Vision

Federico Vitabile

June 2021

1 Introduction

The purpose of this first section is to show the operation of the vision node. Its task is to detect the presence of one, or more, objects and evaluate its position (x, y, z), orientation (ϕ, θ, ψ) and velocity $(v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$. For this task was used the Intel RealSense D415 camera, It uses stereo vision to calculate depth and consists of a pair of depth sensors, RGB sensor, and infrared projector. To realize our purpose were used three topics of the camera, the topic */camera/depth/color/points* were used for detected the object and compute its position and velocity, the topics */camera/depth/camera_info* and */camera/color/image_raw* were used respectively to project the center of the object from the spatial coordinates (x, y, z) to those of the pixels, and to show the scene to display.

2 Object detection

Subscribed to the vision node to the topic */camera/depth/color/points* there is a point cloud available, or a vector contained the XYZ coordinates, and their RGB color, of all points in space detected by the camera. The PCL library was used to work with this points cloud.

But the point cloud is expressed in camera frame, view fig. 1, and the camera can have a generic orientation. Then, indicating with T_c^s the transformation matrix between spatial frame and camera frame, the points cloud was been reported in spatial frame. The spatial frame can be anyone, provided that it has a perpendicular axis to the wall and a perpendicular axis on the floor, or table.

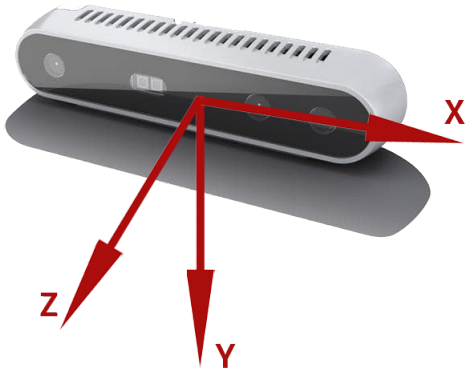


Figure 1: camera frame

With the points cloud expressed in the spatial frame the first step is to filter it to remove the walls and the floor, or table. Therefore, we defined a threshold compared to the camera and have been deleted farther and lower points. To allow moving objects during tests, all black-colored points have been deleted.

The remaining points cloud was then under-damped for a lower computational burden and divided into cluster. Every cluster is the points cloud of a single object.

3 Position and orientation of Objects

For each object detected, or for each cluster, the center and the orientation has been calculated. To calculate the center of the object was made a simple average of the cluster, While the PCA, principal component analysis, was used for the orientation.

For this one we proceed with the creation of a covariance matrix, which, starting from the centroid, indicates in which directions it is more likely to find points. then the eigenvectors have been extracted from this matrix. This eigenvectors represent the axes of the reference system solidarity to the object. Therefore starting from this, the rotation matrix R_o^s, between the object frame and the spatial frame, was built such as the transposed of the eigenvectors matrix. Finally with the Knowledge of R_o^s and the coordinates of the centroid, the transformation matrix T_o^s was built.

4 Velocity of objects

For the calculation of the linear velocity v, as shown in (1), the knowledge of two successive positions and the reading of the time passed between two subsequent activation of the node (ΔT) was exploited.

$$\begin{aligned} v_x(k) &= \tfrac{p_x(k)-p_x(k-1)}{\Delta T} \\ v_y(k) &= \tfrac{p_y(k)-p_y(k-1)}{\Delta T} \\ v_z(k) &= \tfrac{p_z(k)-p_z(k-1)}{\Delta T} \end{aligned} \tag{1}$$

Instead for the calculation of the angular velocity ω the relationship (2) of the quaternions was exploited.

$$\Omega = 2\ \dot{q}\ q^* \tag{2}$$

with

$$\Omega_{4x1} = \begin{bmatrix} \omega_{3x1} \\ 0 \end{bmatrix} \tag{3}$$

and $\dot{q}$ calculated by (4).

$$\begin{aligned} \dot{q}_w &= \tfrac{q_w(k)-q_w(k-1)}{\Delta T} \\ \dot{q}_x &= \tfrac{q_x(k)-q_x(k-1)}{\Delta T} \\ \dot{q}_y &= \tfrac{q_y(k)-q_y(k-1)}{\Delta T} \\ \dot{q}_z &= \tfrac{q_z(k)-q_z(k-1)}{\Delta T} \end{aligned} \tag{4}$$

5 Filtering

Among the measures previously obtained, we are interested in position and linear velocity. These, because of their way of being calculated, will inevitably be affected by noise. So we thought to exploit the knowledge of the type of motion of the object. In fact the object to be taken will move on a conveyor belt with velocity almost constant. It opted for a recursive Kalman filter.

5.1 Model

As a model, the kinematic equations of a body that proceeds with uniform rectilinear motion were used, in the individual components.

Then indicating how filter status the (5).

$$x = \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{5}$$

And as a model the (6).

$$\begin{cases} p_i(k+1) = p_i(k) + v_i(k)\ \Delta T \\ v_i(k+1) = v_i(K) \end{cases} \tag{6}$$

But the ΔT will never be constant, consequently in the prediction phase there will be a mistake due to its non-perfect knowledge. However, we can think to separate the ΔT in a constant component $\bar{\Delta T}$ and in a random component δt, as in (7). With δt a white noise with variance q and average 0.

$$\Delta T = \bar{\Delta T} + \delta t \tag{7}$$

So the filter model can be written in compact form like the (8).

$$x_{k+1} = \begin{bmatrix} I_{3x3} & I_{3x3}\ \bar{\Delta T} \\ 0_{3x3} & I_{3x3} \end{bmatrix}\ x_k + \begin{bmatrix} V_k \\ 0_{3x1} \end{bmatrix}\ \delta t = A_{6x6}\ x_k + D_{6x1}\ \delta t \tag{8}$$

As the vector of measures you have the whole state available, then

$$y_k = I_{6x6}\ x_k + n_k = C\ x_k + n_k \tag{9}$$

With n_k a white noise with covariance matrix R and average 0.

5.2 Kalman Filter equations

The Kalman filter will therefore be made of two phases, one of correction and one of prediction.

Correction:

$$\begin{aligned} e_k &= y_k - C\ \hat{x}_{k|k-1} \\ S_k &= R_k + C\ P_{k|k-1}\ C^T \\ L_k &= P_{k|k-1}\ C^T\ S_k^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + L_k\ e_k \\ P_{k|k} &= (I - L_k\ C_k)\ P_{k|k-1}\ (I - L_k\ C_k)^T + L_k\ R_k\ L_k^T \end{aligned} \tag{10}$$

Prediction:

$$\begin{aligned} \hat{x}_{k+1|k} &= A\ \hat{x}_{k|k} \\ P_{k+1|k} &= A\ P_{k|k}\ A^T + D\ q\ D^T \end{aligned} \tag{11}$$

With initial conditions $\hat{x}_{1|0} = y_0$ and $P_{1|0} = R$.

5.3 Choice of q and R

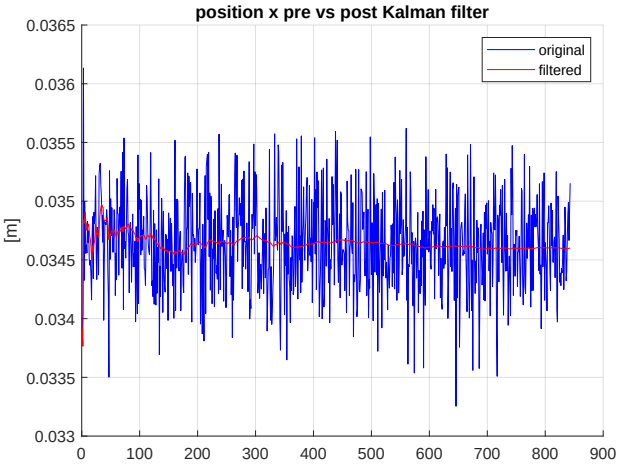
As already anticipated in section 5.1,we have chosen the variance of ΔT like the process noise, then $q = \sigma_t$. To estimate σ_t tests were carried out and the variance of the node frequency was measured.

The 12 was chosen as a covariance matrix R, with $sigma_{p_i}$ and $sigma_{v_i}$ esteemed as the variance of the position measurement and the variance of velocity measurement, with a object with constant velocity.

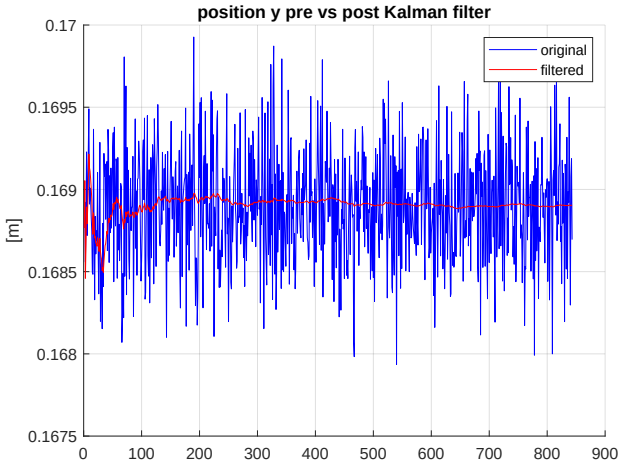
$$R = \begin{bmatrix} \sigma_{p_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{p_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{p_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{v_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{v_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{v_y} \end{bmatrix} \tag{12}$$

6 Test with stationary object

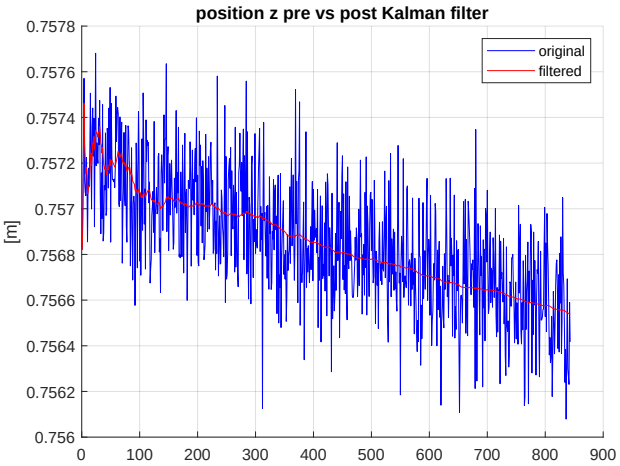
The figure 2 shows the results obtained for the estimate of the position and speed of a stationary object.



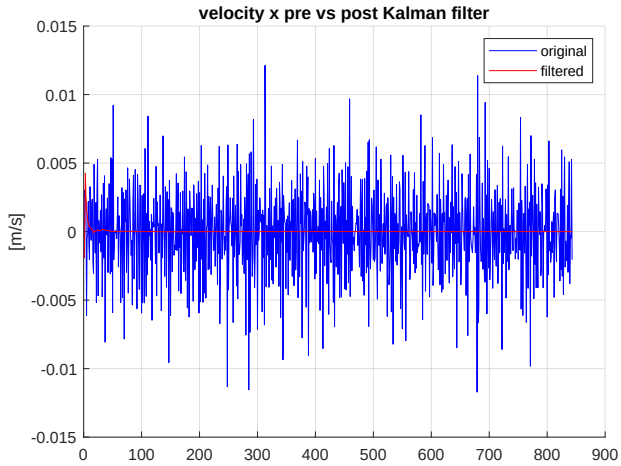
(a) Coordinate x of the position



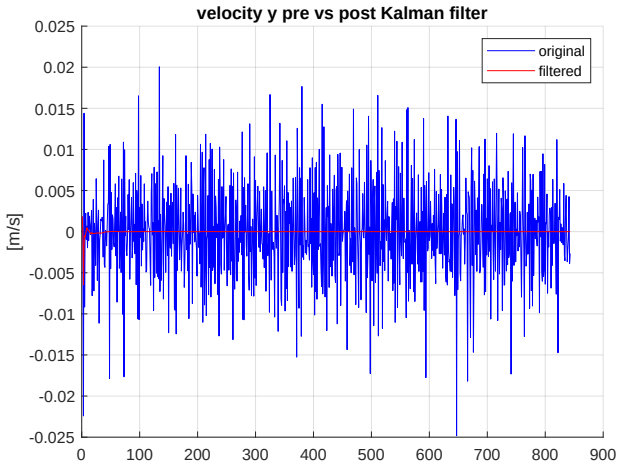
(b) Coordinate y of the position



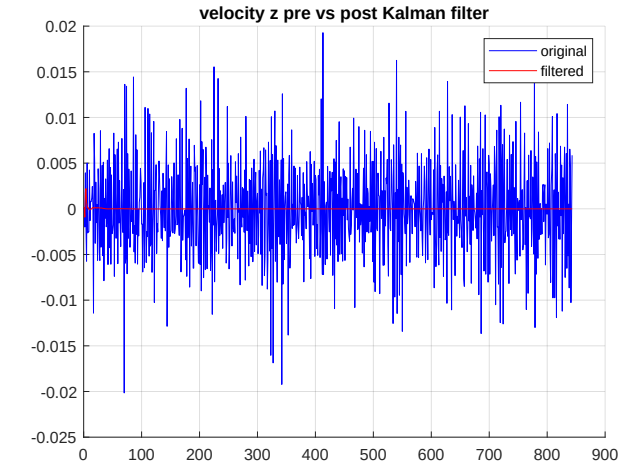
(c) Coordinate z of the position



(d) Coordinate x of the velocity



(e) Coordinate y of the velocity

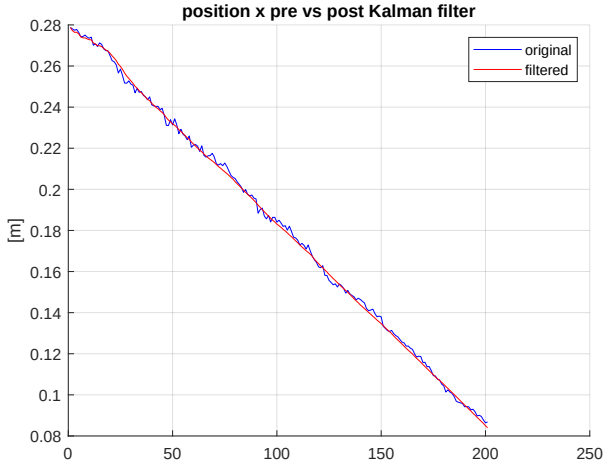


(f) Coordinate z of the velocity

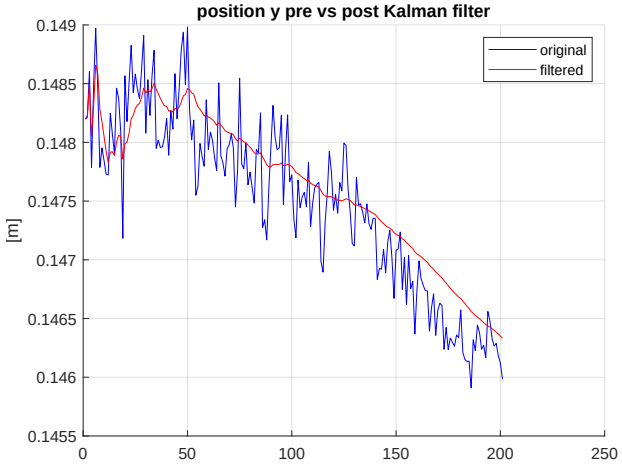
Figure 2: Test with stationary object

7 Test with object with constant velocity

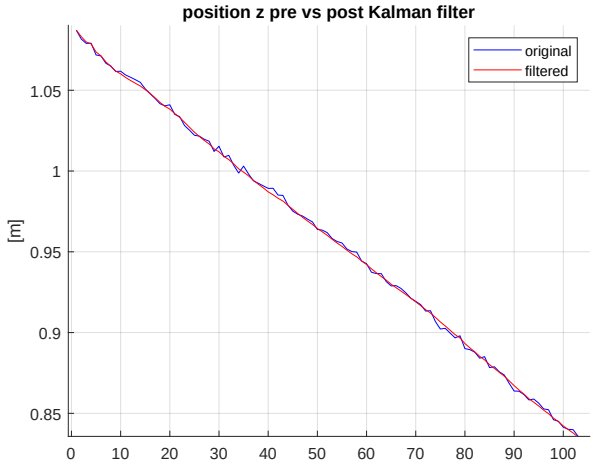
The figure 3 shows the results obtained for the estimate of the position and speed of an object with constant velocity.



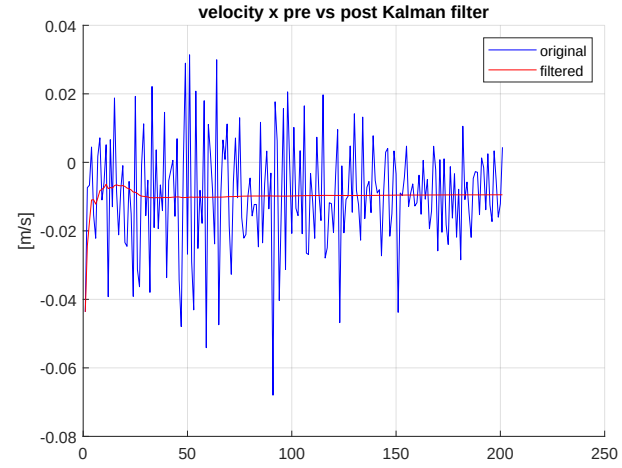
(a) Coordinate x of the position



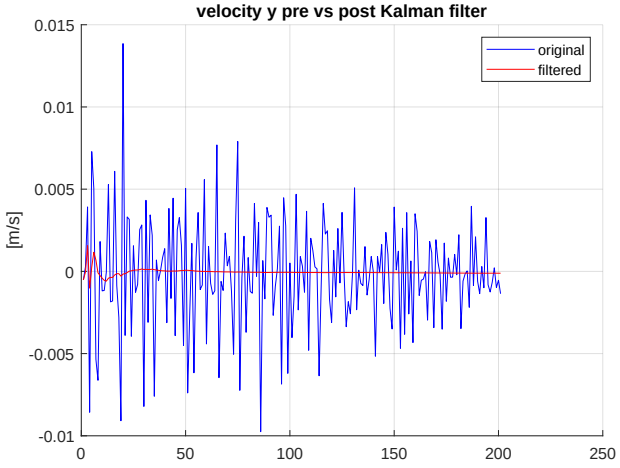
(b) Coordinate y of the position



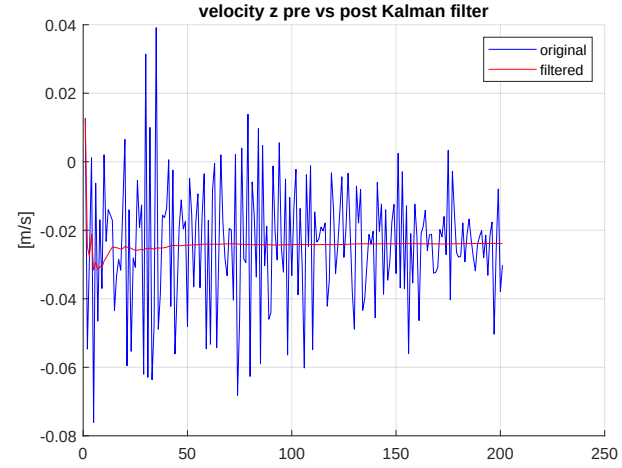
(c) Coordinate z of the position



(d) Coordinate x of the velocity



(e) Coordinate y of the velocity



(f) Coordinate z of the velocity

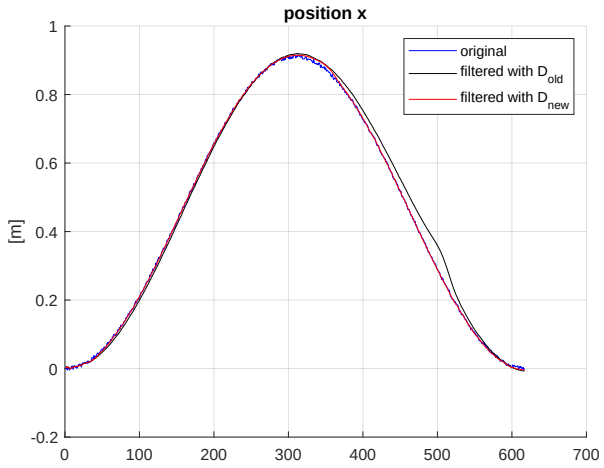
Figure 3: Test with object with constant velocity .

8 Implementation notes

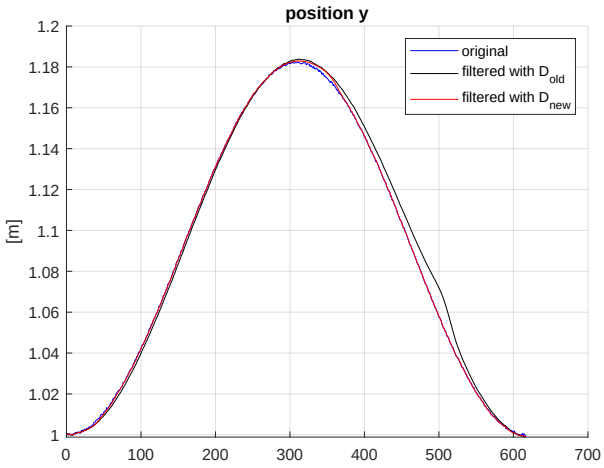
In the filter implementation it was thought to anticipate the prediction phase to that of correction. Doing that, the used ΔT becomes perfectly known. So the choice of $q = \sigma_t$ loses justification. Furthermore the filter previously illustrated, having $D = [v_x \ v_y \ v_z \ 0 \ 0 \ 0]^T$, does not take into account model errors concerning the dynamics of speed. Then, we used $D = [v_x \ v_y \ v_z \ v_x \ v_y \ v_z]^T$ with the idea of making the filter usable with objects that have non constant velocity.

The section 9 shows the comparison between the filter with $D_{old} = [v_x \ v_y \ v_z \ 0 \ 0 \ 0]^T$ and the filter with $D_{new} = [v_x \ v_y \ v_z \ v_x \ v_y \ v_z]^T$, using the simulated data of an object that moves with sinusoidal trajectory.

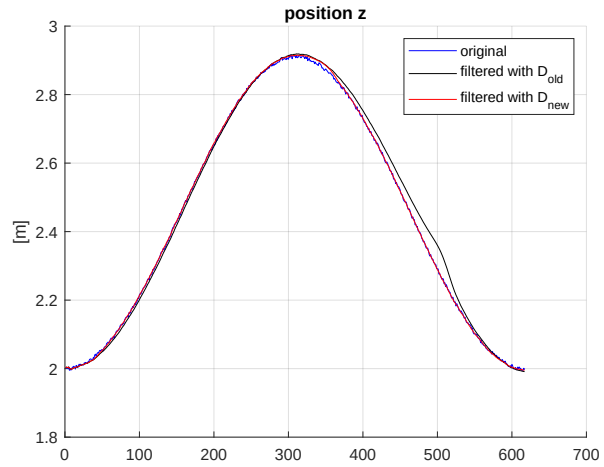
9 Comparison between the two filters



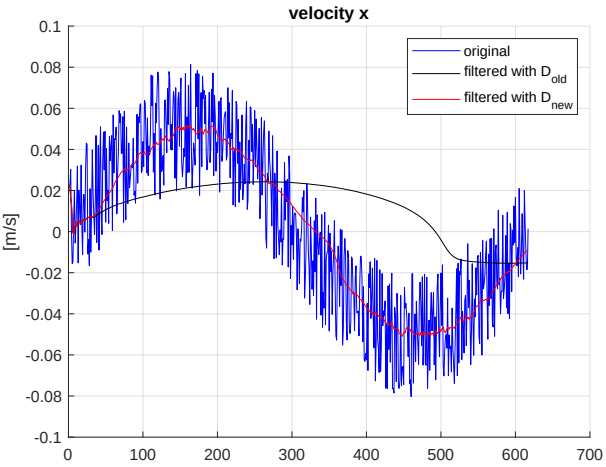
(a) Coordinate x of the position



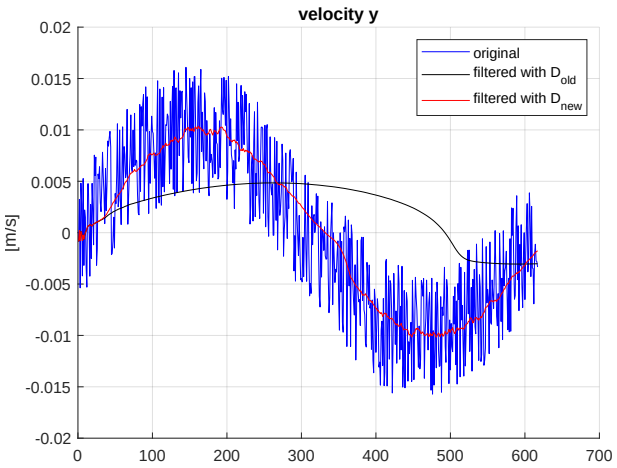
(b) Coordinate y of the position



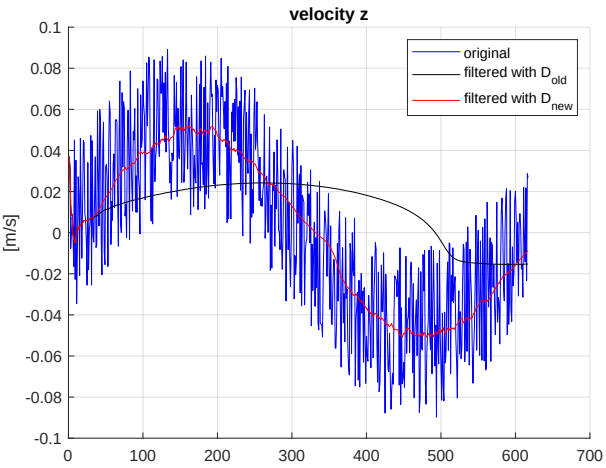
(c) Coordinate z of the position



(d) Coordinate x of the velocity



(e) Coordinate y of the velocity



(f) Coordinate z of the velocity

Figure 4: Comparison between filter with $D_{old} = [v_x \ v_y \ v_z \ 0 \ 0 \ 0]^T$ and filter with $D_{new} = [v_x \ v_y \ v_z \ v_x \ v_y \ v_z]^T$.