Institute for Cognitive Systems
Technische Universität München
Dr.-Ing. Rogelio Guadarrama

Modeling and control of legged robots
Tutorial 5: Balance Control

1 Introduction

In this tutorial we will implement the calculations of the ground reference points covered in Lecture 5 and how to use them for balance control. We will also cover the "virtual" dynamics method used to implement torque-level controllers in robots with position-controlled hardware interfaces.

You will need a robot standing on its two legs with a whole body controller as designed in Tutorial 4. The upper body should be controlled to keep a "home" posture as described in Tutorial 2.

a Ground Reference Points

In Lecture 5 the main ground reference points were introduced as well as their physical meaning and amethod to estimate their position from sensor data. Let us summarize these points for a biped humanoid robot.

a.1 Zero Moment Point (ZMP)

For a flat foot in full contact with flat terrain, the point where the resultant ground reaction wrench's torque is zero is known as the Zero Moment Point (ZMP). From an ankle force torque sensor, this point is defined as

$$\boldsymbol{p}_{x_{foot}} = \frac{-\boldsymbol{\tau}_{y_{ankle}} - \boldsymbol{f}_{x_{ankle}} d}{\boldsymbol{f}_{z_{ankle}}} \tag{1}$$

$$\boldsymbol{p}_{y_{foot}} = \frac{\boldsymbol{\tau}_{x_{ankle}} - \boldsymbol{f}_{y_{ankle}} d}{\boldsymbol{f}_{z_{ankle}}} \tag{2}$$

$$\boldsymbol{p}_{z_{foot}} = 0 \tag{3}$$

where $\boldsymbol{p}_{foot} = \begin{bmatrix} \boldsymbol{p}_{x_{foot}} & \boldsymbol{p}_{y_{foot}} & \boldsymbol{p}_{z_{foot}} \end{bmatrix}^\top$ is the ZMP of the foot with respect to the foot coordinate frame, $\boldsymbol{f}_{ankle} = \begin{bmatrix} \boldsymbol{f}_{x_{ankle}} & \boldsymbol{f}_{y_{ankle}} & \boldsymbol{f}_{z_{ankle}} \end{bmatrix}^\top$ is the force measured by the sensor, and $\boldsymbol{\tau}_{ankle} = \begin{bmatrix} \boldsymbol{\tau}_{x_{ankle}} & \boldsymbol{\tau}_{y_{ankle}} & \boldsymbol{\tau}_{z_{ankle}} \end{bmatrix}^\top$ is the torque measured by the sensor.

For the case of a double support (the robot standing on two flat feet over flat ground), the ZMP can be computed from the ZMP of each foot as

$$\begin{aligned} \boldsymbol{p}_x &= \frac{\boldsymbol{p}_{x_R}\boldsymbol{f}_{z_R} + \boldsymbol{p}_{x_L}\boldsymbol{f}_{z_L}}{\boldsymbol{f}_{z_R} + \boldsymbol{f}_{z_L}} && (4) \\ \boldsymbol{p}_y &= \frac{\boldsymbol{p}_{y_R}\boldsymbol{f}_{z_R} + \boldsymbol{p}_{y_L}\boldsymbol{f}_{z_L}}{\boldsymbol{f}_{z_R} + \boldsymbol{f}_{z_L}} && (5) \\ \boldsymbol{p}_z &= 0 && (6) \end{aligned}$$

where $\boldsymbol{p} = \begin{bmatrix} \boldsymbol{p}_x & \boldsymbol{p}_y & \boldsymbol{p}_z \end{bmatrix}^\top$ is the ZMP expressed in the world coordinate frame, $\boldsymbol{p}_R = \begin{bmatrix} \boldsymbol{p}_{x_R} & \boldsymbol{p}_{y_R} & \boldsymbol{p}_{z_R} \end{bmatrix}^\top$ and $\boldsymbol{p}_L = \begin{bmatrix} \boldsymbol{p}_{x_L} & \boldsymbol{p}_{y_L} & \boldsymbol{p}_{z_L} \end{bmatrix}^\top$ are the ZMPs of the right and left feet respectively, also expressed in the world coordinate frame, and $\boldsymbol{f}_R = \begin{bmatrix} \boldsymbol{f}_{x_R} & \boldsymbol{f}_{y_R} & \boldsymbol{f}_{z_R} \end{bmatrix}^\top$ and $\boldsymbol{f}_L = \begin{bmatrix} \boldsymbol{f}_{x_L} & \boldsymbol{f}_{y_L} & \boldsymbol{f}_{z_L} \end{bmatrix}^\top$ are the ground reaction forces of the right and left feet respectively, expressed in the ZMP of each foot.

a.2 Centroidal Moment Pivot (CMP)

The point where a line parallel to the ground reaction force, passing through the Center of Mass (CoM), intersects with the external contact surface. It can be computed from the CoM position and the ground reaction force. Being the ground reaction force, the sum of all the ground reaction forces, transformed to a coordinate frame with its origin at the ZMP. Then, the CMP is defined as

$$\begin{aligned} \boldsymbol{r}_x &= \boldsymbol{x}_x - \frac{\boldsymbol{f}_x}{\boldsymbol{f}_z}\boldsymbol{x}_z && (7) \\ \boldsymbol{r}_y &= \boldsymbol{x}_y - \frac{\boldsymbol{f}_y}{\boldsymbol{f}_z}\boldsymbol{x}_z && (8) \\ \boldsymbol{r}_z &= 0 && (9) \end{aligned}$$

where $\boldsymbol{r} = \begin{bmatrix} \boldsymbol{r}_x & \boldsymbol{r}_y & \boldsymbol{r}_z \end{bmatrix}^\top$ is the CMP expressed in the world coordinate frame, $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_x & \boldsymbol{x}_y & \boldsymbol{x}_z \end{bmatrix}^\top$ is the position of the CoM also in the world coordinate frame, and $\boldsymbol{f} = \begin{bmatrix} \boldsymbol{f}_x & \boldsymbol{f}_y & \boldsymbol{f}_z \end{bmatrix}^\top$ is the ground reaction force.

a.3 Capture point (CP) / Divergent Component of Motion (DCM)

The Capture point (CP) or Divergent Component of Motion (DCM) is a point on the ground where the robot can step to in order to bring itself to a complete stop. It is defined as a change of variable from the CoM state to reduce one degree of the LIPM dynamics as

$$\boldsymbol{\xi} = \boldsymbol{x}_p + \frac{\dot{\boldsymbol{x}}_p}{\boldsymbol{\omega}} \qquad (10)$$

where $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_x & \boldsymbol{\xi}_y & 0 \end{bmatrix}^\top$ is the DCM in the world coordinate frame, and $\boldsymbol{x}_p = \begin{bmatrix} \boldsymbol{x}_{x_p} & \boldsymbol{x}_{y_p} & 0 \end{bmatrix}^\top$ is the position of the CoM in the world coordinate frame projected to the level where the ZMP is located (for flat ground, its projection to the ground), and $\dot{\boldsymbol{x}}_p$ its first time derivative. $\boldsymbol{\omega} = \sqrt{\frac{g}{\boldsymbol{x}_z}}$ is the natural frequency of the inverted pendulum.

b Balance Control

With the feedback of the points covered in the precious section, we can design controllers that reject or minimize external disturbance to keep a stable posture and fulfill the main balance condition: To keep existence of the ZMP inside the supporting polygon. With feedback control, we can also shift the ZMP to the position we want, for example at the center of the sporting polygon.

As covered in Lecture 6, there are three main reaction mechanisms to keep balance when an external force disturbs the biped posture of a person or a humanoid robot, namely *Ankle strategy*, *Hip strategy*, and *Stepping strategy*. In this tutorial we will only cover the first two because the third one requires a functional walking controller and he have not reach that point in this course.

b.1 Ankle strategy

If an external disturbance is not so big, it can be suppressed by shifting the hip position according to the feedback of the ZMP as

$$\dot{\boldsymbol{x}}_d = \dot{\boldsymbol{x}}_{ref} - \mathbf{K}_x(\boldsymbol{x}_d - \boldsymbol{x}_{ref}) + \mathbf{K}_p(\boldsymbol{p} - \boldsymbol{p}_{ref}) \tag{11}$$

where $\dot{\boldsymbol{x}}_d$ an new desired velocity for the CoM, $\dot{\boldsymbol{x}}_{ref}$ is the reference velocity for the CoM (this value comes from a walking motion generation module. For standing conditions you can assume is zero). $\boldsymbol{x}_d$, and $\boldsymbol{x}_{ref}$ are the desired and reference positions for the CoM. $\boldsymbol{p}$ and $\boldsymbol{p}_{ref}$ are the position and reference position of the ZMP. $\mathbf{K}_x$ and $\mathbf{K}_p$ are constant scalar gains. According to Choi et al., this feedback law is stable if we chose the gains as $0 < \mathbf{K}_p < \omega < \mathbf{K}_x$.

b.2 Hip strategy

To neglect higher disturbances, angular momentum can be produce by shifting the torso/hip orientation, according to ZMP or CMP feedback. The most simple feedback controller to apply the hip strategy can be designed as

$$\Gamma_d = \mathbf{K}_\Gamma \left(\boldsymbol{r} - \boldsymbol{r}_{ref}\right) \tag{12}$$

Where Γ_d is the desired angular momentum to be applied to the upper body, $\boldsymbol{r}$ and $\boldsymbol{r}_{ref}$ are de CMP and reference CMP (once again, for standing conditions you can set it to 0), and $\mathbf{K}_\Gamma$ is a constant scalar gain.

2 Homework

You should build on the solution of the previous tutorials. The goal is to apply the two control methods from above to the humanoid robot `talos` and learn a method used to implement them on a position-controlled hardware interface.

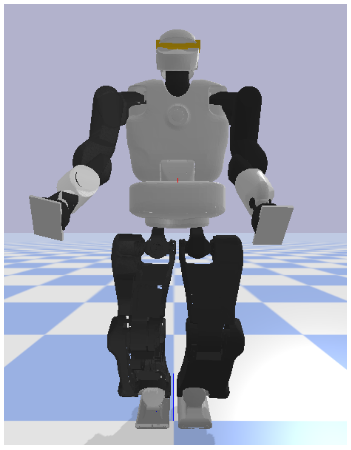
Figure 1: Prepare a home posture standing simulation for the Talos robot

a Exercise 1: Prepare your simulation

1. Prepare a simulation with a robot standing as shown in Figure 1. Use a whole body controller as in Tutorial 4 including a task to control the CoM and the whole-body angular momentum. Save the python script as `t51.py`

2. Prepare a state machine that applies a pushing force $\boldsymbol{f}_{push}$ to the hip of the robot for t_{push} seconds after t_{period} seconds. First from the right, then from the left and finally from the back.

   To apply a force you can use the function

   ```
   robot.applyForce(force)
   ```

   where force is a 3-dimensional `numpy` array with the three components of the pushing force. You can visualize the force using an rviz publisher or with a debugging line in `pybullet` as

   ```
   line_id = simulator.addGlobalDebugLine(p1, p2, line_id, color=[1,0,0])
   ```

   to remove the line use

   ```
   sim.removeDebugItem(line_id)
   ```

   `line_id` is an integer that is assigned by pybullet when creating visualization objects. You can use the initial id -1.

3. Add force-torque sensors to the ankles of your robot. You can add them with the following lines

   ```
   pb.enableJointForceTorqueSensor(robot.id(), robot.jointNameIndexMap()['
       leg_right_6_joint'], True)
   pb.enableJointForceTorqueSensor(robot.id(), robot.jointNameIndexMap()['
       leg_left_6_joint'], True)
   ```

note that the sensors are declared in sixth joint of the leg which is located 10 cm above the sole coordinate frame.

In your main loop, you can read the ankle wrenches as follows

```
wren = pb.getJointState(robot.id(), robot.jointNameIndexMap()['
    leg_right_6_joint'])[2]
wnp = np.array([-wren[0], -wren[1], -wren[2], -wren[3], -wren[4], -wren
    [5]])
wr_rankle = pin.Force(wnp)

wren = pb.getJointState(robot.id(), robot.jointNameIndexMap()['
    leg_left_6_joint'])[2]
wnp = np.array([-wren[0], -wren[1], -wren[2], -wren[3], -wren[4], -wren
    [5]])
wl_lankle = pin.Force(wnp)
```

notice that these wrenches are measured in the ankle joint coordinate frames.

You can get the positions of the ankles and the soles as follows

```
data = robot._model.createData()
pin.framesForwardKinematics(robot._model, data, q)

H_w_lsole = data.oMf[robot._model.getFrameId("left_sole_link")]
H_w_rsole = data.oMf[robot._model.getFrameId("right_sole_link")]

H_w_lankle = data.oMf[robot._model.getFrameId("leg_left_6_joint")]
H_w_rankle = data.oMf[robot._model.getFrameId("leg_right_6_joint")]
```

These homogeneous transformations are from the links and joints with respect to the world.

b Exercise 2: Ground reference points

1. Implement the estimation of the ZMP as described in Section a.1.
2. Implement the estimation of the CMP as described in Section a.2.
3. Implement the estimation of the CP/DCM as described in Section a.3.
4. Prepare the plotting of the x and y components of all the ground reference points and the CoM with the time in the x axis of the plots. You can use any method you want to record and plot the data (eg. record a ROS bag file and plot it with `PlotJuggler` or `Matlab`, use the `matplotlib`, or use `rqt_plot` in ROS).
5. Run the simulation once recording and plotting your data to verify that your implementations are correct.

c Exercise 3: Balance control

1. Implement the ankle strategy described in Section b.1 and tune it to keep balance while being pushed from thew sides and the back. Use $t_{push} = 0.5$ [s], and $t_{period} = 4$ [s]. Hint: Try applying a small force and watch the reaction, and increase it gradually until you make the robot fall, then increase your gains.
2. Implement the hip strategy described in Section b.2 and tune it to keep balance while being pushed from thew sides and the back.
3. When you have the controllers tuned, record and save the plots of each run (i.e. one time with no controllers, one with the ankle strategy, one with hip strategy, and one with both strategies)

d Exercise 4: Position-controlled hardware interface

For robots that do not provide joint-torque control, we can simulate the dynamics and compute the required joint accelerations. Then we integrate them to find the joint velocities and positions which we can command to the hardware interface of the robot.

1. Make a copy of `t51.py` and save it as `t52.py`.
2. Replace the `setActuatedJointTorques` function after computing the TSID solution with the following lines

```
# integrate sol in virtual state
q_tsid, v_tsid = stack.integrate_dv(q_tsid, v_tsid, dv_sol, n_update*
    simulator.stepTime())
robot.setActuatedJointPositions(q_tsid, v_tsid)
```

   `q_tsid` and `v_tsid` are the virtual states which are computed from the TSID solution. Therefore you should use them also as input in the TSID formulation and initialize them with the same initial condition as all the simulator and model. When you call `setActuatedJointPositions` the hardware interface will be automatically set for position.
3. Try increasing the pushing force and re-tuning the balance controllers.
4. Repeat the same recordings with this modality. One time with no controllers, one with the ankle strategy, one with hip strategy, and one with both strategies.

e What to deliver for this tutorial?

1. The `t51.py` and `t52.py` scripts.
2. A report with the 4 dual plots of the Exercise 2 and Exercise 3.
3. A readMe file with the instructions to run your simulations and the answer to these questions:
4. Which ones of the ground reference points can exist outside the supporting polygon?
5. Which modality holds higher pushing forces, torque or position hardware interface?
6. Are the torque and position control modalities equivalent with the proposed method? if not, why?

3 Deliver

Deadline for the second **tutorial package (T4 and T5)** is on **Thursday** the **26.06.2025 at 23:59 PM**.

- Implement the tutorials in different scripts.
- Add a readme.md on how to launch the individual tutorials.
- Send a zip file of your workspace **src folder** to the lecture email address.
- Use the naming convention: `<last_name>_deliverable_2.zip`
- Do not deliver the install, build and log folders of the workspace. Only the contents of the src folder.