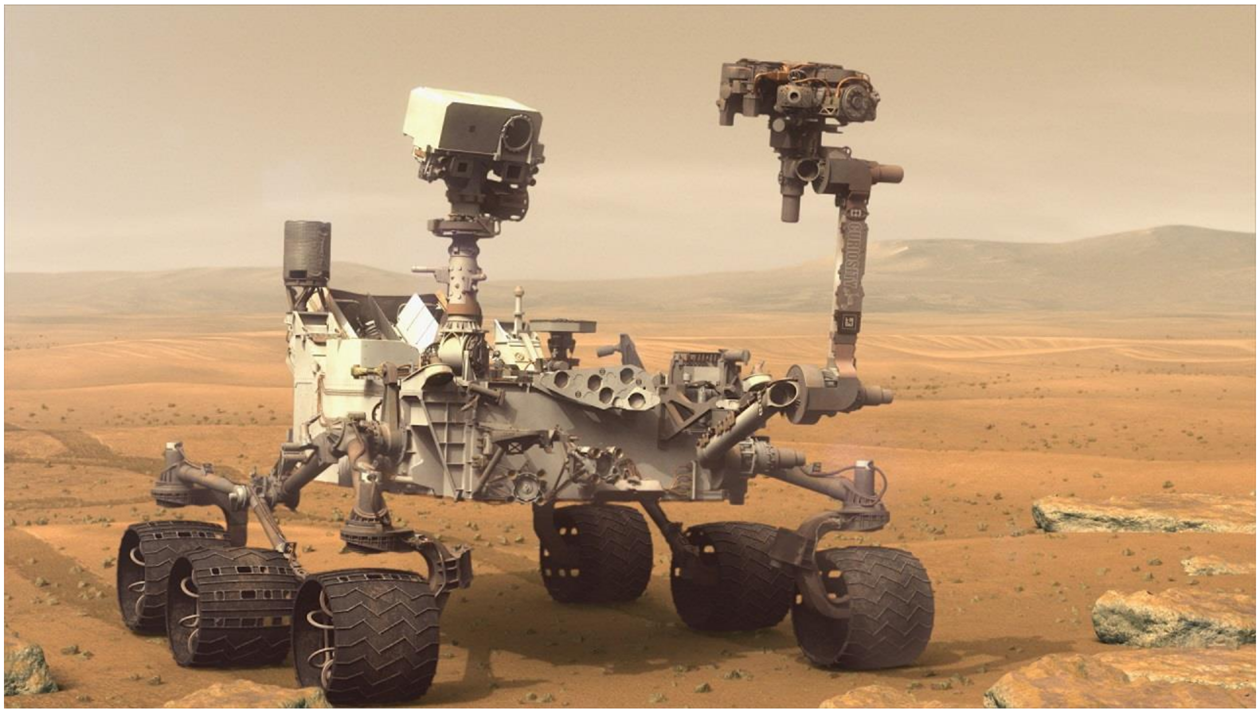
Sensors and Data Fusion

Kalman Filter Development for Martian Rover Tracking

Martin Yanev

May 13, 2018



This is my own unaided work unless stated otherwise.

1. Introduction

This study will concentrate on improving the measurements for planetary rover position using Kalman Filter. The rover is located on Mars surface and its motion is assumed to be in 2D plane. It is moving relatively slow with stop and go technique so it can scan the surroundings and collect important information. The relative velocity of the rover is about 0.01 m/s when moving on flat surfaces, but as it will be seen later on this value deviate due to the 3D effect that is not taken into account. [1]

To obtain rovers position and heading, many navigation techniques could be used. Moreover, the relatively low speed of the rover should simplify its tracking. Unfortunately, the traditional terrestrial methods which include using of global positioning system (GPS), map tracking and magnetic compass are accurate enough and are not able to solve the self-localization problem themselves on other planet. [2] This is because, the Martian magnetic field is too weak to be used by compass system. Similarly, the available digital maps are not fully developed on interplanetary environment and in most cases do not exist or have too low resolution [3]. These conditions limit significantly make rover tracking difficult when located on Martian environment.

The rovers position in our case is measured by number of satellites in synchronous orbit and information for its location is available every 60 seconds. Only a part of rovers path is available in this report. The two sets of data SDF2018a and SDF2018b show the measurements for rover position in X and Y axis. An extra measurement of Range from the initial beacon is available only in SDF2018b. Further, to improve the accuracy of this navigation sensor a Kalman filter will be used. The aim of this is achieving of reliable navigation calculations within the desired goals for rover tracking.

2. Methodology

As it was already mentioned a Standard Kalman Filter has been used firstly and then an Extended Kalman Filter has been applied as the Ranges became available. Both filters include system identification and software programs are needed for process automation. To achieve this, Kalman filter is implemented using MATLAB software.

Standard Kalman Filter

The Kalman Filter in an iterative mathematical process which uses a set of equations and consecutive data input to quickly estimate true position, value or velocity of an object when the measured values contain uncertainty, variation or random error.

Figure 1 shows a block chart of the Standard Kalman Filter. As we can see, there are seven stages to estimate the most accurate variable. Firstly, there is a need for good assumption of initial states of initial position and covariance error. These values should be closer to the real ones and for that reason in this particular case the initial sensor readings are set as initial state. At the first iteration the initial state is equal to the previous state, in order a starting point to be available. An important part of the process is the estimation of new state. The new state vector and covariance matrix are computed here and then used will then be used in the next step.

The Kalman gain is computed next. It determines how much of the new measurement is needed to update the new estimate. It can be simply explained as a ratio of the error in estimate divided by the sum of the measure in the estimate and error in the measurement. Kalman gain is then used as a weighting factor to determine what part of estimate and what part of measurement to use in order to calculate the new state vector. It is usually a number between zero and one. As close as Kalman gain is to unity as more accurate the measurements are and larger part of them will be used in new estimate calculation. The new estimate is then used as a previous state closing the iterative loop, after being saved.

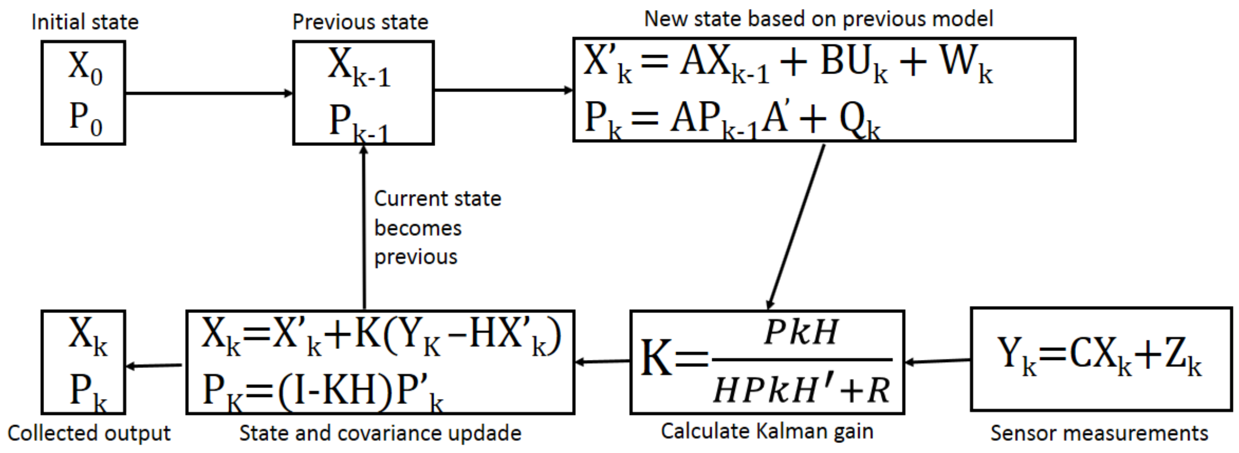


Figure 1. Standard Kalmen Filter Block Diagram

Extended Kalman Filter

While the Range from the beacon becomes available later in this assignment, there is a need of using extended Kalman filter, due to the fact that we cope with non-linear problem. A block diagram of Extended Kalman filter is shown on Figure 2 and as it can be seen the overall flow of algorithm is very similar to Standard Kalman filter. The first noticeable difference in the equation for prediction of the new state $F(X,U)$ and the second is in the state update $h(X')$. This new functions transform the linear algorithm into non-linear.

Theoretically, both standard and extended Kalman filter use linearization, but the Extended Kalman filter takes the X_k value as a reference for linearization.A care should be taken to A and H matrixes that should have alreasy been determined, but in this case are unknown yet. They should be derived using nan-linear model. While the standard Kalman filter solves them by linearization, here they are derived by linearizing the non-linear model [4]. This is done by partial differentiation with respect to x as follows:

$$H \equiv \frac{\partial h}{\partial x} \quad A \equiv \frac{\partial f}{\partial x}$$

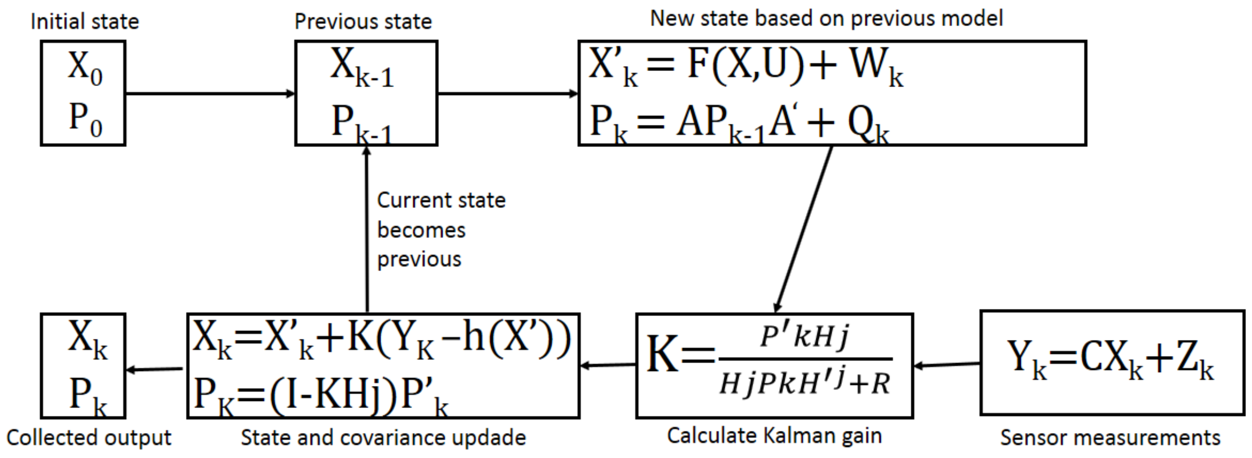


Figure 2. Extended Kalmen Filter Block Diagram

Matrixes definition

Let’s now take a look in more depth for Kalman filter implementation. It is first desirable to obtain the system model. As we have the measurement data for X and Y displacement and the time step of 60 seconds we can then determine the measured velocity corresponding to both axis. This velocities will be added to the state vector for both estimated and measured state. To be exact at this point the acceleration can also be estimated when double differentiate the position values and then can be used in the state space representation of the system shown in Equation 2.

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta T\text{^}2 & 0 \\ 0 & \frac{1}{2}\Delta T^2 \\ \Delta T & 0 \\ 0 & \Delta T \end{bmatrix} \begin{bmatrix} ax \\ ay \end{bmatrix} + W_k$$

The A and B design matrixes can be clearly determined from the equation above as:

$$A = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{2}\Delta T^2 & 0 \\ 0 & \frac{1}{2}\Delta T^2 \\ \Delta T & 0 \\ 0 & \Delta T \end{bmatrix}$$

The term W_k introduce the error in process of prediction of new state and is assumed to be zero for this study. Also, note that the previous state is in form of four by one matrix, due to the four states included in the Kalman filter and marked with k-1.

Thinking about the noise as a variable which can not be predicted, the statistic approach can be introduced here. The noise is assumed to have a normal distribution, so its average will always be zero. An estimate which needs to be done only once is determining the covariance matrixes P, Q_k and R. Here P is the state covariance matrix which determines the error in the estimate, Q_k is process covariance matrix which keeps the state covariance matrix from being too small or going to zero and R is measurement covariance matrix which shows the error in the measurement.

Firstly, the state covariance matrix is determined knowing the number of measurements and estimating the standard deviation of each term. The deviations are then combined and presented in a matrix form as follows:

$$P_k = \begin{bmatrix} \sigma x^2 & \sigma x\sigma y & \sigma x\sigma vx & \sigma x\sigma vy \\ \sigma y\sigma x & \sigma y^2 & \sigma y\sigma vx & \sigma y\sigma vy \\ \sigma vx\sigma x & \sigma vx\sigma y & \sigma vx^2 & \sigma vx\sigma vy \\ \sigma vy\sigma x & \sigma vy\sigma y & \sigma vy\sigma vx & \sigma vy^2 \end{bmatrix} = \begin{bmatrix} 3.65 * 10^3 & 0 & 0 & 0 \\ 0 & 1.48 * 10^4 & 0 & 0 \\ 0 & 0 & 10^{-4} & 0 \\ 0 & 0 & 0 & 10^{-4} \end{bmatrix}$$

Note that the terms that are not located on the diagonal are set to be null due to their similarity. This simplifies the process of computing and do not increase the uncertainty significantly [4].

Q_k matrix is also diagonal one and shows the covariance in the noise W_k. As number of noises increase , as much rolls and columns the matrix will have, due to the increased number of variances. The R covariance follows absolutely the same pattern but it is covariance of measurement error Z_k. While Q_k has to be calculated depending on rover acceleration and the time step, R is already determined in sensor's specification and only need to be written in a matrix form.

The numerical expression of process covariance and measurement covariance are presented below:

$$Qk = \begin{bmatrix} dx & 0 & 0 & 0 \\ 0 & dy & 0 & 0 \\ 0 & 0 & vx & 0 \\ 0 & 0 & 0 & vy \end{bmatrix} = \begin{bmatrix} 0.5 * a * dt^2 & 0 & 0 & 0 \\ 0 & 0.5 * a * dt^2 & 0 & 0 \\ 0 & 0 & a * dt & 0 \\ 0 & 0 & 0 & a * dt \end{bmatrix}$$

$$R = \begin{bmatrix} Delta\,X^2 & 0 & 0 & 0 \\ 0 & Delta\,Y^2 & 0 & 0 \\ 0 & 0 & Delta\,X^2 & 0 \\ 0 & 0 & 0 & Delta\,Y^2 \end{bmatrix} = \begin{bmatrix} 2500 & 0 & 0 & 0 \\ 0 & 2500 & 0 & 0 \\ 0 & 0 & 2500 & 0 \\ 0 & 0 & 0 & 2500 \end{bmatrix}$$

, where a is acceleration magnitude, dt is the time step, Delta_X and Delta_Y are measurement standard deviations.

3. Results and Discussion

This section will show the results of Kalman filter implementation on rover motion measurements and discuss their variance when increasing the propagation covariance. Also, the rover displacement will be calculated using range measurements only and this will then be combined with the information for X and Y position provided by the satellites to improve the solution.

Rover tracking using satellite measurements

Figures 3,4 and 5 show a comparison between measured estimated and calculated by Kalman filter motion of the rover. As an overall trend the estimated and Kalman values was very close to each other and the blue line representing Kalman filter can be barely seen, hidden under estimate line. However, the filter's line have always been between the other two, showing the best estimate. The linear zones of rover motion are well captures and all lines are close to each other, this can be well seen on figures 3 and 5 characterized with larger linear regions.

Taking a look to the regions with large turns, it is obvious that the sensor measurements accuracy decrease as well as the estimate accuracy. The Kalman filter capture this uncertainty and gets further from measurement. The concave geometry of the lines which represent Kalman filter and calculation estimate at the beginning of rover's mission can be explained with lack of accuracy when setting the initial conditions. As mentioned above, the initial conditions used in the Kalman loop are chosen to be the same as the one from sensor measurements.

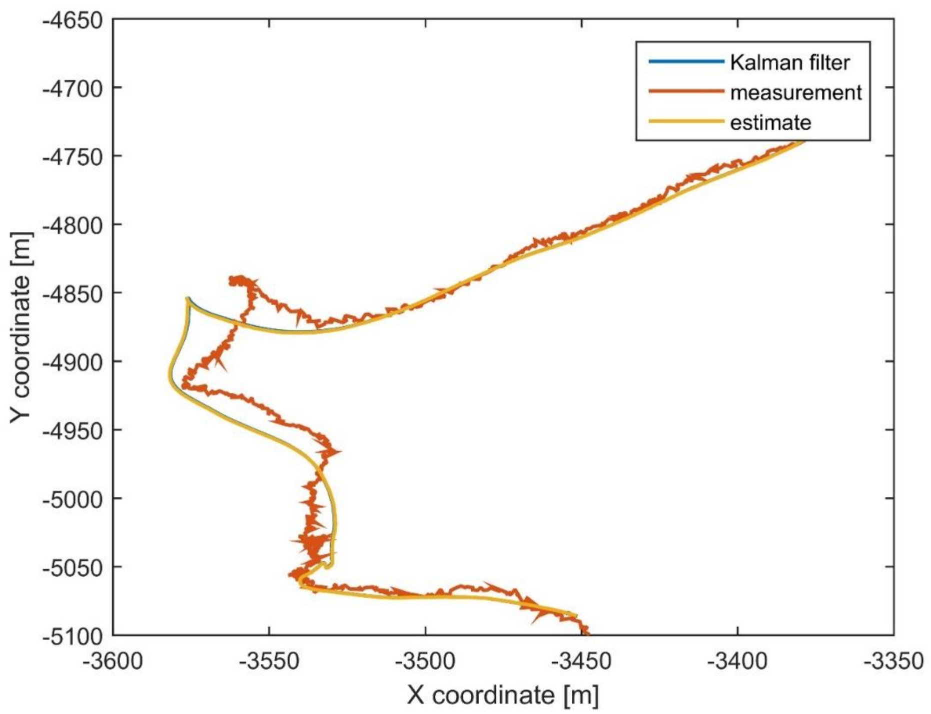


Figure 3. Measurement, estimation and Kalman filter prediction for Martian rover position using data set SDF2018a

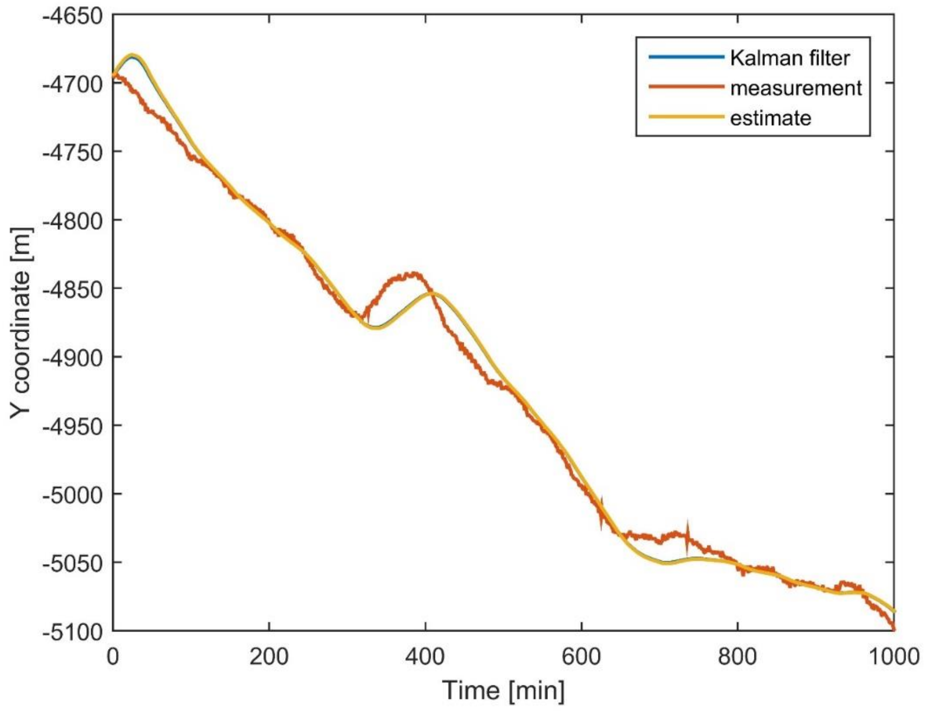


Figure 4. Displacement of Martian rover in Y direction using data set SDF2018a

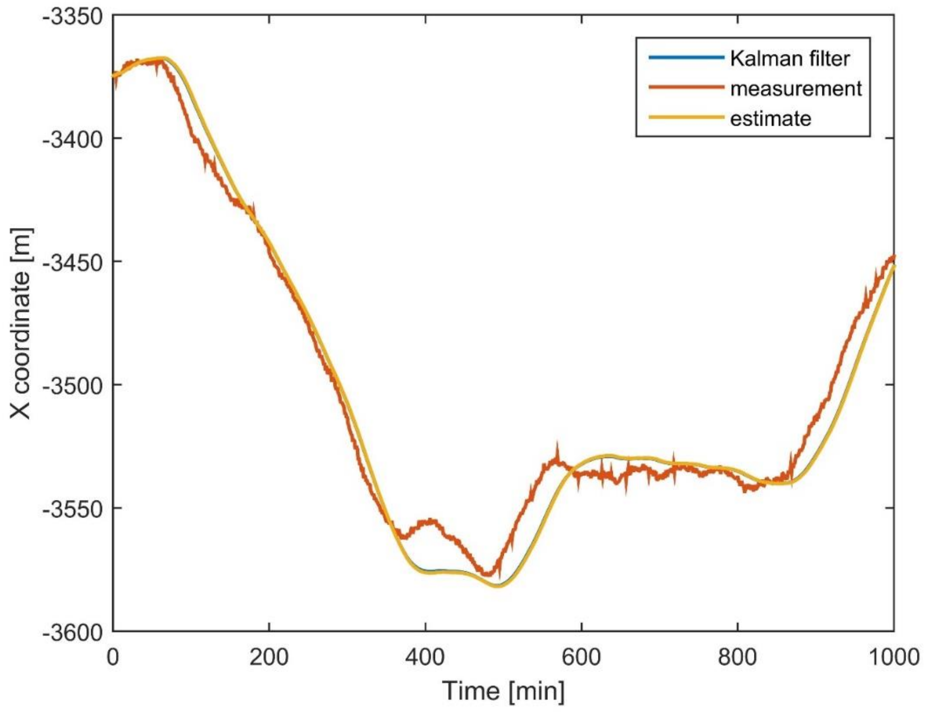


Figure 5. Displacement of Martian rover in X direction using data set SDF2018a

To compute the Kalman filter a single script has been created on MATLAB. The computing time was as much as few seconds and the code can be found in Appendix A. The acceleration portion in the state space equation has later been removed, due to the great uncertainty which it introduced and for that reason the results with added acceleration portion will not be discussed here. Now the state equation becomes:

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \triangle T & 0 \\ 0 & 1 & 0 & \triangle T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + W_k$$

Effect of propagation covariance adjustment

Figures 6 to 9 show the effects of adjusting the process covariance matrix. As already been mentioned, the Kalman gain is a ratio of measured covariance and the sum of measured and estimated covariance. Also, when the process covariance matrix increase this means that the uncertainty in the estimate is greater. Consequently, the Kalman gain increase in size allowing the Kalman filter to estimate the true value closer to the measurement value, as more error have been introduced in the estimate. Exactly this pattern is seen in the figures below.

As it can be seen on Figure 6, as W is increased a closer to the measurements the Kalman filter is. The rover turns are better captured and at W=10 both lines are very close to each other at all regions.

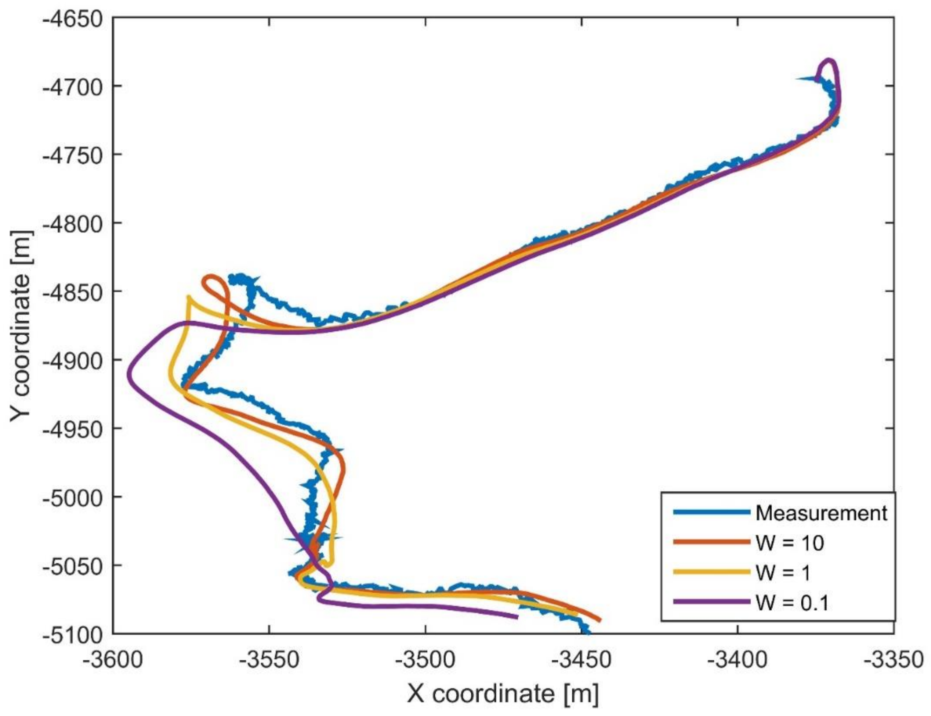


Figure 6. Rover motion 2D estimate comparison in terms of propagation covariance of 0.1, 1 and 10

The same pattern is followed when plotting X and Y displacement in terms of time. At Figure 7 the displacement peak highly over estimated by W=0.1 and its size gradually reduce until matching the measured value at W=10. Observing the Y position graph, when W=10 the measurement and Kalman filter almost perfectly match.

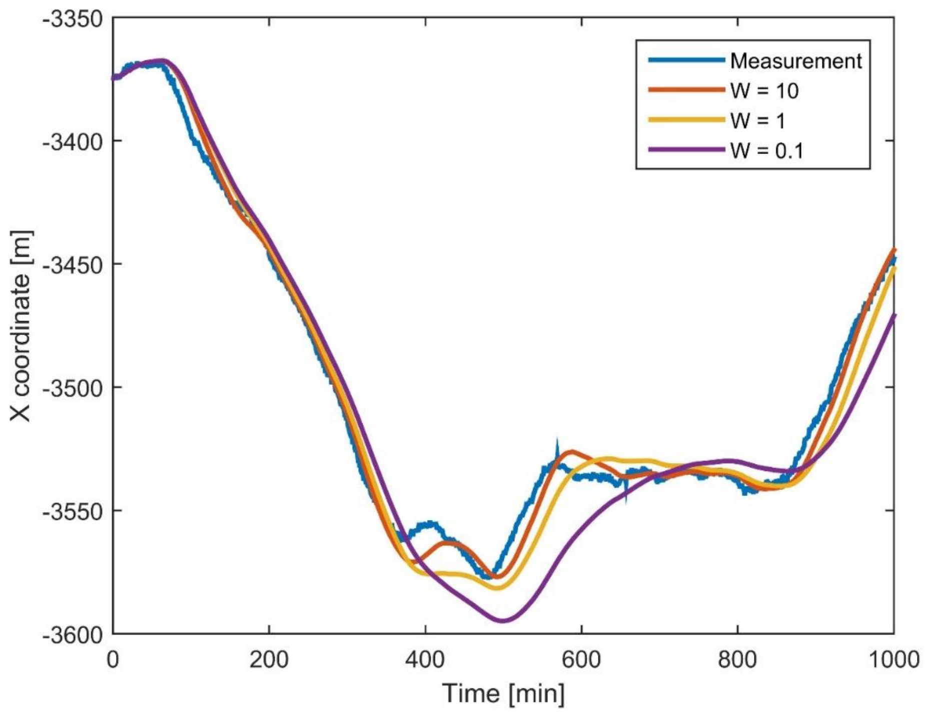


Figure 7. Rover X axis movement comparison in terms of propagation covariance of 0.1, 1 and 10

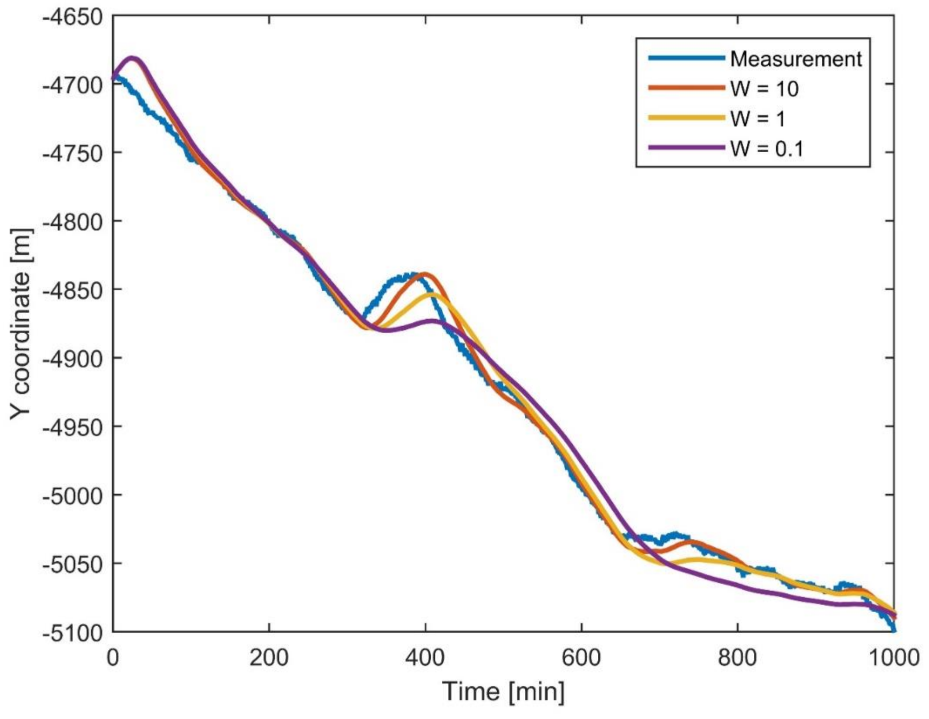


Figure 8. Rover Y axis movement comparison in terms of propagation covariance of 0.1, 1 and 10

An attempt has been done to capture the rover's velocity in terms of time, but as you can see from Figure 9 there were insufficient and no matter what the value of propagation covariance was the velocity couldn't be captured accurately. Obviously, this is due to the rapid velocity change from positive to negative value, because as seen above, Kalman filter hardly cope with turns and sharp changes of measured values.

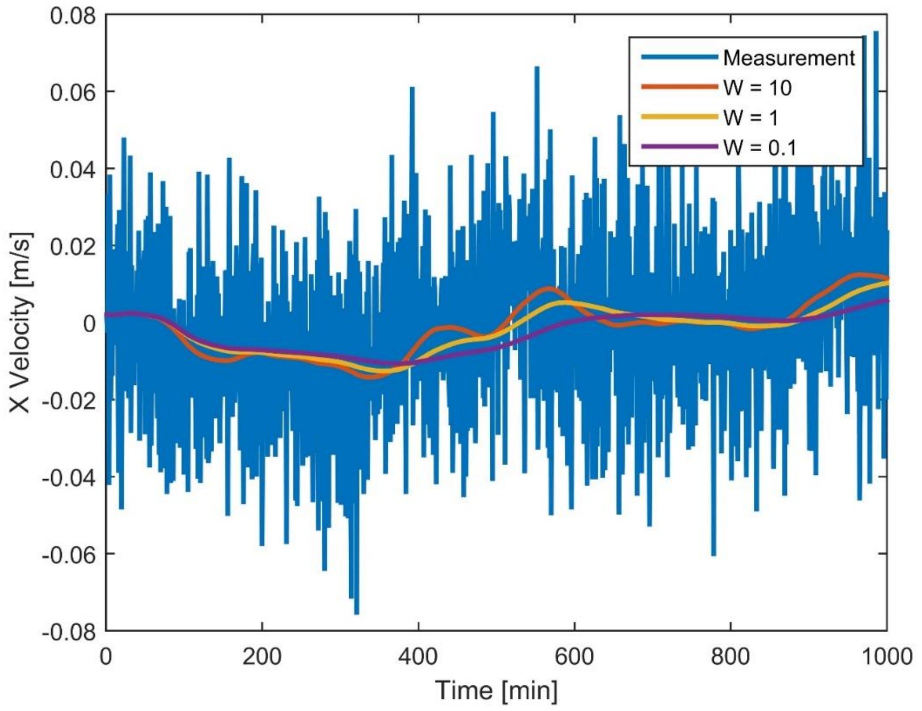


Figure 9. Velocity change in terms of propagation covariance

Range measurement implementation

The data pack consisting a measurements for X, Y coordinates and Range from the beacon has also been used and a Kalman Filter was created in that case as well. The extra range measurement was used to improve the Kalman Filter solution.

Firstly, according to [4] an extended Kalman filter that uses only range measurement has been developed. Here the H matrix is not identity as in the Standard Kalman filter ,but was set as a function of X and Y estimates and written in Jacobian form:

$$H_j = [\frac{X}{\sqrt{X^2+Y^2}} \quad \frac{Y}{\sqrt{X^2+Y^2}} \quad 0]$$

, where the first two terms correspond to the estimations for X and Y and velocity is assumed to be zero.

Obviously, the term $\sqrt{X^2 + Y^2}$ is the estimate of range. This is used in the equation of state update and it becomes from:

$$X_k = X'_k + K(Y_K - HX'_k)$$

,to

$$X_k = X'_k + K(\text{Range} - h(X'))$$

, where h(X') is the estimation for Range and "Range" corresponds to measured Range.

Applying those modifications to a set of MATLAB functions gives the X and Y displacement of the rover by having only Range measurements. Figures 10 and 11 show a comparison between Radar estimate and satellite measurement of rover motion in X and Y direction. In both cases the error becomes greater when the time increase, which seems to be due to non-linearity. Moreover, radar estimates give the same portion of movement for both X and Y states. For that reason, if we plot them on the same figure, the result will be a straight line.

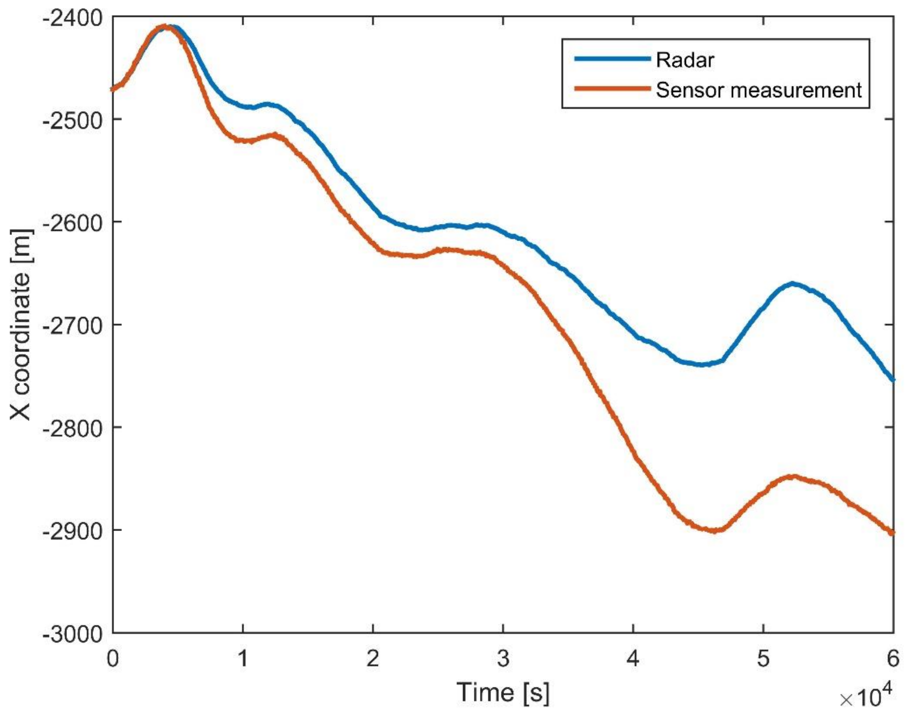


Figure 10. Comparison between Extended Range Kalman Filter estimates and Sensor measurements for X coordinate

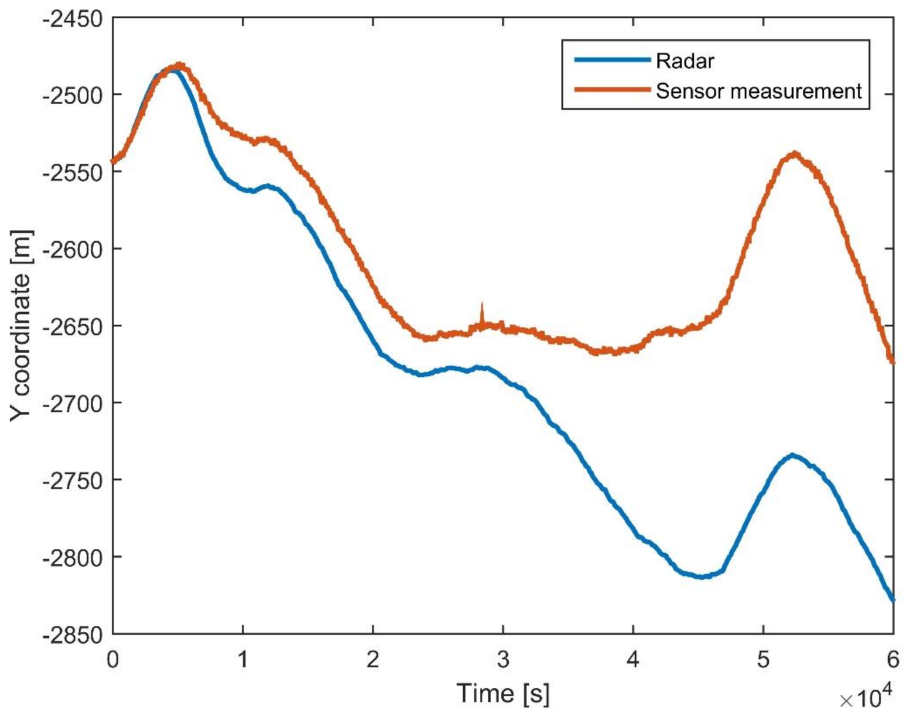


Figure 11. Comparison between Extended Range Kalman Filter estimates and Sensor measurements for Y coordinate

Clearly, there is a need of combination of radar measurements and satellite measurements in order to find the best estimate for rover position. This is done by applying the additional set of data to the Standard Kalman Filter code, resulting 5x5 matrix. Also, the previously discussed Jacobian matrix will now be function to all these 5 states introducing Extended Kalman Filter function.

Figure 12 compares the results for Extended and Standard Kalman filter. It is desirable so be mentioned that the propagation covariance has not been adjusted and both filters has the same order of uncertainty. Obviously, the Extended Kalman filter tends to be much closer to the measurement data and overlap it at almost all regions. The figure shows that noise of the measurement is almost gone and the rate of change of rover position due to its motion is well presented.

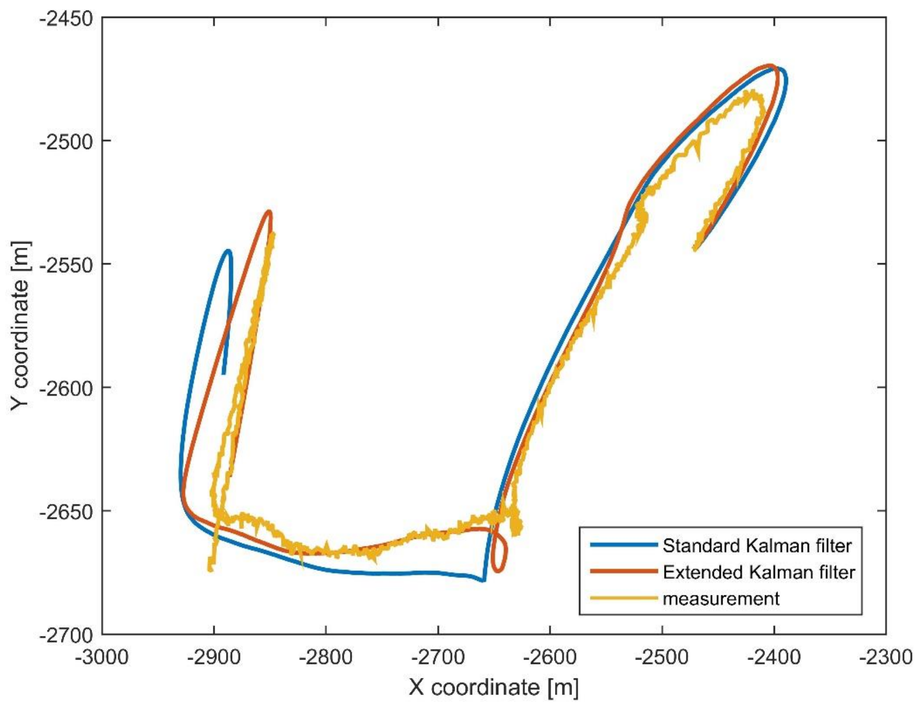


Figure 12. Comparison between Standard and Extended Kalman filter with Range correction

4. Conclusion

In this work, a methods for predicting and estimation of rover position by different Kalman Filter algorithms have been presented. At all cases the Kalman Filter solution tended to be closer to the estimated one and an adjustment in propagation covariance corrected filter solution to be closer to measured data. The Kalman filter was then evolved to Extended Kalman filter adding a couple of new functions to the Standard one. This has been done due to the non- linearity introduced from the added Range measurement.

The Extended Kalman filter improved significantly the results achieved by the standard one, but took more time to compute. Also, it was proven that estimating the rover position, by range only we not a sufficient approach, as the results deviate significantly from the true one. Further, literature review showed that range measurement only can be sufficient in case of 1D, or where two of the three position degrees of freedom are constants.

Finally, at all cases the results presented by Kalman filter were significantly less noisy comparing to the measurements, which is another advantage because of witch the filter is widely used.

References

[1] M. Maimone and J. Biesiadeck, "Surface navigation and mobility intelligence on," *Intelligence for Space Rover*, pp. 45-69, 2006.

[2] G. Bekey, "Autonomous Robots: From Biological Inspiration to Implementation and Control," The MIT Press: Cambridge, 2005.

[3] D. H. Titterton and J. I. Weston, "Strapdown Inertial Navigation Technology," The Institution of Engineering 2004.

[4] P. Kim, Kalman Filter for Beginners, A-JIN, 2010.

Appendix

Standard Kalman filter script

```
%estimate a new state from predicted
%and measured one
clear all
close all
%% Load files
load ('SDF2018a.mat')
%% Constants
P=1:1:1000;
N=1001;
dt=60;% time step
a=0.005*10^(-3); % acceleration magnitude
u=0.01; % velocity magnitude

delta_X= 50; % measurement standard deviation
delta_Y=50; % measurement standard deviation
% delta_Vx=a*dt; delta_Vy=a*dt;
%% initial conditions
Xk=[X(1);Y(1);Vx(1);Vy(1)]; %added manualy
uk=[Vx(1)/dt;Vy(1)/dt]; %added manualy

dx=0.5*a*dt^2;
dy=dx;
dvx=a*dt;
```

```
dvy=dvx;

A=[1 0 dt 0;
    0 1 0 dt;
    0 0 1 0;
    0 0 0 1]; % velocity design

B=[0.5*dt^2 0;
    0 0.5*dt^2;
    1 0
    0 1]; % acceleration design

H=[1 0 0 0;
    0 1 0 0;
    0 0 1 0;
    0 0 0 1];


R=[ delta_X^2 0 0 0;
    0 delta_Y^2 0 0;
    0 0 delta_X^2 0;
    0 0 0 delta_Y^2];
C=[1 0 0 0;
    0 1 0 0;
    0 0 1 0;
    0 0 0 1];

I=[1 0 0 0;
    0 1 0 0;
    0 0 1 0;
    0 0 0 1];
%% Errors
Wk=0;
Zk=0;
Qk=[dx^2 0 0 0;
    0 dy^2 0 0;
    0 0 dvx^2 0;
    0 0 0 dvy^2];
w=1;

%% Find initial process covarience matrix
Pk= [ 3.6563*10^3 0 0 0;
       0 1.483*10^4 0 0;
       0 0 1*10^-4 0;
       0 0 0 1*10^-4];

%% Trajectory
XX = zeros(4, size(1:1:1000,2)); %KF data
YY = zeros(4, size(1:1:1000,2)); %sensor data
ZZ = zeros(4, size(1:1:1000,2)); %predicted data
%% Step state
for i=1:numel(Step)
 % to be changed
%% Predicted state
% Xkp=A*Xk+Wk; %estimated prediction
```

```
Xkp=A*Xk+Wk+B*uk ;
uk=[Xkp(3)/dt; Xkp(4)/dt];
%%  Find predicted process covarience matrix
Pkp=(A*Pk*A'+w*Qk);
%% Find Kalman gain
K=(Pkp*H')/(H*Pkp*H'+R);
%% New observation (no Z no this step)
Yk=[X(i); Y(i); Vx(i); Vy(i)]; % observations correct
%% Calcolating the current state (use Xkp Yk)
Xk=Xkp+K*(Yk-H*Xkp);
%% Update process covarience metrix
Pk=(I-K*H)*Pkp;
%% Colect data
 XX(:,i) = Xk;
 ZZ(:,i) = Xkp;
 YY(:,i) = Yk;
end



% Plot data
figure
plot(1:1:1000, XX(1,:)), hold on;
plot(1:1:1000, YY(1,:)), hold on;
plot(1:1:1000, ZZ(1,:));
xlabel('Time [min]');ylabel('X coordinate [m]')
legend('Kalman filter','measurement','estimate');

figure
plot(1:1:1000, XX(2,:)), hold on;
plot(1:1:1000, YY(2,:)), hold on;
plot(1:1:1000, ZZ(2,:));
xlabel('Time [min]');ylabel('Y coordinate [m]')
legend('Kalman filter','measurement','estimate');


figure
plot(1:1:1000, XX(3,:)), hold on;
plot(1:1:1000, YY(3,:)), hold on;
plot(1:1:1000, ZZ(3,:));
xlabel('Time [min]');ylabel('X Velocity [m/s]')
legend('Kalman filter','measurement','estimate');

figure
plot(1:1:1000, XX(4,:)), hold on;
plot(1:1:1000, YY(4,:)), hold on;
plot(1:1:1000, ZZ(4,:));
xlabel('Time [min]');ylabel('Y Velocity [m/s]')
legend('Kalman filter','measurement','estimate');

figure
plot( XX(1,:), XX(2,:)), hold on;
plot( YY(1,:), YY(2,:)), hold on;
plot( ZZ(1,:), ZZ(2,:));
xlabel('X coordinate [m]');ylabel('Y coordinate [m]')
legend('Kalman filter','measurement','estimate');
```