# CoM Estimation on the HRP4 Humanoid (draft)

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

December 11, 2018

## 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 of $\boldsymbol{x}$ 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 the 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 $\mathcal{F}_s$:

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

$$
\begin{aligned}
&= \begin{pmatrix} \boldsymbol{R}_z(\boldsymbol{o}_s)\,{}^s\boldsymbol{p}_t \\ {}^s\boldsymbol{o}_{t,x} \\ {}^s\boldsymbol{o}_{t,y} \\ \boldsymbol{o}_s + {}^s\boldsymbol{o}_{t,z} \end{pmatrix} \\
&= \begin{bmatrix} \boldsymbol{R}_z(\boldsymbol{o}_s) & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \begin{pmatrix} {}^s\boldsymbol{p}_t \\ {}^s\boldsymbol{o}_t \end{pmatrix} + \begin{pmatrix} \boldsymbol{0}_5 \\ \boldsymbol{o}_s \end{pmatrix} \\
&= \begin{bmatrix} \boldsymbol{R}_z(\boldsymbol{o}_s) & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \boldsymbol{f}(\boldsymbol{q}_s) + \begin{pmatrix} \boldsymbol{0}_5 \\ \boldsymbol{o}_s \end{pmatrix} \qquad (2)
\end{aligned}
$$

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

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. In the C++ library `Eigen`, $\boldsymbol{\Omega}$ is implemented with `t2v`, while $\boldsymbol{\Omega}^{-1}$ is implemented with `v2t`.

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

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

In particular, the position $\boldsymbol{p}_t$ is computed by doing trilateration exploiting known position of the landmarks, while the orientation $\boldsymbol{o}_t$ is computed by simply integrating the data obtained from the IMU. Details are explained in subsections 2.3 and 2.4. [TODO: ADD COMMENT ABOUT NOTATION OF $q_s$ AND $q_n$ WHICH IS CONSISTENT WITH THE PAPER BUT NOT ACTUALLY USED IN THE IMPLEMENTATION].

## 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:

$$
\begin{aligned}
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + T\boldsymbol{J}(\boldsymbol{q}_{s,k}, \boldsymbol{o}_s)\dot{\boldsymbol{q}}_{s,k} + \boldsymbol{v}_k \qquad (5) \\
\boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{q}_{n,k}) + \boldsymbol{w}_k \qquad (6)
\end{aligned}
$$

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 $\hat{\boldsymbol{x}}_k$:

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

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:

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

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 [TODO]:

$$\begin{aligned} \boldsymbol{V} &= \operatorname{diag}\{\cdot\} && (17)\\ \boldsymbol{W} &= \operatorname{diag}\{\cdot\} && (18) \end{aligned}$$

## 2.3 Gyroscope Integration

Gyroscope integration (Euler).

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

Gyroscope integration (Trapezoidal).

$$\boldsymbol{o}_{t,k+1} = \boldsymbol{o}_{t,k} + T\frac{\dot{\boldsymbol{o}}_{t,k} + \dot{\boldsymbol{o}}_{t,k-1}}{2} \tag{20}$$

Note that the implementation of the gyroscope returns, at each timestep $k$, the angular velocity ${}^t\dot{\boldsymbol{o}}_{t,k}$ 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:

$$\dot{\boldsymbol{o}}_{t,k} = \left[\boldsymbol{T}_{RPY}\left(\hat{\boldsymbol{x}}_{k+1|k}\right)\right]^{-1} {}^{t}\dot{\boldsymbol{o}}_{t,k} \tag{21}$$

with:

$$\boldsymbol{T}_{RPY}\left(\hat{\boldsymbol{x}}_{k+1|k}\right) = \begin{pmatrix} cos(\beta)cos(\gamma) & -sin(\gamma) & 0 \\ cos(\beta)sin(\gamma) & cos(\gamma) & 0 \\ -sin(\beta) & 0 & 1 \end{pmatrix} \tag{22}$$

where $\beta$ and $\gamma$ are, respectively, the pitch and the yaw angles of $\hat{\boldsymbol{x}}_{k+1|k}$.

## 2.4 Trilateration

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

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:

$$^{cam}x = \frac{2p_x - w}{w} {}^{cam}z \cdot tan\left(\frac{\phi}{2}\right) \tag{23}$$

$$^{cam}y = \frac{2p_y - h}{h} {}^{cam}z \cdot tan\left(\frac{\phi}{2}\right) \tag{24}$$

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:

$$^{cam}z = (\lambda_f - \lambda) \cdot p_z + \lambda \tag{25}$$

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:

$$r_i = \left\| \begin{pmatrix} {}^{cam}x_i \\ {}^{cam}y_i \\ {}^{cam}z_i \end{pmatrix} \right\|^2 + \bar{r} \quad (i = 1, 2, \dots, n) \tag{26}$$

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).

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

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:

$$(p_{h,x} - x_i)^2 + (p_{h,y} - y_i)^2 + (p_{h,z} - z_i)^2 = r_i^2 \quad (i = 1, 2, \dots, n) \tag{27}$$

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

$$\boldsymbol{A} = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ \vdots & \vdots & \vdots \\ x_n - x_1 & y_n - y_1 & z_n - z_1 \end{pmatrix}, \quad \boldsymbol{x} = \boldsymbol{p}_h - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} b_{21} \\ b_{31} \\ \vdots \\ b_{n1} \end{pmatrix} \tag{28}$$

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

$$b_{k1} = \frac{1}{2}\left[r_1^2 + r_k^2 + (x_k - x_1)^2 + (y_k - y_1)^2 + (z_k - z_1)^2\right] \quad (k = 2, 3, \dots, n) \tag{29}$$

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

$$\boldsymbol{p}_h = \boldsymbol{x} + \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \tag{30}$$

[TODO: DESCRIBE THE USE OF THE PSEUDOINVERSE (MINIMIZATION PROBLEM), THE MINIMUM NUMBER OF POINT (4) NEEDED TO DO TRILATERATION]

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

$$\boldsymbol{p}_t = \boldsymbol{p}_h - {}^{w}\boldsymbol{R}_t(\hat{\boldsymbol{x}}_{k+1|k})({}^{t}\boldsymbol{p}_h - {}^{t}\boldsymbol{p}_t) \tag{31}$$

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.

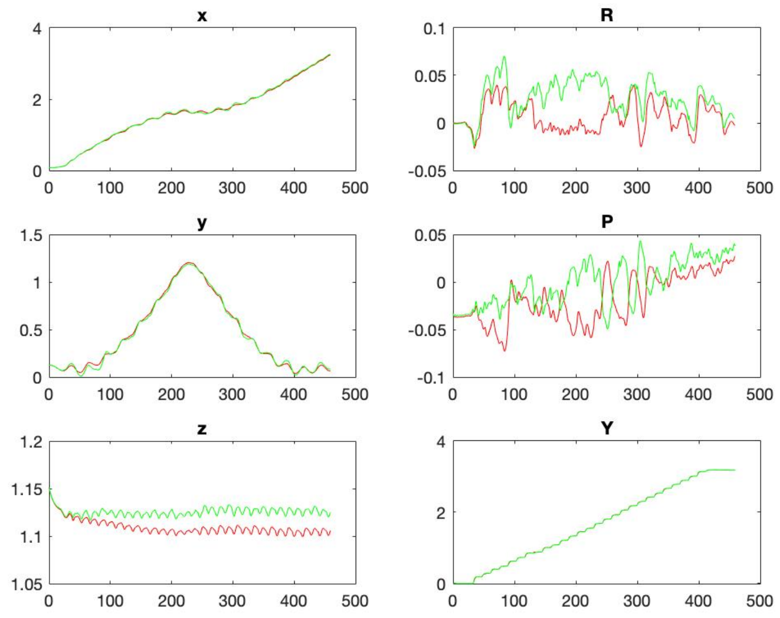

Figure 1: Comparison of the pose of the torso estimated with the EKF (in red) with respect to the ground truth (in green). $x$, $y$, $z$ stands for the position of the torso expressed in the reference frame of the world. $R$, $P$, $Y$ stands for the rotations roll, pitch, yaw around the respective axes of the reference frame of the world.

## 3 MPC Loop Closure

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

## 4 Experiments

[TODO: DIVIDE THIS SECTION IN SUBSECTIONS].

Figures 1 and 2 for the torso pose estimation (`simulationType = 1`) [TODO: ADD DESCRIPTION].

[TODO: ADD FIGURES OF THE ROBOT FOLLOWING A STRAIGHT LINE].

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

Figures 3 and 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.

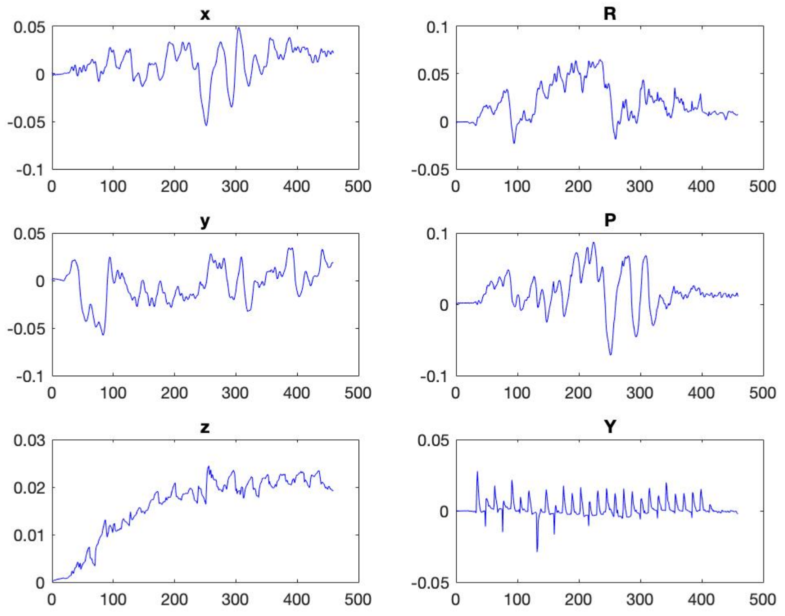

Figure 2: Difference of the pose of the torso estimated with the EKF (in red) with respect to the ground truth (in green). $x$, $y$, $z$ stands for the position of the torso expressed in the reference frame of the world. $R$, $P$, $Y$ stands for the rotations roll, pitch, yaw around the respective axes of the reference frame of the world.

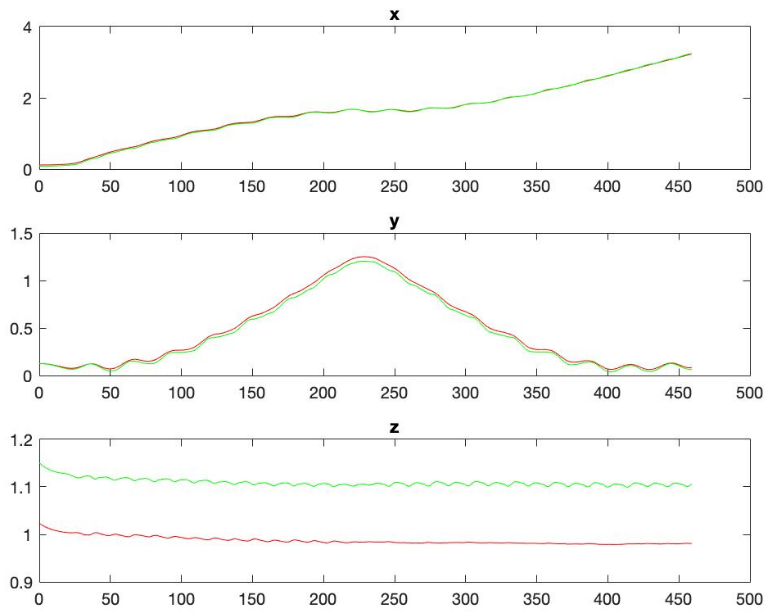

Figure 3: Comparison of the position of the torso estimated with the EKF (in red) with respect to the CoM (in green). $x$, $y$, $z$ stands for the position of the torso expressed in the reference frame of the world. $R$, $P$, $Y$ stands for the rotations roll, pitch, yaw around the respective axes of the reference frame of the world.

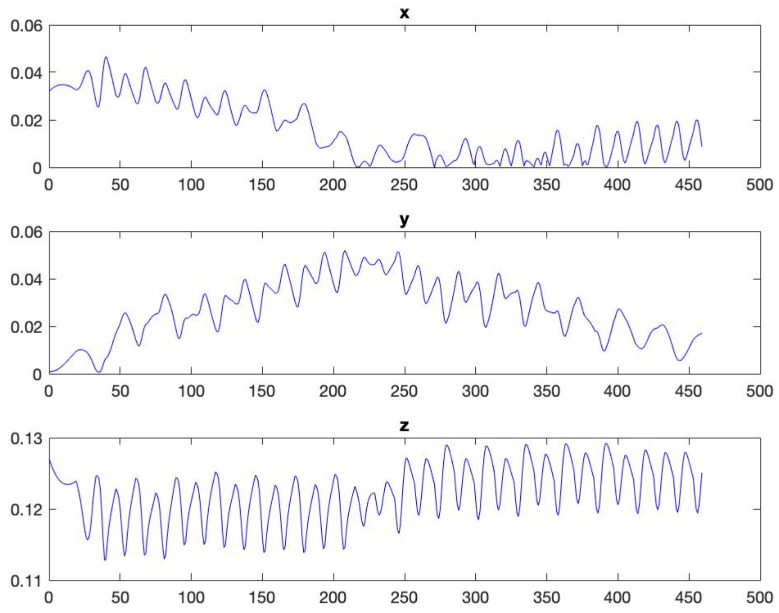

Figure 4: Difference of the position of the torso estimated with the EKF (in red) with respect to the CoM (in green). $x$, $y$, $z$ stands for the position of the torso expressed in the reference frame of the world. $R$, $P$, $Y$ stands for the rotations roll, pitch, yaw around the respective axes of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

## 5 Conclusions

Summary of the project, possible future developments and conclusions.

## 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.