SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER, CONTROL AND MANAGEMENT ENGINEERING

FP2
Torso Pose Estimation on the HRP4 Humanoid Robot

AUTONOMOUS AND MOBILE ROBOTICS
ROBOTICS 2

Professors:
Giuseppe Oriolo
Alessandro De Luca

Students:
Michele Cipriano
Godwin K. Peprah
Lorenzo Vianello

Supervisor:
Nicola Scianca

Academic Year 2017/2018

Contents

1 Introduction

The aim of the project is the estimate the pose of the torso on the HRP4 humanoid robot following the methodologies introduced in [1]. Knowing the pose of the torso is extremely important because it can be used to localize any other point of the robot using kinematic computations based on joint encoder readings. The idea is to use this estimate in order to obtain the position of the CoM of the robot so that it can be used to close the feedback loop in the MPC gait generator.

In our approach, we have developed an Extended Kalman Filter (EKF) to estimate the pose of the torso, using the joint encoders, the Inertial Measurement Unit (IMU) and a RGBD camera as sensors. We first used only the accelerometer and the gyroscope of the IMU as measurements, obtaining good results on the estimate of the orientation but diverging results on the position. We then improved the EKF by filtering the linear velocities of the torso (previously obtained by integrating the linear acceleration read by the accelerometer sensor) obtaining convergence on the x, y axes and reducing the error on the z axis. The introduction of the RGBD camera, less susceptible to noise and drift, further improved the estimate of the position of the torso, obtaining convergence on the z axis as well. In this last step we used the trilateration algorithm.

We tested the behaviour of the EKF by modeling the robot as a unicycle in order to make it perform a cartesian and a posture regulation task, correctly achieving the desired final configuration in both cases.

In the end, we tried to close the feedback loop in the MPC gait generator obtaining unsuccessful results.

The project has been entirely developed in C++ using the HRP4 humanoid robot model in a V-REP environment.

2 Torso Pose Estimation

This section introduces the kinematic model used for the Extended Kalman Filter, describing in details its whole implementation and the different methods used to provide the measurements to the filter itself.

2.1 Kinematic Model

Let $\boldsymbol{x} = (\boldsymbol{p}_t^T, \boldsymbol{o}_t^T)^T$ be the pose of the torso frame $\mathcal{F}_t$ with respect to the world frame $\mathcal{F}_w$. We want to develop a filter that estimates the state $\boldsymbol{x}$ while it moves around the environment. Let $\mathcal{F}_s$ be the support foot frame with respect to the world frame and let $\boldsymbol{o}_s$ be its orientation. Let $\boldsymbol{J}(\boldsymbol{q}_s, \boldsymbol{o}_s)$ the Jacobian matrix of the kinematic map from the support frame $\mathcal{F}_s$ to the torso frame $\mathcal{F}_t$.

Let's use the following kinematic model to describe the evolution of the state $\boldsymbol{x}$ through time:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q}_s, \boldsymbol{o}_s)\dot{\boldsymbol{q}}_s \tag{1}$$

with $\dot{\boldsymbol{q}}_s$ velocities of the support joints acting as control inputs. Note that the Jacobian $\boldsymbol{J}(\boldsymbol{q}_s, \boldsymbol{o}_s)$ does not depend on the position of $\mathcal{F}_s$:

$$\begin{aligned}
\boldsymbol{f}(\boldsymbol{q}_s, \boldsymbol{o}_s) &= \boldsymbol{\Omega}\left(\begin{bmatrix} \boldsymbol{R}_z(\boldsymbol{o}_s) & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \boldsymbol{\Omega}^{-1}\left(\begin{bmatrix} {}^s\boldsymbol{p}_t \\ {}^s\boldsymbol{o}_t \end{bmatrix}\right)\right) \\
&= \boldsymbol{\Omega}\left(\begin{bmatrix} \boldsymbol{R}_z(\boldsymbol{o}_s + {}^s\boldsymbol{o}_{t,z})\boldsymbol{R}_y({}^s\boldsymbol{o}_{t,y})\boldsymbol{R}_x({}^s\boldsymbol{o}_{t,x}) & \boldsymbol{R}_z(\boldsymbol{o}_s){}^s\boldsymbol{p}_t \\ \boldsymbol{0}^T & 1 \end{bmatrix}\right) \\
&= \begin{bmatrix} \boldsymbol{R}_z(\boldsymbol{o}_s) & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \boldsymbol{f}(\boldsymbol{q}_s) + \begin{pmatrix} \boldsymbol{0}_5 \\ \boldsymbol{o}_s \end{pmatrix}
\end{aligned} \tag{2}$$

$$\begin{aligned}
\boldsymbol{J}(\boldsymbol{q}_s, \boldsymbol{o}_s) &= \frac{\partial \boldsymbol{f}(\boldsymbol{q}_s, \boldsymbol{o}_s)}{\partial \boldsymbol{q}_s} \\
&= \begin{bmatrix} \boldsymbol{R}_z(\boldsymbol{o}_s) & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \frac{\partial \boldsymbol{f}(\boldsymbol{q}_s)}{\partial \boldsymbol{q}_s} \\
&= \begin{bmatrix} \boldsymbol{R}_z(\boldsymbol{o}_s) & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \boldsymbol{J}(\boldsymbol{q}_s)
\end{aligned} \tag{3}$$

which definition is important for development purposes. In fact, $\boldsymbol{J}(\boldsymbol{q}_s)$ is easily accessible thanks to the C++ implementation of the HRP4 kinematics. Note that, unless specified otherwise, $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ represents the identity matrix, $\boldsymbol{O} \in \mathbb{R}^{3\times3}$ represents the zero matrix, while $\boldsymbol{0} \in \mathbb{R}^3$ represents the zero vector. $\boldsymbol{\Omega}$ is a function that returns a minimal representation of a transform function.

The robot is equipped with a RGBD camera and an IMU, used to measure the pose of the torso frame:

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{q}_n) = \begin{pmatrix} \boldsymbol{p}_t \\ \boldsymbol{o}_t \end{pmatrix} \tag{4}$$

In particular, the position $\boldsymbol{p}_t$ is computed by doing either trilateration exploiting known position of the landmarks or integrating the data coming from the accelerometer, while the orientation $\boldsymbol{o}_t$ is computed by simply integrating the data obtained from the gyroscope. The measurements of the pose of the torso $\boldsymbol{y}$ depend on the estimation of the pose of the torso $\boldsymbol{x}$ and the configuration of the neck joints $\boldsymbol{q}_n$. Details are explained in subsections 2.3–2.5. Note that the notation $\boldsymbol{q}_s$ and $\boldsymbol{q}_n$ have been kept in order to be consistent with the main reference paper but are not actually used in the implementation. The whole configuration of the robot $\boldsymbol{q}$ is used instead.

2.2 Extended Kalman Filter

It is now possible to define a discrete-time stochastic system using Equations 1-4, with T sampling interval of the filter and k current timestep:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T\boldsymbol{J}(\boldsymbol{q}_{s,k}, \boldsymbol{o}_s)\dot{\boldsymbol{q}}_{s,k} + \boldsymbol{v}_k \tag{5}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{q}_{n,k}) + \boldsymbol{w}_k \tag{6}$$

with $\boldsymbol{v}_k \sim \boldsymbol{\mathcal{N}}(\boldsymbol{0}, \boldsymbol{V}_k)$ and $\boldsymbol{w}_k \sim \boldsymbol{\mathcal{N}}(\boldsymbol{0}, \boldsymbol{W}_k)$ zero-mean white Gaussian noises and $\boldsymbol{V}_k \in \mathbb{R}^{6\times6}$, $\boldsymbol{W}_k \in \mathbb{R}^{6\times6}$ their respective covariance matrices.

At each timestep, a prediction $\hat{\boldsymbol{x}}_{k+1|k}$ is generated using the current estimate $\hat{\boldsymbol{x}}_k$:

$$\hat{\boldsymbol{x}}_{k+1|k} = \hat{\boldsymbol{x}}_k + \boldsymbol{J}(\boldsymbol{q}_{s,k}, \boldsymbol{o}_s)\Delta\boldsymbol{q}_{s,k}, \quad \Delta\boldsymbol{q}_{s,k} = \boldsymbol{q}_{s,k+1} - \boldsymbol{q}_{s,k} \tag{7}$$

with $\Delta\boldsymbol{q}_{s,k} \approx T\dot{\boldsymbol{q}}_{s,k}$ obtained using encoder readings. The covariance prediction matrix is updated accordingly:

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{P}_k + \boldsymbol{V}_k \tag{8}$$

In the same way as before, the predicted output associated to $\hat{\boldsymbol{x}}_{k+1|k}$ is computed as well:

$$\hat{\boldsymbol{y}}_{k+1|k} = \boldsymbol{h}(\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{q}_{n,k+1}) \tag{9}$$

To correct the predicted state we need to compute the innovation using the measurements and the predicted output computed in the previous step:

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{y}_{k+1} - \hat{\boldsymbol{y}}_{k+1|k} \tag{10}$$

It is, hence, possible to determine the corrected state estimate by computing:

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_{k+1|k} + \boldsymbol{G}_{k+1}\boldsymbol{\nu}_{k+1} \tag{11}$$

with $\boldsymbol{G}_{k+1}$ Kalman gain matrix, defined as

$$\boldsymbol{G}_{k+1} = \boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^T \left(\boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^T + \boldsymbol{W}_{k+1}\right)^{-1} \tag{12}$$

$$= \boldsymbol{P}_{k+1|k}(\boldsymbol{P}_{k+1|k} + \boldsymbol{W}_{k+1})^{-1} \tag{13}$$

since the partial derivative of the observation function with respect to the state is the identity matrix:

$$\boldsymbol{H}_{k+1} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k+1|k}} = \boldsymbol{I} \tag{14}$$

The corrected covariance matrix is updated as well:

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_{k+1|k} - \boldsymbol{G}_{k+1}\boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k} \tag{15}$$

$$= \boldsymbol{P}_{k+1|k} - \boldsymbol{G}_{k+1}\boldsymbol{P}_{k+1|k} \tag{16}$$

Note that, in our implementation, we defined the covariance matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ as:

$$\boldsymbol{V} = \text{diag}\{5, 5, 5, 100, 100, 100\} \cdot 10^{-6} \tag{17}$$

$$\boldsymbol{W} = \text{diag}\{5, 5, 5, 5 \cdot 10^{-2}, 5 \cdot 10^{-2}, 5 \cdot 10^{-2}\} \cdot 10^{-2} \tag{18}$$

2.3 Accelerometer Integration

As said before, the measurement of the position of the torso $\boldsymbol{p}_t$ can be computed by either doing trilateration, exploiting the known position of the landmarks, or by integrating the data coming from the accelerometer. Let's discuss the second approach considering constant acceleration $\ddot{\boldsymbol{p}}_{t,k}$ in an interval $[t_k, t_{k+1})$:

$$\dot{\boldsymbol{p}}_{t,k+1} = \dot{\boldsymbol{p}}_{t,k} + \ddot{\boldsymbol{p}}_{t,k} T \tag{19}$$

$$\boldsymbol{p}_{t,k+1} = \boldsymbol{p}_{t,k} + \dot{\boldsymbol{p}}_{t,k} T + \frac{1}{2} \ddot{\boldsymbol{p}}_{t,k} T^2 \tag{20}$$

Note that the implementation of the accelerometer returns, at each timestep k, the linear acceleration ${}^{t}\boldsymbol{a}_{t,k}$ expressed in the current reference frame of the torso, hence, it must be transformed to the reference frame of the world in order be able to perform the integration:

$$\ddot{\boldsymbol{p}}_{t,k} = {}^{w}\boldsymbol{R}_t(\hat{\boldsymbol{x}}_k) {}^{t}\boldsymbol{a}_{t,k} + \boldsymbol{g} \tag{21}$$

with $\boldsymbol{g} = [0, 0, -g]^T, g = 9.81\text{m/s}^2$ and with:

$${}^{w}\boldsymbol{R}_t(\hat{\boldsymbol{x}}_k) = \boldsymbol{R}_z(\gamma)\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\alpha) \tag{22}$$

where α, β and γ are, respectively, the roll, the pitch and the yaw angles of $\hat{\boldsymbol{x}}_k$.

2.4 Gyroscope Integration

In a similar way, the gyroscope data can be integrated to obtain the orientation of the robot $\boldsymbol{o}_t$ at each timestep. Considering a constant angular velocity in an interval $[t_k, t_{k+1})$:

$$\boldsymbol{o}_{t,k+1} = \boldsymbol{o}_{t,k} + T\dot{\boldsymbol{o}}_{t,k} \tag{23}$$

Note that the implementation of the gyroscope returns, at each timestep k, the angular velocity ${}^{t}\boldsymbol{\omega}_{t,k}$ expressed in the current reference frame of the torso, hence, it must be transformed to the reference frame of the world in order to be able to perform the integration:

$$\dot{\boldsymbol{o}}_{t,k} = \boldsymbol{T}(\hat{\boldsymbol{x}}_k) {}^{t}\boldsymbol{\omega}_{t,k} \tag{24}$$

with:

$$\boldsymbol{T}(\hat{\boldsymbol{x}}_k) = \begin{pmatrix} \cos(\gamma)/\cos(\beta) & \sin(\gamma)/\cos(\beta) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ \cos(\gamma)\tan(\beta) & \sin(\gamma)\tan(\beta) & 1 \end{pmatrix} \tag{25}$$

where β and γ are, respectively, the pitch and the yaw angles of $\hat{\boldsymbol{x}}_k$.

2.5 Trilateration

Let's now discuss how trilateration [2] can be used with a RGBD camera in order to obtain the position of the torso $\boldsymbol{p}_t$. We will find the position $\boldsymbol{p}_h$ of the camera in order to obtain the position of the torso.

First, it's important to have landmarks that can be easily recognizable. To achieve this, we have added to the scene $n = 6$ spheres of different colors. The camera of the robot is pointing towards the sky in order to simplify the recognition of the spheres themselves avoiding the colors of the terrain. Moreover, the specular, the emissive and the auxiliary components of the spheres are the same as their ambient component in order to avoid shades. This allows to have pixels of the same color when seeing a sphere. It is, hence, possible to obtain the pixel coordinates $(p_x, p_y)^T$ of each sphere in the image of the camera by computing the centroid (a simple weighted mean in our implementation) of the sphere itself.

Let w and h be the width and the height of the images coming from the RGB camera. Let λ be the focal length (length to the nearest clipping plane), λ_f be the length to the furthest clipping plane and ϕ be the perspective angle of the camera. It is possible to obtain the position (expressed with respect to the reference frame of the camera) of each sphere with centroid at coordinates $(p_x, p_y)^T$ by simply considering the proportion between the ratio of the pixel coordinates (translated so that the origin is at $(0, 0)^T$) and half the size of the image (in particular width when computing the x component, height when computing the y component), and the ratio between the coordinates of the sphere (in camera frame) and the distance between the considered point with the axes x and y of the camera frame. This results in a simple computation:

$${}^{cam}x = \frac{2p_x - w}{w} {}^{cam}z \cdot tan\left(\frac{\phi}{2}\right) \tag{26}$$

$${}^{cam}y = \frac{2p_y - h}{h} {}^{cam}z \cdot tan\left(\frac{\phi}{2}\right) \tag{27}$$

with $({}^{cam}x, {}^{cam}y, {}^{cam}z)^T$ coordinates of the considered point in the camera frame. Note that ${}^{cam}z$ can be computed by considering the distance from the clipping planes and the depth p_z of the object at position $(p_x, p_y)^T$, which can be obtained by using the depth sensor of the RGBD camera:

$${}^{cam}z = (\lambda_f - \lambda) \cdot p_z + \lambda \tag{28}$$

Moreover, note that the depth sensor returns a value between 0 and 1. In particular, it returns 0 if a point is exactly on the nearest clipping plane, 1 if a point is exactly on the furthest clipping plane.

Given that each sphere has radius $\bar{r}$, it is possible to obtain the distance from the camera to each sphere i by computing:

$$r_i = \left\| \begin{pmatrix} {}^{cam}x_i \\ {}^{cam}y_i \\ {}^{cam}z_i \end{pmatrix} \right\|^2 + \bar{r} \quad (i = 1, 2, \ldots, n) \tag{29}$$

where the subscript i specifies the index of the sphere. Note that adding $\bar{r}$ to the norm is an approximation of the real distance since the centroid could not be along the line passing through the position of the camera and the center of the sphere (e.g. when a sphere is only partially visible because occluded by another sphere or clipped).

Since the position of the spheres $(x_i, y_i, z_i)^T$ is known and it is possible to associate the distance r_i to each sphere because of the unique color, the problem of finding the position of the camera $\boldsymbol{p}_h$ is reduced to solving the following system of equations:

$$(p_{h,x} - x_i)^2 + (p_{h,y} - y_i)^2 + (p_{h,z} - z_i)^2 = r_i^2 \quad (i = 1, 2, \ldots, n) \tag{30}$$

which can we rewritten as a linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where:

$$\boldsymbol{A} = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ \vdots & \vdots & \vdots \\ x_n - x_1 & y_n - y_1 & z_n - z_1 \end{pmatrix}, \quad \boldsymbol{x} = \boldsymbol{p}_h - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} b_{21} \\ b_{31} \\ \vdots \\ b_{n1} \end{pmatrix} \tag{31}$$

and each element of the vector b is defined as:

$$b_{k1} = \frac{1}{2}\left[r_1^2 + r_k^2 + (x_k - x_1)^2 + (y_k - y_1)^2 + (z_k - z_1)^2\right] \quad (k = 2, 3, \ldots, n) \tag{32}$$

Hence, the position of the camera can be determined by:

$$\boldsymbol{p}_h = \boldsymbol{x} + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \tag{33}$$

At this point, the position of the torso can be obtained by a simple transformation:

$$\boldsymbol{p}_t = \boldsymbol{p}_h - {}^w\boldsymbol{R}_t(\hat{\boldsymbol{x}}_k)({}^t\boldsymbol{p}_h - {}^t\boldsymbol{p}_t) \tag{34}$$

with ${}^t\boldsymbol{p}_h - {}^t\boldsymbol{p}_t$ constant and known since the camera has been defined as child of the torso in the hierarchical model of the robot in order to simplify the computation of the direct kinematics.

3 MPC Loop Closure

Given the estimate of the pose of the torso $(\hat{\boldsymbol{p}}_t, \hat{\boldsymbol{o}}_t)^T$, it is possible to obtain the estimate of the pose of the support foot $(\hat{\boldsymbol{p}}_s, \hat{\boldsymbol{o}}_s)^T$ by computing:

$$\begin{pmatrix} \hat{\boldsymbol{p}}_s \\ \hat{\boldsymbol{o}}_s \end{pmatrix} = \boldsymbol{\Omega} \left(\boldsymbol{\Omega}^{-1} \begin{pmatrix} \hat{\boldsymbol{p}}_t \\ \hat{\boldsymbol{o}}_t \end{pmatrix} \boldsymbol{\Omega}^{-1} \begin{pmatrix} {}^t\boldsymbol{p}_s \\ {}^t\boldsymbol{o}_s \end{pmatrix} \right) \tag{35}$$

Note that, since in the C++ framework of the HRP4 the pose of the support foot expressed with respect to the torso is not available, we compute it in the following way:

$$\boldsymbol{\Omega}^{-1} \begin{pmatrix} {}^t\boldsymbol{p}_s \\ {}^t\boldsymbol{o}_s \end{pmatrix} = \left[\boldsymbol{\Omega}^{-1} \begin{pmatrix} {}^s\boldsymbol{p}_t \\ {}^s\boldsymbol{o}_t \end{pmatrix} \right]^{-1} \tag{36}$$

At this point it is possible to compute the position the estimate of a reference frame attached to the CoM by computing:

$$\begin{pmatrix} \hat{\boldsymbol{p}}_{CoM} \\ \hat{\boldsymbol{o}}_{CoM} \end{pmatrix} = \boldsymbol{\Omega} \left(\boldsymbol{\Omega}^{-1} \begin{pmatrix} \hat{\boldsymbol{p}}_s \\ \hat{\boldsymbol{o}}_s \end{pmatrix} \boldsymbol{\Omega}^{-1} \begin{pmatrix} {}^s\boldsymbol{p}_{CoM} \\ {}^s\boldsymbol{o}_{CoM} \end{pmatrix} \right) \tag{37}$$

Note that these steps are necessary since, as before, in the C++ framework of the HRP4 it is not possible to directly express a reference frame attached to the CoM with respect to the torso.

Given $\hat{\boldsymbol{p}}_s$, $\hat{\boldsymbol{p}}_{CoM}$ and the yaw angle γ of $\hat{\boldsymbol{x}}$, it is possible to express the position of the CoM in a rotated reference frame of the support foot $\mathcal{F}_{s'}$ which has the z-axis orthogonal to the floor:

$${}^{s'}\hat{\boldsymbol{p}}_{CoM} = \boldsymbol{R}_z^T(\gamma)(\hat{\boldsymbol{p}}_{CoM} - \hat{\boldsymbol{p}}_s) \tag{38}$$

The position of the CoM ${}^{s'}\hat{\boldsymbol{p}}_{CoM}$ can be used as a reference signal in the MPC in order to generate the gait of the humanoid robot.

4 Regulation

Another way to test the correctness of the EKF is to make the robot perform a regulation task, hence, make it move to a certain configuration $(x_g, y_g, \theta_g)^T$. Note that the problem can be simplified by modeling the robot as a unicycle and by considering its coordinates in a reference frame $\mathcal{F}_g$ fixed at a position $(x_g, y_g)^T$ and rotated by θ_g around the reference frame of the world. In this way, it is possible to express the generalized coordinates of the robot in $\mathcal{F}_g$ by computing:

$$\begin{pmatrix} {}^g x \\ {}^g y \\ {}^g\theta \end{pmatrix} = \boldsymbol{R}_z^T(\theta_g) \begin{pmatrix} x - x_g \\ y - y_g \\ \theta - \theta_g \end{pmatrix} \tag{39}$$

4.1 Proportional Controller

Let's first make the robot move to a desired position with any orientation θ_g developing a proportional controller. Let's define the kinematic model of the unicycle:

$$\dot{x} = v\cos(\theta) \tag{40}$$

$$\dot{y} = v\sin(\theta) \tag{41}$$

$$\dot{\theta} = \omega \tag{42}$$

with v and w respectively linear and angular velocity of the unicycle. A simple way to obtain the desired behaviour is to control the linear velocity v by making it decrease while moving closer to the goal and to control the angular velocity ω so that the unicycle will be oriented towards the goal:

$$v = k_1 \left\| \boldsymbol{p}_g - \hat{\boldsymbol{p}}_t \right\| \tag{43}$$

$$\omega = k_2 e_\theta \tag{44}$$

where $\boldsymbol{p}_g$ is the desired position, $\hat{\boldsymbol{p}}_t$ is the estimated position $(x, y)^T$ of the torso and e_θ is the angle between the sagittal vector of the unicycle and the vector pointing from the unicycle towards the goal:

$$e_\theta = \operatorname{sgn}\left(\begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \times (\boldsymbol{p}_g - \hat{\boldsymbol{p}}_t) \right) \cdot \operatorname{acos}\left(\frac{\begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}^T (\boldsymbol{p}_g - \hat{\boldsymbol{p}}_t)}{\left\| \boldsymbol{p}_g - \hat{\boldsymbol{p}}_t \right\|} \right) \tag{45}$$

Of course, since the humanoid robot has an oscillatory motion due to the gait, we decided to set the velocities v and w to 0 when the distance $\left\| \boldsymbol{p}_g - \hat{\boldsymbol{p}}_t \right\|$ from the desired position $(x_g, y_g)^T$ is less than a certain threshold (set to 0.25 in our experiment).

4.2 Cartesian Regulation

Another way to achieve the same goal is to exploit the reference frame $\mathcal{F}_g$ defined above in order to drive the unicycle to a configuration $(0, 0)^T$ in $\mathcal{F}_g$ itself. Let's use the same kinematic model defined above and consider the following control law:

$$v = -k_1({}^g x\cos({}^g\theta) + {}^g y\sin({}^g\theta)) \tag{46}$$

$$\omega = k_2(\operatorname{Atan2}({}^g y, {}^g x) - {}^g\theta + \pi) \tag{47}$$

The robot will move to $(0, 0)^T$ in $\mathcal{F}_g$ and hence to the desired position $\boldsymbol{p}_g$ with any orientation. As before, since the humanoid robot has an oscillatory

motion due to the gait, we decided to set the velocities v and w to 0 when the distance $\|\boldsymbol{p}_g - \hat{\boldsymbol{p}}_t\|$ from the desired position $(x_g, y_g)^T$ is less than a certain threshold (set to 0.2 in our experiment).

4.3 Posture Regulation

Let's now make the robot move to a desired configuration specifying the orientation θ_g as well. Let's define the kinematic model of the unicycle using polar coordinates:

$$\dot{\rho}_r = -v\cos(\gamma_r) \tag{48}$$

$$\dot{\gamma}_r = \frac{\sin(\gamma_r)}{\rho_r} v - \omega \tag{49}$$

$$\dot{\delta}_r = \frac{\sin(\gamma_r)}{\rho_r} v \tag{50}$$

with v and w respectively linear and angular velocity of the unicycle. The polar coordinates can be obtained from the generalized coordinates of the unicycle $(x, y, \theta)^T$ by computing:

$$\rho_r = \sqrt{{}^g x^2 + {}^g y^2} \tag{51}$$

$$\gamma_r = \text{Atan2}({}^g y, {}^g x) - {}^g\theta + \pi \tag{52}$$

$$\delta_r = \gamma_r + {}^g\theta \tag{53}$$

By considering the following control law:

$$v = k_1 \rho_r \cos(\gamma_r) \tag{54}$$

$$w = k_2 \gamma_r + k_1 \frac{\sin(\gamma_r)\cos(\gamma_r)}{\gamma_r}(\gamma_r + k_3 \delta_r) \tag{55}$$

it is possible to prove [3] that ρ_r, γ_r and δ_r converge to zero, which implies that the configuration of the unicycle $(x, y, \theta)^T$ converges to the desired configuration $(x_g, y_g, \theta_g)^T$. Again, since the humanoid robot has an oscillatory motion due to the gait, we decided to set the velocities v and w to 0 when the distance ρ_r from the desired position $(x_g, y_g)^T$ is less than a certain threshold (set to 0.2 in our experiment).

5 Experiments

We tested our EKF implementation doing multiple experiments. Initially, we considered only the IMU measurements without achieving convergence. Then, we improved the estimate by filtering the linear velocities of the robot, correctly localizing it in the environment. We equipped the robot with a RGBD testing our trilateration algorithm, improving the estimate of the pose of the torso. We

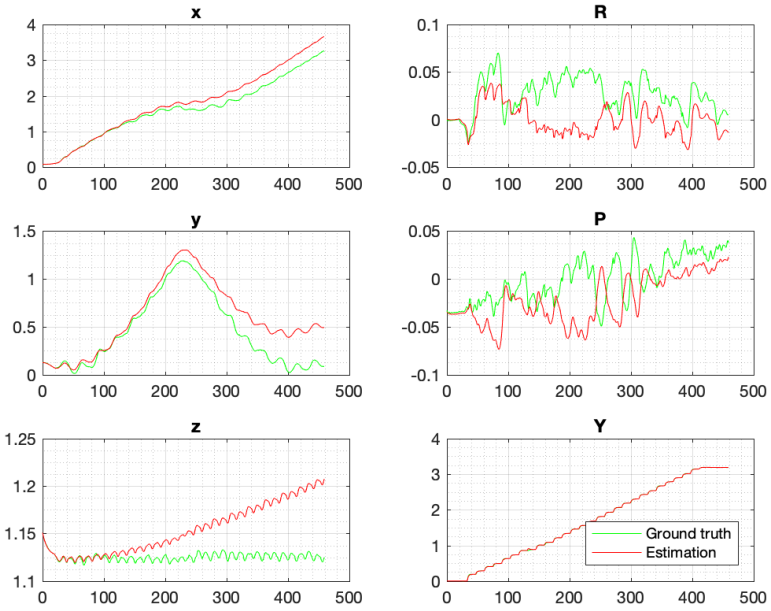

Figure 1: Comparison between the ground truth (in green) and the estimate (in red) of the torso pose when using a double integrator on the data of the accelerometer and the gyroscope. x, y, z stands for the position of the torso expressed in the reference frame of the world. R, P, Y stands for the rotations roll, pitch, yaw around the respective axes of the reference frame of the world.

made the robot perform a cartesian and a posture regulation task using the estimate of the EKF, correctly reaching the desired configuration. In the end we tried to close the feedback loop in the MPC gait generator making the robot move a few steps before falling.

5.1 Inertial Measurement Unit

The first test mainly focused on the integration of the data coming from the accelerometer and the gyroscope (IMU). Considering the EKF described in subsection 2.2, we defined the measurement function (Eq. 4) at each timestep k as:

$$\boldsymbol{y}_k = \boldsymbol{h}(\hat{\boldsymbol{x}}_k, \boldsymbol{q}_{n,k}) = \begin{pmatrix} \hat{\boldsymbol{p}}_{t,k} \\ \hat{\boldsymbol{o}}_{t,k} \end{pmatrix} \tag{56}$$

where $\boldsymbol{p}_{t,k}$ and $\boldsymbol{o}_{t,k}$ have been obtained by integrating the data of the IMU, hence, using Equations 20 and 23. Results are shown in Fig. 1. As it is possible to see from the plots, while the orientation has good results, the estimates of the position are not converging to the real value.

Camera Params.	Description
$w = 512$	width of the image
$h = 512$	height of the image
$\lambda = 0.01[m]$	nearest clipping plane
$\lambda_f = 10.0[m]$	furthest clipping plane
$\phi = \pi/3[rad]$	perspective angle

Table 1: Parameters of the RGBD camera used to do trilateration.

The main reason why the method is not working is because the integration of the linear acceleration is subject to drift, hence, the velocity term computed in Eq. 19 is accumulating an error at each timestep.

5.2 Filtering Linear Velocities

One way to fix the problem of drift accumulation on the integration of the acceleration is to add the linear velocity to the EKF. Nevertheless, since this requires an observation function for the linear velocity and since the velocity from a timestep to the following one are similar, we decided to approximate it using the difference of the position of the robot between two successive timesteps, which are already filtered:

$$\dot{\boldsymbol{p}}_{t,k} \approx \dot{\hat{\boldsymbol{p}}}_{t,k} \approx \dot{\hat{\boldsymbol{p}}}_{t,k-1} = \frac{\hat{\boldsymbol{p}}_{t,k} - \hat{\boldsymbol{p}}_{t,k-1}}{T} \tag{57}$$

This approximation, which has been used in Eq. 20 in our implementation,, improves the previous results (Fig. 2), estimating correctly the coordinates $(x, y)^T$ and reducing the error on z. Even if the error on z has been improved, it does not seem to be converging like the other two coordinates. This suggests that another approach is needed in order to further improve the estimate.

5.3 Trilateration

In our third experiment we equipped the robot with a RGBD camera positioned on top of its head and we added 6 spheres of different colors to the scene as already explained in subsection 2.5. The parameters of the camera are those defined in Table 1. All the spheres have radius $\bar{r} = 0.05$.

We then redefined the measurement function so that the position of the torso observed by the robot is the one defined by Eq. 34. Fig. 3 shows the results obtained when doing trilateration for the measurement of the position. It is clearly possible to notice that not only the estimate of z has improved with respect to the previous experiment, but also x and y are closer to the ground truth. As in the previous experiments, the measurement of the orientation has been computed using the gyroscope.

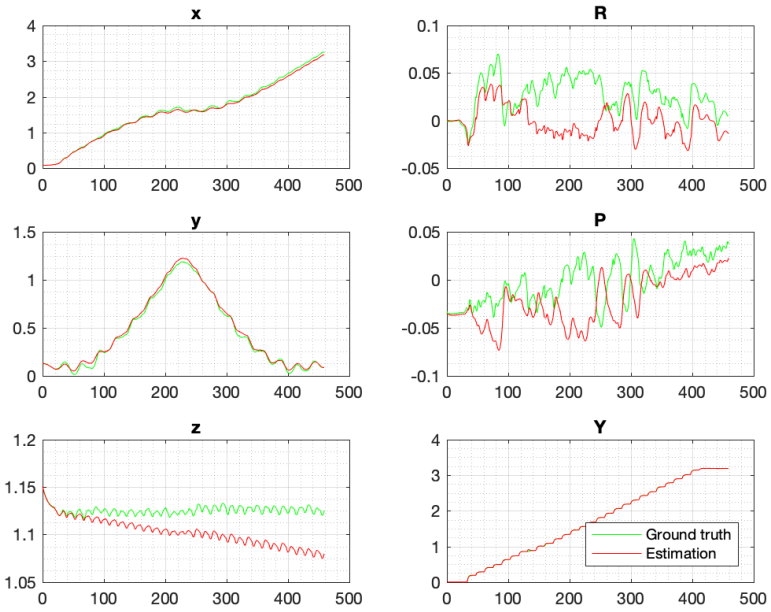

Figure 2: Comparison between the ground truth (in green) and the estimate (in red) of the torso pose when using a double integrator on the data of the accelerometer and the gyroscope. Here the velocity is approximated with the velocity of the previous step, which is in turn computed using the positions coming from the EKF. This should give an idea about what happens if the velocities are filtered as well. x, y, z stands for the position of the torso expressed in the reference frame of the world. R, P, Y stands for the rotations roll, pitch, yaw around the respective axes of the reference frame of the world.

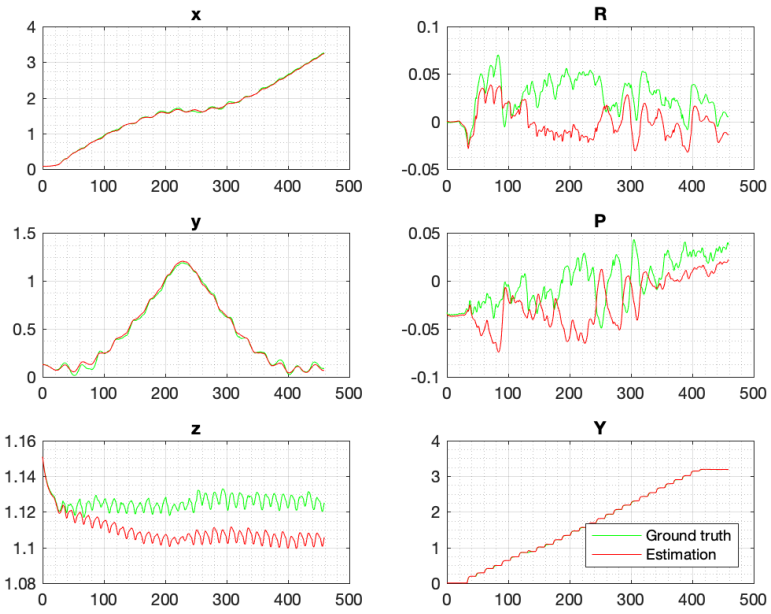

Figure 3: Comparison between the pose of the torso estimated with the EKF (in red) with respect to the ground truth (in green) when using trilateration and the data coming from the gyroscope. x, y, z stands for the position of the torso expressed in the reference frame of the world. R, P, Y stands for the rotations roll, pitch, yaw around the respective axes of the reference frame of the world.

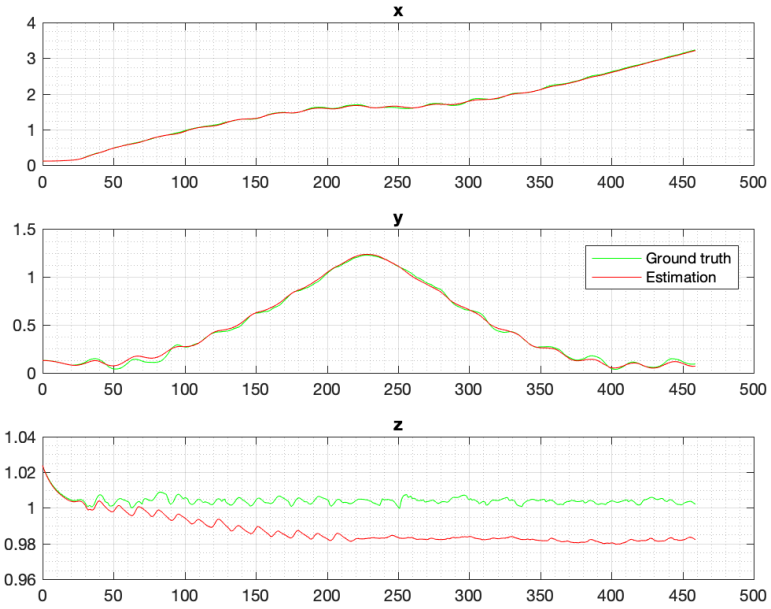

Figure 4: Comparison between the estimate of the position of the CoM obtained with the EKF (in red) and the ground truth of the CoM (in green) when using trilateration and the gyroscope. x, y, z stands for the position of the torso expressed in the reference frame of the world. R, P, Y stands for the rotations roll, pitch, yaw around the respective axes of the reference frame of the world.

5.4 MPC Loop Closure

Given the estimate of the pose of the torso $\hat{\boldsymbol{x}}$ we checked whether it could be used to estimate also the position of the CoM of the robot $\hat{\boldsymbol{p}}_{CoM}$ with respect to the world frame (Eq. 37), correctly converging to the ground truth. The results are shown in Fig. 4.

We then expressed the position of the CoM in a rotated reference frame of the support foot $\mathcal{F}_{s'}$, as described in section 3, without achieving the desired behaviour. The robot, in fact, falls after a few steps even when using a very low weight[1] in the MPC implementation.

5.5 Proportional Controller

We tested the behaviour of the EKF when the robot is performing cartesian regulation (subsection 4.1). We modeled the HRP4 as a unicycle (Equations 40-42) and we made it move from its initial configuration $\boldsymbol{q}_s$ to a final

[1]The weight establishes the importance of the estimate with respect to the optimal value.

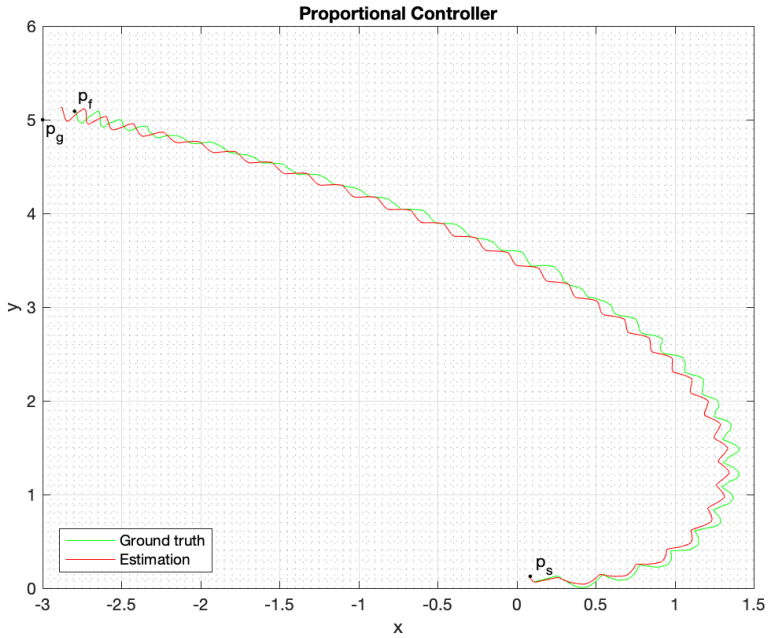

Figure 5: Trajectory followed by the robot when moving to a desired configuration $\boldsymbol{q}_g = (-3, 5, \cdot)^T$ using the proportional controller described in subsection 4.1. Here, the measurements of the Kalman filter are obtained using the accelerometer and the gyroscope. In green the ground truth of the coordinates (x, y) of the robot, in red the estimate of the EKF. The robot start at the position $\boldsymbol{p}_s$ and it stops at the position $\boldsymbol{p}_f$. Note that, as explained in section 4.1, when $\|\boldsymbol{p}_g - \hat{\boldsymbol{p}}_t\| < 0.25$, v are w are set to 0.

configuration $\boldsymbol{q}_g = (-3, 5, \cdot)^T$, making it stop when $\|\boldsymbol{p}_g - \hat{\boldsymbol{p}}_t\| < 0.25$ in order to avoid problems due to the oscillation of the humanoid (Fig. 5).

The control law defined by Equations 43-44, when using gains $k_1 = 0.18$, $k_2 = 0.014$ makes the robot stop at the final configuration $\boldsymbol{q}_f = (-2.798, 5.090, 0.98\pi)^T$, near the estimate $\hat{\boldsymbol{q}}_f = (-2.885, 5.127, 0.979\pi)^T$ and near the desired final configuration $\boldsymbol{q}_g$.

5.6 Cartesian Regulation

We then tested cartesian regulation (subsection 4.2) on a different control law, modelling the HRP4 as in the previous experiment. We made the robot move from its initial configuration $\boldsymbol{q}_s$ to a final configuration $\boldsymbol{q}_g = (-2, 3.2, \cdot)^T$, making it stop when $\|\boldsymbol{p}_g - \hat{\boldsymbol{p}}_t\| < 0.2$ as it is possible to see from Fig. 6.

The control law defined by Equations 46-47, when using gains $k_1 = 0.07$, $k_2 = 0.01$ makes robot stops at the final configuration $\boldsymbol{q}_f =$

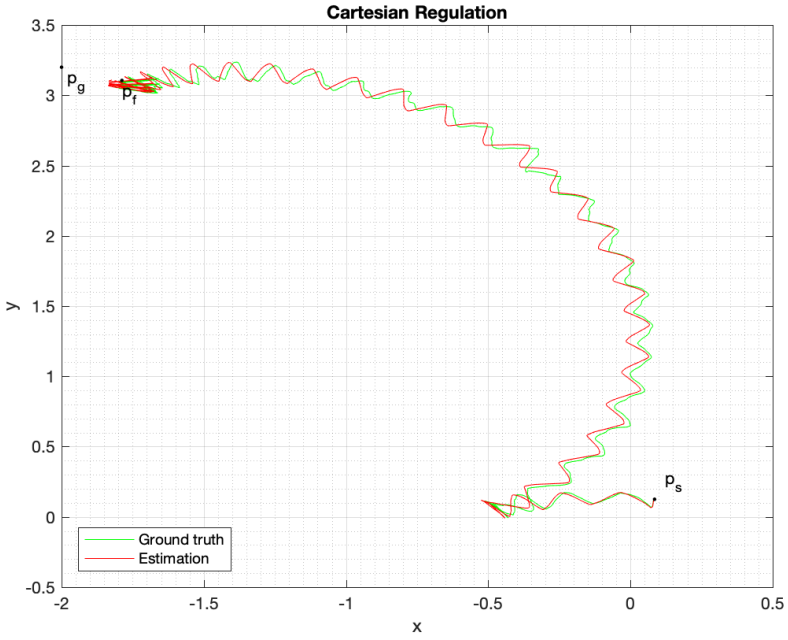

Figure 6: Trajectory followed by the robot when moving to a desired configuration $\boldsymbol{q}_g = (-2, 3.2, \cdot)^T$ using the cartesian regulation task described in subsection 4.2. Here, the measurements of the Kalman filter are obtained using the accelerometer and the gyroscope. In green the ground truth of the coordinates (x, y) of the robot, in red the estimate of the EKF. The robot start at the position $\boldsymbol{p}_s$ and it stops at the position $\boldsymbol{p}_f$. Note that, as explained in section 4.3, when $\|\boldsymbol{p}_g - \hat{\boldsymbol{p}}_t\| < 0.2$, v are w are set to 0.

$(-1.788, 3.105, 1.562\pi)^T$, near the estimate $\hat{\boldsymbol{q}}_f = (-1.823, 3.108, 1.562\pi)^T$ and near the desired final configuration $\boldsymbol{q}_g$. It is interesting to notice that the robot starts oscillating on the spot when it reaches a configuration which is close to the desired one.

5.7 Posture Regulation

In our last experiment we tested the behaviour of the EKF when the robot is performing posture regulation (subsection 4.3). We modeled, again, the HRP4 as a unicycle and we made it move from its initial configuration $\boldsymbol{q}_s$ to a final configuration $\boldsymbol{q}_g = (-2.0, 3.2, \pi)^T$, making it stop when $\delta_r < 0.2$ in order to avoid problems due to the oscillation of the humanoid.

As it is possible to see from Figures 7-8, the robot is able to move to the desired configuration by first going backward (keeping the same orientation) and then by simultaneously moving forward and rotating until the desired final orientation is reached.

The control law defined by Equations 54-55, when using gains $k_1 = 0.1$, $k_2 = 0.007$ and $k_3 = 0.004$, generates the velocities v and w shown in Fig. 9. The robot stops at the final configuration $\boldsymbol{q}_f = (-1.797, 3.194, 1.024\pi)^T$, near the estimate $\hat{\boldsymbol{q}}_f = (-1.824, 3.222, 1.024\pi)^T$ and near the desired final configuration $\boldsymbol{q}_g$. It is interesting to notice from the plots how the lack of the correction step (given by the exteroceptive sensors) of the Kalman filter, namely the odometry, makes the robot move to a configuration $(-1.282, 3.424, 0.912\pi)^T$ which is far away from the desired one.

6 Conclusion

In this project we developed an Extended Kalman Filter to localize the HRP4 humanoid robot in a V-REP environment. We equipped the robot with a gyroscope, an accelerometer and a RGBD camera, considering different approaches in order to make the estimate as accurate as possible. We first tested the EKF when using only IMU measurements, not achieving convergence. We then filtered the linear velocities of the robot improving the estimate and correctly localizing it in the environment. Then we developed a trilateration algorithm and used a RGBD camera further improving the estimate. We validated our implementation on cartesian and posture regulation tasks, correctly achieving the desired configuration. In the end we used the estimate of the CoM of the robot to close the feedback loop of the MPC gait generator but we did not achieved the desired behaviour, suggesting that a deeper study is needed in the implementation of the MPC.

The behaviour of the EKF could be further improved by fusing the sensors used in the project. Moreover, developing a proper VSLAM system could make

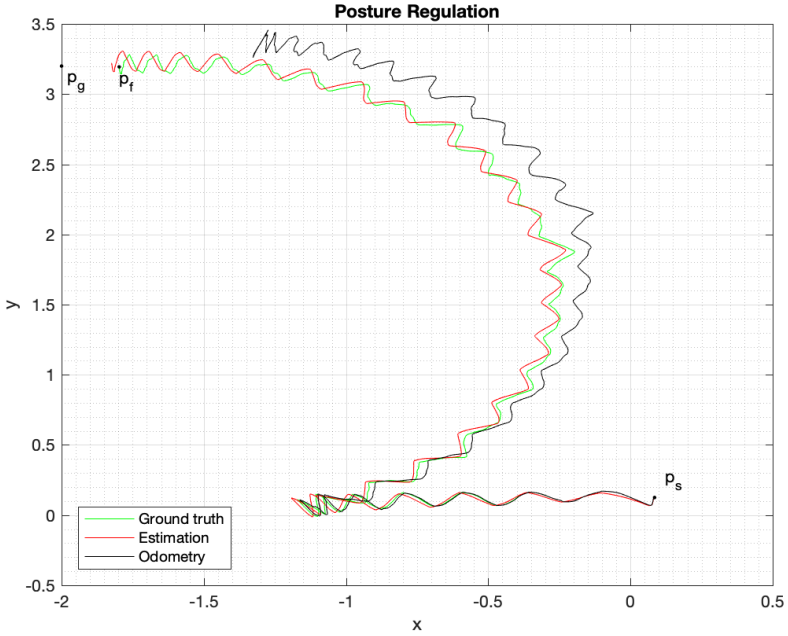

Figure 7: Trajectory followed by the robot when moving to a desired configuration $\boldsymbol{q}_g = (-2, 3.2, \pi)$ while performing the posture regulation task described in subsection 4.3. Here, the measurements of the Kalman filter are obtained using the accelerometer and the gyroscope. In green the ground truth of the coordinates (x, y) of the robot, in red the estimate of the EKF, in black the odometry. The robot start at the position $\boldsymbol{p}_s$ and it stops at the position $\boldsymbol{p}_f$. Note that, as explained in section 4.3, when $\rho_r < 0.2$, v are w are set to 0.

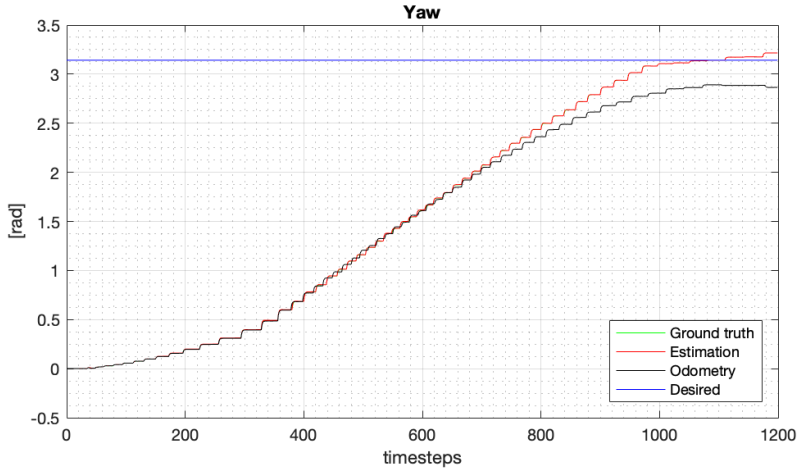

Figure 8: Orientation followed by the robot when moving to a desired configuration $\boldsymbol{q}_g = (-2, 3.2, \pi)^T$ while performing the posture regulation task described in subsection 4.3. Here, the measurements of the Kalman filter are obtained using the accelerometer and the gyroscope. In green the ground truth of the coordinate θ of the robot (unicycle), in red the estimate of the EKF (note that they coincide for the whole path). In black the odometry, in blue the desired final orientation $\theta_g = \pi$.

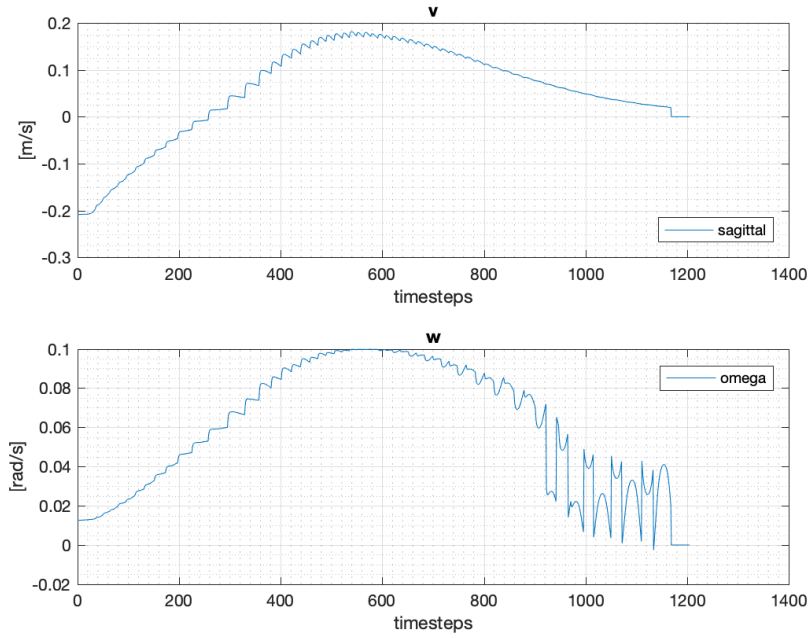

Figure 9: Linear and angular velocity of the HRP4 while performing the posture regulation task described in subsection 4.3. Note that when $\rho_r < 0.2$, v are w are set to 0.

the robot localize in more complex environments, enabling advanced behaviours and increasing the chances of making the MPC work correctly.

References

[1] G. Oriolo, A. Paolillo, L. Rosa, and M. Vendittelli, "Humanoid odometric localization integrating kinematic, inertial and visual information," *Auton. Robots*, vol. 40, no. 5, pp. 867–879, 2016.

[2] W. Hereman and J. William S. Murphy, "Determination of a Position in Three Dimensions Using Trilateration and Approximate Distances," 1995.

[3] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st ed., 2008.