Memorias del Congreso Nacional de Control Automático
ISSN: 2594-2492

Video object tracking using Kalman and Unbiased FIR Filters

Eli G. Pale-Ramon, Yuriy S. Shmaliy, Jorge A. Ortega-Contreras, Miguel A. Vázquez-Olguín.
Electronics Engineering Dept.
Universidad de Guanajuato
Salamanca, Gto., Mexico
{eg.paleramon, shmaliy, ja.ortegacontreras, miguel.vazquez} @ugto.mx

Luis J. Morales-Mendoza, Mario González-Lee
Electronics Engineering Dept.
Universidad Veracruzana
Poza Rica, Ver., Mexico
{javmorales, mgonzalez01} @uv.mx

Abstract: Tracking of moving objects is a well-known problem of estimating the trajectory of a target in a video sequence. The video object tracking generally is accompanied by variations in the size and position of image frames; that is, consecutive video frames do not follow the object with precision in the tracking process. These variations can be considered colored measurement noise (CMN) caused by the object and camera frame dynamics. In this paper, we treat such variations as a Gauss-Markov color measurement noise. A recursive strategy in object tracking is used for the unbiased finite impulse response (UFIR) and Kalman filter (KF); each recursion has two recognized phases: predict and update. Filters are showed to be able to produce a high precision in object tracking under CMN. The standard Kalman and UFIR algorithms are tested in video sequences with different factors to affectation to demonstrate the best performance.

Keywords: Video object tracking, colored measurement noise, Kalman filter, unbiased FIR filter.

1. INTRODUCTION

Visual object tracking remains a complex problem to tackle and of great study interest. Object tracking is a widely researched field within computer vision, mainly due to the many practical applications such as video surveillance and security, robotics, human-computer interaction, autonomous vehicle navigation, etc. (Yilmaz, et al., 2006; Bishop, et al., 2010; Kang, et al., 2017). It can be defined as the problem of estimating the trajectory of an object in the image plane while moving around a scene (Parekh, et al., 2014; Parmar, 2016).

Likewise, object tracking is accompanied by variations in the size and position of the frames, typically bounding boxes; that is, the target is not exactly followed in the tracking process. These variations can be considered as colored measurement noise (CMN). An example of variations in video object tracking is shown in Fig. 1 for the "blurcar4" benchmark (Computer Vision Lab, 2015), where a desirable frame is shown green. In this case, the target is a pickup truck.

An effective way to avoid significant tracking errors is to use a motion model and state estimators (Simon, 2006; Yoon, et al., 2008; Shmaliy, 2010; Liang, et al., 2015; Shmaliy, et al., 2020). If the model in state space is appropriately specified, it can be represented with great precision the dynamics of the object for different motions. However, the accuracy of the tracking algorithm will continue to depend heavily on residues represent noise measurement data and mismodeling errors.

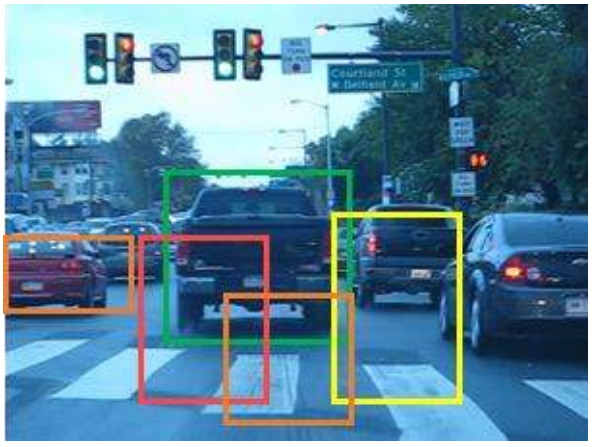

Fig. 1 Example of pickup truck tracking in a video sequence

Therefore, in this paper, the algorithms Kalman filter (KF) and Unbiased Finite Impulse Response filter (UFIR) are using to stabilize the trajectory of the bounding box during object tracking, under the assumption that variations in the image frame can be considered as CMN. For both algorithms, the state estimation strategy was in two different phases in the recursions and iterations: predict and update.

The UFIR and KF algorithms were tested on video sequences with different affectation factors of tracking. We used a classical method of object initialization, a combined method of region labeling and contour search, to test the robustness of both algorithms. We evaluated the performance of KF and UFIR; based on the results, both algorithms showed favorable results, but UFIR produced smaller errors than KF.

2. IMAGE PROCESSING

Before starting to visual tracking process, it is necessary to initialize the target. Image processing operations seek the best recognition of objects in tracking, which involves finding the correct characteristics to differentiate the target of other objects and the background. Subsequently, whether the pixel belongs to the target is verified, a method is known as segmentation, which produces a binary image. A pixel has the value one if it belongs to the object; otherwise, it is zero. The image is divided into regions, and the discontinuities are known as the boundaries between the regions (Jahne, 2004; Burger and Burger, 2009; Solomon & Breckon, 2011).

However, in this way, it is not possible to determine how many objects are in the general scene. Therefore, it is better to analyze connected regions in images. The searching process for binary regions is to find out what belongs to which regions, how many, and where these are located. This process is called region labeling or region coloring (Jahne, 2004; Burger and Burger, 2009). Different methods are known as flood fill, sequential marking, and a combined region labeling and contour search method. Regardless of the method, region marking is required (Jahne, 2004): 0 is the background, 1 is the foreground, 2 and 3 are the region label.

The image content can be described through its properties. In this way, it is necessary to calculate the image properties and use them as a basis for further classification. Therefore, the extraction of shape parameters is essential for image representations. One of the parameters most used in object tracking is the bounding box (Burger and Burger, 2009).

2.1 Bounding box

The bounding box (BB) is a rectangular box that encloses all the objects in an image or scene. It can be represented by giving coordinates of the box´s upper left and lower right corners (Choeychuent, et al., 2006). The object position information in pixels is contained in an array of bounding boxes. The BB matrix contains the position of the bounding box´s minimum and maximum vertices. The distribution of pixels within a frame begins in the upper left corner and ends in the lower right corner (Farhadi and Redmon, 2018). The bounding box matrix is distributed over n rows, representing the number of recognized objects, and four columns, containing the measurements for each BB located as follows:

$$Bounding\ Box = (Xcorner,\ Ycorner,\ Xwidth,\ Yheigth) \quad (1)$$

Where *Xcorner*, *Ycorner, Xwidth,* and *Yheigth* are the coordinates of four corners of the bounding box. In addition to these measurements, the centroid of the Bounding Box can be generated in "*x*" and "*y*" coordinates, called *Cx* and *Cy*.

The algorithm to generate the bounding box in the tracking process must predict the four coordinates and the centroids. The centroids are used for the bounding box prediction. The target´s centroid coordinates are tracked through the movement prediction that identifies each of the interest objects with the same label in all sequences (Murray, 2017).

During the tracking may be mismatches and detection errors; a filtering method can be applied to reduce them. A filtering method is used to predict the centroid of each bounding box in the video sequence and update the bounding box. The prediction and correction methods are used to mitigate the noise present in the object tracking process. The prediction indicates the position of the bounding box in time *t* based on its position at time *t-1*. The Update: it is a correction step. It includes the new object tracking model measurement and helps improve filtering (Deepak and Suresh, 2015).

3. PERFORMANCE OF OBJECT TRACKING ALGORITHM

One of the metrics to evaluate the performance of the tracking algorithms is precision, that is, the percentage of correct predictions. To calculate the precision values is necessary to calculate the Intersection over Union (*IoU*). The equations for calculating precision and *IoU* are (2) and (3), respectively. Variables used in calculating the precision are obtained from a comparison of *IoU* results with an established threshold. (Padilla, et al., 2021).

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

Where *IA* is the intersection area of the true bounding box (*TBB*) and the 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 *FN* is false-negative.

A threshold value must be set for *IoU* to determine if object detection is valid or not. *IoU* is generally set to 0.5. Assuming the *IoU* threshold is 0.5: If $IoU \geq 0.5$, it is classified as True Positive (*TP*). If $IoU < 0.5$, it is a false detection, and it is 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. Understanding that 1 would be an exact overlap of the BB.

4. STATE-SPACE MODEL OF A MOVING OBJECT

We consider a linear time-varying system with the moving object represented in discrete-time state-space with the following state and observation equations:

$$x_n = A_n x_{n-1} + E_n u_n + Bw_n \quad (4)$$

$$y_n = H_n x_n + v_n \quad (5)$$

Where $x_n \in \mathbb{R}^k$ is the state vector, $y_n \in \mathbb{R}^M$ is the observation vector. A is the state transition model. E is the control-input model, u_n is the input control. B is the noise matrix. H 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, respectively, and the property $E\{w_n v_k^T\} = 0$ for all n and k. The two noise sequences and the initial state are assumed uncorrelated and independent of each sampling instant. (Bar-Shalom, et al., 2001).

5. KALMAN FILTER

The Kalman filter uses the equation of state of the linear system to estimate the system´s state through observation of input and output. Using KF requires knowledge of the system parameters, initial values, and measurement sequences (Brown & Hwang, 2012). The KF can estimate the state sequences of the system iteratively. It consists of two phases: predict, where the optimal state $\hat{x}_n^-$ prior to observing y_n is computed, 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 $P_n = E\{\epsilon_n \epsilon_n^T\}$.

In the predict phase are produced (6) and (7). Since the process noise w_n is assumed white Gaussian with zero mean, the a priori state estimate, and the a priori error covariance matrix are defined as:

$$\hat{x}_n^- = A\hat{x}_{n-1} + E_n u_n \quad (6)$$

$$P_n^- = A_n P_{n-1} A_n^T + B_n Q_n B_n^T \quad (7)$$

In the update phase, the current a priori predictions are combined with the observation of the current state to redefine the state estimate and error covariance matrix. The improved estimate, which uses the current observation, is called a posteriori estimation of the state.

The measurement y_n is corrupted by the noise v_n. Since v_n is white with zero mean, this becomes (8), and the measurement residual by (9).

$$y_n = H_n \hat{x}_{n-1} \quad 8)$$

$$z_n = y_n - H_n \hat{x}^- \quad (9)$$

The residual covariance matrix is given as shown below.

$$S_n = H_n P_n^- H_n^T + R_n \quad (10)$$

The optimal Kalman gain is obtained by:

$$K_n = P_n^- H_n^T S_n^{-1} \quad (11)$$

The a posteriori state estimate is given as:

$$\hat{x}_n = \hat{x}_n^- + K_n(z_n - H\hat{x}_n^-) \quad (12)$$

The a posteriori error covariance matrix is calculated as:

$$P_k = (I - K_n H) P_k^- \quad (13)$$

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

Algorithm 1: Optimal 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, ... ***do***
    $\hat{x}_n^- = A\hat{x}_{n-1} + E_n u_n$
    $P_n^- = A_n P_{n-1} A_n^T + B_n Q_n B_n^T$
    $S_n = H_n P_n^- H_n^T + R_n$
    $K_n = P_n^- H_n^T S_n^{-1}$
    $\hat{x}_n = \hat{x}_n^- + K_n(y_k - H_n \hat{x}_n^-)$
    $P_k = (I - K_n H) P_k^-$
  end for
End

6 UFIR FILTER

In contrast to the KF, the UFIR does not require any information about noise, except for the zero-mean assumption (Shmaliy, et al., 2017; Pale-Ramon, et al., 2020). Therefore, the UFIR is more suitable for object tracking, where the measurement and process noises are unknown. However, the UFIR requires an optimal horizon length, N_{opt}, from $m = n - N_{opt} + 1$ 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:

$$\hat{x}_l^- = A\hat{x}_{l-1} + E_l u_l \quad (14)$$

In the update phase, the state estimate is combined with the observation of the current state to refine the state. The estimate is iteratively updated to the a posteriori state estimate using (15) -(19) as shown below:

Generalized noise power gain (GNPG) is computed as:

$$G_l = [H_l^T H_l + (A_L G_{l-1} A_l^T)^{-1}]^{-1} \quad (15)$$

The measurement residual is obtained as follow:

$$z_l = y_l - H_l \hat{x}^- \quad (16)$$

The bias correction gain is defined as:

$$Gain_l = y_l - H_l \hat{x}^- \quad (17)$$

The a posteriori state estimate is given by:

$$\hat{x}_l = \hat{x}_l^- + Gain_l(z_l - H\hat{x}_l^-) \quad (18)$$

A pseudo-code of the UFIR algorithm is listed as Algorithm 2. To initialize iterations, the algorithm requires a short measurement vector $y_{m.k} = [y_m \dots y_k]^T$, and matrix:

$$C_{m,k} = \begin{bmatrix} H_m(A_k^{m+1})^{-1} \\ H_{m+1}(A_k^{m+2})^{-1} \\ . \\ . \\ . \\ H_{k-1}A_k^{-1} \\ H_k \end{bmatrix}$$

Algorithm 2: UFIR filter

Data: y_n, u_n, N
Result: $\hat{x}_n$
Begin
  For *n= N-1, N, ...* ***do***
    $m = n - N + 1, s = n - N + K$
    $G_S = (C_{m,s}^T C_{m,s})$
    $\tilde{x}_s = G_s C_{m,s}^T (y_{m,s} - L_{m,s}U_{m,s}) + S_{m,s}^k U_{m,s}$
    For l=*s + k*: **do**
    $\tilde{x}_l^- = A\hat{x}_{l-1} + Eu$
    $G_l = [H_l^T H_l + (A_l G_{l-1} A_l^T)^{-1}]^{-1}$
    $Gain_l = G_l H_l^T$
    $\tilde{x}_l = \hat{x}_l^- + Gain_l(y_l - H_i\tilde{x}_l^-)$
    end for
  end for
  $\hat{x}_n = \tilde{x}_n$
End

Where $S_{m,s}$ and $L_{m,s}$ are obtained by (19) and (20) respectively, $S_{m,s}^k$ is the *Kth* row vector.

$$S_{m,s} = \begin{bmatrix} E_0 & 0 & \dots & 0 & 0 \\ A_{m+1}E_m & E_{m+1} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ A_{s-1}^{m+1}E_m & A_{s-1}^{m+2}E_{m+1} & \dots & E_{s-1} & 0 \\ A_s^{m+1}E_m & A_s^{m+2}E_{m+1} & \dots & A_sE_{s-1} & E_s \end{bmatrix} \quad (19)$$

$$L_{m,s} = diag(C_{m,s})S_{m.s} \quad (20)$$

7. VIDEO TRACKING OF MOVING OBJECT

The KF and UFIR algorithms were tested on video sequences. The videos are available in (Pale-Ramon, 2021).

7.1 Test on video with low illumination and change direction

The first test of UFIR and KF algorithms was in a video with low illumination and directions changes; the video is called "Remotecar1". The object also has color characteristics similar to the background. The purpose was to track a remote-control car throughout its trajectory. We supposed that an object is disturbed by white Gaussian acceleration noise with the standard deviation of $\sigma_w = 10m/s^2$. The data noise (CMN) originates from white Gaussian $\sigma_v = 2m$. With sample time T= 0.1 seconds, $P_0 = 0$, $Q = \sigma_w^2$, $R = \sigma_v^2$, and the matrices:

$$A = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; B = \begin{bmatrix} \frac{T^2}{2} & 0 & T & 0 \\ 0 & \frac{T^2}{2} & 0 & T \end{bmatrix}, H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Since the UFIR filter requires an optimal averaging horizon $[m, n]$ of N_{opt} points. Following, we determine the N_{opt} (Ramirez-Echeverria, et al., 2014):

$$N_{opt} = \sqrt{\frac{12\sigma_v}{T^2\sigma_w}} \cong 15$$

Next, we examine the results of both algorithms in the object tracking process using the bounding box measurements. In Fig. 2, the true trajectory of the object is shown with a black line; the results of KF and UFIR with a blue and red line, respectively. Given that the N_{opt} for the UFIR was 15, the estimates started from this. The trajectory of the moving object was from right to left. The KF presented differences compared to the true trajectory. At the same time, UFIR presented an adequate tracking of the trajectory. To provide a complete view, we calculated the root mean square error (RMSE). For the KF, the RMSE in the x-position was 3.6859 and 0.0068 in the y-position. While for UFIR the RMSE in the *x* and *y*-position was 0.0808.

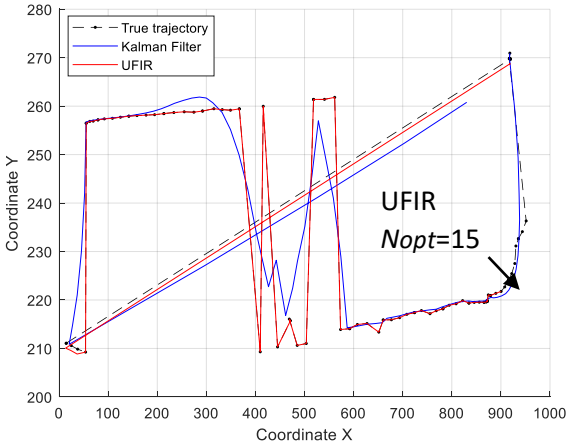

Fig. 2 Estimation of the Remotecar1 trajectory in the x-y plane using Kalman and UFIR filters)

7.2 Test on video with fast direction changes and motion blur

The algorithms UFIR and KF were tested in a video sequence called Remotecar3, in which the target makes rapid changes of direction. Also, during the trajectory, the target became blur due to its fast movement and camera movement. In this case, there were clearly differentiated the background and the target. We considered the standard deviation of $\sigma_w = 10m/s^2$. The data noise (CMN) originates from white Gaussian $\sigma_v = 3m$. The sample time is T= 0.03 seconds. The

matrices used were the same as in the previous test, and the $N_{opt} \cong 63$.

The results of this test are shown in Fig. 3. The object's trajectory is shown with a black line and the estimates made by the algorithms with blue and red for KF and UFIR, respectively. The object trajectory was from left to right. Given that the N_{opt} for the UFIR was 63, the estimates made with this algorithm started from it. The UFIR algorithm provided a more accurate estimate compared to the true trajectory of the target. While the KF presented slightly varying estimates from the true trajectory. The RMSE for the KF was the RMSE in the x-position was 0.1706, and 0.3432 in the y-position. While for UFIR the RMSE in the x-position and y-position was 0.

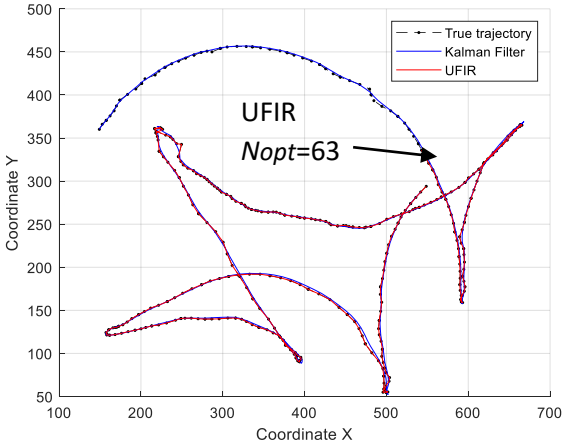

Fig. 3 Estimation of the Remotecar3 trajectory in the x-y plane using Kalman and UFIR filters

7.3 Performance evaluation of algorithms

Although the estimated trajectory between the true bounding box (TBB) and predicted bounding box (PBB) was similar. The dimensions of the BB can vary, and the area of intersection varies or does not intersect. To evaluate the algorithm performance in object tracking, we used the precision metric, correct predictions percentage. The precision was calculated using equations (2) and (3) with *IoU* thresholds from 0 to 1, as shown in the *x*-axis of Fig. 4 and Fig. 5.

The precision results of the test video RemoteCar1 are shown in Fig. 4. The KF algorithm produced a high precision on the thresholds from 0 to 0.5 and then decayed. The precision was greater than 0.7 until the threshold of 0.8. The UFIR algorithm presented a high precision until the threshold of 0.8. That is, each detection covers 80% of the TBB area. Since that, the most used threshold values are 0.5% and 0.75%. (Padilla, et al., 2021), we considered that UFIR and KF gave favorable results in the most widely used threshold *IoU* range. Also, the average precision was above 80%.

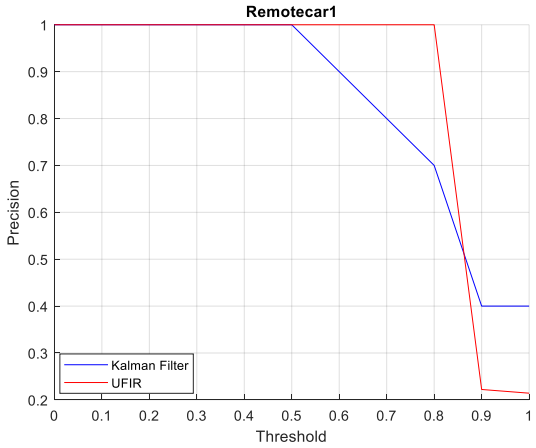

Fig. 4 Precision of Remotecar1 test video

The results for the test in video RemoteCar3 are shown in Fig. 5. The KF and UFIR produced a high precision threshold from 0 to 0.9 threshold and then decayed. Both algorithms presented a high precision until threshold 0.9. That is, each detection covers 90% of the TBB area. The overlap was not exact; otherwise, with a value of 1 threshold, the precision would have been equal to 1. Using 0.5 *IoU* threshold, we considered that both algorithms gave favorable results with an average precision above 80%.

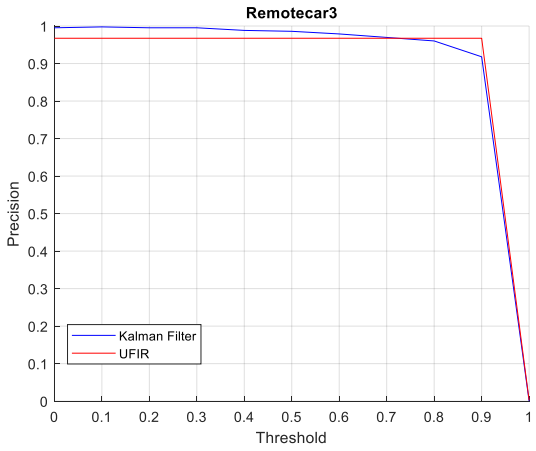

Fig. 5 Precision of Remotecar3 test video

The UFIR an KF algorithms performed efficiently in the tracking process. However, the UFIR filter showed better results. In addition, for UFIR, it is not required to know information about the initial state and noise. Therefore, it is more suitable for practical applications because not all information is usually available for object tracking models. Furthermore, the object tracking study is still a complex problem to tackle due to the diversity of application areas, where the impact factors vary from one application to another. Given the good performance of the UFIR filter, we consider that it can help treat current research problems such as vehicle navigation, security, robotics, etc.

8. CONCLUSIONS

The KF and UFIR estimation algorithms appear to be efficient in the video object tracking task. The object tracking tests showed a good performance of both algorithms. The UFIR presented a better performance and provided an estimate of the state with greater precision, which can be helpful in many visual tracking applications since it gives good results. It does not require knowing the initial state and noise information. Therefore, we conclude that visual object tracking algorithms can be further developed by incorporating state estimators for a wide area of applications.

Referring to the good performance of the UFIR filter, we now design practical modified algorithms for visual tracking of singular and multiple objects to mitigate CMN under the affectation factors occlusion and scale changes.

REFERENCES

Bar-Shalom, Y., Rong Li, X., and Kirubarajan, T. (2001). *Estimation with applications to tracking and navigation*. John Wiley and Sons, Inc., Hoboken, N.J.

Bishop, A. N., Savkin, A. V., and Pathirana, P. N. (2010). Vision-based target tracking and surveillance with robust set-valued state estimation, *IEEE Signal Process. Let.*, 17(3), pp. 289–292.

Brown, R. G., and Hwang, P. Y. (2012). *Introduction to random signals and applied Kalman filtering: with MATLAB exercises*, 4th ed. J Wiley and Sons, N.J.

Burger, W. and Burger, M. (2009). *Principles of digital image processing*. Springer, London.

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

Computer Vision Lab (2015). *Datasets-visual tracker benchmark*. Available at: http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html. (Accessed: 25 February 2021).

Deepak, P. and Suresh, S. (2015). *Design and Utilization of Bounding Box in Human Detection and Activity Identification*. Springer, Cham.

Farhadi, A. and Redmon, J. (2018). Yolov3: An incremental improvement, *Cornell University*. Available at: https://arxiv.org/abs/1804.02767. (Accessed: 16 March 2021).

Grewal, M. S., and Andrews, A. (2014). *Kalman filtering: Theory and Practice with MATLAB*. John Wiley and Sons, Hoboken, NJ.

Jahne, B. (2004). *Practical handbook on image processing for scientific and technical applications*. CRC press, FL.

Kang, T. K., Mo, Y. H., Pae, D. S., Ahn, C. K., and Lim, M. T. (2017). Robust visual tracking framework in the presence of blurring by arbitrating appearance- and feature-based detection. *Measurement*, 95, pp. 50-69.

Liang, P., Blasch, E., and Ling, H. (2015). Encoding color information for visual tracking: Algorithms and benchmark, *IEEE Trans. Image Process*, 24(12), pp. 5630–5644.

Murray, S. (2017). Real-time multiple object tracking-a study on the importance of speed. *Cornell University*. Available at: https://arxiv.org/abs/1709.03572. (Accessed: 16 March 2021).

Padilla, R., Passos, W., Dias, T. L. B., Netto, S. L., and da Silva, E. A. B. (2021). A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*, 10(279).

Pale-Ramon, E. G. (2021). *Database*. Available at: https://drive.google.com/drive/folders/1emoP9O7R0vYNGd343TSvkkUd2eitBS1h?usp=sharing. (Accessed: 08 July 2021).

Pale-Ramon, E. G., Contreras, J., Lucio, J., Shmaliy, Y.S., Zhao, S. (2020). Recursive Algorithms for Visual Target Tracking Under Colored Measurement Noise. In *2020 24th Int. Conf. on Circuits, Syst., Commun. and Computers (CSCC)*, Crete Island, Greece, pp. 5-10.

Parekh, H., Thakore, D., and Jaliya, U. (2014). A Survey on Object Detection and Tracking Methods. I*nt. J. of Innovative Research in Comp. and Commun. Eng.*, 2(2), pp. 2970-2978.

Parmar, M. (2016). A survey of video object tracking methods. *GIT-J. of Eng. and Technol.*, 9, pp. 519-524.

Ramirez-Echeverria, F., Sarr, A. and Shmaliy, Y. S. (2014). Optimal memory for discrete-time fir filters in state-space. *IEEE Trans. on Signal Process.*, 62(3), p. 557–561.

Shmaliy, Y. S. (2010). An iterative Kalman-like algorithm ignoring noise and initial conditions. *IEEE Trans. on Signal Process.*, 59(6), pp. 2465–2473.

Shmaliy, Y. S., Zhao, S., and Ahn, C. K. (2020). Kalman and UFIR state estimation with colored measurement noise using backward Euler method. *IET Signal Process.*, 14(2), pp. 64-71.

Shmaliy, Y. S., Zhao, S., and C. K. Ahn (2017). Unbiased finite impulse response filtering: An iterative alternative to Kalman filtering ignoring noise and initial conditions. *IEEE Control Syst. Mag.*, 37(5), pp. 70-89.

Simon, D. (2006). *Optimal state estimation: Kalman, H_∞, and nonlinear approaches*. John Wiley and Sons, Hoboken, N.J.

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

Yilmaz, A., Javed, O. and Shah, M. (2006). Object tracking: A Survey. *ACM Computing Surveys*, 38(4), pp. 1- 45.

Yoon, Y., Kosaka, A., and Kak, A. (2008). A new Kalman-filter-based framework for fast and accurate visual tracking of rigid objects. *IEEE Transactions on Robotics*, 24(5), pp. 1238-1251.

Zhao, S., Shmaliy, Y.S., and Ahn, C. (2018). Bias-constrained optimal fusion filtering for decentralized WSN with correlated noise sources. *IEEE Trans. on Signal and Inform. Process. over Networks*, 4(4), pp. 727-735.

