integration for finding CoM position.

Calculation of ZMP

For measuring ZMP, we can use `getContactPoints()` function of pybullet to receive the normal force of contact points and contact position. By having normal contact force, we can use the following equation for ZMP calculation:

$$ZMP_x = \frac{\sum_{j=1}^{N} p_{jx} f_{jz}}{\sum_{j=1}^{N} f_{jz}} \tag{10}$$

$$ZMP_y = \frac{\sum_{j=1}^{N} p_{jy} f_{jz}}{\sum_{j=1}^{N} f_{jz}} \tag{11}$$

The f_{jz} is the vertical component of contact force at j contact point and p_{jx} is the contact position of j contact point.

Report

Please upload your code, a video of your results, and create a short report including details on the method, plots, and answers to the following questions:

Questions

1. Based on equation (5), which physical parameters will affect the rate of divergence of the DCM dynamics?
2. In the DCM motion planning, how do we guarantee dynamic balancing conditions?
3. If we do have dynamic balancing guarantees, why is the robot not able to walk without parameter tuning?
4. In order to achieve stable locomotion, which parameters did you tune and what are their values?
5. Please calculate and plot the real ZMP(CoP) using contact force from pybullet during locomotion.
6. Please tune DCM parameters to achieve faster locomotion. What is the fastest walking speed you can achieve and what are the corresponding parameters?
7. Consider the scenario in which we would like to use the robot to explore the surface of the moon. If you consider the gravity condition of the moon, what is the fastest walking speed you can achieve? What are the corresponding parameters?
8. What could be the effect of considering double support during DCM planning?