Engineering Science and Technology, an International Journal 33 (2022) 101084

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Full Length Article

FPGA implementation of multi-dimensional Kalman filter for object tracking and motion detection

Praveenkumar Babu, Eswaran Parthasarathy *

Department of Electronics and Communication Engineering, SRM Institute of Science and Technology, Kattankulathur, 603 203 Chennai, India

ARTICLE INFO

Article history:
Received 30 April 2021
Revised 22 September 2021
Accepted 28 November 2021
Available online 23 December 2021

Keywords:
Kalman filter
Object tracking
Motion detection
FPGAs
GPUs
SOC

ABSTRACT

Object tracking and motion detection are the major challenges in the real-time image and video processing applications. There are several tracking and prediction algorithms available to estimate and predict the state of a system. Kalman filter is the most widely used prediction algorithm as it is very simple, efficient and easy to implement for linear measurements. However, these types of filter algorithms are customized on hardware platforms such as Field-Programmable Gate Arrays (FPGAs) and Graphic Processing Units (GPUs) to achieve design requirements for embedded applications. In this work, a multi-dimensional Kalman filter (MDKF) algorithm is proposed for object tracking and motion detection. The numerical analysis of proposed tracking algorithm achieves competitive tracking performance in contrast with state-of-the-art tracking algorithms trained on standard benchmarks. Furthermore, MDKF is implemented on Xilinx Zynq™-7000 System-on-a chip (SoC). The implementation of MDKF on SoC performs 2× times tracking speed than that of software approach. The experimental results provide resource utilization of about 61.43% of Block RAMs (BRAMs), 90.09% of DSPs, 83.27% of Look-up tables (LUTs) and 82.35% of logic cells operating at 140 MHz with power consumption of 780 mW which outperforms previous related methods.

1. Introduction

Kalman filter is a classical recursive and optimal algorithm which uses set of mathematical equations and data inputs to estimate the factors such as position, velocity and true values of measurements of an object once the measurement values are uncertainty. Kalman filter was first introduced by R. E. Kalman [1]. This linear quadratic estimation (LQE) is widely used for surveillance, navigation, control of aircraft and spacecraft. The common applications of adapting Kalman filters are object tracking and motion detection in dynamically positioned vehicles [2]. Kalman filter produces an estimate of the state of the system as an average of the system's predicted state and of the new measurement using the weights calculated from the measure of the estimated uncertainty of predicted system's state termed as covariance [3]. There are two processes involved in Kalman filtering algorithm are shown in Fig. 1. Since, Kalman filters are recursive in nature, this process is repeated at every time step which results in a new estimate and updated covariance with prediction established in the subsequent iterations. For non-linear systems, Extended Kalman filters (EKF) and Unscented Kalman filters (UKF) can be applied [4].

The key aspects in the field of computer vision are tracking and detection of moving objects. There are several algorithms and implementation of such tracking algorithms have been developed over the years. The improved tracking algorithm with EKF using multi-feature extraction based on 3D sensors is proposed to solve the tracking error in gesture recognition [5]. An adaptive Kalman filter with convolutional Siamese network (SiamFC) is developed to improve tracking efficiency in complex scenes at 43 frames per second (fps) [6]. However, the advancements in deep learning techniques for tracking algorithms is increasing, the computation task is still high and the background information for bounding boxes will degrade the performance of the whole system. The tracking problem for extended objects is addressed and EKF is used for tracking [7]. C. G. Prevost et al. demonstrated the state estimation and trajectory predictions using EKF for Unmanned Aerial Vehicles (UAVs) [8].

Several Kalman filter based tracking multiple objects have been explained in [9–12]. The challenging task in the multiple object tracking is the occlusion impediments which is identified and resolved using different types of Kalman filtering algorithms

* Corresponding author.
E-mail addresses: praveenb2@srmist.edu.in (P. Babu), eswaranp@srmist.edu.in (E. Parthasarathy).

https://doi.org/10.1016/j.jestch.2021.101084
2215-0986/

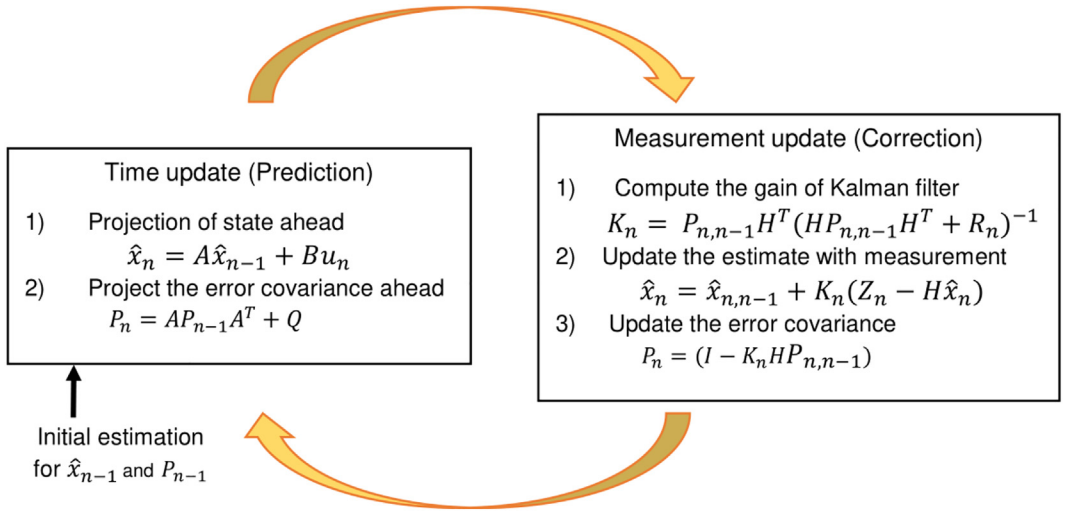

Fig. 1. Kalman filter algorithm. [3].

[13,14]. Zhou et al. presented multiple instance learning (MIL) method which is robust to track the occluded multiple objects [15]. MIL tracking algorithm is based on the principle of using samples form the previous frames to update the model which can avoid update error of the previous frame such that the classifier tracks the new samples [16]. Tracking algorithms based on deep convolutional neural networks (CNN) have gained reasonable performance over the years [17–20]. Yuan et al. proposed self-supervised deep correlation tracking method for manually annotated samples to enhance the performance of feature extraction network [21]. Dong et al. proposed object method using hyperparameter optimization algorithm based on deep Q- learning [22]. However, these CNN based tracking methods are time-consuming and they require a network with multiple layers to perform feature extraction. For these reasons, these trackers are failed to achieve required tracking speed and accuracy. To overcome the issues addressed earlier, these types of tracking algorithms can be implemented on hardware platforms. FPGAs provide practical solutions for various applications [23,24]. As mentioned earlier, FPGAs can deliver parallelism and flexibility to meet the design requirements. FPGA as a reconfigurable platform can solve numerous solutions related to system performance and designers are able to custom the resources with respect to the applications and demand [25]. These features have motivated the researchers to implement tracking and prediction on FPGAs. Some of the hardware based Kalman filtering algorithms are as follows. FPGA based Kalman filter to improve the performance in moving rockets is demonstrated in [26]. Several hardware implementation of different types of Kalman filters are proposed in the existing literature [27–30]. L. Bossuet et al. illustrated the modeling of FPGA architectures [31]. A novel approach of implementation of Kalman filter on Zynq SoC using piece-wise affine modelling is demonstrated to overcome software workload and communication overhead [32]. Later, Jarrah et al. proposed implementing EKF on Xilinx Artix® 7 FPGA board and the experimental results showed better utilization of FPGA resources but computational complexities for FPGA implementation of such tracking algorithms involved in the process is not addressed [33]. Soh and Wu developed implementation of UKF on Zynq SoC for SoC applications without optimizing logic resources [34]. Zhang et al. implemented a recursive tracking algorithm termed as multi-frame Track Before-Detect (MF-TBD) on FPGA and showed that hardware implementations outperform software simulation in terms of speed and performance [35]. Iqbal et al. developed tracking algorithm based on adaptive video sampling for single object tracking methods by merging mean shift (MS) algorithm and Kalman filter. MS-KF achieved tracking speed of 38 fps at 75 MHz when implemented on FPGA [36]. The major contributions of this work are as follows.

- A multi-dimensional Kalman filter for object tracking and motion detection is proposed and trained on customized dataset containing 42 challenging videos in different locations.
- The experimental evaluations of proposed tracking algorithm are performed on the following benchmarks: OTB-100 [37], MOT-16 [38] and UAVDT [39] and also compared with other state-of-the-art tracking methods.
- The proposed tracking method is implemented on Zynq™-7000 SoC to estimate the performance of object tracking with optimal resource utilization.

Table 1 gives the description of notations quoted in this work. The multi-dimensional Kalman filter algorithm is analyzed and its parametric equations are derived in Section 2. The numerical analysis of proposed algorithm and its hardware implementation are described in Section 3. FPGA implementation results and

Table 1
Nomenclature and description.

Notation	Description
$\hat{x}_{n\|n}$	State vector for estimation
$\hat{x}_{n\|n+1}$	Predicted state vector for (n + 1) step
$\hat{u}_{n\|n}$	Deterministic variable
ω	Process Noise vector
F	State Transition matrix
G	Control matrix
$P_{n\|n}$	Estimation of uncertainty for present state
$P^{+}_{n\|n}$	Estimation of uncertainty for next state
R_n	Measurement uncertainty matrix
Q	Process noise matrix
K	Kalman gain
H	Observation matrix
$\hat{x}_{n\|n-1}$	Predicted state vector for (n-1) step
Z_n	Measurement state vector
A	Dynamics System matrices
B	Input matrix
C	Output matrix
D	Feedback or Input matrices
n	Discrete Time Index

resource utilization are discussed in Section 4 followed with concluding remarks and future scope of this work.

2. Multi-dimensional Kalman filter and its parameters

Kalman filter provides the process estimation by means of feedback using noisy measurements. In this robust algorithm, prediction and correction equations are the two update equations resulting in estimated state process. Prediction or extrapolation equations result in the propagation of estimated present state and correction or update equations are responsible for the introduction of new measurements with the estimation of *a posteriori state* $\hat{x}_{n|n+1}$ *and updating error covariance.*

The functional description of Kalman filter's block diagram is shown in Fig. 2. There are several steps involved in the process of estimating measurement values and uncertainty. The initialization is performed only once to estimate initial system state $\hat{x}_{0|0}$ and initial state uncertainty $\widehat{P}_{0|0}$. It is followed by prediction. It involves in the measurement process which provides Z_n and R_n. The state update process is responsible for the estimation of present system state. Based on these inputs, Kalman gain (K_n) can be calculated. The prediction process extrapolates the current system state and the uncertainty of the current system state estimate to the next system state, based on the system's dynamic model.

The mathematical equations of multi-dimensional Kalman filters are derived in the matrix form because the implementation of Kalman Filter in matrix form yields faster computation run time [32]. The analysis of filter algorithm with updated state extrapolation and covariance estimation with Kalman gain is explained briefly in the next section. Assume that the state vector that describes the airplane position, velocity and acceleration is nine-dimensional which is given by,

$$\begin{cases} x_n = x_{n-1} + \dot{x}_{n-1}\Delta t + 0.5\ddot{x}_{n-1}\Delta t^2 \\ y_n = y_{n-1} + \dot{y}_{n-1}\Delta t + 0.5\ddot{y}_{n-1}\Delta t^2 \\ z_n = z_{n-1} + \dot{z}_{n-1}\Delta t + 0.5\ddot{z}_{n-1}\Delta t^2 \\ \dot{x}_n = \dot{x}_{n-1}t + \ddot{x}_{n-1}\Delta t \\ \dot{y}_n = \dot{y}_{n-1}t + \ddot{y}_{n-1}\Delta t \\ \dot{z}_n = \dot{z}_{n-1}t + \ddot{z}_{n-1}\Delta t \\ \ddot{x}_n = \dot{x}_{n-1} \\ \ddot{y}_n = \ddot{y}_{n-1} \\ \ddot{z}_n = \ddot{z}_{n-1} \end{cases}$$

2.1. State extrapolation

Using the state extrapolation equation, we can predict the next system state, based on the knowledge of the current state. It extrapolates state vector from the present state (n) to the next step $(n+1)$. The general form of the state extrapolation equation in a matrix notation is represented as,

$$\hat{x}_n^+ = F\hat{x}_{n|n} + G\hat{u}_{n|n} + \omega_n. \tag{1}$$

The state space representation of linear time invariant (LTI) system is expressed by,

$$\dot{x}(t) = Ax(t) + Bu(t). \tag{2}$$

$$y(t) = Cx(t) + Du(t). \tag{3}$$

Consider an object is moving with constant velocity. If there is no external force is applied, then u(t)=0. The state space variable x(t) with displacement p(t) and speed v(t) is termed by $x(t)=\begin{bmatrix} p \\ v \end{bmatrix}$. Eq. 2 becomes,

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = A\begin{bmatrix} p \\ v \end{bmatrix} + B.0, \tag{4}$$

$$\begin{bmatrix} \dot{p} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} p \\ v \end{bmatrix} \tag{5}$$

The output state space variable $(y(t))$ is illustrated by,

$$p = C\begin{bmatrix} p \\ v \end{bmatrix} + D.0, \tag{6}$$

$$p = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} p \\ v \end{bmatrix} \tag{7}$$

Consider the dynamic system model with nth order differential equation is given by,

$$a_n\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \ldots + a_2\frac{d^2 y}{dt^2} + a_0 y = u, \tag{8}$$

In order to reduce the governing equation, isolate higher-order derivative is expressed as,

$$\frac{d^n y}{dt^n} = -\frac{a_0}{a_n}y - \frac{a_1}{a_n}y\frac{d^1 y}{dt^1} - \ldots - \frac{a_{n-1}}{a_n}y\frac{d^{n-1}y}{dt^{n-1}} + \frac{1}{a_n}u, \tag{9}$$

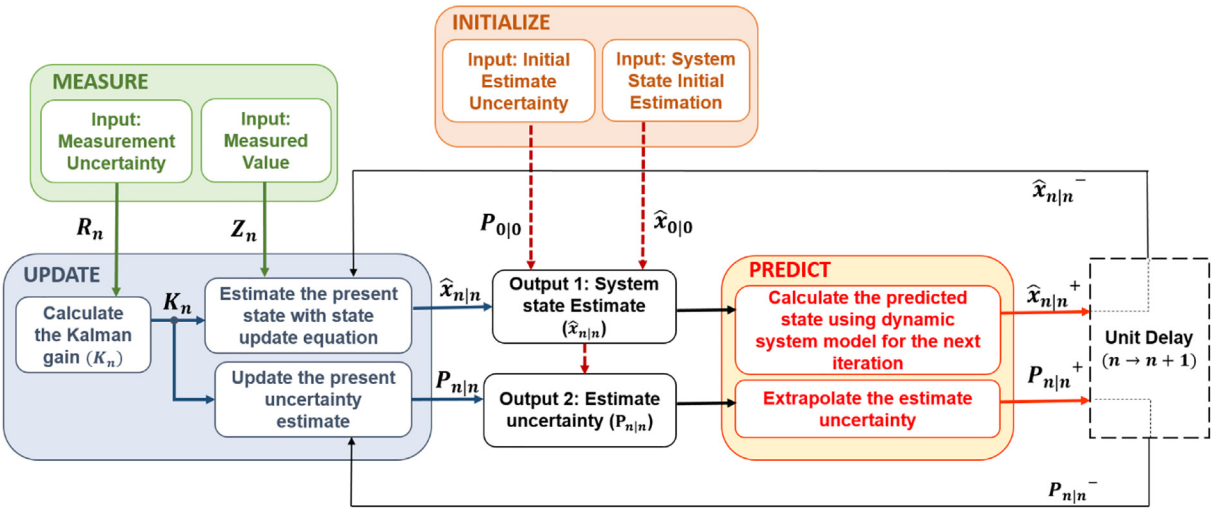

Fig. 2. Functional block diagram of proposed multi-dimensional Kalman filter.

$$\frac{dx_{n-1}}{dt} = x_n(t), \tag{10}$$

$$\frac{dx_n}{dt} = \frac{d^n y}{dt^n}, \tag{11}$$

$$\frac{dx_n}{dt} = -\frac{a_0}{a_n}x_1 - \frac{a_1}{a_n}x_2 - \ldots - \frac{a_{n-1}}{a_n}x_n + \frac{1}{a_n}u. \tag{12}$$

The resulted system of equations using vector–matrix notation is given by,

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_{n-1}}{dt} \\ \frac{dx_n}{dt} \end{bmatrix} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_{n-1} \\ \hat{x}_n \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_{n-1} \\ \hat{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & -\frac{a_2}{a_n} & \cdots & -\frac{a_{n-2}}{a_n} & -\frac{a_{n-1}}{a_n} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{a_n} \end{bmatrix} .u \tag{14}$$

Compare Eq. (2) and Eq. (14), we get,

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -\frac{a_0}{a_n} & -\frac{a_1}{a_n} & -\frac{a_2}{a_n} & \cdots & -\frac{a_{n-2}}{a_n} & -\frac{a_{n-1}}{a_n} \end{bmatrix}; B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{a_n} \end{bmatrix}$$

The State Update Equation in the matrix form is given by,

$$\hat{x}_{n|n} = \hat{x}_{n|n}^{-} + K_n\left(Z_n - H\hat{x}_{n|n}^{-}\right). \tag{15}$$

2.2. Covariance extrapolation

In the multi-dimensional Kalman Filter, the process noise is a covariance matrix denoted by Q. The general form of the Covariance Extrapolation Equation is given by,

$$P_{n|n}^{+} = FP_{n|n}F^T + Q, \tag{16}$$

where Q = $\begin{bmatrix} q_1 & 0 & \ldots & 0 \\ 0 & q_1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & q_n \end{bmatrix}$

The covariance update equation to measure uncertainty is given by,

$$P_{n|n} = (I - K_nH)P_{n|n}^{-}(I - K_nH)^T + K_nR_nK_n^T. \tag{17}$$

2.3. Kalman gain

Kalman gain gives the actual measurements and present state estimate. The Kalman Gain in matrix notation is expressed as,

$$K_n = P_{n|n}^{-}H^T\left(HP_{n|n}^{-}H^T + R_n\right)^{-1}. \tag{18}$$

In many cases it performs well, however, even the smallest error in computing the Kalman Gain can lead to huge computation errors. The subtraction $(I - K_nH)$ can lead to non-symmetric matrices results due to the floating-point errors. Hence, this equation is numerically unstable. In order to get the simplified form of covariance update equation, substitute Eq. 18 in Eq. 17, we get

$$P_{n|n} = \left(P_{n|n}^{-} - P_{n|n}^{-}K_nH - P_{n|n}^{-}K_n^TH^T\right) + K_n\left(HP_{n|n}^{-}H^T + R_n\right)K_n^T, \tag{19}$$

$$P_{n|n} = \left(P_{n|n}^{-}P_{n|n}^{-}K_n^TH^T\right) - K_nHP_{n|n}^{+}P_{n|n}^{-}K_n^TH^T, \tag{20}$$

$$P_{n|n} = \left(P_{n|n}^{-}P_{n|n}^{-}K_nH\right), \tag{21}$$

$$P_{n|n} = (I - K_nH)P_{n|n}. \tag{22}$$

Eq. 22 gives the simplified form of updated covariance equation. The steps involved in Kalman filtering are based on these update equations.

Algorithm 1: Kalman Filtering

1: **INITIALIZE** the system state $\hat{x}_{n|n}$ and state uncertainty $P_{n|n}$.
2: **MEASURE** the system state Z_n and uncertainty R_n.
3: **ESTIMATE** the state equation $\hat{x}_n^{+} = F\hat{x}_{n|n} + G_{n|n}^{u} + \omega_n$.
4: **CALCULATE** the covariance extrapolation $P_{n|n}^{+} = FP_{n|n}F^T + Q$.
5: **CALCULATE** Kalman gain $K_n = P_{n|n}^{-}H^T\left(HP_{n|n}^{-}H^T + R_n\right)^{-1}$.
6: **UPDATE** the state equation $\hat{x}_{n|n} = \hat{x}_{n|n}^{-} + K_n\left(Z_n - H\hat{x}_{n|n}^{-}\right)$ and covariance equation,
7: $P_{n|n} = (I - K_nH)P_{n|n}^{-}(I - K_nH)^T + K_nR_nK_n^T$.
8: **PREDICT** the extrapolation from present state to next state.

3. Numerical analysis

The numerical analysis are carried out based on proposed tracking algorithm and are compared with other state-of-the-art trackers on OTB-100 [37], MOT16 [38] and UAVDT [39] datasets. The numerical simulation analysis are carried out using MATLAB® 2019a [40] on a PC powered by a 3.20 GHz i5-4570 CPU with 12 GB RAM for further analysis and study.

3.1. Evaluation parameters

Precision and success rates are considered as evaluation parameters to estimate the tracking performance. Precision refers to the average distance between the predicted position and manually labelled ground-truths [37]. Success rate gives the overlap score by means of area under curve [41]. Tracking speed in frames per second (fps) is also taken into account to address the performance of tracking algorithm. The most widely adapted methods for eval-

uation are one-pass evaluation (OPE), temporal robust evaluation (TRE) and spatial robustness evaluation (SRE) [42]. However, we follow OPE to compare with state-of-the-art trackers to evaluate performance for convenience.

3.1.1. Analysis on OTB-100 benchmark

The numerical analysis are performed on OTB-100 dataset [37] plotted in 3. The performance of proposed tracking algorithm is verified on comparing with other state-of-the-art tracking algorithms include ATOM [43], SDCT [21], ECO [20], SRDCF [44], SiamFC [6] and MetaCREST [45]. From this analysis, the proposed tracking algorithm is effective than SDCT [21], SiamFC [6], ECO [20] and SRDCF [44]. However, proposed tracker is less effective against [43] and MetaCREST [45], due to the reason that these two tracking algorithms follow accurate target estimation strategy and meta-learning respectively.

3.1.2. Analysis on UAVDT benchmark

The ablation study on UAVDT dataset [39] is carried out to estimate the tracking speed, precision and AUC shown in 4. The analytical results and illustrations on proposed MDKF tracking algorithm alongside other trackers including STRCF [46], CFNET [47], UDT [48], SiamFC [18] Staple CA [49] and SDCT [21] is listed in Table 2. The tracking speed of the proposed multi-dimensional Kalman filter algorithm performs the third best score and second best score in terms of precision and AUC trained on UAVDT [39]. MDKF provides better tracking performance as compared to SDCT [21] and Staple CA [49]. Therefore, the results demonstrate that the proposed algorithm provides qualitative tracking results presented in 2.

3.2. Hardware implementation

The experimental setup for the implementation of proposed filtering algorithm on Xilinx Zynq SoC is depicted in 5. The objective of this section is to demonstrate the implementation of multi-dimensional Kalman filter on FPGA. The targeted hardware platform for the implementation is Xilinx Zynq XC7Z020 SoC [50]. This hardware comprises of Dual-core ARM Cortex-A9 processing system (PS) along with Artix-7 FPGA programmable logic (PL). It also includes on-chip memory, external memory interfaces, DDR3 component memory and other peripheral connectivity interfaces [51]. In the FPGA architecture, Application Processor Unit (APU) consists of NEON™ media processing engine for Single Instruction Multiple Data (SIMD) support with double precision floating point for each processor. FPGA architecture can support for the customization of logic resources and software components in PL and PS respectively. In this architecture, Advanced Microcontroller Bus Architecture (AMBA) based AXI interconnects provide high bandwidth connectivity within PS and between PL and PS.

Xilinx provides Vivado® High Level Synthesis (HLS) for application development on both standard and specialized processors [52]. The design flow of algorithm design to FPGA implementation using MATLAB/Simulink is illustrated in 6. The codes for FPGA hardware design and implementation are written in VHDL. Using HDL™ Coder, interfacing and generating codes can be obtained.

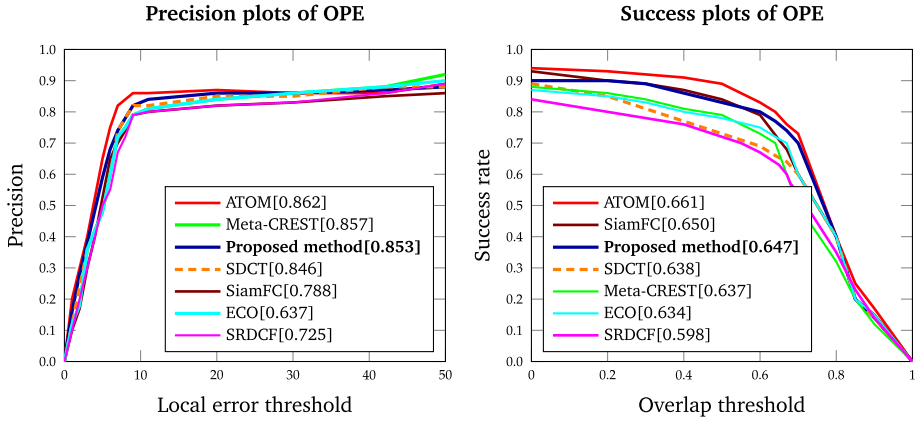

Fig. 3. Precision and Success plots of OPE on OTB-100[37] dataset.

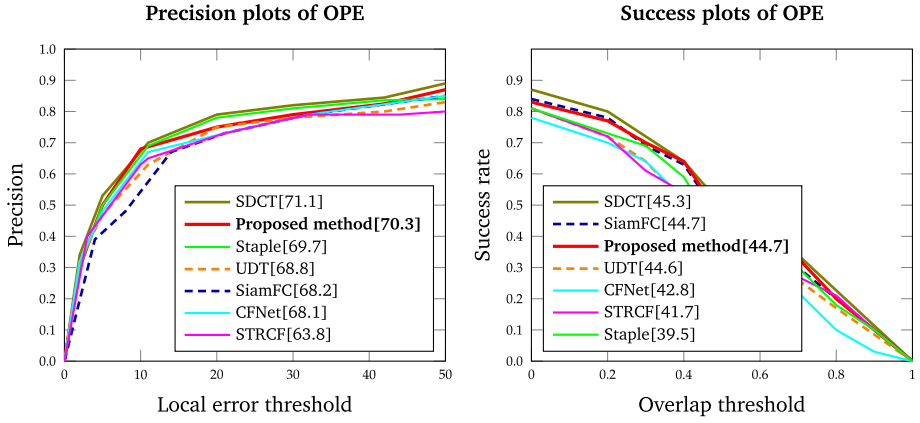

Fig. 4. Precision and Success plots of OPE on UAVDT [39] dataset.

Table 2
Comparison with state-of-the-art tracking methods in terms of evaluation parameters on UAVDT[39] dataset.

Tracking algorithms	Precision (%)	AUC (%)	Tracking Speed (fps)
STRCF [46]	63.8	41.7	29
CFNET [47]	68.1	42.8	65
SiamFC [18]	68.2	44.7	58
UDT [48]	68.8	44.6	45
Staple CA [49]	69.7	39.5	35
SDCT [21]	71.1	45.3	48
Proposed tracker	70.3	44.7	49

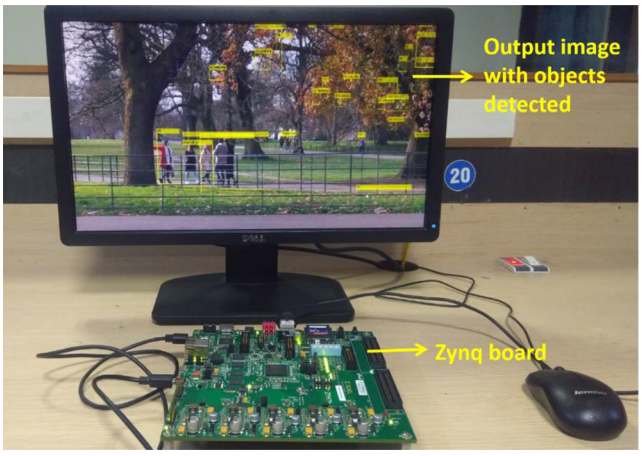

Fig. 5. Hardware setup.

4. Results and discussion

The implementation results are synthesized by using Vivado Design Suite on Zynq XC7Z020 SoC platform. The outputs of multi-dimensional Kalman filtering includes the object tracking performed on MOT16 benchmark dataset [38] and customized dataset using multi-dimensional Kalman filter are shown in 7.

The proposed tracking algorithm provides qualitative results trained on other standard benchmark datasets. An illustration of FPGA implementation of multi-dimensional Kalman filter for object tracking and motion detection performed on customized dataset is shown in 8.

4.1. Resource utilization

The resource utilization for the implementation of proposed multi- dimensional Kalman filter on Xilinx Zynq SoC platform using Vivado® Design Suite (2020.2) is plotted in 9. Some of the major system resources utilized for this implementation are BRAMs, Flip-flops, Look-up Tables (LUTs), programmable logic cells and DSPs are considered. The other constraint is the latency caused by pipelining is one of the trade-offs to consider during FPGA design. FPGA pipelining increases the latency for number of clock cycles and also increases performance to customize the circuit design to operate at maximum clock frequency [53]. Vivado HLS provides hardware co-processor with control algorithm to improve latency or processing speed in the design. The implementation results show that the resources are utilized effectively but the usage of DSPs are directly involving in the trade-off between flexibility and performance of the implementation. In any cases, for DSPs' utilization, the important constraint to be satisfied is represented as, $n_{dsp}(2n-1) \leqslant n_{dspmax}$ [32]. The resource consumption of LUTs and BRAMs are linear in nature represented in terms of n and $2n$ respectively (n is the number of states). The number of clock cycles taken for the iterations to complete can be determined by the clocks required for matrix multiplication and addition (d_m), clocks required for division (d_d) and number of steps to complete Kalman algorithm (K_s) is given by, $k = (d_m(2n^2-2)+ d_m(2n-2)K_s + d_d$. The overall implementation of our proposed multi-dimensional Kalman filter is operated at 140 MHz (max. 500 MHz). The power consumption can be obtained from Xilinx Power estimator tool. The total power consumed for this implementation is about 780 mW. The tracking speed of real-time object detection when proposed tracking algorithm is implemented on Xilinx Zynq-7000 SoC is faster than that of previous approaches

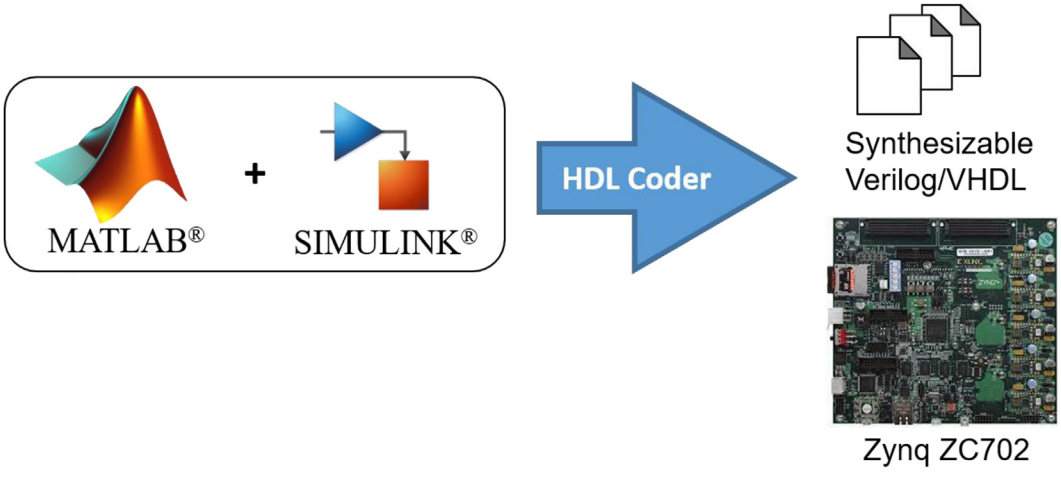

Fig. 6. Work Flow: algorithm design to FPGA implementation.

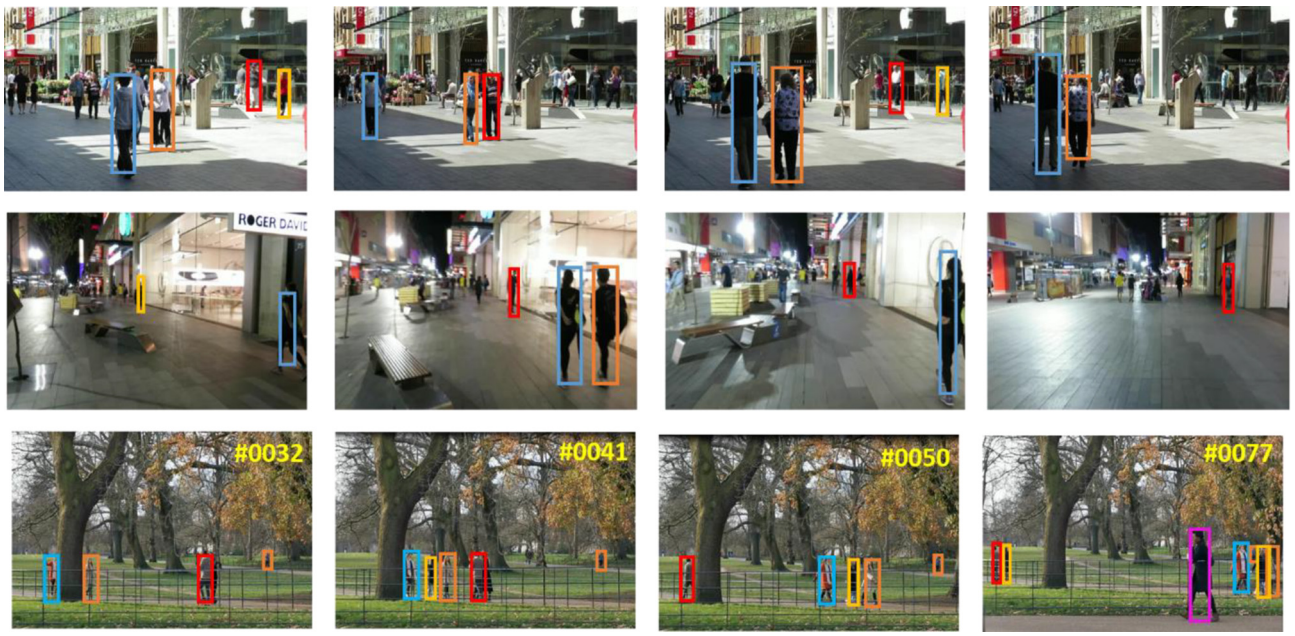

Fig. 7. Objects tracking results on MOT16 and MOT17 [38] datasets (top row and middle row respectively) and our own dataset (bottom row) using multi-dimensional Kalman filter.

Fig. 8. Motion detection (left) and object tracking (right) using multi-dimensional Kalman filter implemented on FPGA.

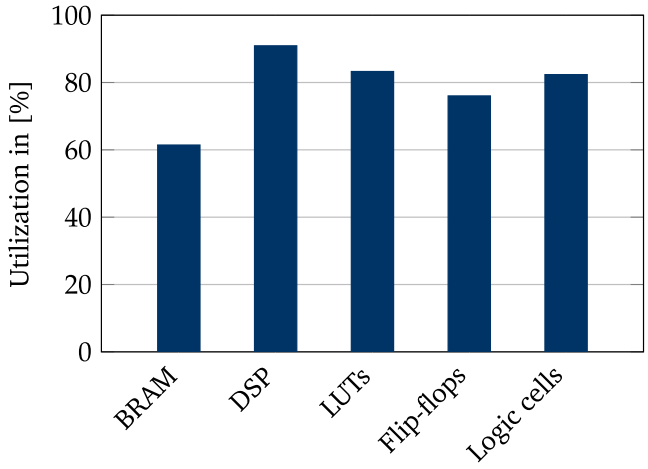

Fig. 9. Zynq SoC Resource Utilization.

illustrated in Table 3. The implementation results of proposed tracker is compared with PWAKF [32], EKF [33], UKF [34], MF-TBD [35] and Mean-shift + KF [36] in terms of resource utilization, power consumption, tracking speed and operating frequency. The experimental results prove that the implementation of MDKF achieves effective utilization of hardware resources.

5. Conclusion

A multi-dimensional Kalman filter for linear systems with updated state vector and covariance equations is presented in this paper for multiple objects tracking and detection. The hardware implementation of multi-dimensional Kalman filter on Zynq XC7Z020 SoC with effective resource utilization is also proposed. MDKF tracking algorithm is performed on various benchmark datasets to estimate performance and accuracy. The experimental results showed that the tracking speed of proposed tracking algorithm is about 49 fps which is productive than the state-of-the-art algorithms. The hardware implementation of multi-dimensional Kalman filter provided the breakthrough in terms of effective resource utilization in contrast with previous implementation methods and achieved 91 fps operating at 140 MHz with power consumption of 780 mW. In resource utilization, the only constraint is the usage of DSP blocks in the Zynq SoC, which may be exhausted due to the increase in the number of states for measuring uncertainties. Moreover, the proposed implementation will be highly efficient and suitable for many embedded platforms in the

Table 3
Comparison of hardware resource utilization for the implementation of different tracking algorithms and their tracking performances.

Parameters	Zhang et al. [35]	Iqbal et al. [36]	Soh et al. [34]	Jarrah et al. [33]	Mills et al. [32]	**Proposed method**
Algorithm	MF-TBD	Mean-shift + KF	UKF	EKF	PWAKF	MDKF
Platform	XC7S50	ZCU102	XC7Z045	XC7A100T	XC7Z020	XC7Z020
Device Family	Spartan-7	UltraScale + MPSoC	Kintex-7	Artix-7	Artix-7	Artix-7
BRAM (%)	85.33	32.89	22	NA	60	61.43
DSP (%)	21.67	16.15	20	12.08	99	90.9
LUTs (%)	48.97	8.79	24	30	92	83.27
Flip-flops (%)	NA	NA	11	23.55	NA	76.017
Logic cells (%)	NA	42.92	NA	NA	NA	82.35
Power consumption (mW)	1493	NA	1084	4100	722	780
Max. Frequency (MHz)	80	NA	100	12.89	45	140
Tracking speed (fps)	NA	38	NA	NA	NA	91

fields of real-time image and video processing applications. This work can be extended to implement on SoC device with full or partial reconfiguration in order to provide trade-off between flexibility and parallelism.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors would like to thank SRM Institute of Science and Technology for providing facilities to carry out the research work.

References

[1] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME–J. Basic Eng. 82 (1) (1960) 35–45.
[2] Q. Li, R. Li, K. Ji, W. Dai, Kalman filter and its application, in: 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), IEEE, 2015, pp. 74–77.
[3] F.A. Faruqi, R.C. Davis, Kalman filter design for target tracking, IEEE Trans. Aerospace Electron. Syst. 16 (4) (1980) 500–508, https://doi.org/10.1109/TAES.1980.308980.
[4] S.J. Julier, J.K. Uhlmann, New extension of the Kalman filter to non-linear systems, in: Proc. SPIE 3068, Signal Processing, Sensor Fusion, and Target Recognition VI, IEEE, 1997..
[5] Y. Fang, L. Yu, S. Fei, An improved moving tracking algorithm with multiple information fusion based on 3D sensors, IEEE Access 8 (2020) 142295–142302, https://doi.org/10.1109/ACCESS.2020.3008435.
[6] Y. Wang, X. Mu, Dynamic Siamese network with adaptive kalman filter for object tracking in complex scenes, IEEE Access 8 (2020) 222918–222930, https://doi.org/10.1109/ACCESS.2020.3043878.
[7] S. Yang, M. Baum, Extended Kalman filter for extended object tracking, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 4386–4390, https://doi.org/10.1109/ICASSP.2017.7952985.
[8] C.G. Prevost, A. Desbiens, E. Gagnon, Extended Kalman filter for state estimation and trajectory prediction of a moving object detected by an unmanned aerial vehicle, 2007 American Control Conference (2007) 1805–1810, https://doi.org/10.1109/ACC.2007.4282823.
[9] S. Feng, K. Hu, E. Fan, L. Zhao, C. Wu, Kalman Filter for Spatial-Temporal Regularized Correlation Filters, IEEE Trans. Image Process. 30 (2021) 3263–3278, https://doi.org/10.1109/TIP.2021.3060164.
[10] I.A. Iswanto, T. Choa, B. Li, Object tracking based on meanshift and particle-Kalman filter algorithm with multi features, Proc. Comput. Sci. 157 (2019) 521–529, https://doi.org/10.1016/j.procs.2019.09.009.
[11] S. Xuan, S. Li, X.W.M. Han, G. Xia, Object tracking in satellite videos by improved correlation filters with motion estimations, IEEE Trans. Geosci. Remote Sens. 58(2) (2020) 1074–1086. doi:10.1109/TGRS.2019.2943366..
[12] F. Farahi, H.S. Yazdi, Probabilistic Kalman filter for moving object tracking, Signal Process.: Image Commun. 82 (10) (2020), https://doi.org/10.1016/j.image.2019.115751 115751.
[13] J.-M. Jeong, T.-S. Yoon, J.-B. Park, Kalman filter based multiple objects detection-tracking algorithm robust to occlusion, in: 2014 Proceedings of the SICE Annual Conference (SICE), 2014, pp. 941–946, https://doi.org/10.1109/SICE.2014.6935235.
[14] M. Heimbach, K. Ebadi, S. Wood, in: 2018 52nd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, 2018, pp. 1499–1502, https://doi.org/10.1109/ACSSC.2018.8645175.
[15] Z. Zhou, X. Gao, J. Xia, Z. Zhu, D. Yang, J. Quan, Multiple instance learning tracking based on Fisher linear discriminant with incorporated priors, Int. J. Adv. Robotic Syst. 15(1) (2018) 1–19. doi:10.1177/1729881417750724..
[16] Z. Zhou, J. Wang, Y. Wang, Z. Zhu, J. Du, X. Liu, J. Quan, Visual tracking using improved multiple instance learning with co-training framework for moving robot, KSII Trans. Internet Inf. Syst. 12 (11) (2018) 5496–5521.
[17] E. Gundogdu, A.A.Alatan, Good features to correlate for visual tracking, IEEE Trans. Image Process. 27(5) (2018) 2526–2540. doi:10.1109/TIP.2018.2806280..
[18] L. Bertinetto, J. Valmadre, J.F. Henriques, A. Vedaldi, P.H.S. Torr, Fully-convolutional Siamese networks for object tracking, Proc. ECCV Workshop (2016) 850–865.
[19] H. Li, Y. Li, F. Porikli, Deeptrack: Learning discriminative feature representations online for robust visual tracking, Proc. BMVC (2014) 1–12.
[20] M. Danelljan, G. Bhat, F.S. Khan, M. Felsberg, ECO: Efficient convolution operators for tracking, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6931–6939.
[21] D. Yuan, X. Chang, P.Y. Huang, Q. Liu, Z. He, Self-supervised deep correlation tracking, IEEE Trans. Image Process. 30 (2021) 976–985, https://doi.org/10.1109/TIP.2020.3037518.
[22] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, F. Porikli, Hyperparameter Optimization for Tracking with Continuous Deep Q-Learning, in: Proc. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 518–527.
[23] K. Vipin, S.A. Fahmy, FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures, Methods, and Applications, ACM Comput. Surveys 51(4) (2018) 1–39. doi:10.1145/3193827..
[24] P. Babu, E. Parthasarathy, Reconfigurable FPGA Architectures: A Survey and Applications, ACM Comput. Surveys 102 (2021) 143–156. doi:10.1007/s40031-020-00508-y.
[25] C. Lee, Z. Salcic, High-performance fpga-based implementation of kalman filter, Microprocessors Microsyst. 21 (4) (1997) 257–265, https://doi.org/10.1016/S0141-9331(97)00040-9.
[26] J.V. Fonseca, R.C.L. Oliveira, J.A.P. Abreu, E. Ferreira, M. Machado, Kalman filter embedded in FPGA to improve tracking performance in ballistic rockets, in: 2013 UKSim 15th International Conference on Computer Modelling and Simulation, 2013, pp. 606–610, https://doi.org/10.1109/UKSim.2013.149.
[27] Al-Rababah, A.A. Qadir, Embedded architecture for object tracking using Kalman filter, J. Comput. Sci. 12(5) (2016) 241–245. doi:10.3844/jcssp.2016.241.245..
[28] W. Liu, H. Chen, L. Ma, Moving object detection and tracking based on Zynq FPGA and ARM SoC, IET International Radar Conference (2015) 1–4, https://doi.org/10.1049/cp.2015.1356.
[29] A. Sudarsanam, Analysis of Field Programmable Gate Array-based Kalman Filter Architectures, url:http://digitalcommons.usu.edu/etd/788. (2010)..
[30] P. Rao, M.A. Bayoumi, An efficient vlsi implementation of real-time Kalman filter, IEEE International Symposium on Circuits and Systems (1990) 2353–2356, https://doi.org/10.1109/ISCAS.1990.112482.
[31] L. Bossuet, G. Gogniat, J. Diguet, J. Philippe, A modeling method for Reconfigurable Architectures, in: System-on-Chip for Real-Time Applications. The Kluwer International Series in Engineering and Computer Science, vol. 711, 2003. doi:10.1007/978-1-4615-0351-4_16..
[32] A. Mills, P.H. Jones, J. Zambreno, Parameterizable FPGA-based Kalman Filter Coprocessor using Piecewise Affine Modeling, in: IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2016, pp. 139–147, https://doi.org/10.1109/IPDPSW.2016.101.
[33] A. Jarrah, A. Al-Tamimi, T. Albashir, Optimized parallel implementation of extended kalman filter using FPGA, J. Circuits Syst. Comput. 27(1) (2017) 1850009(1–22). doi:10.1142/S0218126618500093.
[34] J. Soh, X. Wu, An fpga-based unscented Kalman filter for System-On-Chip Applications, IEEE Trans. Circuits Syst. II: Express Briefs 64(4) (2017) 447–451. doi:10.1109/TCSII.2016.2565730..
[35] P. Zhang, W. Li, X. Yang, Efficient implementation of recursive multi-frame track-before-detect algorithm based on fpga, in: Proc. 2019 International Conference on Control, Automation and Information Sciences (ICCAIS), 2019, pp. 1–6.
[36] Q. Iqbal et al., Design and fpga implementation of an adaptive video subsampling algorithm for energy-efficient single object tracking, in: Proc.

2020 IEEE International Conference on Image Processing (ICIP), 2009, pp. 3065–3069.
[37] Y. Wu, J. Lim, M.H. Yang, Online Object Tracking: A Benchmark, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2411–2418.
[38] A. Milan, et al., MOT16: A Benchmark for Multi-Object Tracking, url:https://arXiv.org/abs/1603.00831 (2016)..
[39] D. Du et al., The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking, Proc. ECCV (2018) 370–386.
[40] MATLAB, Xilinx zynq support from matlab and simulink, URL: https://in.mathworks.com/hardware-support/zynq.html..
[41] J. Yin, W. Wang, Q. Meng, R. Yang, J. Shen, A Unified Object Motion and Affinity Model for Online Multi-Object Tracking, in: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 6767–6776.
[42] X. Dong, J. Shen, W. Wang, L. Shao, H. Ling, F. Porikli, Dynamical Hyperparameter Optimization via Deep Reinforcement Learning in Tracking, IEEE Trans. Pattern Anal. Mach. Intell. 43(5) (2021) 1515–1529. doi:10.1109/TPAMI.2019.2956703..
[43] M. Danelljan, G. Bhat, F.S. Khan, M. Felsberg, ATOM: Accurate tracking by overlap maximization, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4660–4669..
[44] M. Danelljan, G. Hager, F.S. Khan, M. Felsberg, Learning spatially regularized correlation filters for visual tracking, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4310–4318.
[45] E. Park, A. Berg, Meta-Tracker: Fast and Robust Online Adaptation for Visual Object Trackers, url:https://arXiv.org/abs/1801.03049 (2018)..
[46] F. Li, C. Tian, W. Zuo, L. Zhang, M.-H. Yang, Learning spatial-temporal regularized correlation filters for visual tracking, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4904–4913.
[47] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, P.H.S. Torr, End-to-End Representation Learning for Correlation Filter Based Tracking, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5000–5008.
[48] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, H. Li, Unsupervised Deep Tracking, in: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1308–1317.
[49] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, P. Torr, Staple: Complementary Learners for Real-Time Tracking, in: Proc. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1401–1409..
[50] Xilinx, ZC702 Evaluation Board for the Zynq-7000 XC7Z020 SoC: User Guide (2017). URL: https://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/ug850-zc702-eval-bd.pdf..
[51] Xilinx, Zynq-7000 SoC Data Sheet: Overview (2019). URL: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf..
[52] Xilinx, Introduction to FPGA Design with Vivado High-Level Synthesis (2019). URL: https://www.xilinx.com/support/documentation/sw_manuals/ug998-vivado-intro-fpga-design-hls.pdf..
[53] S. Chandrakar, A. Clements, A. Sudarsanam, A. Dasu, Memory architecture template for fast block matching algorithms on fpgas, in: IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2009, pp. 1–8, https://doi.org/10.1109/IPDPSW.2010.5470751.

B. Praveenkumar received his B.E Degree in Electronics and Communication Engineering from Anna University, Chennai in 2011 and M.E Degree in VLSI Design from Anna University, Chennai in 2014. He is currently working towards his Ph.D Degree in Department of ECE, SRM Institute of Science and Technology, Tamil Nadu, India. He worked as an Assistant Professor in the Department of ECE in Muthayammal Engineering College (Autonomous), Tamil Nadu, India. He is an Associate Member of IETE and graduate student member of IEEE. His research interests include VLSI System Design, Embedded Systems, Reconfigurable Computing and Architectures.

P. Eswaran received his Bachelor degree in Electronics and Telecommunication Engineering from the Institute of Engineers (India) in 2000 and, the Masters in Mechatronics in 2003 from Madras Institute of Technology (India) and Ph.D. in Electronics and Communication Engineering from SRM University in 2014. Currently working as Professor in Department of Electronics and Communication Engineering at SRMIST (Formerly known as SRM University). His areas of interest are MEMS, VLSI, Device Modeling, EV and PV systems, embedded systems and Industry 4.0. He has published over 30 reviewed international journal/conference papers, and two Indian patents. He has served as a reviewer for peer reviewed journals. He is also Fellow Member of IE (India), Life Member of IETE and ISTE professional bodies.