Spaceship Tracking - Kalman Filter

Università di Genova

Simone Campisi[1]
[1]Genoa University, Master in Artificial Intelligence

Student ID:4341240
January 21, 2021

Abstract

In this project it is presented an application of the *kalman* filter, which it is used to track a spaceship in a simulation of a trip from the Earth to the Moon. The trajectory implemented is not realistic, is a sort of "toy" trajectory in which it is was assumed that the spaceship has a motion uniformly accelerated.

1 Introduction

1.1 Kalman Filter - Overview

Kalman filtering is an algorithm that takes a series of measurements over time, containing statistical noise, and produce an estimate that tends to be more accurate than a single measurement. This is very important, because the the sensors give us always noisy information or the environment could makes data collection difficult, and the predictions made using kalman filtering help or make a better estimate. There are many applications of the kalman filtering, and in this case, it was used to track a spaceship in a simulated trip from the Earth to the Moon.

1.2 Kalman Filter - Algorithm

The Kalman filter for tracking moving objects estimates a state vector, s_k comprising the parameters of the target, such as position, velocity and acceleration, based on a dynamic/measurement mode.

The algorithm of Kalman filtering consists of two phases: *update* and *estimation*. It was assumed that the upper "-" on the vectors and matrices denotes: "before the acquisition of the k-th measure".

1. **Update:** First, it is necessary to acquire the new measurement m_k, which is modeled as

$$m_k = Hs_k + r_k$$

where **H** is the *M x N measurement matrix*, and $\boldsymbol{r_k}$ is *the measurement noise* with the *M x M measurement noise covariance matrix* **R**. Then, it is possible to compute the **Kalman Gain**, $\boldsymbol{K_k}$ that minimizes the error of the estimates:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

. Now it is possible update the state $\hat{s_k}$:

$$\hat{s_k} = \hat{s_k}^- + K_k(m_k - H\hat{s_k}^-)$$

and the and the covariance P_k

$$P_k = (I - K_k H)P_k^-$$

In which **P** is the *N x N Posterior covariance matrix.*

2. **Estimation:** Now it is possible to make the two estimations: the state estimate projection

$$\hat{s}_{k+1}^- = \Phi\hat{s_k}$$

and the covariance estimate projection

$$\hat{P}_{k+1}^- = \Phi P_k \Phi^T + Q$$

where $\mathbf{\Phi}$ is the *transition matrix*, which allows to pass from the state k to the state k+1, and **Q** is the *covariance matrix of the process noise*, that allows some variability in the model.

[3]

1.3 Data

In order to build a simulation of a trajectory from the Earth to the Moon, has been generated synthetic data to build "toy" trajectory from the Earth to the Moon. The Kalman filter model is a *constant acceleration (CA)* model, in fact, it is assumed that the spaceship has a uniformly accelerated motion. But, in order to build a trajectory, that it may remotely resemble to a trip from the Earth to the Moon, it was necessary to add the acceleration and other factors to generate the data. The data are generated in a 3-dimensional space, so, at the end of the generation, the data are composed by the three coordinates of the position (x, y, z), the velocities of the spaceship in the 3 directions (vx, vy, vz) and also the acceleration in the 3 directions (ax, ay, az). Then, in order to simulate measurements obtained by a sensor, it was necessary add noise to the data. The Fig 1 shows the results of the data generation. As mentioned before, the measurements simulate a sensor that provides measures with uncertainty, and this measures are represented with a scatter plot.

2 Methods - Initialization and Execution

Now it's possible to model the kalman filter in order to track the spaceship from the Earth to the moon. The Kalman Filter algorithm has not been implemented from scratch, but has

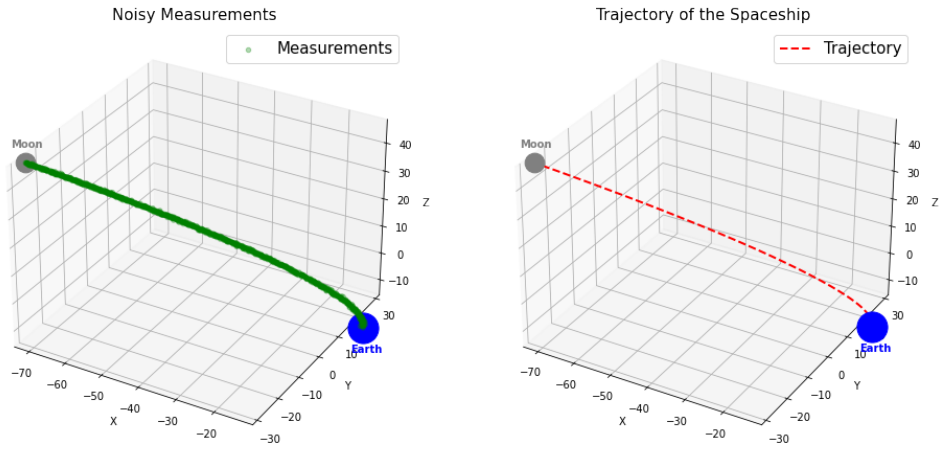


Fig. 1. Noisy Measurements (on the left) and real trajectory (on the right)

been used the implementation of the algorithm provided by the module *filterPy* [1]. First it was necessary to initialize the matrices Φ, H, P, R, Q and to define the state and, as mentioned before, it was assumed that the model is uniformly accelerated. The dynamics of a moving object in one-dimension can be described by the Newtonian distance equation:

$$x_t = \frac{1}{2}\ddot{x}\Delta t^2 + \dot{x}_{t-1}\Delta t + x_{t-1} \tag{1}$$

$$\dot{x}_t = \ddot{x}\Delta t + \dot{x}_{t-1} \tag{2}$$

and since the motion is uniformly accelerated:

$$\ddot{x}_t = \ddot{x}_{t-1} \tag{3}$$

In which $x_t, \dot{x}_t, \ddot{x}_t$ denote respectively the position, the velocity and the acceleration at time t.So, the dynamics of a moving object in one-dimension can be modeled by the position and the first derivation of it. Now, this notation can extended for a 3D object, that is described by x_t, y_t, z_t for the position, $\dot{x}_t, \dot{y}_t, \dot{z}_t$, for the velocity, and $\ddot{x}_t, \ddot{y}_t, \ddot{z}_t$ for the acceleration. So, for the 3D object, the state vector

$$s_t = \begin{bmatrix} x_t & y_t & z_t & \dot{x}_t & \dot{y}_t & \dot{z}_t & \ddot{x}_t & \ddot{y}_t & \ddot{z}_t \end{bmatrix}^T \tag{4}$$

which, at start has been initialized with the starting position, velocity and acceleration.

Hence, with this assumptions the transition matrix Φ can be defined as the following 9x9 matrix:

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Since the aim of this project is to track the position of the spaceship, x, y, and z, the measurement matrix H, 9 x 3, will be:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

Then the process noise Q must to be set in order o achieve tracking errors that are as small as possible. In the conventional tracking systems, the most commonly process noise matrix used is the *random acceleration (RA) process noise*, which is often selected because it has a better performance [2]. The 9x9 random acceleration process noise matrix is defined as following:

$$Q = GG^T\sigma_a^2 \tag{7}$$

in which σ_a^2 is the *acceleration process noise*, and G, for a constant acceleration model is:

$$G = \begin{bmatrix} \frac{1}{2}\Delta t & \frac{1}{2}\Delta t & \frac{1}{2}\Delta t & \Delta t & \Delta t & \Delta t & 1 & 1 & 1 \end{bmatrix} \tag{8}$$

Finally, the matrix P can be set to random values, instead R, can be empirically carried out, in this case is a diagonal matrix 3 x 9, with all ones in the diagonal. All the above matrices are used to initialize an object *KalmanFilter()* of filterpy, which represents the tracker for the spaceship, and provides two fundamental methods:

- *predict()*, which Predict next state
- *update(z)*, which add a new measurement, z, to the Kalman filter.

This two methods are recalled in a loop for each measurements, and, at each iteration are saved the predicted state ($\hat{s}_t$), and the predicted covariance matrix (P). At the end of

the loop, are returned a matrix N x M, in which N is the total number of measures and M is the size of the state vector, that contains all the state vector for each measure acquired. Moreover, is returned also a matrix N x M x M, which contains all the covariance matrix P for each measure acquired.

3 Results

Initializing the matrices and iterating as mentioned in the section 2, the results obtained are showed in figure 2. As you can notice, the filter takes noisy measures (the small circles in Fig. 2a and Fig. 2b) and makes an estimation that is better than the measurements, allowing the spaceship to travel with the correct trajectory from the Earth to the Moon, and the estimated trajectory is very close to the original one without noise.

Then, in order to evaluate the quality of the predictions, has been computed the residuals, that is the difference between the positions measured and the position estimated by the filter. The results of the residual are showed in figure 3.As you can notice the oscillations of the residuals are in a range close to 0. In Fact the variance, σ^2 and the mean, μ, of the residuals, are very close to 0, especially for the x and y position. Since the "ideal" residual should be as much as possible close to 0, is possible to say that the model has produced a good prediction of the trajectory.

3.1 Experimental Results - Signal Loss Simulation

The tracker modeled has been tested, in order to show the behavior of the tracker with different kind of loss signal.

3.1.1 Filter updated with only the first half of the measurements

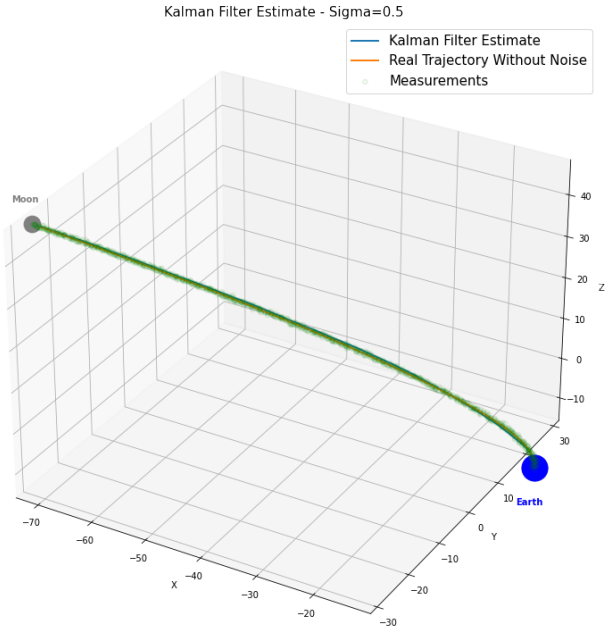
This first experiment shows the worst case: the spaceship at half of the trip looses totally the signal. In fact in the loop explained in section 2, the tracker is updated with only the first half of the measurements. The Results are showed in figure 4. As you can notice, at half trip, the spaceship deviates totally from route and fails to reach the moon.

3.1.2 Filter updated with only the even indices of the measurements

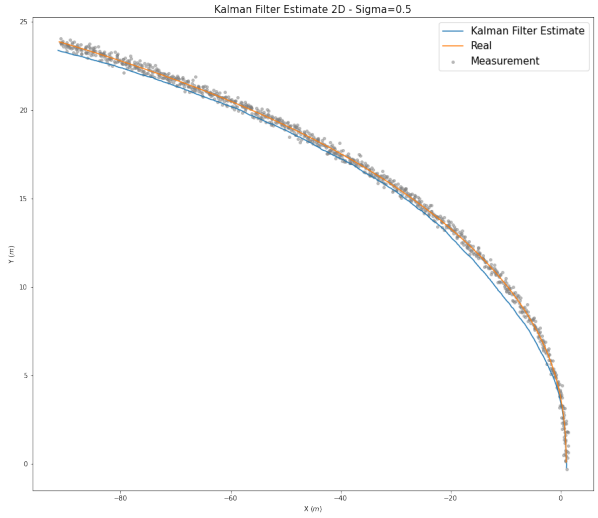
In this case, the situation is different from the experiment 3.1.1, in fact, updating the tracker with only the even indices of the measurements, the spaceship is able to reach correctly the Moon with a trajectory very similar to the one predicted with all the measurements. The figure 5 shows the obtained results.

3.1.3 Filter updated with only one measurement every 5

Here the filter is updated with a measurement every five. Also in this case, the spaceship is able to reach correctly the Moon following a trajectory very similar to the one estimated with all the measurements, as showed in figure 6.



(a) Kalman Filter Estimate - 3D



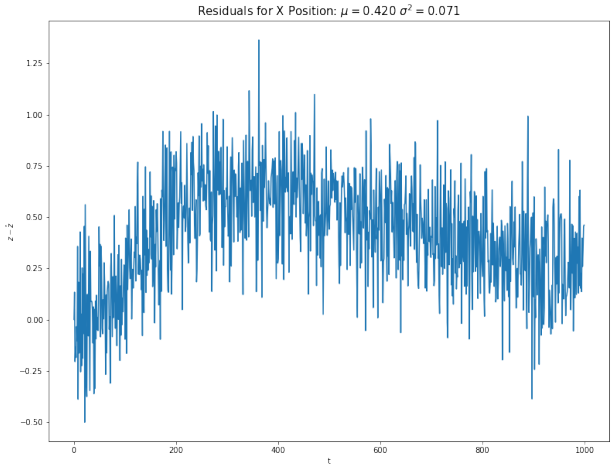
(b) Kalman Filter Estimate 2D - Visualization of axis x and z

Fig. 2. Kalman Filter Estimate 3D - 2D

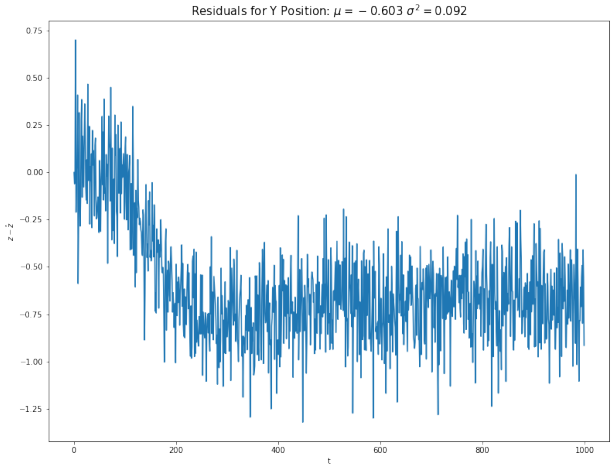
3.1.4 Spaceship looses the signal halfway for a certain period

Here has been simulated a situation in which the spaceship looses the signal for a certain period at about half trip. In this case, as showed in figure 7, the spaceship deviates a bit from

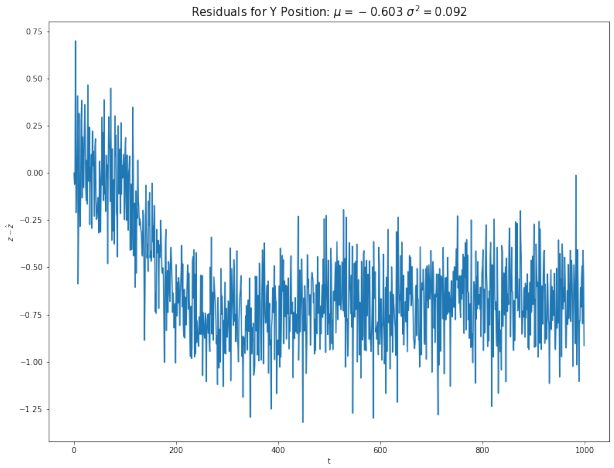
the route, but then, when the tracker acquires again the measures, the spaceship return in the original route and reaches the Moon.



(a) Residual between x positions measured and estimate - $\mu = 0.420$, $\sigma^2 = 0.071$

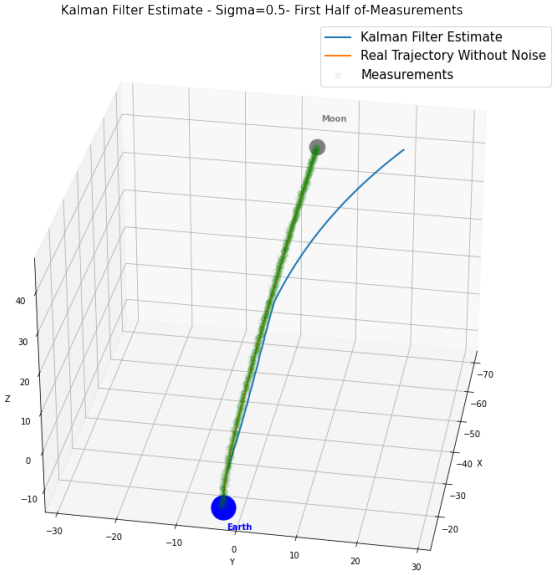


(b) Residual between y positions measured and estimate - $\mu = -0.603$, $\sigma^2 = 0.092$

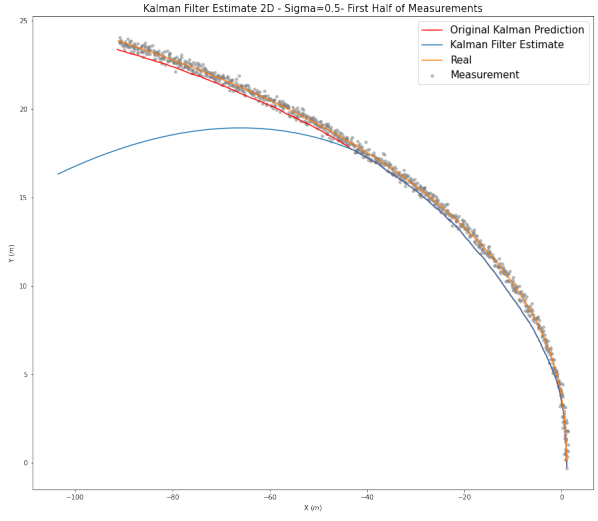


(c) Residual between z positions measured and estimate - $\mu = 0.305$, $\sigma^2 = 0.059$

Fig. 3. Residuals for the positions x,y, z

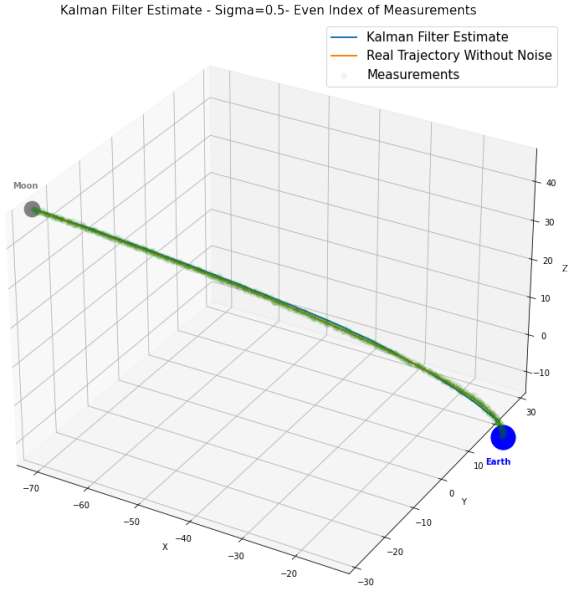


(a) Kalman Filter Estimate - 3D

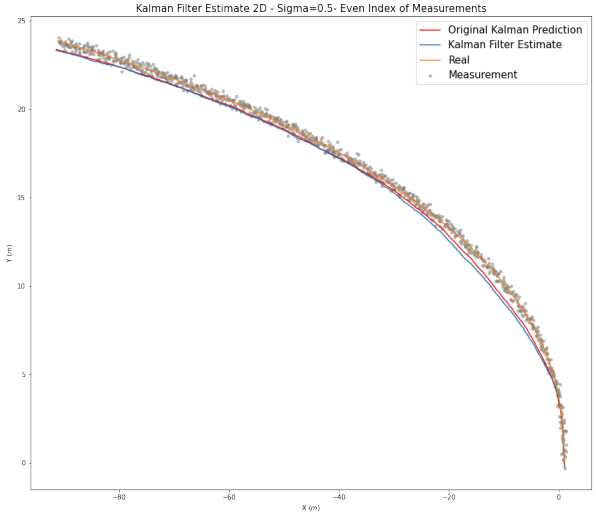


(b) Kalman Filter Estimate 2D - Visualization of axis x and z

Fig. 4. Kalman Filter Updated with only the first half of measurements

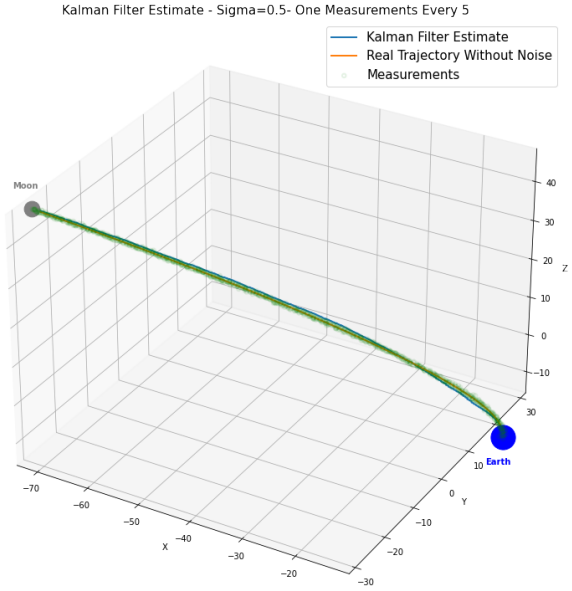


(a) Kalman Filter Estimate - 3D

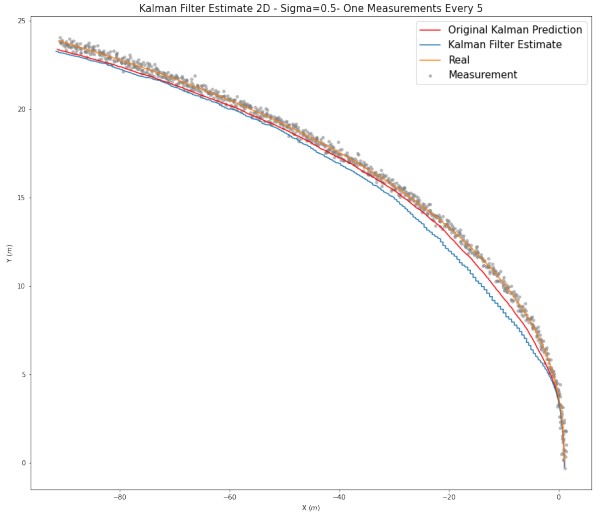


(b) Kalman Filter Estimate 2D - Visualization of axis x and z

Fig. 5. Kalman Filter Updated with only the even indices of measurements

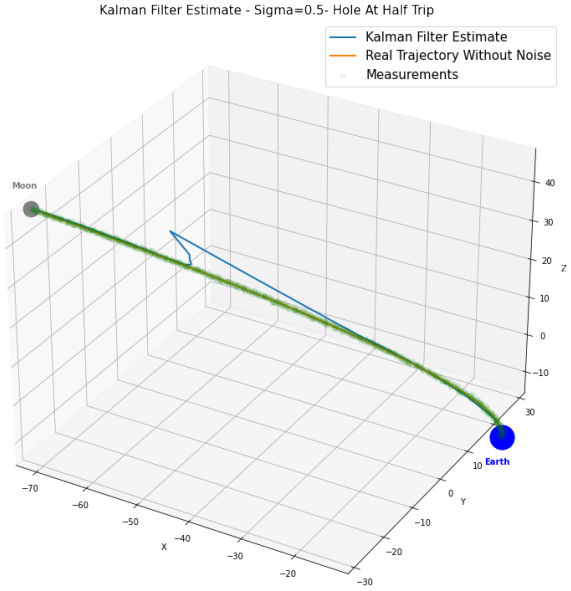


(a) Kalman Filter Estimate - 3D

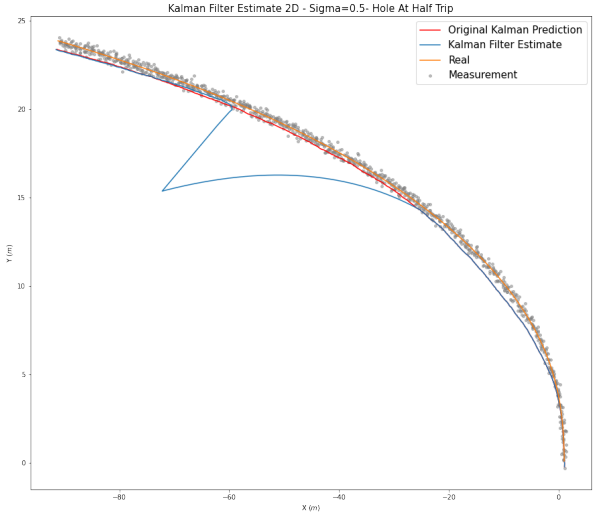


(b) Kalman Filter Estimate 2D - Visualization of axis x and z

Fig. 6. Kalman Filter updated with only one measurement every 5



(a) Kalman Filter Estimate - 3D



(b) Kalman Filter Estimate 2D - Visualization of axis x and z

Fig. 7. Spaceship looses the signal halfway for a certain period

References

[1] *FilterPy Documentation*. URL: https://filterpy.readthedocs.io/en/latest/.

[2] Kenshi Saho. "Kalman Filter for Moving Object Tracking: Performance Analysis and Filter Design". In: (2017). DOI: http://dx.doi.org/10.5772/intechopen.71731.

[3] Alessandro Verri and Annalisa Barla. "Notes of the course Digital Signal Image Processinng". In: (2020).