# Torso Pose Estimation on the HRP4 Humanoid Robot (draft)

Michele Cipriano, Godwin Joey, Lorenzo Vianello
Supervised by Nicola Scianca

January 10, 2019

**Abstract**

Todo.

## 1 Introduction

General overview of the project, what has been implemented, etc [1].

## 2 Torso Pose Estimation

Brief introduction to the section.

### 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 $\mathcal{F}_s$:

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

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

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 $\mathbf{0} \in \mathbb{R}^3$ 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 $\boldsymbol{v}_k \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{V}_k)$ and $\boldsymbol{w}_k \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{W}_k)$ 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

$$\begin{aligned} \boldsymbol{G}_{k+1} &= \boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^T \left(\boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^T + \boldsymbol{W}_{k+1}\right)^{-1} && (12)\\ &= \boldsymbol{P}_{k+1|k}(\boldsymbol{P}_{k+1|k} + \boldsymbol{W}_{k+1})^{-1} && (13) \end{aligned}$$

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:

$$\begin{aligned} \boldsymbol{P}_{k+1} &= \boldsymbol{P}_{k+1|k} - \boldsymbol{G}_{k+1}\boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k} && (15)\\ &= \boldsymbol{P}_{k+1|k} - \boldsymbol{G}_{k+1}\boldsymbol{P}_{k+1|k} && (16) \end{aligned}$$

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

$$\begin{aligned} \boldsymbol{V} &= \mathrm{diag}\{5, 5, 5, 100, 100, 100\} \cdot 10^{-6} && (17)\\ \boldsymbol{W} &= \mathrm{diag}\{5, 5, 5, 5 \cdot 10^{-2}, 5 \cdot 10^{-2}, 5 \cdot 10^{-2}\} \cdot 10^{-2} && (18) \end{aligned}$$

## 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 $[t_k, t_{k+1})$:

$$\begin{aligned} \dot{\boldsymbol{p}}_{t,k+1} &= \dot{\boldsymbol{p}}_{t,k} + \ddot{\boldsymbol{p}}_{t,k}T && (19)\\ \boldsymbol{p}_{t,k+1} &= \boldsymbol{p}_{t,k} + \dot{\boldsymbol{p}}_{t,k}T + \frac{1}{2}\ddot{\boldsymbol{p}}_{t,k}T^2 && (20) \end{aligned}$$

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:

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

with:

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

where $\alpha$, $\beta$ and $\gamma$ 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 $\beta$ and $\gamma$ 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 $\lambda$ be the focal length (length to the nearest clipping plane), $\lambda_f$ be length to the furthest clipping plane and $\phi$ 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:

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

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:

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

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

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

where each element of the vector $b$ is defined as:

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

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

[TODO: DESCRIPTION OF THE MPC AS DONE ABOVE WITH EKF].

## 4 Posture Regulation

Another way to test the correctness of the EKF is to make the robot perform the posture 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 $\theta_g$ 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:

$$\begin{pmatrix} {}^g x \\ {}^g y \\ {}^g \theta \end{pmatrix} = \boldsymbol{R}_z^T(\theta_g) \begin{pmatrix} x - x_g \\ y - y_g \\ \theta - \theta_g \end{pmatrix} \tag{35}$$

Let's define kinematic model of the unicycle using polar coordinates:

$$\dot{\rho}_r = -v cos(\gamma_r) \tag{36}$$

$$\dot{\gamma}_r = \frac{sin(\gamma_r)}{\rho} v - w \tag{37}$$

$$\dot{\delta}_r = \frac{sin(\gamma_r)}{\rho} v \tag{38}$$

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:

$$\rho_r = \sqrt{{}^g x^2 + {}^g y^2} \tag{39}$$

$$\gamma_r = \text{Atan2}({}^g y, {}^g x) - {}^g \theta + \pi \tag{40}$$

$$\delta_r = \gamma_r + {}^g \theta \tag{41}$$

By considering the following control law:

$$v = k_1 \rho_r cos(\gamma_r) \tag{42}$$

$$w = k_2 \gamma_r + k_1 \frac{sin(\gamma_r) cos(\gamma_r)}{\gamma_r} (\gamma_r + k_3 \delta_r) \tag{43}$$

it is possible to prove [3] that $\rho_r$, $\gamma_r$ and $\delta_r$ converge to zero, which implies that the configuration of the unicycle $(x, y, \theta)^T$ converges to the desired configuration $(x_g, y_g, \theta_g)^T$. 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 $\rho_r$ from the desired position $(x_g, y_g)^T$ is less than a certain threshold (set to 0.2 in our experiment).

## 5 Experiments

[TODO: DESCRIBE PROBLEMS WITH DOUBLE INTEGRATION OF THE ACCELEROMETER].

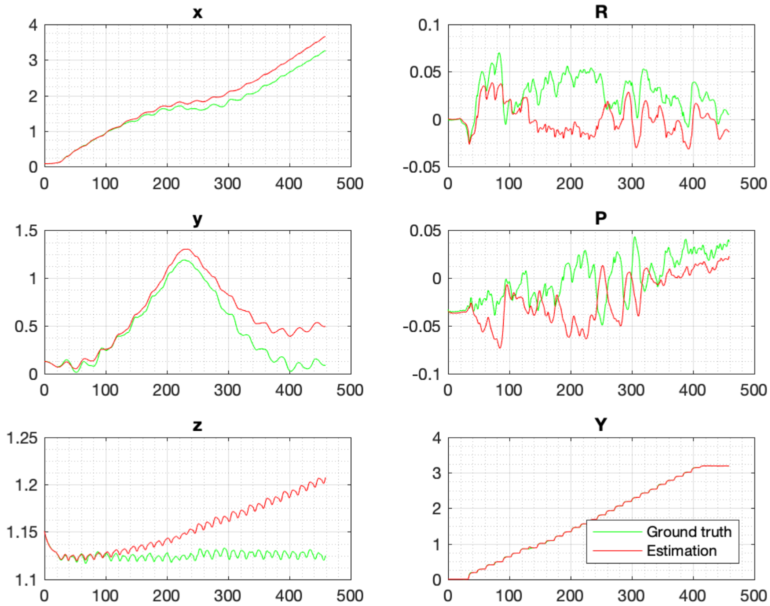


Figure 1: Comparison between the ground truth (in green) and the estimation (in red) of the torso pose when using a double integrator on the data of the accelerometer and the gyroscope. $x$, $y$, $z$ stands for the position of the torso expressed in the reference frame of the world. $R$, $P$, $Y$ stands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

[TODO: DIVIDE THIS SECTION IN SUBSECTIONS].

[TODO: SAY WHICH ARE THE VALUES USED IN THE IMPLEMENTATION - CAMERA TRILATERATION].

Figures 3 for the torso pose estimation (`simulationType = 1`) [TODO: ADD DESCRIPTION].

[TODO: DOUBLE CHECK THAT THE TORSO POSE ESTIMATION IS ACTUALLY USEFUL TO FOLLOW THE RIGHT TRAJECTORY] (figure 5).

Figure 4 for the torso position estimate and the CoM. Is there actually a way to use this for the MPC loop closure? Note, moreover, that the implementation of the MPC is using the position of the CoM w.r.t. the reference frame of the support foot, which is independent from the pose of the torso.

# 6 Conclusions

Summary of the project, possible future developments and conclusions.

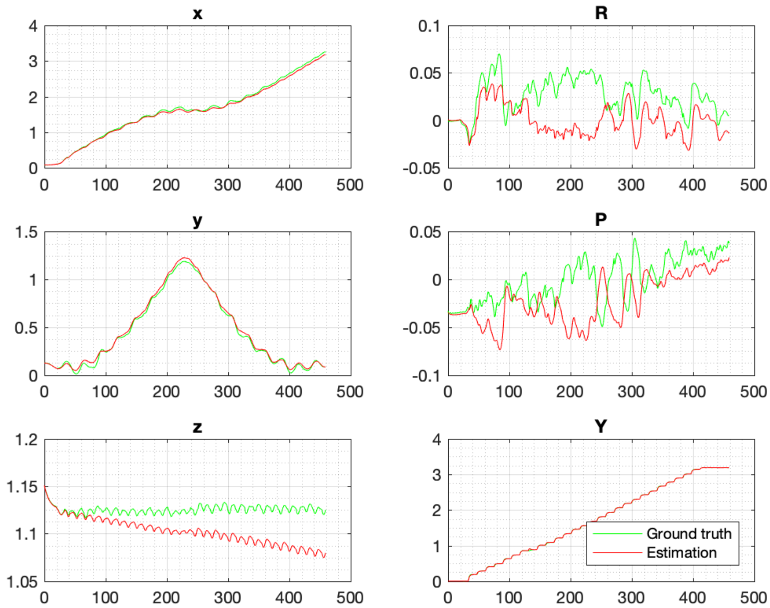


Figure 2: Comparison between the ground truth (in green) and the estimation (in red) of the torso pose when using a double integrator on the data of the accelerometer and the gyroscope. Here the velocity is approximated with the velocity of the previous step, which is in turn computed using the positions 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.

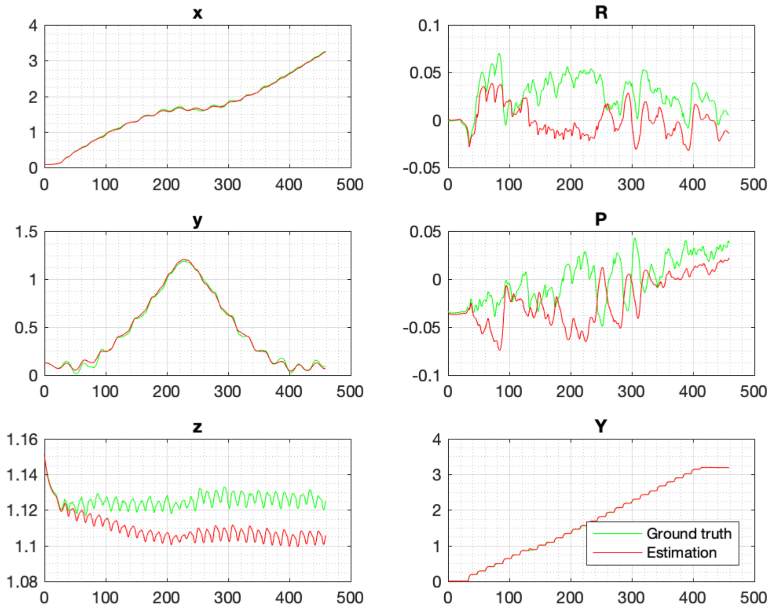


Figure 3: Comparison of the pose of the torso estimated with the EKF (in red) with respect to the ground truth (in green) when using trilateration and the data coming from the gyroscope. $x$, $y$, $z$ stands for the position of the torso expressed in the reference frame of the world. $R$, $P$, $Y$ stands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

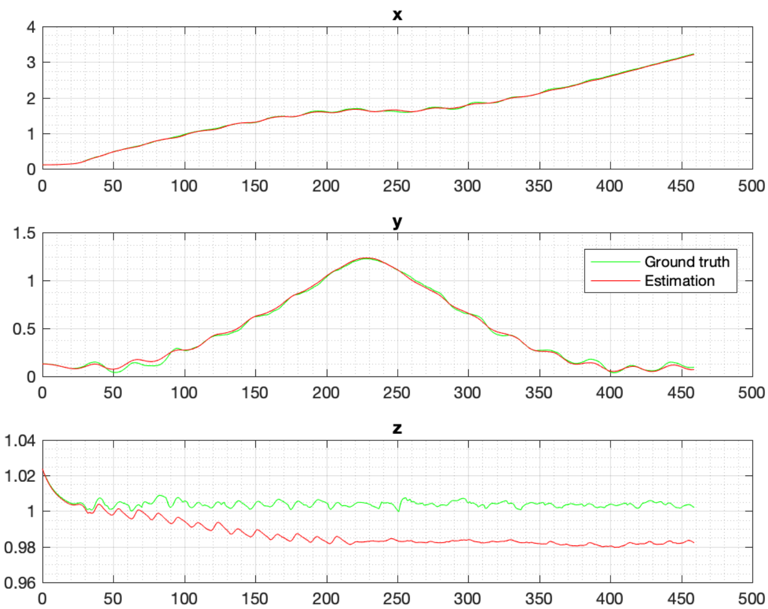


Figure 4: Comparison of the position of the torso estimated with the EKF (in red) with respect to the CoM (in green) when using trilateration and the gyroscope. $x$, $y$, $z$ stands for the position of the torso expressed in the reference frame of the world. $R$, $P$, $Y$ stands for the rotations roll,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

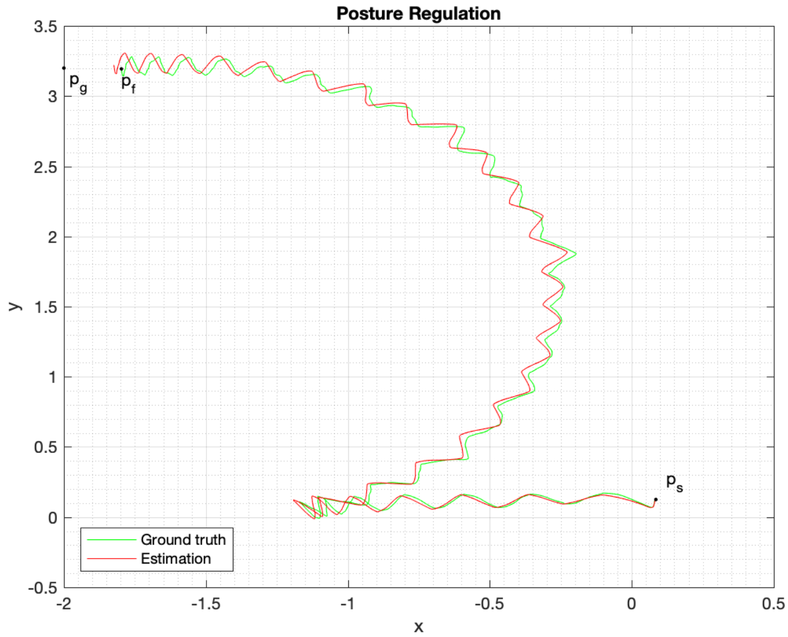


Figure 5: Trajectory followed by the robot when moving to a desired configuration $\boldsymbol{q}_g = (-2, 3.2, \pi)$. Here, the measurements of the Kalman filter are obtained using the accelerometer and the gyroscope. In green the ground truth of the coordinates $(x, y)$ of the robot, in red the estimate 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, when $\rho_r < 0.2$, $v$ are $w$ are set to 0.

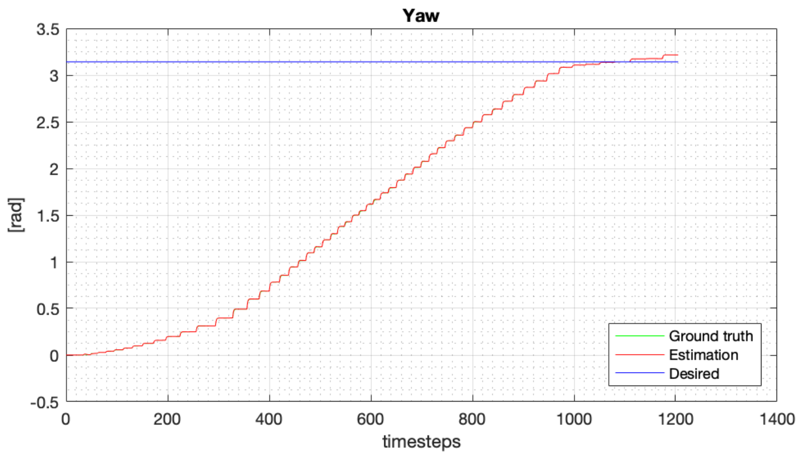


Figure 6: Orientation followed by the robot when moving to a desired configuration $\boldsymbol{q}_g = (-2, 3.2, \pi)$. Here, the measurements of the Kalman filter are obtained using the accelerometer and the gyroscope. In green the ground truth of the coordinate $\theta$ of the robot (unicycle), in red the estimate of the EKF (note that they coincide for the whole path). In blue the desired final orientation $\theta_g = \pi$.

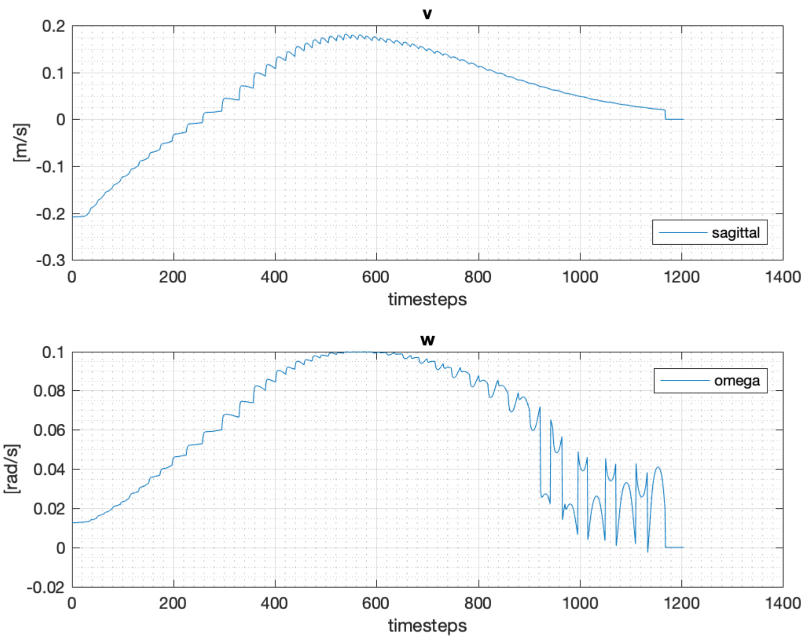


Figure 7: Linear and angular velocity of the HRP4 while performing posture regulation.

## 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.