Bounding box stabilization for visual object tracking using Kalman and FIR filters

ELI G. PALE-RAMON[1], YURIY S. SHMALIY[1], LUIS J. MORALES-MENDOZA[2] AND MARIO GONZÁLEZ-LEE[2]
[1] Electronics Engineerig Dept.
Universidad de Guanajuato
Km 3.5 +1.8 Salamanca-Valle Santiago road, Salamanca, Guanajuato, 36885
MEXICO
[2] Electronics Engineerig Dept.
Universidad Veracruzana
Ing. Guillermo Alvizouri, Poza Rica, Veracruz, 93390
MEXICO

Abstract: -In visual object tracking, the estimation of the trajectory of a moving object is a widely studied problem. In the object tracking process, there are usually variations between the real position of the objet in the scene and the estimated position, that is, the object is not exactly followed throughout its trajectory. These variations can be considered as color measurement noise (CMN) caused by the object and the camera frame movement. In this paper, we treat such differences as Gauss-Markov coloring measurement noise. We use Finite Impulse Response filters and Kalman filter with a recursive strategy in tracking: predict and update. To demonstrate the best performance, tests were carried out with simulated trajectories and with benchmarks from a database available online. The OUFIR and UFIR algorithms showed favorable results with high precision and accuracy in the object tracking task.

Key-Words: - Object tracking, state estimation, FIR filters, Kalman filter, bounding box

Received: February 26, 2021. Revised: December 17, 2021. Accepted: January 22, 2022. Published: February 11, 2022.

1 Introduction

The visual object tracking is a topic widely studied by various researchers, mainly due to multiple practical applications such as video surveillance and security, autonomous vehicle navigation, robotics, etc. [1] [2] [3]. Visual object tracking is a topic of interest in signal and image processing, in which the coordinates of the frame sequences are considered input data for the trajectory estimation [3] [4] [5].

Visual object tracking can be defined as the process of estimating the object trajectory in the image plane as it moves around a scene [5] [6]. However, during the tracking process the object is not followed exactly. There are variations in the estimated, that is, there is a discrepancy between the real position and the estimated position. These variations can be considered as colored measurement noise (CMN) which is not white [7]. An example of differences between real and estimated position is shown in Fig. 1 for the "Human2" benchmark [8], where a desirable frame is shown red and estimation errors in yellow, in this case, the target is the person dressed in blue.

It has been demonstrated that the use of a motion model and state estimators is a effective in avoiding large tracking errors [7] [9] [10] [11] [12] [13] [14].

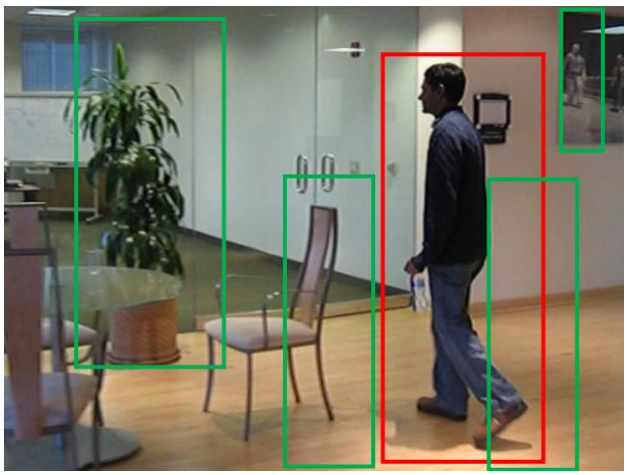

Fig. 1 Example Human tracking in a video sequence

In various investigations, it has been shown that if the model is correctly specified in the state space, it can represent the object dynamics for different movements with great precision [7] [9] [10] [11] [12] [13] [14].

For that reason, in this paper, we use the Kalman filter (KF), Optimal Finite Impulse Response (OFIR), Unbiased FIR (UFIR), and Optimal Unbiased FIR (OUFIR) are used in object tracking process to stabilize the bounding box trajectory. The state estimation method was developed is two steps: predict and update.

The FIR and Kalman filters are tested on simulated trajectories and benchmark data available on [8]. Based on these tests, we show that FIR and KF algorithms showed favorable results on simulated data with low data and process noise values, but with larger values, OUFIR and UFIR produced lower errors than KF and OFIR. Whereas, the results using the benchmark data showed a better performance of OUFIR.

2 Image processing

In order to carry the visual object, it is necessary to identify the object to be tracked. Image processing operations look for the best object recognition in tracking task, which involves finding the correct features to differentiate the target from other objects and the scene background. The image is divided into regions and the discontinuities are known as the boundaries between the regions [15] [16] [17].

An image can be described through its properties. To do this, it is necessary to calculate the mathematical properties of an image or region and use them as a basis for further classification [16]. Therefore, shape parameter extraction is necessary for image representation. One of the most commonly used shape parameters in object tracking is bounding box.

2.1 Bounding box

The Bounding box is a rectangular box that enclose an object in an image or scene. It can be represented by the coordinates of the upper left and lower right corners of the box [18]. When using the bounding box (BB) as a shape parameter in target tracking, information about the position of objects is contained in an array of the minimum and maximum vertices of the box that encloses the detected object within the scene. The distribution of pixels within a frame starts at the upper left corner and ends at the lower right corner [19]. The bounding box matrix is distributed over n rows and 4 columns, the rows represent the number of recognized objects and the columns contain the measurements for each bounding box located as follows:

$$BoundingBox=(Xc, Yc, Xw, Yh) \quad (1)$$

Where X_c, Y_c, X_w, and Y_h are the coordinates of 4 corners of the bounding box: corners, weight, and height. The algorithm to generate the bounding box in the tracking process must predict the four coordinates, X corner, Y corner, width, and height, for each bounding box.

In the tracking process there may be errors in the position estimation, an effective method to reduce them is to apply a filtering method. A filtering method us used to predict the coordinates of a point of bounding box. The aim of using prediction and correction methods is to mitigate the noise present in the object tracking process, the CMN. The prediction indicates the posterior position of the bounding box based on its previous position. The update is a correction step. It includes the new tracking model measurement and helps improve filtering [20] [21].

3 Tracking performance evaluation

The performance of tracking algorithms can be evaluated using metrics called precision and accuracy. Accuracy is the percentage of correct object detections, and the accuracy can be measured from the F-score which is an option metric to measure accuracy.

3.1 Precision

The precision is calculated from the other parameter, intersection over union (IoU). The equations for calculating precision and IoU are (2) and (3), respectively. The variables used in the calculation of the precision are obtained from the comparison of the IoU result with an established threshold [22].

$$IoU= \frac{IA}{(TBB- PBB)-IA} \quad (2)$$

Where IA is the intersection area of the true bounding box (TBB) and the estimated/predicted bounding box (PBB). The IoU value is calculated in each position of the bounding boxes.

$$Precision= \frac{\sum TP}{(\sum TP + \sum FP)} \quad (3)$$

Where TP is true-positive, and FP is false-positive.

To determine the correct object detection of the target, a threshold value of IoU must be established. IoU is generally set to 0.5 [22]. Assuming the IoU threshold is 0.5, if the value is greater or equal to 0.5, the detection is classified as True Positive (TP). If IoU value is less than 0.5 it is considered as a wrong detection and classified as False Positive (FP). The IoU threshold can be set to a value of 0.5 or more, such as 0.75. 0.9, 0.95, or 1.

3.2 Accuracy

One metric to measure the accuracy in object tracking model is F-score [23] [24]. The F-score use the precision and recall and it can provide important information about the model performing at various threshold values. Recall can be calculated as the number of correct detected objects divided by the total number of detections in the ground truth. This metric is based on the bounding box overlap obtained between the algorithm and the real trajectory to calculate the accuracy with which the algorithm operates on an object displacement sequence, the F-score is computed by equation (4).

$$F-score = 2\frac{Precision * Recall}{Precision + Recall} \qquad (4)$$

4 State-space model

We consider a moving object represented in discrete-time state-space with the following state and observation equations (5) and (6).

$$x_n = F_n x_{n-1} + E_n u_n + B w_n \qquad (5)$$
$$y_n = C_n x_n + v_n \qquad (6)$$

Where $x_n \in \mathbb{R}^k$ is the state vector, $y_n \in \mathbb{R}^M$ is the vector of observation. F is the model of the state transition, which is applied to project the previous state x_{n-1} to x_n. E is the input control model, u_n is the input control, B is the noise matrix. C is the observation model. $w_n \in \mathbb{R}^P$ is the process noise $v_n \in \mathbb{R}^M$ is the colored Gauss-Markov noise with white Gaussian with zero mean $w_n \sim N(0, Q_n) \in \mathbb{R}^P$ and $v_n \sim N(0, R_n) \in \mathbb{R}^M$ have the covariances Q_n and R_n, and the property $E\{w_n v_k^T\}=0$ for all n and k. Under the assumption that the two noise sequences and the initial state are uncorrelated and independent of each testing instant [25].

5 Kalman filter

The Kalman filter uses the equation of state of the linear system to estimate the state of the system through observation of input and output. The KF requires knowledge of the system parameters, initial values, and measurement sequences. The KF can estimate the state sequences of the system iteratively [26].

The Kalman filter calculate the optimal state estimate by recursively combining previous estimates with new observations. It consists of two steps: predict, where the optimal state $\hat{x}_n^-$ previous to observing y_n is calculated and update, where after observing y_n the optimal posterior state $\hat{x}_k$ is calculated. Additionally, it computes the prior estimation error $\epsilon_n^- = x_n - \hat{x}_n^-$, the posterior estimation $\epsilon_n = x_n - \hat{x}_n$, a priori estimate error covariance $P_n^- = E\{\epsilon_n^- \epsilon_n^{-T}\}$, and posterior estimate error covariance $P_n = E\{\epsilon_n \epsilon_n^T\}$.

The a priori error covariance matrix is produced in the predict step. Since the process noise w_n is assumed white Gaussian with zero mean, the a priori state estimate is given by (7), and the a priori error covariance matrix is estimated by (8).

$$\hat{x}_n^- = F\hat{x}_{n-1} + E_n u_n \qquad (7)$$
$$P_n^- = F_n P_{n-1} F + B_n Q_n B_n^T \qquad (8)$$

Then, in the update stage, the current a priori predictions are combined with the current state observation to redefine the state estimate and the matrix of error covariance. The current observation is used to improve the estimation, and it is called a posteriori estimation of the state.

The measurement y_n is corrupted by the noise v_n. Since v_n is assumed white with zero mean, this becomes (9), and the measurement residual (10).

$$y_n = C\hat{x}_{n-1} \qquad (9)$$
$$z_n = y_n - C_n \hat{x}^- \qquad (10)$$

The residual covariance matrix is given by:

$$S_n = C_n P_n^- C_n^T + R_n \qquad (11)$$

The optimal Kalman gain:

$$K_n = P_n^- C_n^T S_n^{-1} \qquad (12)$$

A posteriori state estimate:

$$\hat{x}_n = \hat{x}_n^- + K_n(z_n - C\hat{x}_n^-) \qquad (13)$$

A posteriori error covariance matrix:

$$P_n = (I - K_n C)P_n^- \qquad (14)$$

A pseudo-code of the Kalman filter is listed as Algorithm 1.

Algorithm 1: Kalman Filter

Data: $y_n, u_n, \hat{x}_0, P_0, Q_n, R_n$
Result: $\hat{x}_n, P_n$
Begin
  for $n = 1,2, \ldots$ **do**
    $\hat{x}_n^- = F\,\hat{x}_{n-1} + E_n u_n$
    $P_n^- = F_n P_{n-1} F_n^T + B_n Q_n B_n^T$
    $S_n = C_n P_n^- C_n^T + R_n$
    $K_n = P_n^- C_n^T S_n^{-1}$
    $\hat{x}_n = \hat{x}_n^- + K_n(y_k - C_n \hat{x}_n^-)$
    $P_n = (I - K_n C)P_n^-$
  end for
End

6 Optimal Finite Impulse Response (OFIR)

For faster processing, the OFIR algorithm is used in its iterative form. Its iterative computation on horizon [m, k] for given $\hat{x}_m$ and P_m provided by Kalman filter (Algorithm 1) if we change the auxiliary time-index *i* from *m +1* to *n* and take output when *i =n*. The iterative form for the OFIR filter has been developed and tested in **[27] [28]**.

The pseudo-code of the a posterior iterative OFIR filter is listed as Algorithm 2. Given $\hat{x}_m$ and P_m, this algorithm is iteratively updates $\hat{x}_i$ values from *i = m +1* to *i = n* using optimal recursions of KF (Algorithm 1) and obtains $\hat{x}_n$ and P_n, when i = *n.* The number of iterations can be limited by optimal horizon length, N_{opt} of the Unbiased Finite Impulse Response (UFIR) filter.

Algorithm 2: Iterative OFIR filter

Data: $y_n, u_n, \hat{x}_m, P_m, Q_n, R_n, N$
Result: $\hat{x}_n$
Begin
  For *n=1,2, ...* ***do***
    $m = n - N + 1$ if $k > N - 1$ and $m = 0$ otherwise
    For *i=m+1,m+2,..., n* ***do***
      Algorithm 1: $\hat{x}_i, P_i$
    end for
  end for
  $\hat{\boldsymbol{x}}_{\boldsymbol{n}}, P_n$
End

7 Unbiased Finite Impulse Response (UFIR)

Unlike the KF and iterative OFIR, the Unbiased FIR does not require any information about initial conditions and noise, except for the zero mean assumption [9] [14] [29] [30] [31]. Therefore, the Unbiased FIR filter is more suited for object tracking, where measurement and process noises are not exactly known. However, the UFIR filter requires an optimal horizon length, N_{opt}, from $m = n - Nopt + 1$ to n, to minimize the Mean Squared Error, and cannot ignore the CMN v_n, which violates the zero-mean assumption on short horizons.

Since the UFIR algorithm does not require noise statistics, the prediction phase calculates only one value, a priori state.

$$\hat{x}_l^- = F\hat{x}_{l-1} + E_l u_l \tag{15}$$

In the update step, the state estimate is combined with the current observation state to refine the state. The estimate is iteratively updated to the a posteriori state estimate using (16) -(19).

Generalized noise power gain:

$$G_l = \left[C_l^T C_l + (F_L G_{l-1} F_l^T)^{-1}\right]^{-1} \tag{16}$$

The measurement residual:

$$z_l = y_l - C_l \hat{x}^- \tag{17}$$

The UFIR gain:

$$Gain_l = y_l - C_l \hat{x}^- \tag{18}$$

The a posteriori state estimate is given by:

$$\hat{x}_l = \hat{x}_l^- + Gain_l(z_l - C\hat{x}_l^-) \tag{19}$$

A pseudo-code of the UFIR algorithm is listed as Algorithm 3. To initialize iterations, the algorithm requires a short measurement vector $y_{m.k} = [y_m \dots y_k]^T$ and matrix (20).

$$H_{m,s} = \begin{bmatrix} C_m\left(F_k^{m+1}\right)^{-1} \\ C_{m+1}\left(F_k^{m+2}\right)^{-1} \\ \vdots \\ C_{k-1}F_k^{-1} \\ C_k \end{bmatrix} \tag{20}$$

Algorithm 3: Unbiased FIR filter

Data: y_n, u_n, N
Result: $\hat{x}_n$
Begin
  For *n= N−1, N, ...* ***do***
    $m = n - N + 1, s = n - N + K$
    $G_S = (H_{m,s}^T H_{m,s})$
    $\tilde{x}_s = G_s H_{m,s}^T (Y_{m,s} - L_{m,s} u_{m,s}) + S_{m,s}^k u_{m,s}$
    For *l=s+k*: **do**
    $\tilde{x}_l^- = F\hat{x}_{l-1} + Eu$
    $G_l = \left[C_l^T C_l + \left(F_l G_{l-1} F_l^T\right)^{-1}\right]^{-1}$
    $Gain_l = G_l C_l^T$
    $\tilde{x}_l = \hat{x}_l^- + Gain_l(y_l - C\tilde{x}_l^-)$
    end for
  end for
  $\hat{x}_n = \tilde{x}_n$
End

Where $S_{m,s}$ and $L_{m,s}$are given by (21) and (22) respectively, $S_{m,s}^k$ is de *Kth* row vector in (21).

$$S_{m,s} = \begin{bmatrix} E_0 & 0 & \dots & 0 & 0 \\ F_{m+1}E_m & E_{m+1} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ F_{s-1}^{m+1}E_m & F_{s-1}^{m+2}E_{m+1} & \dots & E_{s-1} & 0 \\ F_s^{m+1}E_m & F_s^{m+2}E_{m+1} & \dots & F_s E_{s-1} & E_s \end{bmatrix} \tag{21}$$

$$L_{m,s}=diag(C_{m,s})S_{m.s} \quad (22)$$

8 Optimal Unbiased Finite Impulse Response (OUFIR)

Generally, in real object tracking applications, not all information about the initial conditions of the model is available. Since the OUFIR filter is very indifferent to the initial conditions. In this case, we apply an Optimal Unbiased iterative filter (OUFIR) [32].

The iterative OUFIR algorithm is stated below. The prediction phase calculates a single value, a priori state, considering input (u) equal to zero the a priori state is computed by (23).

$$\tilde{x}_{\boldsymbol{l}} = F_l\hat{x}_l^- + Gain_l(y_l - C_i\tilde{x}_l^-) \quad (23)$$

In the update stage, the state estimated is combined with the current observation state to refine the state. In the same way as UFIR, the estimate is iteratively updated to the a posterior state estimate using equations (24) -(28).

The residual covariance matrix is given by:

$$S_n=C_nP_n^-C_n^T+R_n \quad (24)$$

The OUFIR filter gain:

$$K_l=P_n^-C_n^TS_n \quad (25)$$

$$K_l^-=(I-K_lC_l)\ G_lN_lG_l^TC_l^TS_n \quad (26)$$

$$Gain=K_l+K_l^- \quad (27)$$

The a posteriori state estimate:

$$\tilde{x}_{\boldsymbol{l}} = \hat{x}_l^- + Gain_l(y_l - C_i\tilde{x}_l^-) \quad (28)$$

A pseudo-code of the Optimal Unbiased FIR filter is listed as Algorithm 4.

9 Object tracking tests

The KF and FIR algorithms were tested on simulated data and benchmark data available in [8].

9.1 Test on simulated data

In this section, we test the algorithms numerically by different simulated data. Our main goal is to evaluate the performance in object tracking using precision and F-score metrics. We consider the two-state model and suppose an object is disturbed by white Gaussian acceleration noise with a given value of standard deviation. The model of a moving target in a two-dimensional space was specified by (5) and (6) with matrices:

$$\mathrm{F}=\begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix},\qquad B=\begin{bmatrix}T\\ 1\end{bmatrix},\qquad C=[1\ 0].$$

In simulation data two scenarios will be considered:

Algorithm 4: Iterative OUFIR filter

Data: y_n, Q_n, R_n, N
Result: $\hat{x}_n$
Begin
  For $n = N-1, N, \ldots$ ***do***
    $m = n - N + 1$, if $k > N-1$ and $m = 0$ otherwise
    Compute $\hat{x}_{m+1}$ and P_{m+1}
    For *l=m+2, m+3, … n* **do**
    $P_l^- = F_lP_{l-1}F_l^T + B_lQ_lB_l^T - F_lP_lF_l^TS_lC_lP_nF_l^T$
    $G_l=F_l(I-P_n^-C_l^TS_lC_l^T)G_{l-1}$
    $S_l=C_lP_l^-C_l^T+R_l$
    $N_l=\left(N_{l-1}^{-1}+G_l^TC_l^TS_lC_lG_l\right)^{-1}$
    $K_l=P_n^-C_n^TS_n$
    $K_l^-=(I-K_lC_l)\ G_lN_lG_l^TC_l^TS_n$
    $Gain=K_l+K_l^-$
    $\tilde{x}_{\boldsymbol{l}} = \hat{x}_l^- + Gain_l(y_l - C_i\tilde{x}_l^-)$
    end for
  end for
  $\hat{x}_{\boldsymbol{n}} = \tilde{x}_{\boldsymbol{n}}$
End

1) Simulated data 1. An object target is disturbed by white Gaussian acceleration noise with a standard deviation of $\sigma_w = 4m/s^2$. The for the data noise originates from white Gaussian $\sigma_v = 3m$. The simulation of the trajectory was 1000 points with sample time T= 0.05 seconds, $P_0 = 0$, $Q = \sigma_w^2$, $R = \sigma_V^2$, on a short horizon N_{opt} = 15.

2) Simulated data 2. A trajectory is simulated at 1000 points $\sigma_w = 45m/s^2$, $\sigma_v = 25m$. With sample time T= 0.05 seconds, $P_0 = 0$, $Q = \sigma_w^2$, $R = \sigma_V^2$, on a short horizon N_{opt} = 10.

9.1.1 Test on simulated data 1

We examine the results of algorithms in the object tracking simulation using the bounding box coordinates as a metric of evaluation. In Fig. 2 the true trajectory of the object and the estimations made by the algorithms are shown, where the black line is the true trajectory, the blue line is KF, the red line is UFIR, the yellow line is OUFIR, and the green line is OFIR. Given that the N_{opt} for the FIR filters was 15, the estimates started from this.

With low values of white Gaussian acceleration noise and data noise, the OUFIR and UFIR filters showed similar behavior. To provide a more complete view, we calculate the root mean square error (RMSE). The RMSE values were 929.98 for KF, 929.93 for OFIR, 444.55 for OUFIR, and 478.21 for UFIR. According to these results, we

consider that the OUFIR algorithm presented the best performance with the lowest RMSE value.

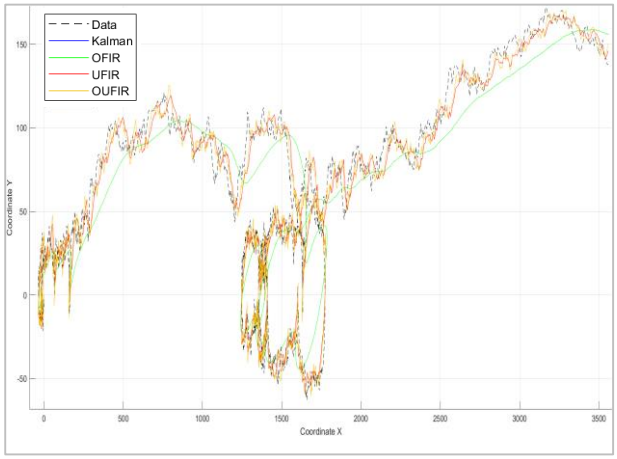

Fig. 2 Trajectory estimation of data 1 using Kalman and FIR filters.

The test precision results of simulated data 1 are shown in Fig. 3. The OFIR and KF algorithms produced a low precision with similar behavior on the all thresholds range. According to the results, it can be inferred that in each detection the overlap of the Predicted Bounding Box (PBB) on the True Bounding Box (TBB) was poor. It can be inferred that each detection of the Kalman and OFIR filters covers less to 50% of the TBB area. Since the most used threshold values are 0.50% and 0.75% [22]. Therefore, we consider that Kalman and FIR filters algorithms gave poor results in the most widely used threshold IoU range. On the other hand, the OUFIR and UFIR filters showed good results with an average precision of 0.87 and 0.85, respectively, that is, they cover at least 80% of the TBB area.

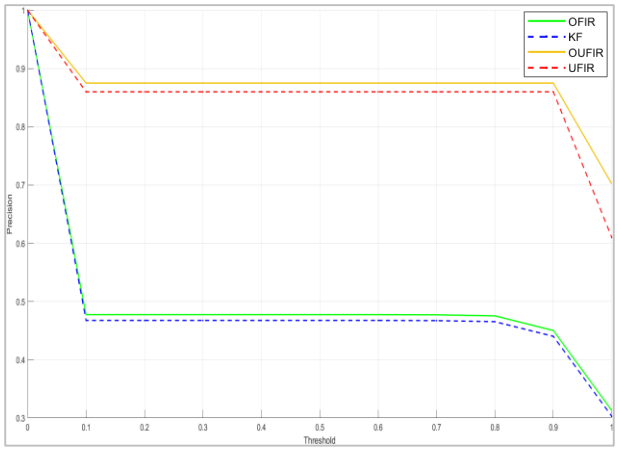

Fig. 3 Precision of simulated data 1

The F-score metric was used to measure the accuracy. This metric is based on the bounding box overlap obtained between the algorithm and the true trajectory to calculate the accuracy with which the algorithm operates on an object trajectory. The results of the F-score for simulated data 1 are shown in Fig. 4. The OUFIR and UFIR algorithms produced a high accuracy from 0 to 0.9 threshold, from which to decay. The OUFIR filter showed the highest accuracy with average of 0.87, closely followed by UFIR with 0.84. OFIR and KF algorithms presented lowest values with average of 0.50 and 0.49 , respectively.

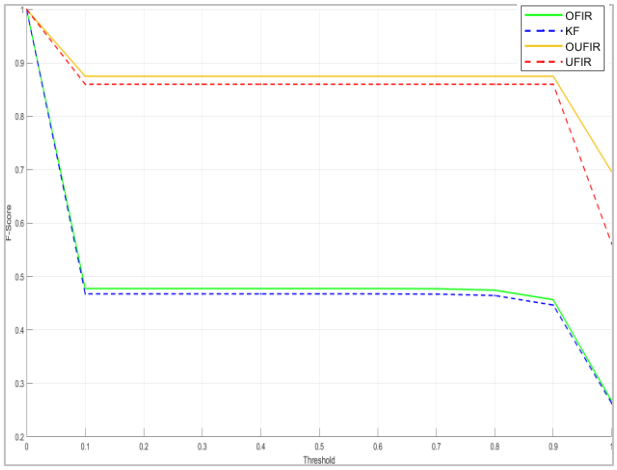

Fig. 4 Accuracy of simulated data 1

9.1.2 Test on simulated data 2

Next, we analyze the simulated data 2 test. In Fig. 5 the real trajectory of the object and the estimations made by the algorithms are shown, where the black line is the true trajectory, the blue line is KF, the red line is UFIR, the yellow line is OUFIR, and the green line is OFIR. Given that the N_{opt} for the FIR filters was 10, the estimates started from this.

With high values of acceleration noise and data noise, the OUFIR and UFIR filters showed similar behavior. To analyze this, we calculate the root mean square error (RMSE). The RMSE values were 6963.56 for KF, 6962.39 for OFIR, 3149.15 for OUFIR, and 3149.15 for UFIR. Therefore we consider the OUFIR and UFIR showed best performance.

The precision results of simulated data 2 test are shown in Fig. 6. OUFIR and UFIR presented a better performance, these produced a precision over 77% from 0 to 0.9 threshold, from which to decay. It can be inferred that each detection covers at least 77% of the TBB area. While OFIR and KF presented low precision values below 40% in the threshold range. The average precision for OUFIR was 0.79, for UFIR was 0.79, for OFIR was 0.29, and for KF was 0.29. As already mentioned, usually the threshold used is 0.5, we consider that the OUFIR and UFIR algorithms gave favorable results.

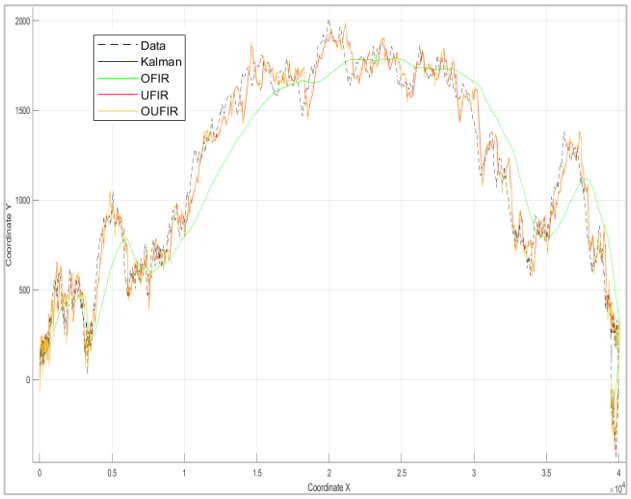

Fig. 5 Trajectory estimation of simulated data 2 using Kalman and FIR filters.

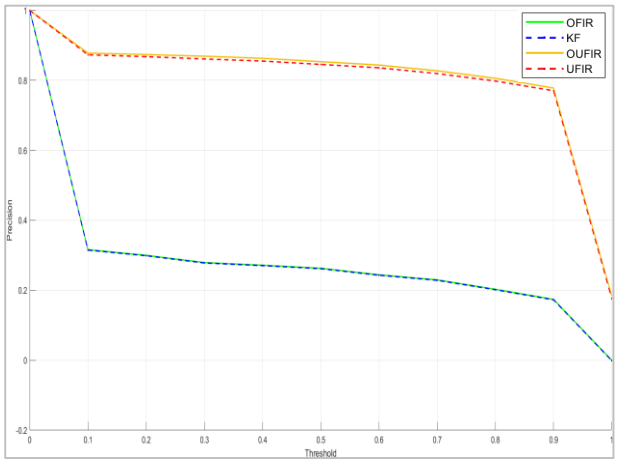

Fig. 6 Precision of simulated data 2

In Fig. 7 the accuracy results of simulated data 2. The OUFIR and UFIR algorithms produced accuracy values over 0.6 from 0 to 0.8 threshold, from which to decayed. The accuracy value towards the 1 threshold is close to 0.2. The OFIR and KF showed a poor performance from 0.1 to 1 threshold, with an accuracy below 0.3. The accuracy value decreases as the threshold value increases. The OUFIR filter showed the highest accuracy with an average of 0.72, closely followed by UFIR with 0.71, the average for OFIR was 0.29, and for KF was 0.28. With the given conditions, and according to the results, we can determine that the OUFIR and the UFIR present a good performance in the object tracking process.

9.2 Test on benchmark trajectory

Then, we realized a test with a benchmark trajectory, called "SUV" , available on [8]. The coordinates of the SUV trajectory are measured by a visual object tracking system. The SUV moves and maneuvers on a highway road.

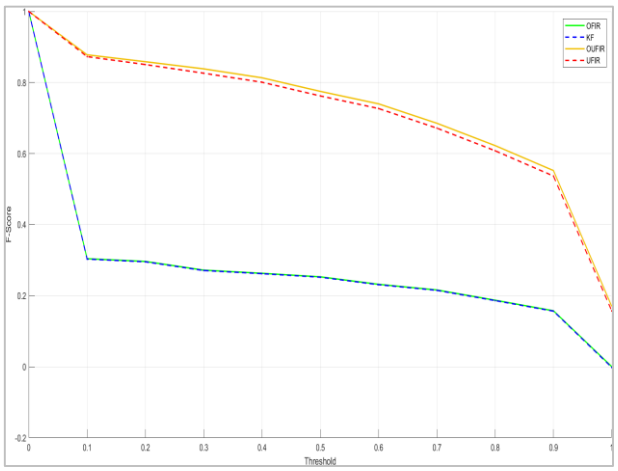

Fig. 7 Accuracy of simulated data 2

For the test, we considered that an object is disturbed by white Gaussian acceleration noise with the standard deviation of $\sigma_w = 10m/s^2$. The for the data noise (CMN) originates from white Gaussian $\sigma_v = 5m$. With sample time T= 0.05 seconds, $P_0 = 0$, $Q = \sigma_w^2$, $R = \sigma_V^2$, and the model of a moving target in a two-dimensional space can be specified by (5) and (6) with:

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ B = \begin{bmatrix} \frac{T^2}{2} & 0 & T & 0 \\ 0 & \frac{T^2}{2} & 0 & T \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Since the UFIR filter requires an optimal averaging horizon [m, n] of N_{opt} points. Following [33], we determine:

$$N_{opt} = \sqrt{\frac{12\sigma_v}{T^2\sigma_w}} \cong 49$$

Finally, we analyze the results obtained for the "SUV" trajectory. The true trajectory and the estimates by FIR and Kalman algorithms are shown in Fig. 8. The identification colors remain as already mentioned previously in this paper. The N_{opt}, as already mentioned for the FIR filters, was 49.

In this test, the estimates of OUFIR and UFIR were the best compared with those obtained by KF and OFIR. Using a quantitative measurement, we calculate the root mean square error (RMSE). The RMSE values were 2316.90 for OUFIR, 2361.90 for UFIR, 4962.39 for OFIR, and 4963.56 for KF. According to these results, we consider that the OUFIR and UFIR algorithms presented a good performance, where OUFIR and UFIR present a the best performance in the object tracking task under the given conditions

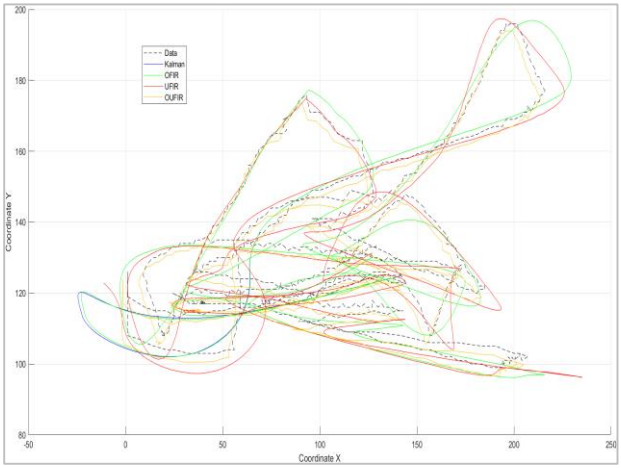

Fig. 8 Trajectory estimation of "SUV" benchmark using Kalman and FIR filters.

Analyzing the precision metric, it is observed that the performance of the algorithms was higher than in the previous simulated tests. The results are shown in Fig. 9. Again, the OUFIR and UFIR showed higher precision compared to OFIR and KF. The precision mean is 0.84 for OUFIR, 0.83 for UFIR. Taking as a reference that the Threshold most used to evaluate the precision is 0.5, in this value the precision is over 80% we can determine that the performance of OUFIR and UFIR were good. In the same way, the performance of KF and OFIR was good, although lower than those already mentioned, with an average precision of 0.68 and 0.66, respectively.

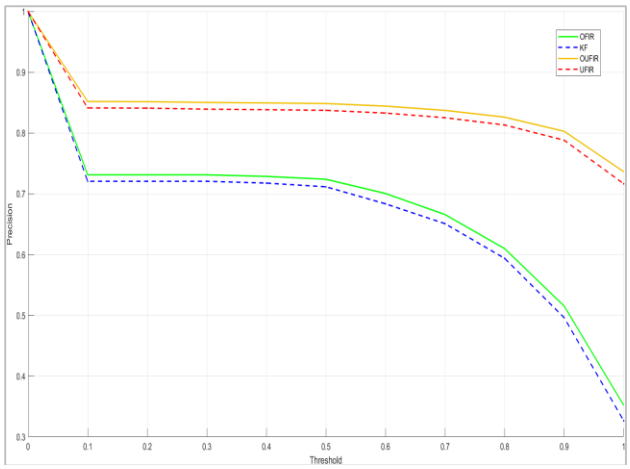

Fig. 9 Precision of "SUV" benchmark

The F-score values, accuracy, are shown in Fig. 10. The OUFIR and UFIR algorithms produced accuracy values over than 0.7 in the 0.1 to 0.8 threshold range. OFIR and KF algorithms produced accuracy values over 0.7 from 0.1 to 0.5 threshold range, from which to decay. The OFIR and KF presented a lower performance than UFIR and OUFIR. According to the results, we consider that OUFIR and UFIR present the best performance in the object tracking process under the given assumptions in a benchmark trajectory.

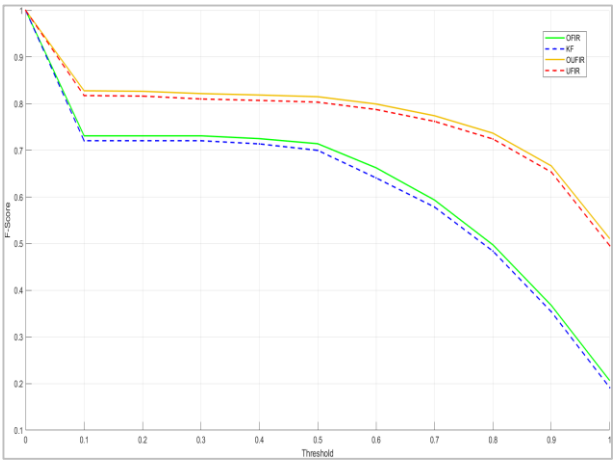

Fig. 10 Accuracy of "SUV" benchmark

10 Conclusion

The KF and OFIR estimation algorithms seem to be less efficient than the OUFIR and UFIR in object tracking process under the conditions given in this paper. On the other hand, the algorithms OUFIR and UFIR showed favourable results in object tracking tests and provided state estimation with higher precision and accuracy, which can be useful in many visual tracking applications such as video surveillance and security, robotics, autonomous vehicle navigation, etc. Remarking that, UFIR does not require noise information and to know the initial position. Likewise, the OUFIR is highly insensitive to initial conditions.

According to the accuracy and precision results, the OUFIR filters showed better performance for tracking objects. Consequently, the OUFIR filter in general shows higher robustness against initial conditions and noise statistics than UFIR, OFIR and KF.

Therefore, we conclude that the incorporation of state estimators and the use of OUFIR and UFIR filtering can provide further development of object tracking algorithms for a wide variety of application areas.

References:

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1– 45, 2006.

[2] T. K. Kang, Y. H. Mo, D. S. Pae, C. K. Ahn, and M. T. Lim, "Robust visual tracking framework in the presence of blurring by

arbitrating appearance- and feature-based detection," *Measurement*, vol. 95, pp. 50 – 69, 2017.

[3] A. N. Bishop, A. V. Savkin, and P. N. Pathirana, "Vision-based target tracking and surveillance with robust set-valued state estimation," *IEEE Signal Process. Lett.*, vol. 17, no. 3, pp. 289 – 292, 2010.

[4] X. Zhou, Y. Li, B. He, and T. Bai, "GM-PHD-based multi-target visual tracking using entropy distribution and game theory," *IEEE Trans. on Ind. Informatics*, vol. 10, no. 2, pp. 1064 – 1076, 2014.

[5] M. Parmar, "A survey of video object tracking methods," *GIT-J. of Eng. and Technol.*, vol. 9, pp. 40 – 46, 2016.

[6] H. S. Parekh, D. G. Thakore, and U.K. Jaliya, "A Survey on Object Detection and Tracking Methods," *Int. J. of Innovative Research in Comp. and Commun. Eng.*, vol. 2, no. 2, pp. 2970 – 2978, 2014.

[7] Y. Yoon, A. Kosaka, and A. Kak, "A new Kalman-filter-based framework for fast and accurate visual tracking of rigid objects," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1238 – 1251, 2008.

[8] Computer Vision Lab. (2015). *Datasets-visual tracker benchmark*
[Online]. Available: http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html.

[9] D. Simon, *Optimal state estimation: Kalman, H_∞, and nonlinear approaches*. Hoboken, N.J.: John Wiley & Sons, 2006.

[10] P. Liang, E. Blasch, and H. Ling, "Encoding color information for visual tracking: algorithms and benchmark," *IEEE Trans. Image Process*, vol. 24, no. 12, pp. 5630–5644, 2015.

[11] Y.S. Shmaliy, S. Zhao, and C. K. Ahn, "Kalman and UFIR state estimation with coloured measurement noise using backward Euler method," *IET Signal Process.*, vol. 14, no. 2, pp. 64 –71, 2020.

[12] Y. S. Shmaliy, "Linear optimal fir estimation of discrete time-invariant state-space models," *IEEE Trans. on Signal Process.*, vol. 58, no. 6, pp. 3086 – 3096, 2010.

[13] S. Zhao, Y.S. Shmaliy, and C. Ahn, "Bias-constrained optimal fusion filtering for decentralized WSN with correlated noise sources," *IEEE Trans. on Signal and Inform. Process. over Networks*, vol. 4, no. 4, pp. 727–735, 2018.

[14] Y. S. Shmaliy, J. Andrade-Lucio, E. G. Pale-Ramon, J. Ortega-Contreras, L. J. Morales-Mendoza, and M. González-Lee, "Visual Object Tracking with Colored Measurement Noise using Kalman and UFIR Filters," in *2020 17th Int. Conf. on Elect. Eng., Computing Sci. and Automat. Control (CCE),* Mexico City, Mexico, 2020, pp. 1– 6.

[15] B. Jahne, *Practical handbook on image processing for scientific and techniucal applications*, 2nd ed. Boca Raton, FL: CRC press, 2004.

[16] W. Burger and M. Burger, *Principles of digital image processing*. London: Springer, 2009.

[17] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.

[18] K. Choeychuent, P. Kumhomtand, and K. Chamnongthait, "An Efficient Implementation of the Nearest Neighbor Based Visual Objects Tracking," in *2006 Int. Symp. on Intelligent Signal Process. and Commun. Syst.*, Tottori, Japan, 2006, pp. 574 – 576.

[19] A. Farhadi and J. Redmon. (2018). *Yolov3: An incremental improvement* [Online]. Available: https://arxiv.org/abs/1804.02767.

[20] P. Deepak and S. Suresh, "Design and Utilization of Bounding Box in Human Detection and Activity Identification," in *Emerging ICT for Bridging the Future-Proc. of the 49th Annu. Conv. of the Comp. Soc. of India CSI Volume 2*, Springer, Cham, 2015, pp. 59–70.

[21] M. S. Grewal and A. Andrews, *Kalman filtering: Theory and Practice with MATLAB*. Hoboken, NJ: John Wiley & Sons, 2014.

[22] R. Padilla, W. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit," *Electronics*, vol. 10, no. 279, 2021.

[23] A. W . Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan and M. Shah. "Visual tracking: An experimental survey*," IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1442-1468, 2013.

[24] B. Karasulu and. K. S. Karasulu, "A software for performance evaluation and comparison of people detection and tracking methods in video processing," *Multimedia Tools and Applications*, vol. 55, nº 3, pp. 677-723, 2011.

[25] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan, *Estimation with applications to*

tracking and navigation. Hoboken, N.J.:John Wiley & Sons, Inc., 2001.
[26] R. G. Brown, and P. Y. Hwang, *Introduction to random signals and applied Kalman filtering: with MATLAB exercises*, 4th ed. Hoboken, N.J: John Wiley & Sons, 2012.
[27] S. Zhao, Y. S. Shmaliy, S. H. Khan and G. Ji, "Iterative Form for Optimal FIR Filtering of Time-Variant Systems.," *Recent Advances on Electroscience and Computers,* pp. 114, 2015.
[28] S. Zhao, Y. S. Shmaliy and F. Liu, "Fast computation of discrete optimal FIR estimates in white Gaussian noise," *IEEE Signal Processing Letters*, vol. 22, no. 6, pp. 718-722, 2014
[29] Y. S. Shmaliy, "An Iterative Kalman-Like Algorithm Ignoring Noise and Initial Conditions," *IEEE Transactions on Signal Processing,* vol. 59, no. 6, pp. 2465-2473, 2011.
[30] Y. S. Shmaliy, S. Zhao and C. K. Ahn, "Kalman and UFIR state estimation with colored measurement noise using backward Euler method," *IET Signal Processing*, vol. 14, no. 2, pp. 64-71, 2020.
[31] Y. S. Shmaliy, S. Khan, and S. Zhao, "Unbiased FIR filtering: An iterative alternative to Kalman filtering ignoring noise and initial conditions," *IEEE Control Syst. Mag.*, vol. 37, no. 5, pp. 70–89, 2017.
[32] S. Zhao, Y. Shmaliy and F. Liu, "Fast Kalman-like optimal unbiased FIR filtering with applications," *IEEE Transactions on Signal Processing,* vol. 64, no. 9, pp. 2284-2297, 2016.
[33] F. Ramirez-Echeverria, A. Sarr, and Y. S. Shmaliy, "Optimal memory for discrete-time fir filters in state-space," *IEEE Trans. on Signal Process.*, vol. 62, no. 3, pp. 557–561, 2014.

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

Eli G. Pale-Ramon has written, reviewed, edited the paper, and implemented the Algorithms in Matlab.
Yuriy S. Shmaliy has developed the methodology; creation of models, and the project administration
Luis J. Morales-Mendoza has supervised and reviewed the paper, was responsible for the validation metrics used
Mario González-Lee was responsible for the validation of the computational methods applied.

Creative Commons Attribution License 4.0 (Attribution 4.0 International , CC BY 4.0)