

# FPGA-base object tracking: integrating deep learning and sensor fusion with Kalman filter

Abdoul Moumouni Harouna Maloum<sup>1</sup>, Nicasio Maguu Muchuka<sup>2</sup>, Cosmas Raymond Mutugi Kiruki<sup>3</sup>

<sup>1</sup>Department of Electrical Engineering, Pan African University Institute for basic Sciences Technology and Innovation, Nairobi, Kenya

<sup>2</sup>Department of Electrical and Control Engineering, Egerton University, Nakuru, Kenya

<sup>3</sup>Department of Electrical and Information Engineering, University of Nairobi, Nairobi, Kenya

## Article Info

### Article history:

Received Jan 18, 2024

Revised Feb 22, 2024

Accepted Feb 23, 2024

### Keywords:

Autonomous vehicle

FPGA

Kalman filter

Object tracking

Sensor fusion

## ABSTRACT

This research presents an integrated approach for object detection and tracking in autonomous perception systems, combining deep learning techniques for object detection with sensor fusion and field programmable gate array (FPGA-based) hardware implementation of the Kalman filter. This approach is suitable for applications like autonomous vehicles, robotics, and augmented reality. The study explores the seamless integration of pre-trained deep learning models, sensor data from a depth camera, real-sense D435, and FPGA-based Kalman filtering to achieve robust and accurate 3D position and 2D size estimation of tracked objects while maintaining low latency. The object detection and feature extraction are implemented on a central processing unit (CPU), and the Kalman filter sensor fusion with universal asynchronous receiver transmitter (UART) communication is implemented on a Basys 3 FPGA board that performs 8 times faster compared to the software approach. The experimental result provides the hardware resource utilization of about 29% of look-up tables, 6% of lookup table RAMs (LUTRAM), 15% of Flip-flops, 32% of Block-RAM, 38% of DSP blocks operating at 100 MHz, and 230400 baud rates for the UART. The whole FPGA design executes at 2.1 milliseconds, the Kalman filter executes at 240 microseconds, and the UART at 1.86 milliseconds.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Abdoul Moumouni Harouna Maloum

Department of Electrical Engineering

Pan African University Institute for basic Sciences Technology and Innovation

Nairobi, Kenya

Email: moumouni.abdoul@students.jkuat.ac.ke

## 1. INTRODUCTION

Autonomous perception systems have a crucial role in various applications, including autonomous vehicles, robotics, and augmented reality. These systems heavily rely on the timely and accurate estimation of object states within their environment [1], [2]. These system face challenges in maintaining accuracy amidst environmental variations, occlusions, and dynamic elements. Real-time processing demands prompt decision-making under latency constraints, while semantic comprehension and context awareness necessitate the interpretation of observed data [3], [4]. Perception systems necessity mandates the use of sensor fusion techniques, enabling the integration of data from multiple sensors. Additionally, leveraging computer accelerators facilitates quick estimation, enhancing the efficiency and responsiveness of these systems [5], [6]. Sensor's fusion improves autonomous perception systems by combining data from multiple sensors, enhancing the system's understanding of the environment. By integrating information from cameras, light detection and ranging (LiDAR), radar, and inertial measurement unit (IMU), sensor fusion compensates for

individual sensor limitations and uncertainties. This comprehensive data enables more robust object perception, even in challenging conditions, and facilitates redundancy and cross-validation for improved reliability. Ultimately, sensor fusion enhances the perception capabilities of autonomous systems, enabling more informed decision-making and confident navigation in complex environments [7], [8].

The Kalman filter is a widely utilized algorithm for multi-sensor fusion, essential for integrating data from various sensors to estimate system states accurately. Employing iterative prediction and correction steps, it navigates the noise and uncertainty inherent in sensor data, offering reliable estimates. Its versatility makes it invaluable in applications with diverse measurement errors or disturbances, notably in dynamic systems like navigation and control. However, its computational intensity, stemming from matrix operations and iterative computations, can be a challenge, particularly in scenarios with large and complex state spaces or limited computational resources where real-time processing is critical [9], [10]. The common method used in implementing sensor fusion algorithms on processing units such as central processing units (CPUs) or microcontroller units (MCUs) faces the challenge of meeting real-time requirements in the dynamic and complex environments of autonomous vehicles, robotics, and control systems. The computational demand arises from the need to process data from diverse environmental sensors, in real-time. These systems must perform complex calculations, such as Kalman filtering for sensor fusion, to accurately perceive, interpret, and respond to the surrounding environment. However, the limitations in processing power and speed of traditional CPUs and MCUs pose constraints on achieving the stringent real-time deadlines required for ensuring the safety and efficiency of these intelligent systems [11], [12]. As a result, there is a growing interest in exploring more advanced hardware solutions, such as field programmable gate array (FPGAs) or dedicated accelerators, to address the evolving demands of real-time sensor fusion applications [13], [14]. FPGAs offer flexibility and adaptability for various applications, such as sensor fusion in autonomous systems. Their parallel processing capabilities enable the execution of various sensor fusion algorithms and manage high-speed data transport for real-time perception. Their ability to interpret complex signals and extract features is crucial for accurate world perception [15].

Kumar *et al.* [16], proposed a point-to-pixel multi-sensor data fusion algorithm for autonomous vehicle forward collision warning system. By combining LiDAR 2D and stereo camera image pixel data, the system detects, classifies, and tracks the objects obstructions in real-time. Results from the proposed algorithm reduces root mean square error (RMSE) to 93.8 mm and MAE to 83.45 mm, and reduces uncertainty and variance. A multi-sensor platform was used in [17] to mitigate the LIDAR and camera semantic fusion problems. A three-dimensional semantic voxelized map that considers the uncertainty of all relevant processes was used. The method offered a probabilistic pipeline, motion compensation, and heuristic label probabilities for semantic images. It also provides a perspective validation mechanism, generating probabilistic projections from camera images to LIDAR point clouds. This approach addressed the challenges of the probabilistic handling of complex issues in camera and LiDAR projection and fusion. A decentralized target tracking method using an asynchronous camera network was studied in [18]. The cameras exchange the line-of-sight vectors and time tags, local decentralized Kalman information and particle filtering for target state estimation. The performance of the technique was measured using the average RMSE of 3D position estimates. Farag [19], developed a real-time road object detection and tracking (LR ODT) technique for self-driving cars by combining radar and lidar sensors data abroad an ego vehicle. They compared the performance of the extended and unscented Kalman filtering processes. Simulation results showed that the unscented Kalman filter outperforms the extended Kalman filter in terms of RMSE.

Jahromi *et al.* [20] proposed a hybrid multi-sensor fusion pipeline for autonomous vehicles, combining conventional methods the extended Kalman filter and encoder-decoder-based fully convolutional neural network. This approach enables environment perception, road segmentation, obstacle detection, and tracking. The framework also includes radar, LiDAR, and camera sensors for object detection and identification. The method achieves an average tracking RMSE of 0.065 and 0.061 for x and y axis. Garcia-Huerta *et al.* [21] studied and compared two linear models for estimating the position and attitude of precision aerial delivery systems (PADSSs) using a 6-DOF model and a double integrator model. Simulation of both models using a sensor fusion approach and a Kalman filter, showed performance in smooth flights and modest acceleration changes. A discrete-time minimum variance unbiased estimator (MVUE) that is resistant to unidentified factors and aids in unbiased state estimates is suggested in [22]. They suggested unknown parameter estimation and a sequential state technique for railway track geometry inspection, inspired by the MVUE. An investigation of the simulation highlights the significance of creating a decoupled unknown parameter estimator, a state estimation framework, and a state-space realization for track geometry inspection. Also, the MVUE's estimation gain aids in unbiased state estimates. Qiu *et al.* [23], presented the interacting multiple models and the adaptive Kalman filter (IMMCAKF) for a target tracking algorithm in underwater acoustic sensor networks (UASNs). The centralized fusion adaptive Kalman filter (CAKF) algorithm is first found by adding an adaptive forgetting factor to the best centralized fusion Kalman filter

method. Numerical simulation offered that IMMCKFAKF provided better tracking accuracy as compared to other single Kalman filter process.

Babu and Parthasarathy [24], the multi-dimensional Kalman filter (MDKF) technique for object tracking and motion detection was proposed by the authors. In comparison to cutting-edge tracking algorithms learned on benchmarks that are accepted by the industry, the numerical evaluation of the proposed tracking method produces tracking results that are competitive. Additionally, the Xilinx Zynq™-7000 System-on-a-Chip was used to implement MDKF. When it was implemented on a SoC, the performance is two times faster than when software is used. The experimentation's finding indicates that at 140 MHz and 780 mW of power usage, the resource utilization of around 61.43% of block RAMs, 90.09% of digital signal processors (DSPs), 83.27% of look-up tables (LUT), and 82.35% of logic cells. Li and Li [25], developed an FPGA system for industrial ultrasonic applications, consisting of an adaptive Kalman filter (AKF) module and a feedback sensing circuit module. The system correlates temperature and pressure, generating the best working frequency based on feedback voltage, current, and phase difference. Results indicated that the system noise was effectively mitigated by the FPGA modules which leads to clear and smooth feedback recognition. Castells-Rufas *et al.* [26], conducted a survey on computer vision in automotive applications, focusing on FPGA-based works. They highlighted the ongoing efforts to improve safety and meet technical and regulatory requirements for self-driving cars. They highlighted the need for complex algorithms, which often require large computing platforms and processing speeds that fall short of real-time requirements. They suggested that latency-critical algorithms are often implemented on FPGA devices.

Based on the reviewed available literature, there are problems of estimation accuracy in real time perspective of autonomous perception systems and robustness against noise interference. This paper investigates the approach that combining the advantage of deep learning object detection, Kalman filter sensor fusion for accurate object state estimation and FPGA-based accelerator that reduce the latency. The proposed FPGA-based sensor fusion aims to guarantee the system's delivery of accurate, rapid, and effective object state estimation. The remaining of the paper contains system modelling and experimentation in section 2. Section 3 presents the results and discussion. Section 4 concludes the work.

## 2. SYSTEM MODELING AND EXPERIMENTATION

The implementation of the system involves modelling and experimentation procedures. The processes of the Kalman Filter sensor fusion designing, sensor calibration, python designing, FPGA implementation and experimentation are explained in detailed. Also, the model equations used in the study are fully provided.

### 2.1. Kalman filter sensor fusion design

The extended Kalman filter state equation [27] is presents in (1):

$$X_k = f(x_{k-1}, U_{k-1}) + W_{k-1} \quad (1)$$

extended Kalman filter measurement information equation [27]:

$$Z_k = h(x_k) + V_k \quad (2)$$

in kinematic constant velocity model [28]:

$$\begin{cases} x_{a,k} = x_{a,k-1} + \Delta T \dot{x}_{a,k-1} \\ \dot{x}_{a,k} = \dot{x}_{a,k-1} \\ \ddot{x}_{a,k} = 0 \end{cases} \quad (3)$$

where  $\ddot{x}_{a,k}$  is acceleration and  $\Delta T$  is sampling period. Since an assumption of perfect constant velocity is unrealistic for real world applications; the realistic process model is given as (4):

$$\begin{cases} x_{a,k} = x_{a,k-1} + \Delta T \dot{x}_{a,k-1} + \frac{\Delta T^2}{2} w_{x,k-1} \\ \dot{x}_{a,k} = \dot{x}_{a,k-1} + \Delta T w_{x,k-1} \\ \ddot{x}_{a,k} = 0 \end{cases} \quad (4)$$

where  $\ddot{x}_{a,k} = w_{x,k-1}$  piecewise constant white acceleration, can be describe as by a zero-mean Gaussian white noise as (5).

$$w_{x,k-1} = \ddot{x}_{a,k-1} \sim N(0, \sigma_{x,k-1}^2) \quad (5)$$

With this assumption of constant velocity, the model state equations of object being tracked are as follows: the estimation state are: 3D position (x, y, x), implicit 3D velocity ( $\dot{x}$ ,  $\dot{y}$ ,  $\dot{z}$ ) and 2D size (height and width):

$$\begin{cases} x_{a,k} = x_{a,k-1} + \Delta T \dot{x}_{a,k-1} + \frac{\Delta T^2}{2} w_{x,k-1} \\ y_{a,k} = y_{a,k-1} + \Delta T \dot{y}_{a,k-1} + \frac{\Delta T^2}{2} w_{y,k-1} \\ z_{a,k} = z_{a,k-1} + \Delta T \dot{z}_{a,k-1} + \frac{\Delta T^2}{2} w_{z,k-1} \\ \dot{x}_{a,k} = \dot{x}_{a,k-1} + \Delta T w_{x,k-1} \\ \dot{y}_{a,k} = \dot{y}_{a,k-1} + \Delta T w_{y,k-1} \\ \dot{z}_{a,k} = \dot{z}_{a,k-1} + \Delta T w_{z,k-1} \\ h_{a,k} = h_{a,k-1} + w_{h,k-1} \\ wh_{a,k} = w_{a,k-1} + w_{wh,k-1} \end{cases} \quad (6)$$

where  $\frac{\Delta T^2}{2} w_{x,k-1}$ ,  $\frac{\Delta T^2}{2} w_{y,k-1}$ , and  $\frac{\Delta T^2}{2} w_{z,k-1}$  are process noise for x, y and z position coordinate,  $\Delta T w_{x,k-1}$ ,  $\Delta T w_{y,k-1}$ , and  $\Delta T w_{z,k-1}$  are process noise for x, y, and z indirect velocity coordinate, and  $w_{h,k-1}$ , and  $w_{wdt,k-1}$  are process noise for height, and width size coordinate.

After equations linearization and matrix formatting, and there is no external control input  $U_{k-1} = 0$ . the extended Kalman filter sensor fusion state as (7):

$$X_k = f(x_{k-1}) + W_{k-1} \quad (7)$$

$$\begin{bmatrix} x_{a,k} \\ y_{a,k} \\ z_{a,k} \\ \dot{x}_{a,k} \\ \dot{y}_{a,k} \\ \dot{z}_{a,k} \\ h_{a,k} \\ wh_{a,k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta T & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{a,k-1} \\ y_{a,k-1} \\ z_{a,k-1} \\ \dot{x}_{a,k-1} \\ \dot{y}_{a,k-1} \\ \dot{z}_{a,k-1} \\ h_{a,k-1} \\ wh_{a,k-1} \end{bmatrix} + \begin{bmatrix} (\Delta T^2/2)w_{x,k-1} \\ (\Delta T^2/2)w_{y,k-1} \\ (\Delta T^2/2)w_{z,k-1} \\ \Delta T w_{x,k-1} \\ \Delta T w_{y,k-1} \\ \Delta T w_{z,k-1} \\ w_{h,k-1} \\ w_{wh,k-1} \end{bmatrix} \quad (8)$$

$$X_k = F_{k-1} X_{k-1} + W_{k-1} \quad (9)$$

process noise covariance  $Q_{k-1}$ :

$$Q_{k-1} = Conv(W_{k-1}) = E[W_{k-1}W_{k-1}^T] \quad (10)$$

where  $E[W_{k-1}W_{k-1}^T]$  is the expected vale or mean of  $W_{k-1}W_{k-1}^T$  and  $W_{k-1}^T$  is the transpose of  $W_{k-1}$ :

$$Q_{k-1} = Conv(W_{k-1}) = E[W_{k-1}W_{k-1}^T] \quad (11)$$

$$= \begin{bmatrix} (\Delta T^4/4)\sigma_{w_x}^2 & 0 & 0 & (\Delta T^3/2)\sigma_{w_x}^2 & 0 & 0 & 0 & 0 \\ 0 & (\Delta T^4/4)\sigma_{w_y}^2 & 0 & 0 & (\Delta T^3/2)\sigma_{w_y}^2 & 0 & 0 & 0 \\ 0 & 0 & (\Delta T^4/4)\sigma_{w_z}^2 & 0 & 0 & (\Delta T^3/2)\sigma_{w_z}^2 & 0 & 0 \\ (\Delta T^3/2)\sigma_{w_x}^2 & 0 & 0 & \Delta T^2\sigma_{w_x}^2 & 0 & 0 & 0 & 0 \\ 0 & (\Delta T^3/2)\sigma_{w_y}^2 & 0 & 0 & \Delta T^2\sigma_{w_y}^2 & 0 & 0 & 0 \\ 0 & 0 & (\Delta T^3/2)\sigma_z^2 & 0 & 0 & \Delta T^2\sigma_{w_z}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_h^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{wh}^2 \end{bmatrix} \quad (12)$$

where from the derivation of  $Q_{k-1}$ :

$$- E[W_{x,k-1}, W_{x,k-1}] = \sigma_{w_x}^2, E[W_{y,k-1}, W_{y,k-1}] = \sigma_{w_y}^2, E[W_{z,k-1}, W_{z,k-1}] = \sigma_{w_z}^2, E[W_{h,k-1}, W_{h,k-1}] = \sigma_{w_h}^2, \text{ and } E[W_{wh,k-1}, W_{wh,k-1}] = \sigma_{w_{wh}}^2;$$

where  $\sigma_{w_x}^2$  is the variance ( $\sigma_{w_x} = \sqrt{\sigma_{w_x}^2}$  the standard deviation).

- $E[W_{x,k-1}, W_{y,k-1}] = E[W_{x,k-1}, W_{y,k-1}] = E[W_{x,k-1}, W_{z,k-1}] = 0$ , because they are uncorrelated, similar to  $E[W_{h,k-1}, W_{wh,k-1}] = 0$ ;

The derivation of  $H_k$  and  $R_k$ ,  $H_k$ , and  $R_k$  is derived using the approach used for  $F_{k-1}$  and  $Q_{k-1}$  derivation.

The observation at time k is described in (13):

$$\begin{cases} Z_{x,k} = x_{a,k} + v_{x,k} \\ Z_{y,k} = y_{a,k} + v_{y,k} \\ Z_{z,k} = z_{a,k} + v_{z,k} \\ Z_{h,k} = h_{a,k} + v_{h,k} \\ Z_{wh,k} = wh_{a,k} + v_{wh,k} \end{cases} \tag{13}$$

where  $Z_{x,k}$ ,  $Z_{y,k}$ , and  $Z_{z,k}$  are position measurement from centroid of an object detection box projected on depth image at time k, and  $Z_{h,k}$ , and  $Z_{wh,k}$  are detected object size (height and width) measurement from detection box in color image (RGB) coordinate at time k,  $v_{x,k}$ ,  $v_{y,k}$ ,  $v_{z,k}$ ,  $v_{h,k}$ , and  $v_{wh,k}$  are observation noise to  $Z_{x,k}$ ,  $Z_{y,k}$ ,  $Z_{z,k}$ , and  $Z_{h,k}$ , and  $Z_{wh,k}$  respectively which are basically zero-mean Gaussian white noise for instance  $v_{x,k} \sim N(0, \sigma_{z_{x,k}}^2)$ , the state space model in the previous equations are represented using the following state space model.

$$\begin{bmatrix} Z_{x,k} \\ Z_{y,k} \\ Z_{z,k} \\ Z_{h,k} \\ Z_{wh,k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{a,k} \\ y_{a,k} \\ z_{a,k} \\ \dot{x}_{a,k} \\ \dot{y}_{a,k} \\ \dot{z}_{a,k} \\ h_{a,k} \\ wh_{a,k} \end{bmatrix} + \begin{bmatrix} v_{x,k} \\ v_{y,k} \\ v_{z,k} \\ v_{h,k} \\ v_{wh,k} \end{bmatrix} \tag{14}$$

$$Z_k = H_k X_k + V_k \tag{15}$$

The  $H_k$  is arranged in such a way that it maps the state space into the observation space  $Z_k$  can be expressed as:  $Z_k = H_k \cdot X_k + v_k$ .

The expression of  $R_k$  can be obtained by taking the covariance of  $V_k$  as in (16) and (17):

$$R_k = cov(V_k) = E[V_k, V_k^T] \tag{16}$$

$$R_k = \begin{bmatrix} \sigma_{v_x}^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{v_y}^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{v_z}^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{v_h}^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{v_{wh}}^2 \end{bmatrix} \tag{17}$$

where  $E[V_{x,k}, V_{y,k}] = 0$ ,  $E[V_{x,k}, V_{y,k}] = 0$ ,  $E[V_{x,k}, V_{z,k}] = 0$ ,  $E[V_{h,k}, V_{wh,k}] = 0$  and so on because they are uncorrelated.

Figure 1 shows function block diagram of Kalman filter sensor fusion used in this design, which combines noisy measurements from RGB and depth sensors after measurement extraction with a predictive model to estimate an optimal and more accurate state of a system. It dynamically adjusts the weight given to each sensor's data, considering uncertainties, and blends this information to enhance overall estimation precision. This iterative process continuously refines the system's state estimate based on the latest sensor inputs and predicted values, making it a powerful tool for object state estimation. The Figure 1 is an extended Kalman filter sensor fusion functional block diagram.

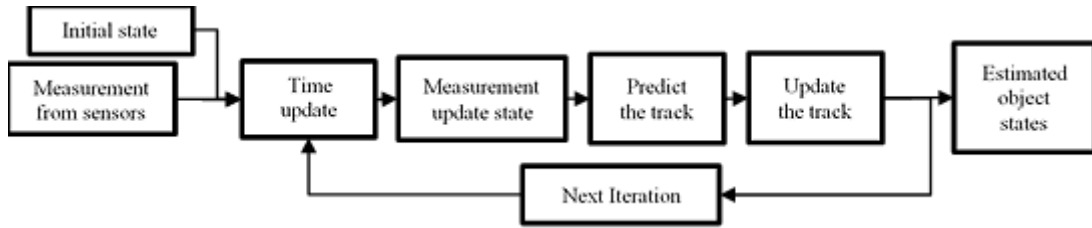


Figure 1. Extended Kalman filter sensor fusion functional block

**2.2. Sensor’s calibration**

In this project, two types of calibration techniques were used to convert 2D sizes to real-world units and determine 3D object positions: camera reference object calibration and calibration using a checkerboard pattern as shown in Figure 2. Camera calibration involves determining the intrinsic and extrinsic parameters of a camera to enhance image measurement accuracy. The first calibration method involves using a reference object to compute the pixel metric ratio. The second method entails capturing images of known geometric patterns, such as a checkerboard, from various angles. Mathematical models are then applied to minimize the disparity between observed and expected image points, refining both the camera’s internal and external parameters.



Figure 2. Reference and checkerboard sensors calibration

**2.3. Python implementation**

In this research, the pretrained Pytorch Mobile-Net V3 SSD-Lite object detection model was utilized for color image object detection. This lightweight model combines the MobileNetV3 architecture for feature extraction with the single shot multi-box detector framework for object detection. It balances accuracy and computational efficiency, predicting bounding boxes and class scores at multiple scales. This model is ideal for on-device applications and scenarios with limited computational resources, maintaining competitive detection performance on standard benchmarks like COCO [29]. The Python part consists of RGB and depth image preprocessing. It starts by receiving images from the sensors, synchronizing them, and configuring the resolution, passing the RGB image for object detection, projecting the detected object on the depth image, extracting the object’s 3D position and 2D size with the sensor calibration parameters, packaging the object measurement to be sent to the FPGA through the UART communication protocol, and then receive the estimated object state for the display as shown in Figure 3.

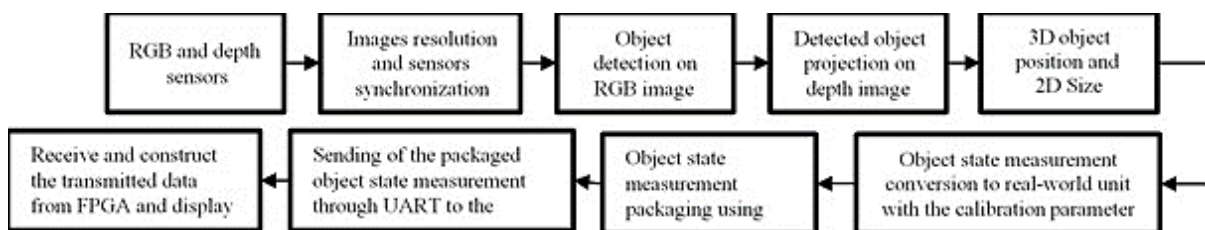


Figure 3. Python part flow diagram





### 3. RESULTS AND DISCUSSIONS

This section presents the results and discussions of the designed and experimentation system. The discussion of the results begins by the system performance evaluation, Vivado implementation results and finally the experimentation setup results. This is achieved from the Integration of the python design and FPGA implementation of Kalman filter sensor fusion.

#### 3.1. System performance evaluation

The graphs in Figures 7-10 are for the overall system performance evaluation, where the blue curve is the noisy input from the sensors, the orange curve is the Python model output, the green curve is for the C++ HLS model, and the red curve for the Verilog model output for estimation and RMSE. Each state of the object has two graphs, the first graph is for all platform (Python, HLS C++, and Verilog) input/output against time and the second graph is for all platform RMSE against time. Figures 7 and 8 are for object height and width states for the three platforms where in the estimation graph the curves start near to the origin (0,0) because of the initialization parameters. Figure 9 is for object distance measurement from sensors and estimation graph from all platform, and Figure 10 is for distance estimation RMSE graph. The performance of the proposed method was evaluated across three different platforms: Python IDE, Vivado HLS C++, and Vivado Verilog. The results analysis revealed that the maximum root means square error (RMSE) for object position ranged from 0 mm to 10 mm across all the platforms as shown in Figure 10, with the Vivado Verilog model achieving an average RMSE close to 5 mm. Similarly, for object width, the maximum RMSE varied from 0 mm to 50 mm, with the Vivado Verilog implementation demonstrating an RMSE approaching 0.0 mm as shown in Figure 8 width RMSE graph. Additionally, the maximum RMSE for object height ranged from 0 mm to 35 mm, again with Vivado Verilog exhibiting an RMSE close to 0.0 mm as shown in Figure 7. These results indicates that the Vivado Verilog implementation consistently achieved superior accuracy compared to the Python IDE and Vivado HLS C++.

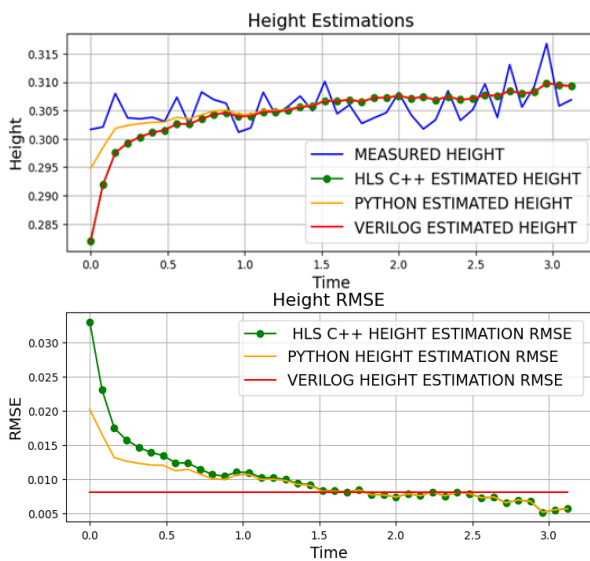


Figure 7. Graphs of object height measurement and estimation vs time and RMSE vs time

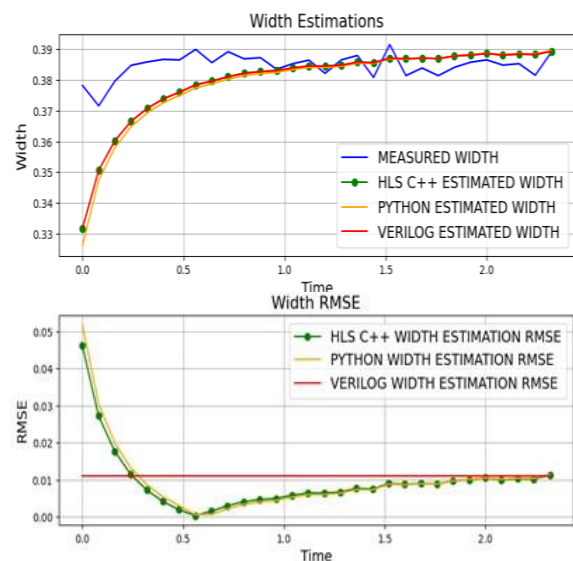


Figure 8. Graphs of object width measurement and estimation vs time and RMSE vs time

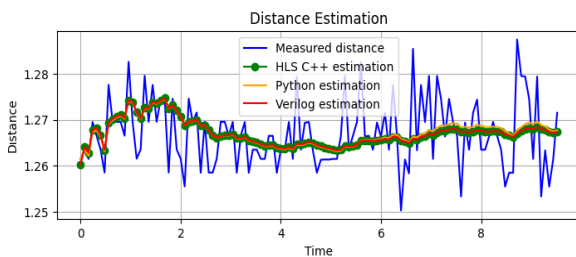


Figure 9. Graph of object distance measurement, and estimation vs time

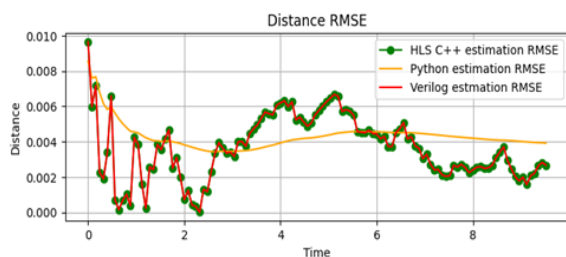


Figure 10. Graph of object distance RMSE vs time



### 3.2. Vivado implementation result

Figure 11 illustrates the Vivado design and implementation simulation process. The 'clk' represents the clock frequency set at 100 MHz, 'rst' indicates reset, 'rx' and 'tx' are designated for UART communication, facilitating data transmission and reception. 'cons\_x' to 'cons\_w' pertain to data construction post-reception, while 'wkf\_start', 'kf\_idle', 'kf\_done', and 'kf\_ready' control the extended Kalman filter IP core. The system begins by receiving data through UART input 'rx'. Upon data reception, the master custom constructs the received data and triggers the Kalman filter with 'wkf\_start' to initiate estimation. When the 'kf\_done' signal is asserted, processing concludes. The data is then packaged into big-endian format and transmitted through UART output 'tx'. Successful simulation demonstrates a latency of 2.1 ms. This encompasses 930 μs for data reception, 240 μs for processing, and an additional 930 μs for transmission. The developed and optimized Vivado HLS C++ equivalent of the Python model underwent validation using recorded data from Python experiments to confirm the design's accuracy. Following C-synthesis and co-simulation, the Kalman filter was exported to Vivado for simulation, synthesis, and implementation. Successful implementation of the UART communication protocol and Kalman filter sensor fusion on a Basys 3 FPGA as presented in Figure 12 utilized efficiently specific hardware resources of 29% of LUT, 6% of LUTRAM, 15% of Flip-flops, 34% BRAM, 38% of DSP, 4% of I/O, and 3% of BUFG while maintaining a clock frequency of 100 MHz the resource allocation demonstrates effective optimization considering the available FPGA resources and operating frequency on FPGA board with total-on-chip power consumption of 1.074 w where 0.074 w is device static power as shown in Figure 13.

The comparison presented in Table 1 demonstrates that the proposed implementation exhibits favourable performance in terms of latency, hardware utilization with consideration of available hardware resource in each used FPGA device, and power consumption. The hardware resource utilization was provided by the Vivado software after synthesis and implementation during the bitstream generation. The estimation latency was measured from the Vivado post synthesis and post implementation simulation, and the power consumption using was measured using the combination of Vivado and Xilinx power estimator (XPE).



Figure 11. Vivado testbench FPGA design simulation

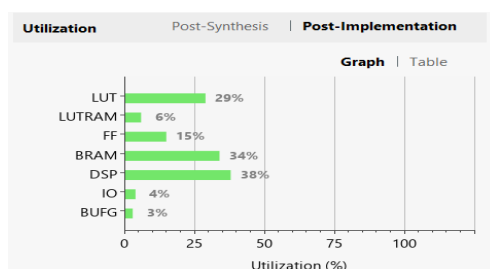


Figure 12. System resource utilisation

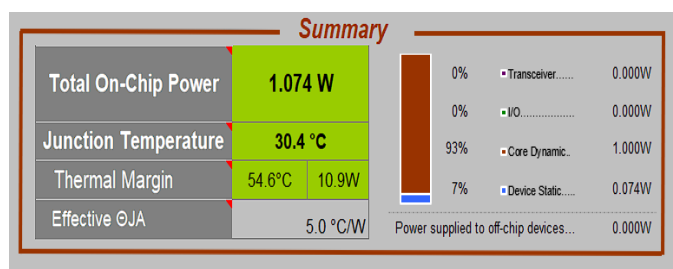


Figure 13. Power consumption summary

Table 1. Results comparison of the suggested method with the other existing works

Literature	Method	Device family	Part number	LUT (%)	DSP (%)	Flip-flops (%)	BRAM (%)	Max working Frequency (MHz)	estimation latency (ms)	Power consumption (mw)
Jerrah <i>et al.</i> [30]	EKF	Artix-7	XC7A100T	30	12.0	23.55	—	12.89	—	4100
Iqbal <i>et al.</i> [31]	Mean-shift + KF	UltraScale+ MPSoC	ZCU102	8.79	16.1	—	32.89	—	26.31	—
Babu and Parthasarathy [24]	MDKF	Artix-7	XC7Z020	83.2	90.9	76.017	61.43	140	10.989	780
Proposed method	EKFSP	Artix-7	XC7A35T CPG236	29	38	15	34	100	2.1	1074

**3.3. Result of integration experiment**

The results of object detection and feature extraction provide measurements by offering information about the 2D size (height, width) and 3D position measurement, as well as classification of detected objects. Figure 14 depicts images captured during the experimentation where the propose method integration underwent testing and validation. The ‘Real\_distance’ represents the ground truth distance, amd ‘Real\_size’ (height and width) denotes the measured ground truth size of the objects during experimentation. The outputs from the Kalman filter sensor fusion FPGA implementation include ‘Est\_size,’ ‘Est\_position,’ and ‘Est\_distance’. The experimentation of the proposed method showed average distance RMSE 0.0036, average height RMSE of 0.01 and average width RMSE of 0.01 for estimation accuracy and robustness during the performance evaluation, while maintaining low execution time in FPGA implementation.

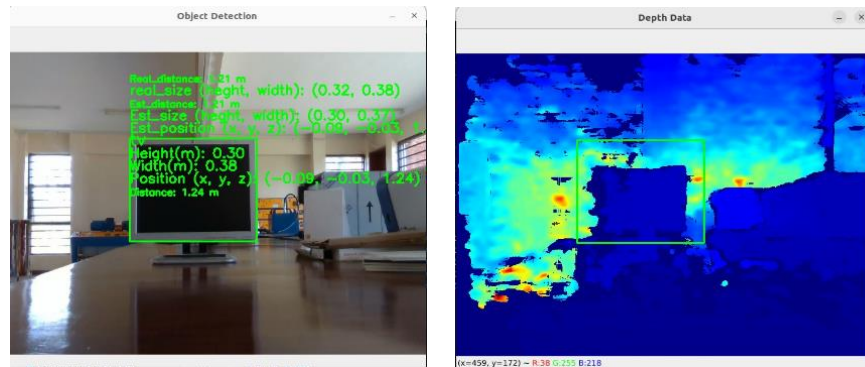


Figure 14. Integration of object detection and FPGA-based tracking

**4. CONCLUSION**

The proposed integration of deep learning object detection and FPGA-based Kalman filter sensor fusion demonstrates the applicability of the Kalman filter on FPGA for autonomous systems. The Kalman filter demonstrates exceptional robustness against noise, as evidenced by its performance in our evaluation. The RMSE for distance estimation is 0.0036, while for height and width, it is 0.01. These low RMSE values indicate that the Kalman filter effectively filters out noise and provides accurate estimates of the object’s position and dimensions. The FPGA accelerated the operation speed eight times faster than the software-based design while maintaining acceptable accuracy. Using the same datatype of a single precision floating point 32 bit, the accuracy remained the same for both FPGA and software-based implementations. Kalman sensor fusion implemented on the Basys 3 FPGA board executes at 240 microseconds and the communication interface at 1.86 milliseconds with a baud rate of 230400, all using efficient hardware resources and running at 100 MHz. In comparison to the study by Babu on FPGA-based object tracking, our proposed study not only confirms similar outcomes but also extends the research by focusing by hardware resource utilization, achieving faster execution time, and enhancing the accuracy of object state estimation. Additionally, to further enhance the flexibility and adaptability of the system, dynamic reconfiguration capabilities of FPGAs can be explored, resulting in specific hardware modules being loaded or unloaded based on the number of objects being detected.

## ACKNOWLEDGEMENTS

The authors would like to express their gratitude to African Union Commission (AUC) for funding this research.




## REFERENCES

- [1] Z. Zheng, Y. Cheng, Z. Xin, Z. Yu, and B. Zheng, "Robust perception under adverse conditions for autonomous driving based on data augmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 13916–13929, 2023, doi: 10.1109/TITS.2023.3297318.
- [2] X. Zhao, P. Sun, Z. Xu, H. Min, and H. Yu, "Fusion of 3D LIDAR and camera data for object detection in autonomous vehicle applications," *IEEE Sensors Journal*, vol. 20, no. 9, pp. 4901–4913, 2020, doi: 10.1109/JSEN.2020.2966034.
- [3] F. Rovira-Mas, V. Saiz-Rubio, and A. Cuenca-Cuenca, "Augmented perception for agricultural robots navigation," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11712–11727, 2021, doi: 10.1109/JSEN.2020.3016081.
- [4] P. Ghorai, A. Eskandarian, Y. K. Kim, and G. Mehr, "State estimation and motion prediction of vehicles and vulnerable road users for cooperative autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 16983–17002, 2022, doi: 10.1109/TITS.2022.3160932.
- [5] K. Wang, Y. Wang, B. Liu, and J. Chen, "Quantification of uncertainty and its applications to complex domain for autonomous vehicles perception system," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–17, 2023, doi: 10.1109/TIM.2023.3256459.
- [6] Y. Zhao, C. Lei, Y. Shen, Y. Du, and Q. Chen, "Improving autonomous vehicle visual perception by fusing human gaze and machine vision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 12716–12725, 2023, doi: 10.1109/TITS.2023.3290016.
- [7] Z. Cai, J. Liu, W. Chi, and B. Zhang, "A low-cost and robust multi-sensor data fusion scheme for heterogeneous multi-robot cooperative positioning in indoor environments," *Remote Sensing*, vol. 15, no. 23, 2023, doi: 10.3390/rs15235584.
- [8] F. Chen *et al.*, "Sensor fusion-based anthropomorphic control of a robotic arm," *Bioengineering*, vol. 10, no. 11, 2023, doi: 10.3390/bioengineering10111243.
- [9] Y. Yin, J. Zhang, M. Guo, X. Ning, Y. Wang, and J. Lu, "Sensor fusion of GNSS and IMU data for robust localization via smoothed error state Kalman filter," *Sensors*, vol. 23, no. 7, 2023, doi: 10.3390/s23073676.
- [10] A. Wondosen, Y. Debele, S. K. Kim, H. Y. Shi, B. Endale, and B. S. Kang, "Bayesian optimization for fine-tuning EKF parameters in UAV attitude and heading reference system estimation," *Aerospace*, vol. 10, no. 12, 2023, doi: 10.3390/aerospace10121023.
- [11] J. Ma, L. Li, and C. Xu, "AutoRS: Environment-dependent real-time scheduling for end-to-end autonomous driving," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 12, pp. 3238–3252, 2023, doi: 10.1109/TPDS.2023.3323975.
- [12] K. Shu, N. D. Dao, W. Shi, and A. Khajepour, "Group frenet frame CAV path planning on highways," *IEEE Internet of Things Journal*, vol. 11, no. 4, pp. 6776–6787, 2024, doi: 10.1109/JIOT.2023.3314373.
- [13] J. Bai, Z. Zeng, T. Wang, S. Zhang, N. N. Xiong, and A. Liu, "TANTO: an effective trust-based unmanned aerial vehicle computing system for the internet of things," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5644–5661, 2023, doi: 10.1109/JIOT.2022.3150765.
- [14] M. K. Somesula, S. K. Mothku, and S. C. Annadanam, "Cooperative service placement and request routing in mobile edge networks for latency-sensitive applications," *IEEE Systems Journal*, vol. 17, no. 3, pp. 4050–4061, 2023, doi: 10.1109/JSYST.2023.3260028.
- [15] Y. Li, S. E. Li, X. Jia, S. Zeng, and Y. Wang, "FPGA accelerated model predictive control for autonomous driving," *Journal of Intelligent and Connected Vehicles*, vol. 5, no. 2, pp. 63–71, 2022, doi: 10.1108/JICV-03-2021-0002.
- [16] A. D. Kumar, R. Karthika, and K. P. Soman, "Stereo camera and LIDAR sensor fusion-based collision warning system for autonomous vehicles," *Advances in computational intelligence techniques*, pp. 239–252, 2020, doi: 10.1007/978-981-15-2620-6\_17.
- [17] J. S. Berrio, M. Shan, S. Worrall, and E. Nebot, "Camera-LIDAR integration: probabilistic sensor fusion for semantic mapping," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7637–7652, 2022, doi: 10.1109/TITS.2021.3071647.
- [18] T. M. Di Gennaro and J. Waldmann, "Sensor fusion with asynchronous decentralized processing for 3D target tracking with a wireless camera network," *Sensors*, vol. 23, no. 3, 2023, doi: 10.3390/s23031194.
- [19] W. Farag, "Kalman-filter-based sensor fusion applied to road-objects detection and tracking for autonomous vehicles," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 235, no. 7, pp. 1125–1138, 2021, doi: 10.1177/0959651820975523.
- [20] B. S. Jahromi, T. Tulabandhula, and S. Cetin, "Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles," *Sensors (Switzerland)*, vol. 19, no. 20, p. 4357, 2019, doi: 10.3390/s19204357.
- [21] R. A. Garcia-Huerta, L. E. González-Jiménez, and I. E. Villalon-Turrubiates, "Sensor fusion algorithm using a model-based Kalman filter for the position and attitude estimation of precision aerial delivery systems," *Sensors (Switzerland)*, vol. 20, no. 18, pp. 1–20, 2020, doi: 10.3390/s20185227.
- [22] P. V. Patil, L. Vachhani, S. Ravitharan, and S. Chauhan, "Sequential state and unknown parameter estimation strategy and its application to a sensor fusion problem," *IEEE Sensors Journal*, vol. 22, no. 21, pp. 20665–20675, 2022, doi: 10.1109/JSEN.2022.3199214.
- [23] J. Qiu *et al.*, "Centralized fusion based on interacting multiple model and adaptive kalman filter for target tracking in underwater acoustic sensor networks," *IEEE Access*, vol. 7, pp. 25948–25958, 2019, doi: 10.1109/ACCESS.2019.2899012.
- [24] P. Babu and E. Parthasarathy, "FPGA implementation of multi-dimensional Kalman filter for object tracking and motion detection," *Engineering Science and Technology, an International Journal*, vol. 33, p. 101084, 2022, doi: 10.1016/j.jestech.2021.101084.
- [25] S. A. Li and C. Li, "FPGA implementation of adaptive Kalman filter for industrial ultrasonic applications," *Microsystem Technologies*, vol. 27, no. 4, pp. 1611–1618, 2021, doi: 10.1007/s00542-019-04456-6.
- [26] D. Castells-Rufas *et al.*, "A survey of FPGA-based vision systems for autonomous cars," *IEEE Access*, vol. 10, pp. 132525–132563, 2022, doi: 10.1109/ACCESS.2022.3230282.
- [27] F. Govaers, *Introduction and implementations of the Kalman filter*. IntechOpen, 2019.




- [28] T. Li *et al.*, “An adaptive control method and learning strategy for ultrasound-guided puncture robot,” *Electronics (Switzerland)*, vol. 13, no. 3, 2024, doi: 10.3390/electronics13030580.
- [29] A. Howard *et al.*, “Searching for mobileNetV3,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, vol. 2019-October, pp. 1314–1324, doi: 10.1109/ICCV.2019.00140.
- [30] A. Jarrah, A. K. Al-Tamimi, and T. Albashir, “Optimized parallel implementation of extended Kalman filter using FPGA,” *Journal of Circuits, Systems and Computers*, vol. 27, no. 1, p. 1850009, 2018, doi: 10.1142/S0218126618500093.
- [31] O. Iqbal *et al.*, “Design and FPGA implementation of an adaptive video subsampling algorithm for energy-efficient single object tracking,” in *Proceedings - International Conference on Image Processing, ICIP*, 2020, vol. 2020-October, pp. 3065–3069, doi: 10.1109/ICIP40778.2020.9191146.

## BIOGRAPHIES OF AUTHORS






**Abdoul Moumouni Harouna Maloum**    is a research student who is currently pursuing a Master’s degree at the Pan African University, Institute for basic Sciences, Technology and Innovation, Kenya. He obtained the B.Sc. degree in Electrical Engineering (Automation and Industrial computing) from the University Institute of Technology of Dan Dicko Dankoulodo of Maradi, Niger. His research interest includes autonomous vehicle, computer accelerator, computer vision, and VLSI. He can be contacted at email: moumouni.abdoul@students.jkuat.ac.ke.



**Nicasio Maguu Muchuka**    is a lecturer in the department of electrical and control engineering at Egerton University Kenya. He obtained his Ph.D. in Electronic Engineering from Stellenbosch University, Stellenbosch South Africa, an M.Sc. in Electronics from the University of Mysore, Mysore India, and a B.Sc. in Physics, Maths, and Electronics from Dr. Babasaheb Ambedkar Marathwada University, Aurangabad, India. His current research interest is in VLSI design, embedded systems design and applied artificial intelligence. He can be contacted at email: nmuchuka@egerton.ac.ke.



**Cosmas Raymond Mutugi Kiruki**    is a Lecturer at the department of Electrical and Information Engineering at the University of Nairobi. He holds a B.Sc. and M.Sc. in Electrical and Electronics Engineering from the University of Nairobi, having graduated in 2014 and 2017, respectively. He graduated with a Ph.D. in Electrical and Space Systems Engineering from Kyushu Institute of Technology, Japan in 2021. His main research interests include embedded and digital systems, nanosatellites and space systems, electronics, power systems, and renewable energies. He can be contacted at email: rkiruki@uonbi.ac.ke.