Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 28, S. 152-156, Kasım 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 28, pp. 152-156, November 2021

Research Article

Kalman Filter Implementation on Field-programmable Gate Array for Navigation Applications of Unmanned Aerial Vehicles

Metin Mert Deniz[1*,2], Ufuk Sakarya[3]

[1*] Turkish Aerospace Industries, İstanbul, Turkey, (ORCID: 0000-0002-6370-4887), metinmert.deniz@tai.com.tr
[2] Yıldız Technical University, Graduate School of Science and Engineering, Department of Avionics Engineering, İstanbul, Turkey
[3] Yıldız Technical University, Faculty of Applied Sciences, Department of Aviation Electronics, İstanbul, Turkey, (ORCID: 0000-0002-8365-3415), usakarya@yildiz.edu.tr

(1st International Conference on Applied Engineering and Natural Sciences ICAENS 2021, November 1-3, 2021)

(**DOI:** 10.31590/ejosat.992118)

ATIF/REFERENCE: Deniz, M. M. & Sakarya, U., (2021). Kalman Filter Implementation on Field-programmable Gate Array for Navigation Applications of Unmanned Aerial Vehicles. *European Journal of Science and Technology*, (28), 152-156.

Abstract

In recent years, unmanned aerial vehicle (UAV) applications have been widely used in various manufacturing areas for the purpose of material handling or monitoring tasks. This situation increased the importance of proper estimation of UAVs' location. This paper presents hardware based Kalman Filter implementation for UAVs to accurately locate/detect its positions. To maintain high performance and compact form factor, Field-programmable Gate Array (FPGA) has been used as a hardware source. However, Kalman Filter algorithm needs lots of matrix computation and the typical implementation of matrix computations in hardware is complex and requires more effort than traditional software-based approaches. Matrix inversion computation in the Kalman gain formula is one of the most difficult matrix calculations in Kalman Filter algorithm and Chebyshev type inversion is used as a matrix inversion method to simplify hardware implementation. The proposed method simulated on both Matlab and Vivado based on the same scenario and numerical results of Kalman Filter and Chebyshev algorithm compared between these two simulation platforms. According to experimental results, the proposed solution serves compact and high performance standalone solution via FPGA for Kalman Filter implementation for UAVs.

Keywords: Unmanned Aerial Vehicle, Kalman Filter, Chebyshev Inversion, Field-programmable Gate Array, Navigation Application, Autonomous Systems.

İnsansız Hava Araçlarının Seyrüsefer Uygulamaları İçin Sahada Programlanabilir Kapı Dizisinde Kalman Filtresi Gerçekleştirmesi

Öz

Son yıllarda insansız hava aracı (İHA) uygulamaları, malzeme taşıma veya izleme görevleri amacıyla çeşitli imalat alanlarında yaygın olarak kullanılmaktadır. Bu durum İHA'ların yerinin doğru tahmin edilmesinin önemini arttırmıştır. Bu makale, İHA'ların konumlarının doğru bir şekilde konumlandırılması/tespit edilmesi için donanım tabanlı Kalman Filtresi uygulamasını sunmaktadır. İHA'ların yüksek performans ve kompakt form faktörünü korumak için, Alanda Programlanabilir Kapı Dizisi (FPGA) donanım kaynağı olarak kullanılmıştır. Bununla birlikte, Kalman Filtre algoritması çok sayıda matris hesaplamasına ihtiyaç duyar. Matris hesaplamalarının donanımda tipik uygulaması karmaşıktır ve geleneksel yazılım tabanlı yaklaşımlardan daha fazla çaba gerektirir. Kalman kazanç formülündeki matris ters çevirme hesaplaması, Kalman Filtre algoritmasındaki en zor matris hesaplamalarından biridir ve donanım uygulamasını basitleştirmek için bir matris ters çevirme yöntemi olarak Chebyshev tipi ters çevirme metodu kullanılmıştır. Önerilen yöntem, aynı senaryoya dayalı olarak hem Matlab hem de Vivado üzerinde simülasyonu yapılmıştır ve Kalman Filtresi ve Chebyshev algoritmasının sayısal sonuçları bu iki simülasyon platformu arasında karşılaştırılmıştır. Deneysel sonuçlara göre, önerilen çözüm, İHA'lara yönelik Kalman Filtre uygulaması için FPGA üzerinden kompakt ve yüksek performanslı bağımsız bir çözüm sunmaktadır.

Anahtar Kelimeler: İnsansız Hava Aracı, Kalman Filtresi, Chebyshev Matris Tersi Alma, Sahada Programlanabilir Kapı Dizisi, Navigasyon Uygulaması, Otonom Sistemler.

[*] Corresponding Author: metinmert.deniz@tai.com.tr

1. Introduction

The number of UAV implementations increase with the demand of autonomous systems in production areas that try to implement Industry 4.0 solutions. Because of that, the fast and accurate state estimation for UAVs became more necessary and important for these production areas such as factories. Study of Khosiawan & Nielsen (2016) can be given as indoor application for UAVs.

Kalman Filter is much known estimation algorithm and it has been used in diverse areas for navigation and control purposes (Kim & Bang, 2019). Hence it is suitable algorithm for state estimation of UAVs.

There are many software based Kalman Filter implementation has been in literature; but, UAV applications desires high performance in a light and compact form and thus, it can be a problem in software-based approaches (Soh & Wu, 2017).

Hardware implementation approaches offer high performance over software-based approaches in terms of a power usage and an execution-time. However, hardware implementation approaches tend to increase complexity and development time of overall design. In addition, they decrease the flexibility with its application specific structure. Field-programmable gate arrays (FPGAs) are suitable for reducing these disadvantages in contrast to traditional application-specific-integrated-circuit (ASIC) approaches. Nevertheless, they still need more development time than software based approaches (Soh & Wu, 2017).

Kalman Filter algorithm includes lots of matrix computations such as matrix multiplication and matrix inversion. As the dimension of matrixes increase, computation complexity increases. Performing matrix inversion calculation on FPGA is complex and takes large area. There are two implemented methods that are frequently used to calculate the inversion of a matrix in hardware: One of them is CORDIC algorithm (Lu et al., 2010). The other is QR decomposition (QRD) (Bai et al., 2012), (Stanislaus & Mohsenin, 2013). However, developing and implementing these algorithms in FPGA is not easy. Hence, Chebyshev algorithm have been used to find matrix inverse to purpose of decreasing complexity and development time (Rico-Aniles et al., 2014), (Rawal, 2015).

In this paper, Kalman Filter implementation on FPGA for navigation applications of UAVs is presented for the selected scenario. The matrix inversion has been implemented on FPGA with Chebsyhev type matrix inversion method that makes easier to development process of hardware implementation (Rico-Aniles et al., 2014), (Rawal, 2015). VHSIC (Very High Speed Integrated Circuit) Hardware Description Language (VHDL) design on FPGA is demonstrated by using Xilinx Vivado Design (WebPack) Program (Xilinx, 2021). The selected scenario is simulated using Matlab Program (Mathworks, 2021). There are two input sources for the selected scenario: The position and velocity values coming from GNSS receiver and the accelerometer values coming from the inertial measurement unit (IMU). These are generated by using Matlab and these are given into the FPGA simulation. The simulation results which are obtained by using Matlab are accepted as ground truth. The simulation results which are obtained by using FPGA simulation are compared with the ground truth results.

The rest of the paper is planned as follows: The next section presents the related works from literature. In Section 3, the proposed method is introduced. The selected scenario and the simulation results are demonstrated in Section 4. Finally, the last section gives some concluding remarks and future issues.

2. Related Works

Kalman Filter is an estimation method (Kim & Bang, 2019). It takes series of measurements as inputs and it generates estimates of unknown variables as outputs. It can be used in several areas for the purpose of navigation using IMU/GNSS, terrain-referenced navigation (TRN), battery-range estimation, target tracking, control systems and much more. In this work, our interest is Kalman Filter for the navigation application by using the position and velocity values coming from GNSS receiver and the accelerometers values coming from the inertial measurement unit (IMU).

To use a Kalman Filter in GNSS application, first step is establishing a proper model with state vector and measurement vector. In established model, the UAV state (position and velocity) are estimated. Position and velocity (in 3-D) construct the state vector below that dimension is 6 (Kim & Bang, 2019):

$$x = [p^T, v^T]^T \qquad \text{Eq. (1)}$$

where p is position vector, v is the velocity vector in 3 dimensional space. Then, state vector in time k can be estimated by using the previous state vector in time k-1 as;

$$x_k = Fx_{k-1} + Ba_{k-1} + w_{k-1} \qquad \text{Eq. (2)}$$

where B is a matrix (control), F is a matrix (state transition), w is a noise generated by process.

Measurement vector can be formed as;

$$z_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix} + v_k \qquad \text{Eq. (3)}$$

$$z_k = Hx_k + v_k \qquad \text{Eq. (4)}$$

where v_k is measurement noise and H is measurement matrix.

After filter model is established, filter algorithm can be started. Kalman Filter comprises of two steps that are prediction and update. In prediction step, estimations are made according to previous measurement and estimations Eq. (5) and Eq. (6). In update step, measurement residual and Kalman gain are calculated according to new measurements Eq. (7) and Eq. (8). Then, state estimate and error is updated based on Kalman gain Eq. (9) and Eq. (10). The prediction and update formulas are below (Kim & Bang, 2019):

$$x_k = Fx_{k-1} + Bu_{k-1} \qquad \text{Eq. (5)}$$

$$P_k = FP_{k-1}F^T + Q \qquad \text{Eq. (6)}$$

$$y_k = z_k - Hx_k \qquad \text{Eq. (7)}$$

$$K_k = P_kH^T(R + HP_kH^T)^{-1} \qquad \text{Eq. (8)}$$

$$x_k = x_k + K_ky \qquad \text{Eq. (9)}$$

$$P_k = (I - K_kH)P_k \qquad \text{Eq. (10)}$$

where I is identity matrix and R is measurement noise covariance matrix. In order to obtain detail information for this issue, can be studied from Kim & Bang (2019).

Matrix inversion implementation on FPGA is not easy. One of the solution approaches for matrix inversion implementation on FPGA is Chebyshev algorithm. The Chebyshev-type matrix inversion method is given by Rico-Aniles et al.(2014) and Rawal (2015).

$$N_{m+1} = N_m(3I - AN_m(3I - AN_m)) \qquad \text{Eq. (11)}$$

where $N_{m+1}$is a next inverse approximation, N_m is a previous inverse approximation and A is a matrix to be inverted.

This algorithm starts with an initial guess and continues iteratively until to make true estimation. Hence it is important to give a proper initial estimate, otherwise it cannot converge.

Proper initial guess that assures the method's convergence can be made with Eq. (12) (Rico-Aniles et al., 2014), (Rawal, 2015) :

$$N_0 = \frac{A^T}{\|A\|_1 \|A\|_\infty} \qquad \text{Eq. (12)}$$

where N_0 is initial guess, A^T is transpose of A, $\|A\|_1$ is the maximum value of the summation of the elements on the each column, $\|A\|_\infty$ is the maximum value of the summation of the elements on the each row.

The entire algorithm can be summarized as below (Rico-Aniles et al., 2014), (Rawal, 2015):

Input: Matrix A

Precondition: $N_0 = \frac{A^T}{\|A\|_1 \|A\|_\infty}$

Iteration:

$N_{m+1} = N_m(3I - AN_m(3I - AN_m))$

Verification:

If $A * N_{m+1} \approx I$ stop the algorithm

Else $N_m = N_{m+1}$ continue iterative stage

Output: N_{m+1} as A^{-1}

3. Proposed Method

In this section, the proposed state estimation method for UAV applications is presented. The proposed method uses Kalman Filter as an algorithm and uses FPGA as a hardware source.

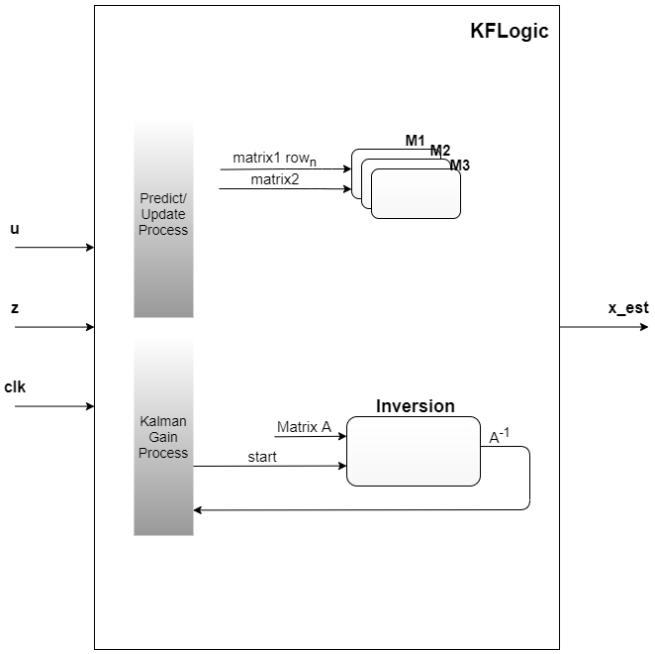

Fig. 1 System blog diagram

System blog diagram is shown in Fig 1. The main inputs of the system are measurements z (position and velocity vector) and u (accelerometer values). The main output of the system is updated state estimate x_est (position and velocity vectors) that represents estimation of new state.

KF Logic refers to Kalman Filter state machine algorithm that manages prediction and update processes. The main Kalman filter calculations are done in block that are state estimate prediction, error covariance prediction, calculation of measurement residual, calculation of Kalman gain, updating state estimate and updating error covariance. The multiplier and inversion sub blocks include matrix calculation algorithms for KF Logic main block.

The M from 1 to 3 refers multipliers with different sizes for matrix multiplications that are 6x6 prod 6x6, 6x6 prod 6x1, 6x3 prod 3x1. Multipliers use DSP48 slices on FPGA. Because of the DSP48 slice count is limited on FPGA, it is not possible to calculate all row and columns of matrix multiplication simultaneously. Hence, for all multiplier sizes, first matrix is taken row by row and multiplication is performed. In order to perform a matrix multiplication with two 6x6 matrices, this module needs to be used 6 times.

In the proposed method, VHDL is used as a hardware description language and fixed point representation used as a data type. To avoid overflow of fixed point data while sum and multiplication calculations, Q16,16 representation is used that means 16 bits for integer part, 16 bits for fractional part. In this way, it guarantees the represent our calculations in range without overflow. Xilinx fixed point library is used to design for fixed point calculations (Xilinx, 2021).

KFLogic block includes two processes that are predict/update and Kalman gain calculation. Whereas predict/update process includes calculation of x_k (Eq.5), P_k (Eq.6, Eq.9 and Eq.10); Kalman gain calculation process includes calculation of y (Eq.7) and K (Eq. 8). Predict/update process starts with prediction of state estimate and error covariance. These calculations include matrix multiplications and to perform these calculations, multipliers are used in sub-blocks. When the state estimate prediction is done, Predict/update process sends a signal to Kalman gain calculation process in order to start measurement residual and Kalman gain calculation. Matrix inverse calculation is required for Kalman gain (Eq. 8). In the proposed system Chebyshev inverse algorithm from Rico-Aniles et al.(2014) is used to overcome matrix inverse calculation in Kalman gain formula. The inversion block on Fig. 1 includes this algorithm implementation. The Chebyshev inverse algorithm is composed of three parts that are preconditioning, iterative and verification states. When the Kalman gain calculation process sent to start signal to the inversion block to calculate inverse of $(R + HP_kH^T)$ in the Kalman gain formula, the Chebyshev algorithm starts preconditioning process with this flag and firstly performs its initial guess N_0. This is an iterative algorithm, in each iteration find an estimation matrix and check estimation matrix multiply input matrix equals to unit matrix in verification state. If not equals iteration continues. It is observed that generally it takes 8-15 cycle to find inverse matrix. When the inverse calculation is done, it sends a finish signal to KFLogic to continue its Kalman filter process.

When the Kalman gain is calculated, predict/update process updates state estimate and error covariance (Eq. 9 and Eq. 10). One iteration of Kalman filter is completed and the algorithm waits for new measurements to pass a new iteration.

4. Experimental Simulation Results

In order to test the proposed design, testbench feature of Xilinx Vivado Design (WebPack) Program is used (Xilinx, 2021). Same scenario is simulated on both Matlab and Vivado testbench, then compared. In order to simulate the selected scenario on Matlab, it has been benefited from Introduction to Kalman Filter and Its Applications website (2021) and the selected scenario is defined as follows.

In the selected scenario, UAV is assumed to operate in outdoor space in a factory. UAV is moving only in x-axis with constant speed that is 2m/s and located at (0,0,0) as initial true position. For the initial state position, UAV is supposed to located at (1,1,1). Its velocity is 2m/s in x-axis and y-axis, 0m/s in z-axis that compose the initial system state vector as [1; 1; 1; 2; 2; 0].

The main inputs, that are accelerometer, GNSS position and GNSS velocity values, are generated and corrupted with noise with randn function in Matlab at every step. Standard deviation value is selected 0.6 m/s^2 in three axes for accelerometer, selected 3m in three axes for GNSS position and selected 0.06 m/s in three axes for GNSS velocity. Then, these input values are given to Matlab and essential vector values are generated. Thus, these vectors, i.e. u and z, are given to Vivado simulations.

Estimated position and velocity values on Matlab can be seen in Fig. 2 and Fig. 3. Also, Vivado simulation of the same values can be seen in Fig. 4 and Fig.5.

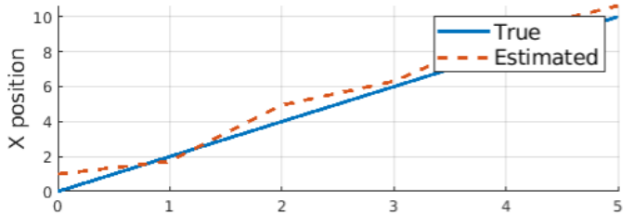

Fig. 2 Estimated position on Matlab

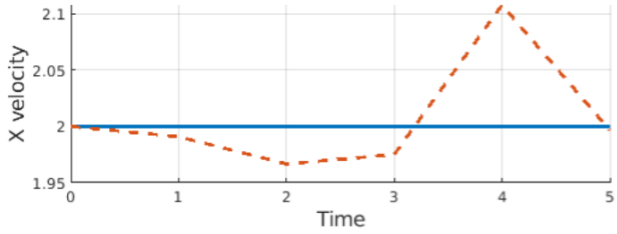

Fig. 3 Estimated velocity on Matlab

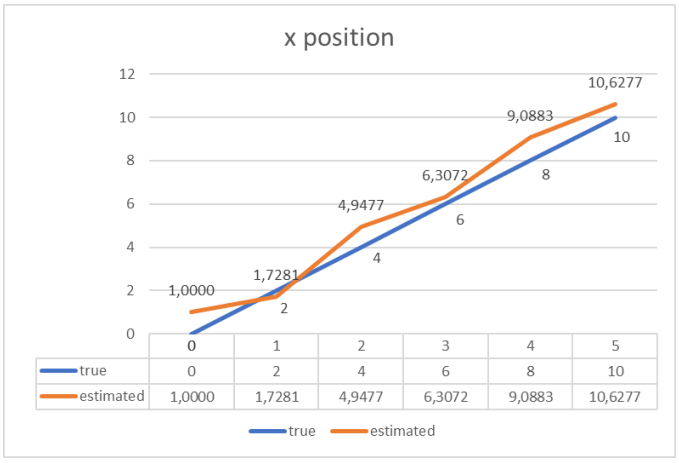

Fig. 4 Estimated position on Vivado

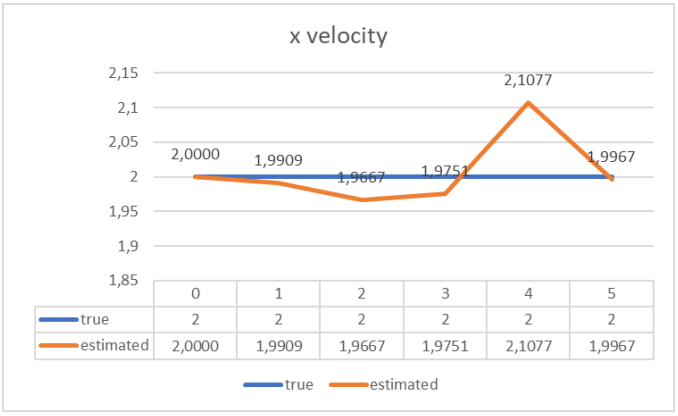

Fig. 5 Estimated velocity on Vivado

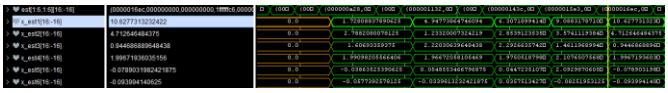

Fig. 6 Vivado testbench simulation of estimated position

On both Matlab and Vivado, estimated values are consistent with true values. Hence, it can be said that the Kalman Filter is working properly on both simulations.

Table 1 Comparison table between Matlab and Vivado

Time	Estimated position		Estimated velocity	
	Matlab	Vivado	Matlab	Vivado
0	1,00000	1,00000	2,00000	2,00000
1	1,72770	1,72808	1,99090	1,99090
2	4,94770	4,94773	1,96670	1,96670
3	6,30820	6,30718	1,97510	1,97505
4	9,08920	9,08830	2,10770	2,10765
5	10,62860	10,62770	1,99670	1,99671

Also, when estimated position and velocity values on both simulations are compared to each other (Table 1), it can be understood that, hardware solution on FPGA works with high precision.

In addition to entire Kalman Filter simulation, Chebyshev algorithm is tested individually on Matlab and its iteration count to find inverse matrix are nearly the same in FPGA approach. However; because of hardware is faster than software, this FPGA implementation saves a huge amount of time as the iteration count increase. In Fig. 7 Inverse of matrix A that size is 6x6 is calculated in Matlab in 12 iterations. In Fig. 8, inverse of same matrix is calculated in Vivado testbench.

```
iteration = 12
A = 6×6
   25.1825         0         0    0.1600         0         0
         0   25.1825         0         0    0.1600         0
         0         0   25.1825         0         0    0.1600
    0.1600         0         0    0.2509         0         0
         0    0.1600         0         0    0.2509         0
         0         0    0.1600         0         0    0.2509

Nest = 6×6
    0.0399         0         0   -0.0254         0         0
         0    0.0399         0         0   -0.0254         0
         0         0    0.0399         0         0   -0.0254
   -0.0254         0         0    4.0019         0         0
         0   -0.0254         0         0    4.0019         0
         0         0   -0.0254         0         0    4.0019

ANest = 6×6
    1.0000         0         0    0.0000         0         0
         0    1.0000         0         0    0.0000         0
         0         0    1.0000         0         0    0.0000
    0.0000         0         0    1.0000         0         0
         0    0.0000         0         0    1.0000         0
         0         0    0.0000         0         0    1.0000
```

Fig. 7 Chebyshev method tested on Matlab

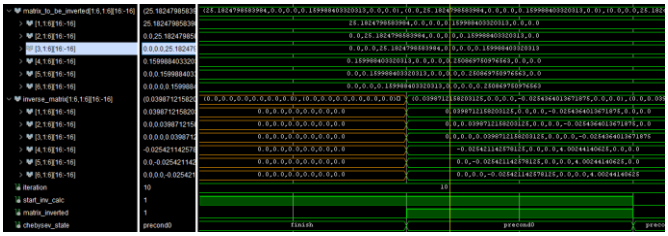

Fig. 8 Chebyshev method tested on Vivado testbench

5. Conclusion

In this paper, Kalman Filter implementation on FPGA for navigation applications of UAVs is demonstrated. Synthesizable VHDL design on FPGA is presented. The selected scenario is simulated and then it is examined on FPGA simulation. Experimental simulation results show that VHDL design on FPGA is validated. On the other hand, there is a future issue that the proposed method can be examined on the hardware by using implementation of the proposed VHDL design.

References

Bai, L., Maechler, P., Muehlberghuber, M., & Kaeslin, H. (2012). High- speed compressed sensing reconstruction on FPGA using OMP and AMP. *2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012)*. doi:10.1109/icecs.2012.6463559

Introduction to Kalman Filter and Its Applications website. (2021). Mathworks.https://www.mathworks.com/matlabcentral/fileexchange/68262-introduction-to-kalman-filter-and-its-applications

ISE WebPACK Design Software website. (2021). Xilinx. https://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.html

Khosiawan, Y., & Nielsen, I. (2016). A system of UAV application in indoor environment. *Production & Manufacturing Research, 4*(1), 2-22. doi:10.1080/21693277.2016.1195304

Kim, Y., & Bang, H. (2019). Introduction to Kalman Filter and Its Applications. *Introduction and Implementations of the Kalman Filter*. doi:10.5772/intechopen.80600

Lu, J., Zhang, H., & Meng, H. (2010). Novel hardware architecture of sparse recovery based on FPGAs. *2010 2nd International Conference on Signal Processing Systems*. doi:10.1109/icsps.2010.5555628

Mathworks website. (2021). https://www.mathworks.com/

Rawal, N. (2015). HDL implementation of Kalman Filter for GNSS receiver. *2015 IEEE International Advance Computing Conference (IACC)*. doi:10.1109/iadcc.2015.7154717

Rico-Aniles, H. D., Ramirez-Cortes, J. M., & Rangel-Magdaleno, J. D. (2014). FPGA-based matrix inversion using an iterative Chebyshev-type method in the context of compressed sensing. *2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*. doi:10.1109/i2mtc.2014.6860890

Soh, J., & Wu, X. (2017). An FPGA-Based Unscented Kalman Filter for System-On-Chip Applications. *IEEE Transactions on Circuits and Systems II: Express Briefs, 64*(4), 447-451. doi:10.1109/tcsii.2016.2565730

Stanislaus, J. L., & Mohsenin, T. (2013). Low-complexity FPGA implementation of compressive sensing reconstruction. *2013 International Conference on Computing, Networking and Communications (ICNC)*. doi:10.1109/iccnc.2013.6504167