Tiny Drone Tracking Framework Using Multiple Trackers and Kalman-based Predictor

Sohee Son[1], Jeongin Kwon[1], Hui-Yong Kim[2] and Haechul Choi[1,*]

[1]*Hanbat National University, South Korea*
[2]*Kyung Hee University, South Korea*
E-mail:choihc@hanbat.ac.kr
[*]*Corresponding Author*

Received 17 September 2020; Accepted 27 July 2021;
Publication 06 November 2021

Abstract

Unmanned aerial vehicles like drones are one of the key development technologies with many beneficial applications. As they have made great progress, security and privacy issues are also growing. Drone tacking with a moving camera is one of the important methods to solve these issues. There are various challenges of drone tracking. First, drones move quickly and are usually tiny. Second, images captured by a moving camera have illumination changes. Moreover, the tracking should be performed in real-time for surveillance applications. For fast and accurate drone tracking, this paper proposes a tracking framework utilizing two trackers, a predictor, and a refinement process. One tracker finds a moving target based on motion flow and the other tracker locates the region of interest (ROI) employing histogram features. The predictor estimates the trajectory of the target by using a Kalman filter. The predictor contributes to keeping track of the target even if the trackers fail. Lastly, the refinement process decides the location of the target taking advantage of ROIs from the trackers and the predictor. In experiments on our dataset containing tiny flying drones, the

Journal of Web Engineering, Vol. 20_8, 2391–2412.
doi: 10.13052/jwe1540-9589.2088

proposed method achieved an average success rate of 1.134 times higher than conventional tracking methods and it performed at an average run-time of 21.08 frames per second.

Keywords: Object tracking, unmanned aerial vehicles, drones, surveillance system.

1 Introduction

Due to the rapid development in the field of unmanned aerial vehicles (UAVs) and the technologies used to construct them, the number of UAVs manufactured for military or commercial purposes has increased sharply [22, 2]. Drones are used for commercial applications in environmental and natural disaster monitoring, border surveillance, delivery of goods, and construction. As a result of the increased commercialization, drones have made headlines in the last few years due to various incidents. These events have created widespread fear of the damage that could be caused by such a small unmanned system when used in malicious ways [27]. Thus, a high demand exists for the development of a surveillance system for drones [12].

The recent interest in surveillance is increasing the need to create and deploy intelligent or automated visual surveillance systems. Recently, the pan-tilt-zoom (PTZ) camera has been extensively used [1]. A fixed camera cannot completely cover the entire space in a single view, but the PTZ camera can be controlled to view changing areas of interest using the pan, tilt, and zoom controls. However, tracking is difficult with a moving camera due to background changes and lighting changes, and it must be implemented in real time [8].

Object tracking is one of the main components in computer vision, such as augmented reality, video surveillance, and human-computer interaction [13, 21]. In object tracking, selecting features for object tracking plays a critical role, and the object must be represented by shape or appearance. Regarding the overall research, according to [30], three approaches to object tracking have primarily been studied: point-based, kernel-based, and silhouette-based tracking methods. The point-based tracking methods represent and track moving objects as feature points in an image. Kernel-based tracking methods generally represent the motion of an object in the form of parametric motion, such as transform, affine, and so on, and track the object by calculating and performing the respective elements for motions that can be expressed [23]. A representative method is a mean-shift tracking method

using a color histogram [9]. Silhouette-based tracking methods represent an object as an outline or silhouette, and these are predominantly used to track complex objects, such as people [3].

Tracking tiny drones with a moving camera is challenging. First, drones fly extremely fast, and these are usually tiny in the image. Second, camera movement causes illumination changes and global motion. Moreover, a tracking system should satisfy a real-time environment of surveillance applications. Considering these conditions, the color feature information, which is frequently used for object tracking, is unsuitable. In addition, edge or silhouette features are also unsuitable because the tiny drones do not have a characteristic edge or silhouette information. To address these problems, this paper proposes a tracking framework comprising two trackers, a predictor, and a refinement process. Trackers use motion flow and histogram features to locate the region of interest (ROI), respectively. Kalman filter is also used to estimate the trajectory of a moving target as the predictor. The predictor predicts the position and size of a target with the trajectory obtained by the Kalman filter, which helps track the target continuously even if the trackers fail. To locate the target from the two ROIs, the ROIs are compared using the trajectory obtained, and the target is localized in the refinement process.

The remainder of this paper is organized as follows. Section 2 outlines the description of the tracking methods using optical flow and the Kalman filter. In Section 3, the proposed tracking framework is introduced. Section 4 reveals the experimental results obtained for the implementation methods. Finally, Section 5 concludes this paper.

2 Methodology

2.1 Optical Flow in Object Tracking

Optical flow is the apparent movement of an image brightness pattern, which is used to recover the image movement of each pixel with the image brightness change between two consecutive frames. That is, the optical flow represents a distance vector in which the point of interest moves between the previous and current frames. Representative methods to obtain the optical flow are the Gunnar Farneback [11], Horn Schunck [15], and Lucas–Kanade [20] methods. The Lucas–Kanade method is based on three assumptions: brightness constancy, temporal persistence, and spatial coherence. Brightness constancy indicates that the brightness values of the pixels within a tracked window are constant. The temporal persistence indicates that

the amount of movement of an object within a frame is not great, even if the movement of the object is very fast. The spatial coherence refers to the precondition that spatially adjacent points are likely to belong to the same object and have the same movement. The Lucas–Kanade method operates by finding the best-matching window in the next frame. In other words, for the problem of finding a point v in an image J that is very similar to point u in image I, a vector must be determined that minimizes Equation (1) [16]:

$$\epsilon\left(d_x, d_y\right) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I(x,y) - J(x+d_x, y+d_y)\right)^2. \quad (1)$$

In the Lucas–Kanade method, the size of the window is critical. A large window causes a problem that reduces performance due to the smoothing effect, whereas a small window often causes the tracking to fail when the motion of an object is larger than the window. An iterative Lucas–Kanade method is proposed to resolve this problem [7]. The iterative Lucas–Kanade method generates an image pyramid according to the image scale from the current frame and repeatedly applies the Lucas–Kanade method from the smallest pyramid image to the largest pyramid image. In other words, the optical flow is estimated at the smallest pyramid image, as shown in Figure 1, and it passes to the lower level. The redefined optical flow is recalculated in the Gaussian pyramid image, and it also passes to the lower levels. Therefore, it is possible to calculate large movements while keeping the window small.

The iterative Lucas–Kanade method is frequently used in an object-tracking system because it is easy to track an object that has appearance changes and fast movements, and it can be tracked in real time. Moreover, Kalal proposed a forward-backward error using the iterative Lucas–Kanade method, which is used in a tracking system called MedianFlow [17]. The forward-backward error is defined as the distance between the trajectory from the t-1 frame to the t frame and the trajectory from the t frame to the t-1 frame. The Euclidean distance algorithm is used to obtain the distance between trajectories. In MedianFlow, a set of points is initialized within the bounding box, and these points are then tracked using the iterative Lucas–Kanade method. The quality of the point predictions is estimated by the forward-backward error. It removes 50% of the points that have a large forward-backward error and finds the median over each spatial dimension to estimate the bounding box in the current frame.

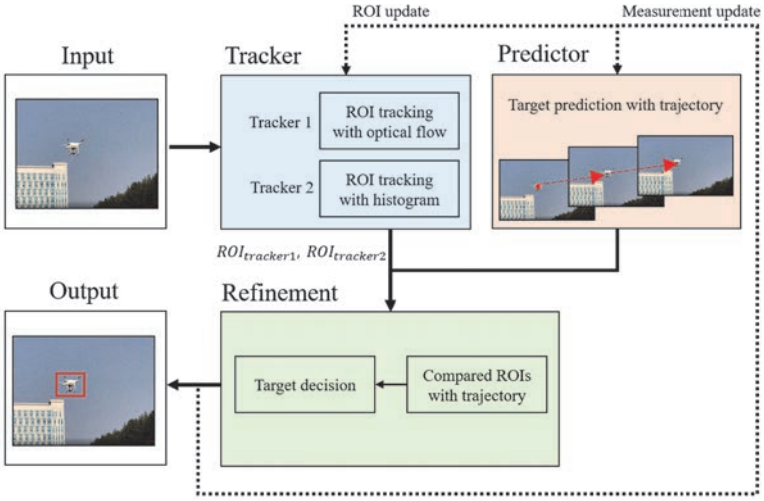

Figure 1 Proposed tracking framework.

2.2 Kalman Filter in Object Tracking

The Kalman filter is an algorithm used to estimate the state of a linear system where the state is assumed to have a Gaussian distribution. The Kalman filter is often used in the field of image processing because of the low computational time cost, and it leads to applications in real time [18]. To detect and track vehicle objects, [5] suggests using Gaussian mixture model (GMM) and Kalman filter. The GMM is used for object detection and the Kalman filter is used to track after the vehicle object is detected. In this work, the Kalman filter provides a new position of the object in the current frame using the information of the detected object in the previous frame. In order to improve the precision of object tracking, [26] proposes a hybrid method using Camshift method and the Kalman filter. The Camshift is applied as the main tracking technique, and the Kalman filter is for prediction and correction. The first step is tracking the object with the Camshift. The centroid of ROI from the result of Camshift uses as an input of the Kalman filter to predict and correct the object location in the next frame. The predicted point from the Kalman filter would be processed by the Camshift method in the next step. In this work, the Kalman filter is proved to make object tracking results become more precise. Although the Kalman filter is often used for object tracking, it has one limitation that is the assumption the state variables are normally distributed [30]. This limitation can be solved

by using the particle filtering [31]. However, the particle filtering requires a considerable computational complexity, which may not be suitable for real-time application.

The Kalman filter estimates a process using a form of feedback control. The filter estimates the process state and then obtains feedback in the form of measurements. As such, Kalman filters fall into two groups: state-update equations and measurement-update equations [19]. State-update equations are responsible for projecting the current state and error covariance estimates to obtain the a priori estimate for the next state-update step. The state update projects the current state estimate ahead in time. The time update equations are shown in Equations (2) and (3), where $\hat{x}_{\bar{k}}$ is a vector representing the predicted process state at time k. In Equation (3), $P_{\bar{k}}$ is the predicted error covariance at time k, and P_{k-1} is a matrix representing the error covariance in the state prediction at time $k-1$. After the state-update process, a measurement-update process is conducted to correct the current object parameter value using the current predicted object parameter value and the current object information, as shown in Equations (4), (5), and (6). In Equation (4), K_k is the Kalman gain using the predicted and measured noise. In Equation (5), the process status $\hat{x}_{\bar{k}}$ is updated using the Kalman gain K_k and the measurement z_k. The last step in Equation (6) is to update the error covariance. After each state-update and measurement-update equation, the process is repeated with the previous posteriori estimates used to project or predict the new a priori estimates [29].

$$\hat{x}_{\bar{k}} = A\hat{x}_{k-1} + Bu_k \tag{2}$$

$$P_{\bar{k}} = AP_{k-1}A^T + Q \tag{3}$$

$$K_k = P_{\bar{k}}H^T \left(HP_{\bar{k}}H^T + R\right)^{-1} \tag{4}$$

$$\hat{x}_{\bar{k}} = \hat{x}_{\bar{k}} + K_k \left(z_k - H\hat{x}_{\bar{k}}\right) \tag{5}$$

$$P_k = \left(1 - K_k H_k\right) P_{\bar{k}} \tag{6}$$

3 Proposed Tracking Framework

As mentioned, tracking tiny drones with a moving camera is challenging. To address this, the tracking framework shown in Figure 1 is proposed. The framework consists of two trackers, a predictor, and a refinement process. The predictor predicts the position and size of the target with the trajectory. Then,

the target is tracked using two tracking methods. To locate the target from the two ROIs, the ROIs are compared using the trajectory obtained, and the target is localized in the refinement process. Lastly, the predictor and trackers are updated for the next frame.

3.1 Optical Flow-based Tracker

The iterative Lucas–Kanade method is often used because it is easy to track an object that has appearance changes and fast movements, and it can also be used in real time. Considering these benefits, this paper uses the iterative Lucas–Kanade method for tracking.

The process of the proposed optical flow-based tracker consists of feature-point-motion estimation, target-motion estimation, bounding-box estimation, and refinement feature points. For tracking, the feature points are extracted in the first frame. The corner-point-detection method proposed by Shi-Tomasi [24] is used for feature extraction. These extracted features are registered as feature points. In the target-motion estimation, each motion flow of the feature point is estimated through the iterative Lucas–Kanade method. To estimate the quality of the predicted feature points, various methods, such as the sum of the squared difference algorithm or a normalized cross-correlation algorithm can be used. In this paper, the feature points are estimated based on the assumption of the spatial coherence of the Lucas–Kanade method. As such, the displacement of each motion flow is calculated, and the proposed tracking method estimates a representative motion flow of the target by averaging the displacements of the motion flow. In the refinement feature-point process, unreliable feature points that have a large difference of displacement relative to the average displacement of all feature points are filtered out from the feature-point set because the outlier points are likely to lead tracking failure. The remaining feature points that have a dominant motion flow are then used to estimate the bounding box of the target. The remaining feature points are updated for the next frame. To maintain sufficient feature points, new corner points are extracted periodically and added to the feature-point set.

3.2 Histogram-based Tracker

Tracking through the optical flow is robust for tiny drone tracking. However, when camera movement happens rapidly, tracking failure often occurs. To address the problem, we supplement the tracking framework by adding the histogram-based tracking method. For tracking a target, the model is set at

the first frame on an ROI. The histogram on an ROI is extracted, and feature points are detected using the Shi-Tomasi method. The histogram of the model is set as the average and standard derivation of the luminance values of feature points. Afterward, tracking is performed in every frame. First, feature-point extraction occurs in the ROI. Then, each histogram of the window in which the feature point is centered is extracted. These histograms are compared using the model histogram. The feature point that has a histogram similar to the model histogram is determined to be a positive point, and it is used to estimate the bounding box. For adaptivity, the proposed tracker conducts the model adjustment. The model is periodically updated by fine-tuning the average and standard derivation of the luminance values of the positive feature points.

3.3 Kalman Filter-based Predictor

The Kalman filter is often used for object tracking because it is easily able to take tracking algorithm outputs as measurements. Tracking often fails when the background and lighting change because of camera movement. To cope with this tracking failure, the Kalman filter is used as the predictor in Figure 1. As such, the predictor detects tracking failures and conducts retracking from the tracking failure. First, the Kalman filter is initialized on the ROI in the first frame. Afterward, the target is predicted in every frame. The bounding box is predicted using the output of the predicted state from the Kalman filter in Equation (7). Where, x, y, $width$, $height$ represent centroid coordinates and sizes of the bounding box, v represents their speed respectively. That is, the Kalman filter predicts the position and size of the target with the trajectory. For the target prediction, the measurement is updated in every frame with four parameters, which are the centroid coordinated and sizes of the bounding box as shown in Equation (8). The predicted bounding box is used in two ways for the refinement process. One method is when all trackers fail, the tracking framework tries to track by predicting the bounding box from the predictor. This causes the tracking framework to track the target continuously even if the trackers fail. Another method is to measure the confidence of the trackers. Hence, it helps the tracking framework to determine the output:

$$\hat{x}_{\bar{k}} = [x, y, v_x, v_y, width, height]^T. \tag{7}$$

$$z_k = [x, y, width, height]^T \tag{8}$$

As shown in Equations (9) to (12), the initialized element is the matrices used to update the values of each vector in performing the Kalman

filter algorithm.

$$A = \begin{bmatrix} 1 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$Q = e^{-2} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$R = e^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

3.4 Refinement Process

In the refinement process, two critical roles are conducted. First, the output is used to update the predictor and trackers for the next frame. The second role is the target decision. This paper proposes a tracking framework that uses two tracking methods. The target decision is determined by comparing it with the known trajectory obtained from the predictor. To measure the error of the outputs from the trackers, the Euclidean distance between the predicted ROI from the predictor (B_P) and the estimated ROI from the tracker (B_T) is used:

$$error(B_P, B_T) = \sqrt{((B_P)_x - (B_T)_x)^2 + ((B_P)_y - (B_T)_y)^2} \tag{13}$$

$$threshold = \sqrt{((B_P)_{width})^2 + ((B_P)_{height})^2} \times \alpha. \tag{14}$$

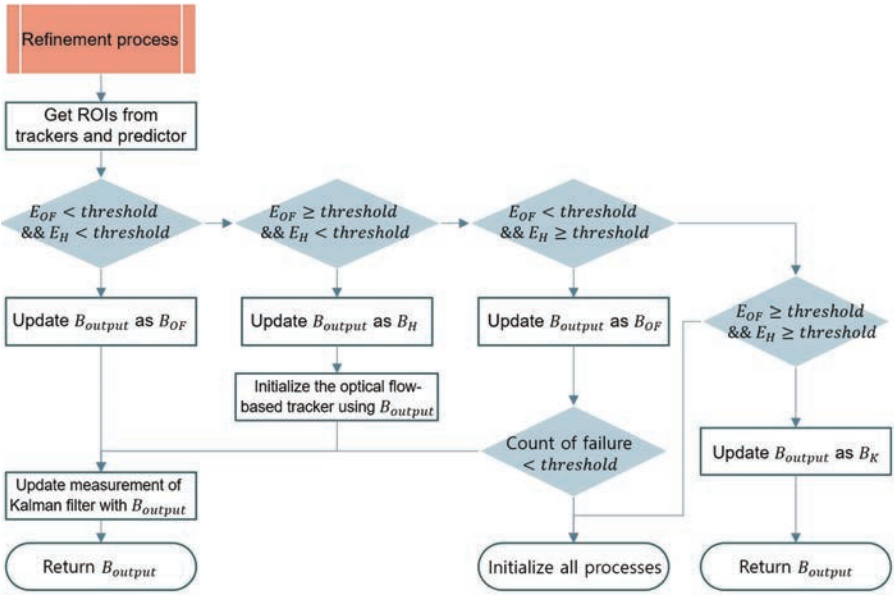

Figure 2 Flowchart of the refinement process.

The flowchart of the refinement process is shown in Figure 2, the output is determined as follows:

- Let B_P, B_{OF}, and B_H denote the output bounding boxes of the predictor, the tracker with optical flow, and the tracker with a histogram, respectively, where $B = [x, y, width, height]$.
- Let E_{OF} and E_H denote the errors compared by B_P with the Equation (13).
- If $E_{OF} <$ threshold and $E_H <$ threshold, the output is set as B_{OF}.
- If $E_{OF} \geq$ threshold and $E_H <$ threshold, B_H becomes the output. The optical flow-based tracker is initialized on B_H.
- If $E_{OF} <$ threshold and $E_H \geq$ threshold, the B_{OF} becomes the output. When this decision occurs continuously, the tracking framework is considered to exhibit a tracking failure by mistracking, and all processes in the tracking framework are initialized.
- If $E_{OF} \geq$ threshold and $E_H \geq$ threshold, the output is set as B_P. Likewise, when this decision occurs continuously, the tracking framework is considered a tracking failure due to mistracking, and all processes are initialized.

Table 1 Test sequences. (Camera Moving (CM) – camera is moving. Global Motion (GM) – the global motion is occured. Background clutters (BC) – the background has the similar color as the target or the background is changed. Occlusion (OC) – the target is occluded. Low Resolution (LR) – the number of pixels inside the ground-truth is less than 400 pixels. Out of View (OV) – the target leaves from the view.)

Resolution	Sequences	Target minimum sizes (pixels)	Target maximum sizes (pixels)	Num. of frames	Attributes
QHD	Inspire1	56	182	1369	CM, BC, OC , LR, GM
	Inspire2	45	2080	3206	CM, BC, LR , GM
	Inspire3	1782	14994	2691	CM, OV, GM
	Inspire4	408	6035	7107	CM, GM
	Inspire5	1334	24738	7061	CM, GM
	Inspire6	99	918	5441	CM, OV, LR , GM
FHD	Phantom1	12	20	586	BC, LR

4 Experimental Results and Discussion

4.1 Dataset and Evaluation Metrics

For evaluation, we built a dataset of natural scene images. The dataset consists of 7 sequences. These sequences are captured by a moving camera, and they involve a flying drone. The dataset includes real-world challenges, such as fast motion, camera movement, scale changes, occlusion, background clutters, and low resolution. Table 1 shows the details of the dataset, which are resolution, target sizes, number of frames, and attributes.

For quantitative evaluation, three criteria are employed, including the center location error, correctly tracked frames, and overlap success rate. The center location error is defined as the average Euclidean distance between the center location of the ground truth and the center location of the tracked bounding box. The correctly tracked frames are computed by the bounding-box overlap with the ground truth. To compute the success rate, the criterion used in the PASCAL VOC challenge [10] is employed. Given the ground truth bounding box r_g and the tracked bounding box r_t, the bounding-box overlap score is defined as $S = \frac{|r_t \bigcap r_g|}{|r_t \bigcup r_g|}$, where $\bigcap$ and $\bigcup$ represent the intersection and union of the two regions, respectively, and $|\cdot|$ denotes the number of pixels in the region [25]. The tracking result in one frame is considered a success when

the score is above t_o (e.g., $t_o = 0.1$). The success rate is computed with all frames of one sequence. As the threshold varies between 0 and 1, the success rate changes, and the resultant curve is presented.

In this paper, the performance of the proposed method is evaluated against the existing methods, such as MedianFlow, MIL, Boosting, KCF, and STAPLE. The experiments are tested on Opencv (v. 3.3). The most common evaluation method is to initialize an algorithm with the ground truth object state in the first frame and report all results. All experiments are measured using this straightforward approach.

4.2 Evaluation

In Figures 3 and 4, the performance of the proposed framework is evaluated against two tracking algorithms, which are conducted through the optical flow. In Figure 3, the center location error is compared frame by frame on the Inspire1 sequence. The optical flow-based tracking (OF) method is the optical flow-based tracker, as mentioned in Section 3.1, and it is compared with the optical flow-based tracking method combined with the Kalman filter (OFKalman). In the figure, the OF and OFKalman methods are track with

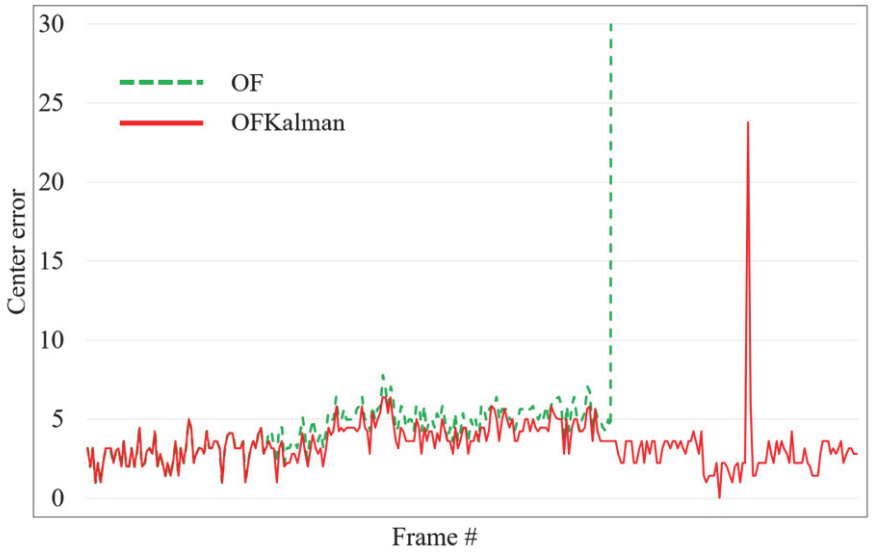

Figure 3 Frame-by-frame comparison of the center location errors (pixels) on the Inspire1 sequence (OF: optical flow-based tracking method, OFKalman: optical flow-based tracking method combined with the Kalman filter).

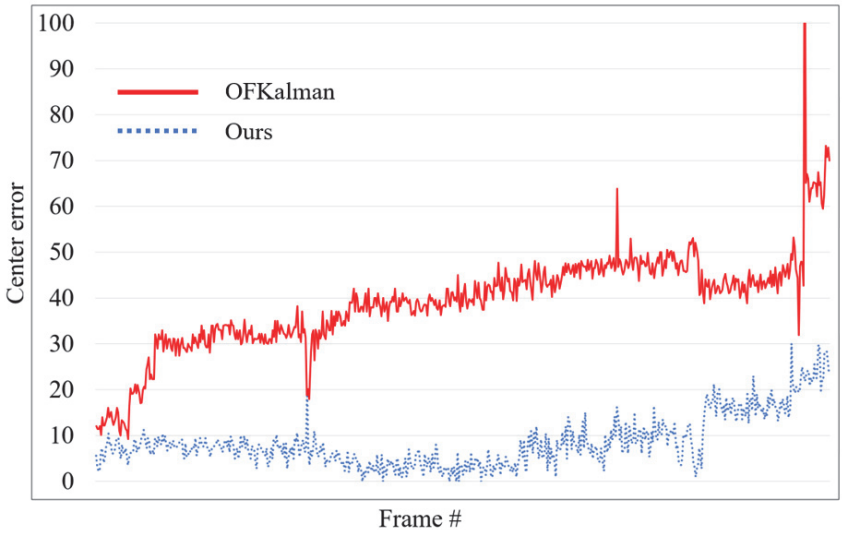

Figure 4 Frame-by-frame comparison of the center location errors (pixels) on the Inspire4 sequence (OFKalman: optical flow-based tracking method combined with Kalman filter, Ours: proposed tracking method in Figure 1).

similar center errors. However, the OF method fails by mistracking becuase of illumination changes, but the OFKalman method continues tracking.

In Figure 4, the proposed framework and the OFKalman are compared. The Inspire4 sequence includes fast camera movement. The proposed framework achieves much better overall results. It indicates that the proposed method manages the changes from fast camera movement.

4.3 Main Comparison Results

In this section, the proposed framework, as shown in Figure 1, is evaluated against existing tracking methods on the dataset. We implemented MedianFlow [17], MIL [4], STAPLE [6], KCF [14], and Boosting [12] trackers.

As shown in Tables 2 and 3, in terms of the average center location errors and the number of succeeding frames. The Table 3 indicates the top three trackers in each sequence. The KCF tracker exhibits the lowest average center errors on all sequences, but it has the least succeeding frames as shown in Table 3. Tables 2 and 3 show the proposed framework tracks better than the

Table 2 Average center errors (pixels) (Ours: proposed tracking method in Figure 1).

Sequences	Ours	MedianFlow	MIL	STAPLE	KCF	Boosting
Inspire1	95.1	176.5	114.4	286	0.3	841.4
Inspire2	89.8	231.3	483.8	376.8	2.9	665.8
Inspire3	10.7	10.3	23.1	6.2	3.1	686.8
Inspire4	6.3	22.9	14.7	18.9	1.9	9.0
Inspire5	4.0	8.6	5.4	6.8	1.4	1065.1
Inspire6	3.5	359.6	2.6	1.9	0.8	1.6
Phantom1	160.7	12.8	292.0	2.7	2.8	2.0
Average	52.9	160.3	133.7	99.9	1.9	467.4

Table 3 The number of succeeded frames. The top three trackers in each sequence are denoted by different colors: red, green, and blue (Ours: proposed tracking method in Figure 1).

Sequences	Ours	MedianFlow	MIL	STAPLE	KCF	Boosting
Inspire1	1046	13	1045	56	2	44
Inspire2	2426	602	198	422	7	150
Inspire3	2690	2691	2377	2692	336	336
Inspire4	7107	6548	6772	2695	785	7107
Inspire5	7061	450	7027	7061	98	113
Inspire6	5435	501	5409	5441	17	5440
Phantom1	134	423	115	586	121	586

other trackers. Although the STAPLE tracker is also a top-three tracker, it has few succeeding frames on Inspire1.

In Figure 5, the proposed framework is compared using the overlap success rate, and the results are reported in the one-pass evaluation (OPE) [28]. The average success rate is also shown. The figure reveals that the proposed framework achieves the highest average success rate of 0.406. The MIL tracker has a high average success rate following the proposed framework, but the tracker does not work well for the Inspire2 sequence. When the overlap threshold is over 0.5, the MIL, STAPLE, and Boosting trackers are slightly better than the proposed method. These experimental results indicate that the proposed method is mostly robust over all range of the threshold for the tiny drone tracking from images captured by a moving camera.

Figures 6 illustrates the continuous tracking results for the Inspire2 sequence. The proposed framework continuously tracks the target, whereas other trackers fail when the camera moves fast. Figure 7 displays examples of the tracking results using the proposed framework.

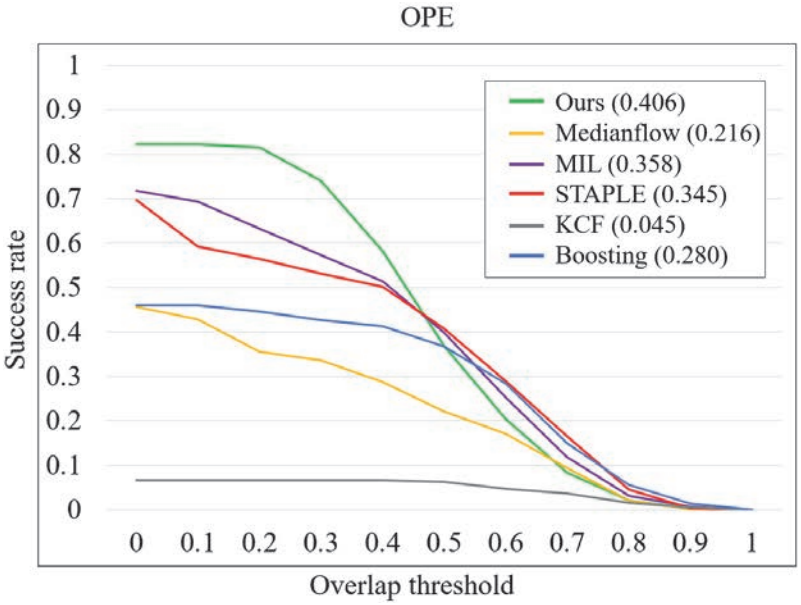

Figure 5 Success plots of OPE on our dataset.

Figure 6 Continuous tracking results on the Inspire2 sequence (green: Ours, yellow: MedianFlow, purple: MIL, red: STAPLE, blue: Boosting).

Figure 7 Examples of tracking results using the proposed framework.

In this work, the proposed framework was implemented on an Intel i7-6700 3.40GHz CPU with 16 GB RAM. To evaluate computational complexity of the proposed framework, a run-time performance in terms of frame-per-second (FPS) is measured. The proposed framework runs at an average of 21.10 FPS on the test sequences in the Table 1. Note that it is implemented using a CPU and not a GPU.

5 Conclusion

In this paper, a tracking framework utilizing two trackers, a predictor, and a refinement process is proposed for fast and accurate drone tracking. Combining the outputs of two trackers, which are the optical flow-based tracker and histogram-based tracker, and a Kalman filter predictor reveals a significant increase in robustness for tracking tiny drones on a moving camera. Experimental results on our dataset containing tiny flying drones show that the proposed method mostly robust against the existing trackers, but an improvement of overall accuracy in terms of success rate is required.

Acknowledgments

This research was supported by the research fund of Hanbat National University in 2019.

References

[1] Luma Issa Abdul-Kreem and Hussam K. Adjul-Ameer. Object tracking using motion flow projection for pan-tilt configuration. volume 10, page 4687. Insitute of Advanced Engineeering and Science, 2020.

[2] Cemal Aker and Sinan Kalkan. Using deep networks for drone detection. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[3] J Athanesious and P Suresh. Implementation and comparison of kernel and silhouette based object tracking. *International Journal of Advanced Research in Computer Engineering & Technology*, pages 1298–1303, 2013.

[4] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *2009 IEEE conference on computer vision and pattern recognition*, pages 983–990. IEEE, 2009.

[5] Rizki Yusliana Bakti, Intan Sari Areni, A Ais Prayogi, et al. Vehicle detection and tracking using gaussian mixture model and kalman filter. In *2016 International Conference on Computational Intelligence and Cybernetics*, pages 115–119. IEEE, 2016.

[6] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr. Staple: Complementary learners for real-time tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1401–1409, 2016.

[7] Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1–10):4, 2001.

[8] PJ Burt, JR Bergen, Rajesh Hingorani, R Kolczynski, WA Lee, A Leung, J Lubin, and H Shvayster. Object tracking with a moving camera. In *[1989] Proceedings. Workshop on Visual Motion*, pages 2–12. IEEE, 1989.

[9] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pages 142–149. IEEE, 2000.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[11] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[12] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *Bmvc*, volume 1, page 6, 2006.

[13] Kun Zheng Guangmin Sun, Junjie Zhang and Xiaohui Fu. Eye tracking and roi detection within a computer screen using a monocular camera. *Journal of Web Engineering*, 19(7–8):1117–1146, 2020.

[14] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014.

[15] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981.

[16] Björn Johansson. Derivation of lucas-kanade tracker. *In other words*, 500:7, 2007.

[17] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *2010 20th International Conference on Pattern Recognition*, pages 2756–2759. IEEE, 2010.

[18] Man-Hyung Lee and Jong-Hwa Kim. Mathematical modelling of moving target and development of real time tracking method using kalman filter. *The Korean Institute of Electrical Engineers*, pages 765–769, 1987.

[19] Xin Li, Kejun Wang, Wei Wang, and Yang Li. A multiple object tracking method using kalman filter. In *The 2010 IEEE international conference on information and automation*, pages 1862–1866. IEEE, 2010.

[20] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[21] Deepika Punj Manisha Mudgal and Anuradha Pillai. Suspicious action detection in intelligent surveillance system using action attribute modelling. *Journal of Web Engineering*, 20(1):129–146, 2021.

[22] Roberto Opromolla, Giancarmine Fasano, and Domenico Accardo. A vision-based approach to uav detection and tracking in cooperative applications. *Sensors*, 18(10):3391, 2018.

[23] Divyani Prajapati and Hiren J Galiyawala. A review on moving object detection and tracking. *International Journal of Computer Application*, 5(3):168–175, 2015.

[24] Jianbo Shi et al. Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE, 1994.

[25] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1442–1468, 2013.

[26] Galandaru Swalaganata, Yessi Affriyenni, et al. Moving object tracking using hybrid method. In *2018 International Conference on Information and Communications Technology (ICOIACT)*, pages 607–611. IEEE, 2018.

[27] Amy R Wagoner, Daniel K Schrader, and Eric T Matson. Survey on detection and tracking of uavs using computer vision. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 320–325. IEEE, 2017.

[28] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

[29] Sheldon Xu and Anthony Chang. Robust object tracking using kalman filters with dynamic covariance. *Cornell University*, pages 1–5, 2014.

[30] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13–es, 2006.

[31] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Multi-task correlation particle filter for robust object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4335–4343, 2017.

Biographies

Sohee Son received her B.S. and M.S. degrees from Hanbat National University, Daejeon, Korea, in 2015 and 2017, respectively, from the department of multimedia engineering. Currently, she is working toward Ph.D. degree in the department of multimedia engineering in Hanbat National University. Her research interests include video coding, parallel processing, and computer vision.

Jeongin Kwon received a B.S degree in industrial management engineering and a M.S degree in multimedia engineering from Hanbat National University, Daejeon, Korea, in 2014 and 2020, respectively. Currently, she is working on a personal project on deep learning. Her research interests include computer vision, image processing and deep learning.

Hui-Yong Kim received his B.S., M.S., and Ph.D. degrees from Korea Advanced Institute of Science and Technology (KAIST) in 1994, 1998, and 2004, respectively. He is currently an Associate Professor in the Dept. of Computer Engineering in Kyung Hee University, Yongin, Korea since Mar. 2020. He was also an Associate Professor in the Dept. of Electronics Engineering in Sookmyung Women's University, Seoul, Korea from Sep. 2019 to Feb. 2020. From 2005 to 2019, prior to joining universities, we worked for Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea as the Managing Director of Realistic Audio & Video Research Group. From 2003 to 2005, he worked for AddPac Technology Co. Ltd., Seoul as the leader of Multimedia Research Team. He was also an affiliate professor in University of Science and Technology (UST) and was a visiting scholar in the Media Communications Lab. at University of Southern California (USC), USA. His current research interest includes video signal processing and compression for realistic media applications such as UHD, 3D, VR, HDR, and Hologram.

Haechul Choi received his B.S. in electronics engineering from Kyungpook National University, Daegu, Korea, in 1997, and his M.S. and Ph.D. in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1999 and 2004, respectively. He is a professor in Information and Communication Engineering at Hanbat National University, Daejeon, Korea. From 2004 to 2010, he was a Senior Member of the Research Staff in the Broadcasting Media Research Group of the Electronics and Telecommunications Research Institute (ETRI). His current research interests include image processing, video coding, and video transmission.