electronics

Article

UAV Object Tracking Application Based on Patch Color Group Feature on Embedded System

Ming-Hwa Sheu [1], Yu-Syuan Jhang [1], S M Salahuddin Morsalin [1], Yao-Fong Huang [1], Chi-Chia Sun [2] and Shin-Chi Lai [3,4,*]

1 Department of Electronic Engineering, National Yunlin University of Science & Technology, Douliu 64002, Taiwan; sheumh@yuntech.edu.tw (M.-H.S.); htshake@gmail.com (Y.-S.J.); s.morsalin10@gmail.com (S.M.S.M.); ajohn83920@gmail.com (Y.-F.H.)
2 Smart Machinery and Intelligent Manufacturing Research Center, Department of Electrical Engineering, National Formosa University, Huwei 632301, Taiwan; ccsun@nfu.edu.tw
3 Department of Computer Science and Information Engineering, Nanhua University, Chiayi 62249, Taiwan
4 Department of Automation Engineering, National Formosa University, Huwei 632301, Taiwan
* Correspondence: chingivan2008@gmail.com or shivan0111@nhu.edu.tw

check for updates

Citation: Sheu, M.-H.; Jhang, Y.-S.; Morsalin, S.M.S.; Huang, Y.-F.; Sun, C.-C.; Lai, S.-C. UAV Object Tracking Application Based on Patch Color Group Feature on Embedded System. *Electronics* **2021**, *10*, 1864. https://doi.org/10.3390/electronics10151864

Academic Editor: Francesco Beritelli

Received: 25 May 2021
Accepted: 6 July 2021
Published: 3 August 2021

Abstract: The discriminative object tracking system for unmanned aerial vehicles (UAVs) is widely used in numerous applications. While an ample amount of research has been carried out in this domain, implementing a low computational cost algorithm on a UAV onboard embedded system is still challenging. To address this issue, we propose a low computational complexity discriminative object tracking system for UAVs approach using the patch color group feature (PCGF) framework in this work. The tracking object is separated into several non-overlapping local image patches then the features are extracted into the PCGFs, which consist of the Gaussian mixture model (GMM). The object location is calculated by the similar PCGFs comparison from the previous frame and current frame. The background PCGFs of the object are removed by four directions feature scanning and dynamic threshold comparison, which improve the performance accuracy. In the terms of speed execution, the proposed algorithm accomplished 32.5 frames per second (FPS) on the x64 CPU platform without a GPU accelerator and 17 FPS in Raspberry Pi 4. Therefore, this work could be considered as a good solution for achieving a low computational complexity PCGF algorithm on a UAV onboard embedded system to improve flight times.

Keywords: unmanned aerial vehicle (UAV); UAV object tracking; Gaussian mixture model (GMM); patch color group feature (PCGF); embedded system

1. Introduction

Unmanned aerial vehicles (UAVs) have rapidly evolved, and there are lots of examples of UAV analysis applications in various fields, such as transportation engineering systems [1], UAV bridge inspection platforms [2], UAV-based traffic analysis [3], and oil pipeline patrol factory inspection [4], etc. In recent years, several object monitoring strategies have been proposed for discriminative object tracking. The solutions for low-cost computational complexity, accurate object tracking, real-time operation speed, and embedded-system implementation must turn out to be necessary problems. In general, the UAV object tracking algorithms are categorized into the deep learning (DL) method and the generic method.

The DL method has been repeatedly used in many works for data compression [5], noise reduction [6], image classification [7], speech recognition [8], disease diagnosis [9], and so on. To extract the important features from input data, the DL model consists of several convolutional layers, activation functions, and pooling layers. The loss function is compared with the DL output error and the ground truth. To reach higher accuracy, the model needs to pre-train before using real cases [10]. In the DL tracking [11–22], the CNN [11–13,16,22] and RNN [18,19,21] are used to achieve higher accuracy on the discriminative object tracking. The DL model presents a powerful tracking performance. In addition, high-end computational platform, high-performance X86-64 multi-core CPU, GPU accelerator and huge computational complexities are required, which are also power-consuming. Therefore, it is hard to achieve DL tracking on UAV onboard embedded systems because of its limitations.

In the generic method, the image features of each frame are obtained through the filters. These features are further used to acquire the object's location. Recently, the discriminative correlation filter (DCF) method has turned out to be a popular and efficient approach to obtain features for discriminative object tracking [23–33]. The objective of DCF is to find a suitable parameter to maximize the features extracted from the object. Because of the cyclic correlation operation, the boundary effect causes periodic expansion in the DCF, which degrades tracking performance [24]. In [25], the regularization condition is applied to suppress the boundary impact and enhance performance to overcome the disadvantage and improve performance. However, when the object moves faster than the variation of local response in the frame, the lack of information and reformation are restricted by the regularization condition. Additionally, more computation increases the complexity, and the decrease in the FPS tends significantly in deformed DCF.

The authors suggested the PCGF algorithm, which is a general approach that works well. It can minimize computational complexity and enhance the object tracking control system's efficiency on UAVs. The PCGF is made up of four GMMs [34] derived from the hue, saturation, and value (HSV) color model. The tracking object is divided into numerous non-overlapping patches, which are then converted into PCGFs, to represent attributes. By comparing the PCGFs from previous and current frames, the object's location is calculated. The backdrop feature of the item has been removed from the picture, ensuring that the current PCGFs have no background feature. In addition, the object is not moved very fast, and difference in the pixels from one frame to another is almost similar. The object position in the sequence frames only needs to be searched around the previous position in order to find the features matching. The main contributions of this work are summarized as follows:

- ❖ An efficient PCGF algorithm for a generic approach is proposed to present the object features and compare the features matching score of the previous frame and the current frame.
- ❖ We introduced a background subtraction technique for PCGF to eliminate background characteristics and ensure the object features are reserved.
- ❖ The proposed method has been implemented on an embedded system, in addition, the proposed approach achieved the real-time speed on a CPU platform without a GPU accelerator.

The rest of this paper is organized as follows: Section 2 revisits the introduction of the previous works. Section 3 describes the proposed method. Section 4 exhibits experimental results as well as related discussions. Finally, the concluding remarks are drawn in Section 5.

2. Related Work

2.1. Deep Learning-Based Tracking

The DL-based method consists of neural network layers, which contain millions of parameters. The DL-based process finds values for every parameter to minimize the error between output and the ground truth. In recent years, the DL-based method has been broadly used for discriminative tracking. The object location is predicted by the features, which are extracted through hidden layers of neural network layers. In [11,12,14,17,18,23], a fully convolutional neural network was used to extract the feature from the input image. The pre-trained VGG Net [7] was implemented as the first layer to improve the object's feature performance [11,12,17,18,21,22]. In [11], two networks, the generic network (GNet) and the specialized network (SNet) were used to produce two heat maps from the input image's area of interest (ROI). The object position was then identified by distracter detection, which was determined by the two heat maps. According to the tracking sequence, Yun et al. [21] presented an action-decision network to anticipate the object's action. The video's recurrent neural network (RNN) shows a strong correlation between frames [19,20,22]. The object feature was used to compare the changing temporal features in multiple spatial LSTM cells in [22]. Multi-directional RNNs were used to calculate the spatial confidence map in [19]. Background suppression is evident in the results. The authors of [20] utilized the quicker R-CNN to discover all probable item candidates in the immediate search area. In addition, the three-stage LSTM cascade decided whether the tracker should update its location based on the search region. The coarse-tracker and fine-tracker [12,15,22] were utilized to execute the coarse-to-fine searching method in [15]. The object location was predicted using a combined response map based on the divided parts of the object in [18]. The Siamese network, which learns the similarity response with template and search region, was proposed in [16]. In addition, various methods to the DL-based tracking system included correlation filters (CF) [12,17–19,22]. The accuracy of discriminative tracking can be improved by combining the CF and DL-based methods. In short, DL-based discriminative tracking methods have high performance, but they are not practical on the UAV onboard platform due to the enormous computing required.

2.2. Generic Method

To enhance the object features in the generic approach, a filter is designed to reinforce the discriminative object characteristics. The object location is determined by the filter response. Furthermore, the filter's coefficient updates every frame to detect object position. For object tracking, the discriminative correlation filters (DCF) are worked as the framework. In [23], the CF process the 256 $\times$ 256 image for object tracking, and a good result was achieved using a single-core 2.4 GHz CPU platform. In [24], the spatially regularized correlation filters (SRDCF) are proposed to eliminate the effect of the boundary effect. Fu. et al. [25] kept images into background-aware correlation filter (BACF) through different levels to obtain multiple features. Shi et al. [26] proposed an effective visual object-tracking algorithm based on locally adaptive regression kernels and implemented it using two support vector machines. In [27], the software de Reconhecimento de Cor e Contorno Ativo (SRCCA) is presented as target recognition and tracking system that uses the hue, saturation, and value color space for color-based object identification in the active component model. The Kalman filter was used for preliminary tracking in [28], and the findings were improved using a saliency map for local detection. Choi et al. [29] created a visual tracking algorithm for attention-modulated disintegration and integration, which they used to disintegrate and integrate target data. A target item is fragmented into various structural characteristic components, such as size, colors, and forms. The dual regularized correlation filter (DRCF) is presented in [30], which is employed to more effectively mitigate the boundary effect. The time slot-based distillation algorithm (TSD) was introduced by Li et al. [31] to improve the BACF. The log-Gabor filter is utilized to encode texture information during tracking in [32], and boundary distortion may be mitigated via saliency-guided feature selection. In conclusion, the DCF-based tracking system achieves a good mix of speed and accuracy. The described method for UAV tracking achieved real-time operation in [24–28]. However, once the target is detected missing, this may affect the tracking in the other frame severely. The accuracy is also affected by object deformation, rapid motion, illumination variation, and background complexity.

3. Proposed Method

Figure 1 depicts the suggested method's working procedure schematic flowchart. Feature extraction, feature matching, and background removal are the three portions of the schematic flowchart, respectively. The feature extraction portion retrieves the target object's characteristics from each current frame. Furthermore, the RGB pixel values are transformed to HSV color space and then divided into numerous non-overlapping blocks.

In addition, the object patches are segmented, and each patch's value combines with GMM to obtain the hue and saturation (μ, σ) values into four groups. The GMM enhances the feature extraction dynamically, and also the K means++ algorithm is developed to acquire the center point of Gaussian distribution before analyzing image features. In addition, the obtained features reinforce for improving target image object precision as well as object deformation, illumination variation, rapid motion, and visual obstruction.

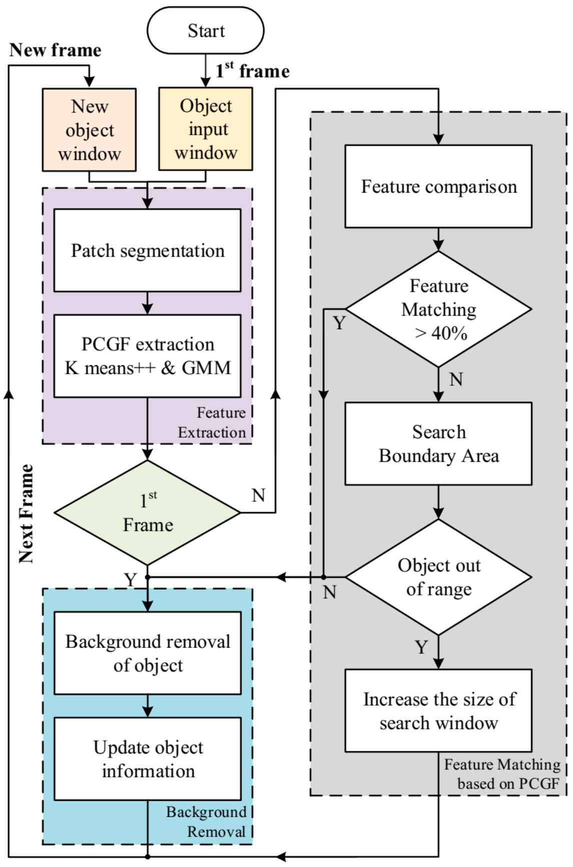


Figure 1. Proposed UAV object tracking algorithm flowchart.

The features of the object are extracted from the first frame to evaluate object characteristics, then block patches segmented for the next operation. Before background subtraction, object features are compared, and the matching score is calculated for the current frame and the previous frame. If similar characteristics were identified, the object information of the current frame is sent into the background removal section; otherwise, the search window is expanded, and the target image object can be attempted to be found again from the input window. The properties of the target object and its background patches were compared to determine whether the patches belonged to the same target object or not. The segmented object image patches are scanned in four directions to eliminate the background from the object window: half-length from up to down and down to up, similarly half-width from left to right and right to left. The target picture object's coordinates, width, height, and patch characteristic values, such as dimensions, colors, and forms, are then changed before proceeding to the next frame assessment.

3.1. The Function of the Input Window

Initially, the object window uses the input window shown in Figure 2 to locate the target item (a). The upper-left side of the input window extracted (Xobj, Yobj) feature and the object window recognized the object's height (Hobj) and width (Wobj). The characteristic values of the target object and its surrounding pixels were retrieved from the input window and object window to decrease computational complexity. The initial frame determines the object's information. Because the target object is moving, the search window is enlarged by 72 pixels on either side of the object window in Figure 2b to identify the target object's current frame.

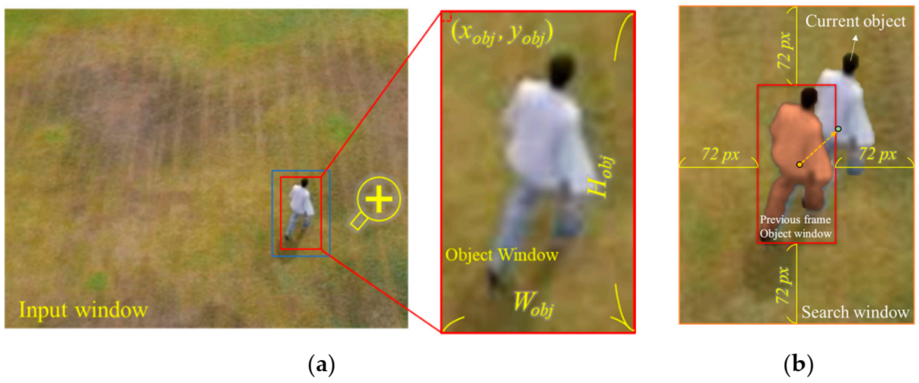


Figure 2. (**a**) Object window defines object from input window; (**b**) 72 pixels expanded search window.

3.2. Patch Segmentation

The object window is captured using the input window, and 12 pixels are placed around it, as seen in Figure 3. Both the object window and the enlarged region are divided into non-overlapping 6 × 6-pixel patches, which are divided into two groups: the object patch (Po) and the outer backdrop patch (Out) (POB). The height and width of the new window are (Hobj+24) and (Wobj+24), respectively. To minimize object tracking complexity and computational cost, and to speed up the performance, the RGB color model has converted to HSV color and just hue and saturation were processed for the computation.

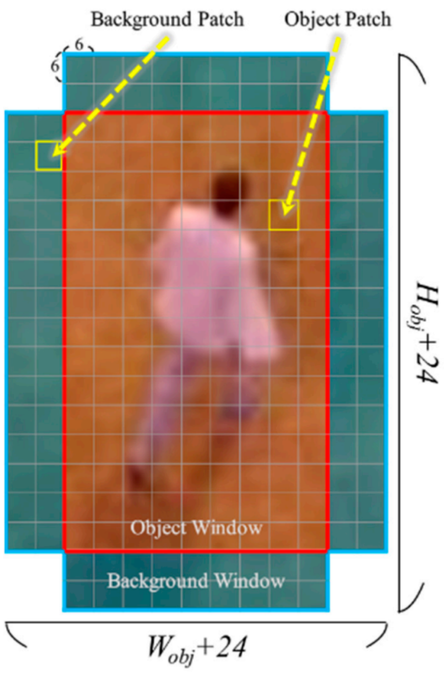


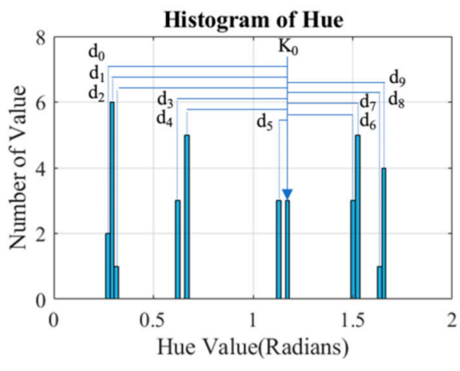
Figure 3. Object patches segmentation.

3.3. PCGF Extraction of K means++ and GMM

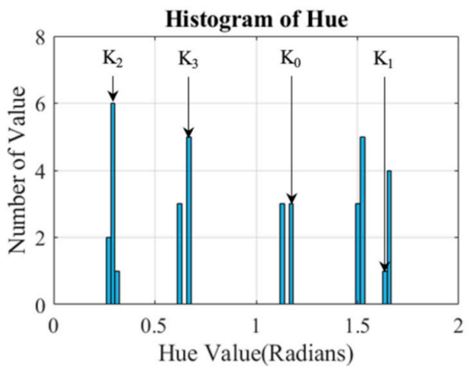
After converting the HSV color gamut and determining the patch color, the next step is the GMM clusters represent several feature values from the patches that are named patch color group feature (PCGF). Afterward, the hue and saturation values of each patch were captured, which requires the initial center value to be defined, in order to analyze the performance. Initial center values were determined by using the K means++ algorithm.

The K means method occasionally produced a poor cluster because of excessively random beginning settings. To solve the problem, Arthur et al. [33] increased the distance between each group's original center locations. We utilized K-means++ to obtain four cluster center points (K0, K1, K2, K3) in each patch, allowing us to determine the maximum distance from each group's starting center location. The detailed procedure is as follows: As illustrated in Figure 4, the HSV histogram is transformed first, then one index of the non-zero value is chosen to represent the initial cluster center point (K0) (a). Following the discovery of the original cluster center point K0, the second, third, and fourth cluster points (K1, K2, K3) were discovered. K1 is the result of calculating the distance between the non-zero index and the center point K0. After determining the center values, the closest distance (Dn) between each pixel's hue and saturation value is calculated using Equation (1) for the non-zero index value and the *i*-th group. Figure 4 depicts the calculated distance, center point, and center values because of the computations (b) (The distance between two locations is denoted by $\mathbf{d}_i$).

$$D_n = \min(HueValue_n - K_i),\ i = 0, 1, \ldots, 3,\ n = NumberOfValue \neq 0 \tag{1}$$



(**a**)



(**b**)

Figure 4. K means++ algorithm. (**a**) The distance between each pixel and the center point is determined; (**b**) Center values are determined after four repeated calculations.

The D_n of each pixel is used to determine the new center point of a group. A higher D_n value is more likely to become the new center value K_i. The patch pixels are divided into four groups, therefore Equations (1) and (2) must be performed four times to calculate the center values of the four groups. The values of these four groups are applied in the GMM to determine the initial conditions.

$$K_i = \mathrm{argmax}(D_n / \sum_n D_n) \tag{2}$$

The GMMs [33] are prevalent in background subtraction, pattern recognition, and performance. The image distribution cannot be described only by using Gaussian distribution, a GMM enables the analysis of data distribution to improve the precision. The GMM consists of multiple Gaussian models and is expressed as follows:

$$P(\mathbf{x}|\theta) = \sum_{k=1}^{K} w_k \phi(\mathbf{x}|\theta_k) \tag{3}$$

where $\mathbf{x}$ represents the input vector data, K denotes the total number of single Gaussian models (four are used to represent the characteristic values of a single block), k is the group index value of a single Gaussian model, w_k represents the probability of an input belonging to the k-th single Gaussian model, and $\phi(\mathbf{x}|\theta_k)$ denotes the Gaussian distribution probability density of the k-th Gaussian model [$\theta_k = (\mu_k, \sigma_k^2)$]. In this study, the color blocks of the input image were grouped. The K means++ algorithm is first employed to identify the group center values of a color block, and the GMM is used to categorize each pixel of the color block to a specific group. To reduce the calculation complexity, the optimization equations proposed by Lin et al. [34], as expressed in Equations (4)–(6), are applied.

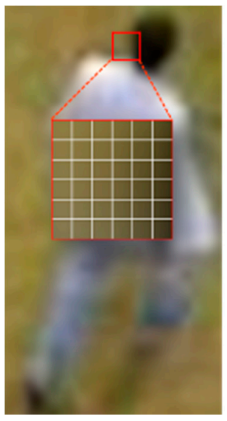
$$w_{k,t+1} = (1 - \alpha_w)w_{k,t} + \alpha_w \tag{4}$$

$$\mu_{k,t+1} = (1 - \alpha_{k,t})\mu_{k,t} + \alpha_{k,t} \cdot px(i) \tag{5}$$

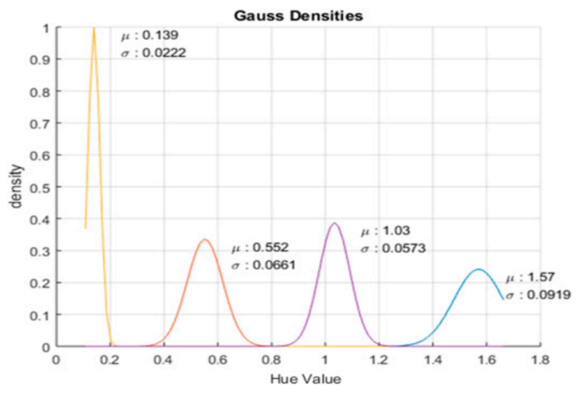
$$\sigma^2_{k,t+1} = (1 - \alpha_{k,t})\sigma^2_{k,t} + \alpha_{k,t}\alpha_w(px(i) - \mu_{k,t})^2 \tag{6}$$

Here, $\alpha_{k,t} = \alpha_w/\omega_{k,t}$, $\omega_{k,t}$ represents the probability value of the k-th single Gaussian model in the t-th frame, $\mu_{k,t}$ represents the mean value of the single Gaussian model, $\sigma^2{}_{k,t}$ represents its standard deviation, and α_w represents the learning rate. Figure 5b shows the hue value computed using the K means++ method and GMM for one of the patches seen in Figure 5a. There is a total of four Gaussian distributions found, and the characteristic values of the patch's hue value are stated in Equation (7), where m and n are the patch's index values. Both hue and saturation values are required for each patch. In the first frame, just the object window and background window's characteristic values are extracted; in subsequent frames, all the search window's characteristic values are extracted.

$$P_{m,n} = \{\mu_0, \mu_1, \mu_2, \mu_3, \sigma_0, \sigma_1, \sigma_2, \sigma_3\} \tag{7}$$



(**a**)



(**b**)

Figure 5. Patch characteristics. (**a**) Patch information; (**b**) GMM grouped color chart.

3.4. Background Patch Removal from the Object Window

After defining the characteristic values ($P_{m,n}$) of every patch, the next step is removing the background patches from the object window. The proposed algorithm extracts patches and finds a threshold window from the background window for every vertical and horizontal patch. The threshold values are calculated by subtracting the background characteristic values of P_{OB} (i.e., $thr(m,n) = |P_{m,n} - P_{m+1,n}|$), which are depicted in Figure 6a,b. The picture is split and scanned in four directions. As indicated in Figure 6, the half-length height from up to down and down to up was scanned at the same time, as was the half-length height from left to right and right to left Figure 6c,d.

Figure 7 illustrates patch acquisition based on the upper half of object characteristics. The upper half part of the object patches is scanned and expressed as the label by the following Equation (8). When the upper half part of the object patch is scanned, (m, n) is the uppermost threshold (*thr*) coordinate, and k represents the displacement value. The *thr* value is compared with the characteristics of the upper half of the object patch (H_{obj}). The $Label_{m,n+k}$ is marked as 1 if the deviation between any characteristic values of the patch and the threshold are greater than the threshold, otherwise it is set to 0. To reduce the computation, when $Label_{m,n+k-1}$ is determined as object patch, the remaining blocks require assessment that can be directly identified as the object patch.

$$Label_{m,n+k} = \begin{cases} 1, & \left|thr_{m,n} - P_{m,n+k}\right| > thr_{m,n} \parallel Label_{m,n+k-1} == 1 \\ 0, & otherwise \end{cases} \tag{8}$$

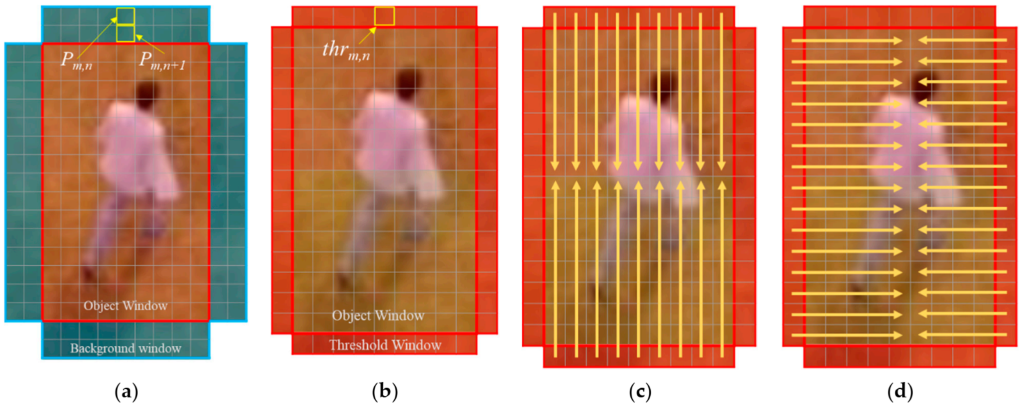


Figure 6. The demonstration of dynamic threshold-value scanning. (**a**,**b**) The threshold value (*thr*) is finding the difference of the characteristics value ($P_{m,n}$ and $P_{m,n+1}$) in the background window; (**c**) vertically scanning process; (**d**) horizontally scanning process.

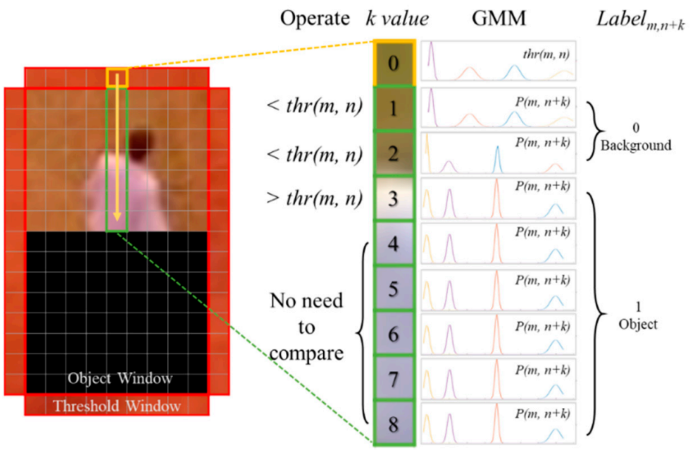
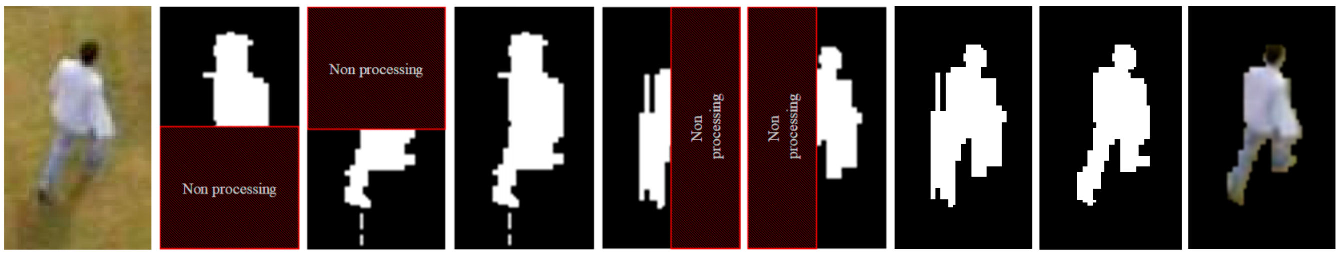


Figure 7. Object marking and difference characteristic calculation.

Figure 8b shows each column pixel after scanning and computing the upper half, with 1 and 0 patches indicated in white and black, respectively. After finishing the upper- and lower-part scanning, the vertical scanning mask is subjected to a "OR" operation, as shown in Figure 8d. As shown in Figure 8, the left and right half of the patches are scanned in the same way, and a "OR" operation is calculated for horizontal mask scanning (Figure 8g). Finally, the entire mask is obtained by performing an "AND" operation on the vertical and horizontal scan results. Figure 8 shows the acquisition outcome of all patches collected by four directional scanning's Figure 8h.

The object's characteristic values are updated to the next frame to monitor the object's location. Figure 8a depicts the original frame, whereas Figure 8i depicts the background removal mask in combination with the image generated from the scanning results. Consequently, obtaining the object's upper-left coordinate, width, and height may be used to define the frame.



(**a**) (**b**) (**c**) (**d**) (**e**) (**f**) (**g**) (**h**) (**i**)

Figure 8. Mask scanning results. (**a**) Original image; (**b**,**c**) upper and lower segmentation, respectively; (**d**) vertical mask scanning; (**e**,**f**) left and right segmentation, respectively; (**g**) horizontal mask scan; (**h**) finalized mask; (**i**) identified image.

3.5. Feature Matching Based on PCGF

After all the characteristic values are defined in the search window, all patch characteristics within the window are compared with the previous frame to determine their similar mask (Figure 9). Equation (9) is used to find the coordinate which makes the minimal difference between the search window and object window in Equation (10).

$$MatchPos = \underset{Q,V}{\operatorname{argmin}} \sum_{m=0}^{I-1} \sum_{n=0}^{J-1} GDiff(OW_{SET(m,n)}, SW_{SET(Q+m,V+n)}) \\ Q = 0, \ldots, W-I-1,\ V = 0, \ldots, H-J-1 \tag{9}$$

$$GDiff(OW_{SET(m,n)}, SW_{SET(m,n)}) = \begin{cases} \left| OW_{SET(m,n)} - SW_{SET(m,n)} \right|, & OW_{SET(m,n)} > 0 \\ 0, & otherwise \end{cases} \tag{10}$$

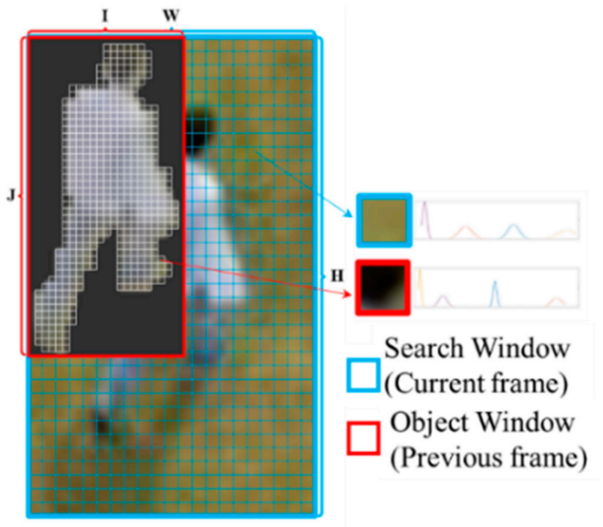


Figure 9. Searching window and object window.

Specifically, OW_{SET} represents the patch set of the object window in the previous frame, SW_{SET} indicates the search window group, (Q, V) is the searching window coordinates, and (m, n) is the block scan coordinate.

After identifying the searching coordinate (*MatchPos*), the feature matching scores are calculated by the following Equation (11). Specifically, *DiffMAX* is the maximum difference for color value (set as 255 in this paper), and *NS* represents the number of object patches defined as an object in the previous frame. If the feature matching score is lower than 40%, then the detection will be invalid, and the searching system requires additional assessment; otherwise, the system enters the background-removal block to update the upper-left coordinate, width, height, and patch characteristics of the object in the current frame.

$$FeatureMatching = 1 - \frac{\min_{m,n} GDiff(OW_{SET(m,n)}, SW_{SET(Q+m,V+n)})}{DiffMAX \times NS} \times 100\% \quad m = 0, \ldots, I-1,\ n = 0, \ldots, J-1 \tag{11}$$

3.6. Object Out of Range (Boundary Area)

If the feature matching is insufficient (<40%), it means the object is out of the boundary area. The searching system assesses whether the object is out of the boundary area or not. While the object is out of bounds, it may appear again from any edge of the boundary area. Therefore, the searching window switches the boundary area, and its height and width are determined using the object window, as shown in Figure 10.

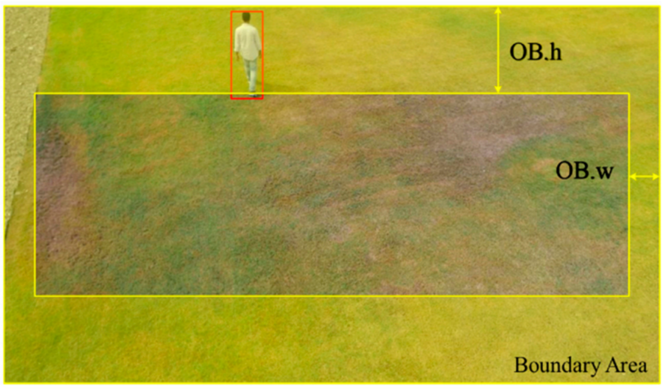


Figure 10. The search range is switching to boundary area when the object is out of bounds.

3.7. Object Occlusion

If the feature matching score is low, but the object is still inside of the window, then the object may be obstructed by other objects illustrated in Figure 11b. The search window then expands to twice the size of the object window to increase the searching range, and the frame maintains its existing specifications to find the object.

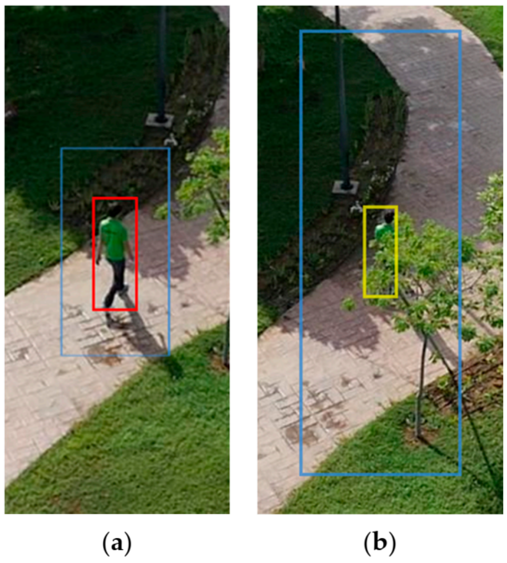


Figure 11. (**a**) The searching window extends 72 px from the object when unobscured; (**b**) object occlusion. The yellow box represents the object label of the last mark which was detected successfully; the blue box represents the size of searching windows increases since the PCGF doesn't detect the object.

4. Experimental Results and Analysis

The method is written in C++ and implemented with the OpenCV library to assess the experimental results. Following that, the code is translated to ARM architecture on the x86-64 platform to evaluate the embedded system's speed and correctness. The correctness of several algorithms was assessed in Section 3.2, and the execution efficiency of the algorithms was evaluated in Section 3.3.

4.1. Measurement Results

The multiple video test performance and tracking results for UAVs are compared with the various algorithms. The UAV123 dataset by Mueller et al. [35] is used to perform this experiment. The most difficult patterns are used to recognize the objects. Figure 12a,b depicts the precise tracking of a fast-moving and deforming object. The method is shown in Figure 12c looking for the pictured item that is out of the window. When the object reenters the window, the algorithm correctly selects the item, as seen in Figure 12d. Figure 13 depicts a case in which the tracking item is covered by a tree. #1399 is the final frame that PCGF can follow the object, as seen in Figure 13a. Because of the item covered by the tree in #1400-1446, the matching score is less than 40. As illustrated in Figure 13, the PCGF expands the search window to discover the item (b). The matching score in #1446 is more than 40. As a result, as illustrated in Figure 13, the object label can be re-marked (c). Figure 14 shows how the system correctly tracked an item while a human was dressed in the same color as the backdrop (grass color), enhancing the object's resemblance to the background. Figure 15 depicts the algorithm accurately identifying an object that is obstructed by leaves and crossing the intersection between sunlight and shade, thereby causing partial obstruction and illumination changes. Figure 15 shows how the algorithm correctly recognized an object that is partially obscured by leaves and crosses the intersection of sunshine and shadow, resulting in partial blockage and lighting variations.

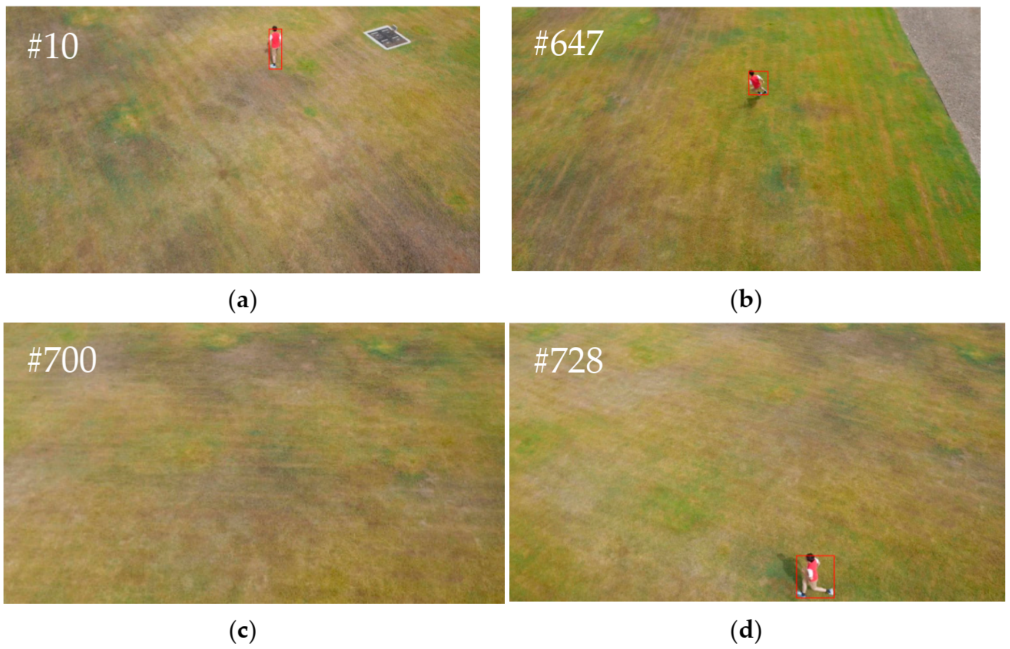
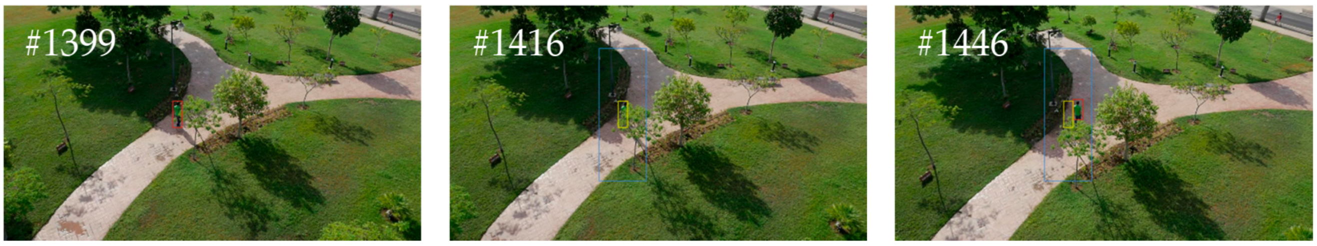
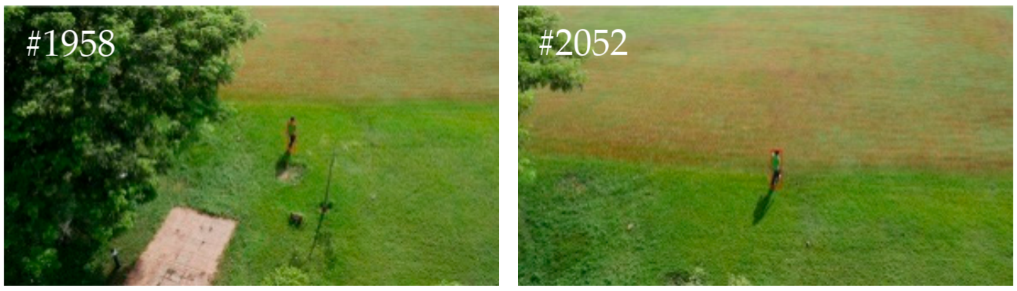


Figure 12. UAV123 datasets tracking test of person7. (**a**) Object in its initial state; (**b**) object in a rolling movement; (**c**) object exiting the line of sight; (**d**) object reentering the line of sight.



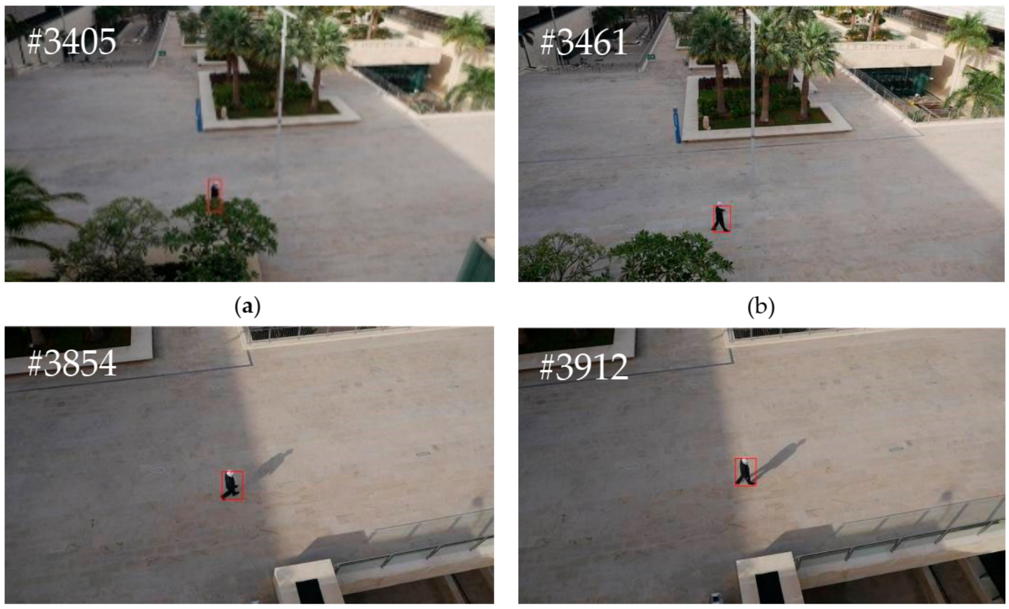
(**a**) (**b**) (**c**)

Figure 13. UAV123 datasets tracking test of person17. (**a**) The last tracked result before the object disappears; (**b**) object disappears, increasing searching window size twice time according to last tracked result; (**c**) object detected again inside the searching window.



(**a**) (**b**)

Figure 14. UAV123 datasets tracking test of person17. (**a**) Object waring similar color cloth with background and complex characteristics; (**b**) object against a background with similar characteristics.



(**c**) (**d**)

Figure 15. UAV123 datasets tracking test of person19. (**a**) Object obstructed; (**b**) object appear after being obstructed; (**c**) object in a location with low illumination; (**d**) object at the intersection of areas with different illumination levels.

Furthermore, we also compared the result of PCGF with some other experiments. This work makes a fair comparison with some state-of-the-art tracking algorithms, which include three DL algorithms: C2FT [14], PBBAT [17], and ADNET [20], and one generic algorithm: SCT [29]. The five test patterns and the results are shown in Figure 16. All of the algorithms have better capabilities for detecting objects with simple backgrounds in person1, person7, and bike1. However, in more complex scenery, PBBAT [17] and SCT [29] obtained a bad tracking and lost the object location in wakeboard4 #0417 because of the complicated waves. In car18, the car moved rapidly, which caused most algorithms to correctly frame the target, but there has a significant error in the size of the image.

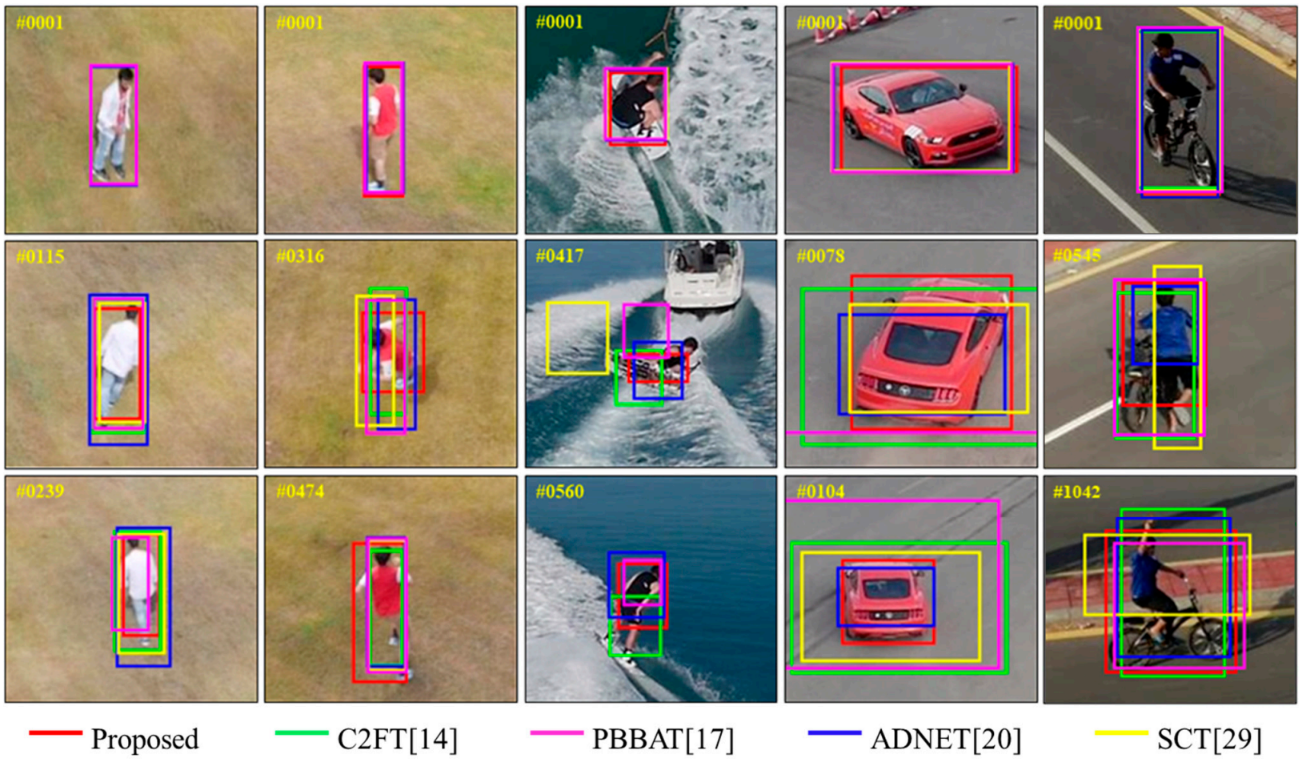


Figure 16. Examples of the UAV datasets tracking results. The first, second, third, fourth, and fifth columns show the challenging image sequences. From left to right are person1, person7, wakeboard4, car18, and bike1, respectively.

4.2. Algorithm Precision Assessment

Wu et al. [36] utilized Equation (12) to define and compute the center point placement error, the ground truth is defined by x_{gt} and y_{gt} in each frame, and the center position of the forecast object window is defined by x_{rs} and y_{rs} the error values of each algorithm. To reduce the error and more accurately track an object, the overlapping rate is calculated by the following Equation (13), whereas the overlapping area between the ground-truth object frame represents by (Win_{gt}) and the tracking object frame (Win_{rs}); a higher overlapping rate indicates that the tracking result is close to the ground truth.

$$CenLocErr = \sqrt{(x_{rs} - x_{gt})^2 + (y_{rs} - y_{gt})^2} \tag{12}$$

$$OverlapRate = \frac{Win_{rs} \cap Win_{gt}}{Win_{rs} \cup Win_{gt}} \tag{13}$$

After defining the center point location error and overlap rate of each frame, and the precision rate is calculated using Equation (14) as a reference, the success rate (15) can be obtained. The label value *FT* indicates whether the center point location error and overlap rate of the frame exceed their threshold values. If so, *FT* is set to 1; otherwise, *FT* is set to 0.

$$\begin{gathered} PrecisionRate(LocErrThr) = \frac{\Sigma FT(CenLocErr, LocErrThr)}{TotalFrame}, \\ \text{where } FT(x, thr) = \begin{cases} 1, & x \leq thr \\ 0, & otherwise \end{cases} \end{gathered} \tag{14}$$

$$SuccessRate(OverlapThr) = \frac{\Sigma FT(OverlapRate, OverlapThr)}{TotalFrame},$$
$$\text{where } FT(x, thr) = \begin{cases} 1, & x \geq thr \\ 0, & otherwise \end{cases} \tag{15}$$

In terms of precision and success, the proposed approach has been compared to those used in C2FT [14], PBBAT [17], ADNet [20], and SCT [29]. One-pass evaluation (OPE) was used in this experiment to compute accuracy, 12 distinct patterns, and various threshold levels for the findings shown in Figure 17. The accuracy is shown by the y-axis, while the threshold value is represented by the x-axis. Except for person18, person20, car18, and wakeboard5, the object center error between the proposed PCGF's projected results and the ground truth supplied by UAV123 is less than 50 pixels. Images of person20, car18, and wakeboard5 reveal that the precision is not closed to 1 for all the compared algorithms, even threshold set as 50 pixels. It means that the object center error is greater than 50 pixels in some frames because the background is complicated and the filming angle of the UAV changes continuously. Table 1 shows the precision result of 5 algorithms when the threshold is set to 20, which is the same evaluation terms as in [20,29]. For relatively simple patterns such as person1, person7, and person16, significant differences were not detected between the algorithms in their precision. However, more complex patterns such as person16 and person20, car18, and boat3, have observed significant differences. The center position predicted by SCT [29] has a big gap with other algorithms and has the same appearance as that shown in frame 417 of wakeboard4 in Figure 16. Under the average rating, although the PBBAT [17] which is a deep learning method has shown the highest average precision that mentions in the green color in Table 1, the operational function for real-time applications is difficult because of the advanced training procedure and highly complex calculation process. However, the proposed generic method has achieved the second-highest average precision that mentions in the blue color in Table 1, and the proposed generic method can operate at real-time applications on the UAV object tracking application with less calculation process.

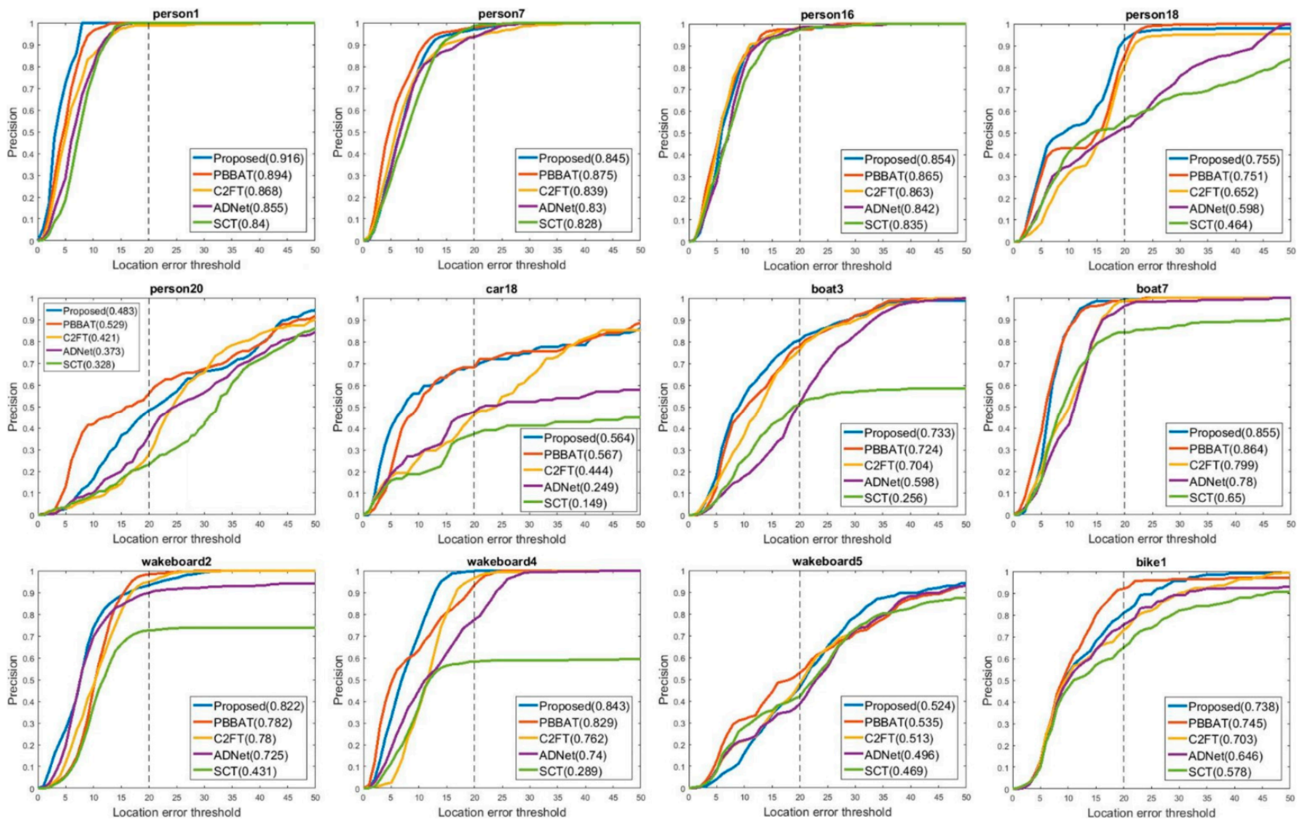


Figure 17. The precision plot of OPE on UAV123 [35].

Table 1. The precision comparison result when threshold is set as 20. The best two precisions for each pattern are shown in blue and green colors, respectively.

	Generic Method		**Deep Learning Based**		
Pattern	**Proposed**	**SCT [29]**	**C2FT [14]**	**PBBAT [17]**	**ADNet [20]**
Person1	0.916	0.830	0.868	0.894	0.855
Person7	0.845	0.828	0.839	0.875	0.830
Person16	0.854	0.835	0.863	0.865	0.842
Person18	0.755	0.464	0.652	0.751	0.598
Person20	0.483	0.328	0.421	0.529	0.373
Car18	0.564	0.149	0.444	0.567	0.249
Boat3	0.733	0.256	0.704	0.724	0.598
Boat7	0.855	0.650	0.799	0.864	0.780
Wakeboard2	0.822	0.431	0.780	0.782	0.725
Wakeboard4	0.843	0.289	0.762	0.829	0.740
Wakeboard5	0.524	0.469	0.513	0.535	0.496
Bike1	0.738	0.579	0.703	0.745	0.646
Average	0.744	0.509	0.695	0.746	0.644

Figure 18 illustrates the success plot of OPE, which investigated the overlapping rates of the algorithms; x represents the threshold value of the overlapping rate, and y represents the success rate. The result shows that in all patterns there exists an intersection between the PCGF result and the UAV123 ground truth. In person18, car18, and boat7, these algorithms reveal significantly different curves in boat3, boat7, wakeboard2, and wakeboard4; the SCT [29] shows that the success rate doesn't reach 1 even though the threshold set in a small value. The overlapping rate is measured by the areas under the curves (AUCs), which were calculated by using the Equation (16). Total AUCs are displayed in Table 2, and it should be mentioned that the proposed algorithm yielded good results, written in green in the table, which are better than the deep learning-based PBBAT [17] method.

$$AUC = \int_0^{\max(x)} SucessRate(x)dx \tag{16}$$

Table 2. The success result of the AUC score. The best two precisions for each pattern are shown in blue and green colors, respectively.

	Generic Method		**Deep Learning-Based**		
Pattern	**Proposed**	**SCT [29]**	**C2FT [14]**	**PBBAT [17]**	**ADNet [20]**
Person1	0.821	0.762	0.633	0.810	0.697
Person7	0.589	0.512	0.575	0.655	0.585
Person16	0.844	0.814	0.845	0.846	0.840
Person18	0.531	0.418	0.348	0.498	0.419
Person20	0.514	0.394	0.464	0.539	0.498
Car18	0.709	0.469	0.507	0.748	0.542
Boat3	0.571	0.177	0.484	0.569	0.488
Boat7	0.674	0.356	0.587	0.630	0.585
Wakeboard2	0.518	0.315	0.395	0.511	0.429
Wakeboard4	0.607	0.202	0.538	0.579	0.509
Wakeboard5	0.559	0.518	0.543	0.543	0.511
Bike1	0.728	0.696	0.695	0.725	0.665
Average	0.638	0.469	0.551	0.637	0.564

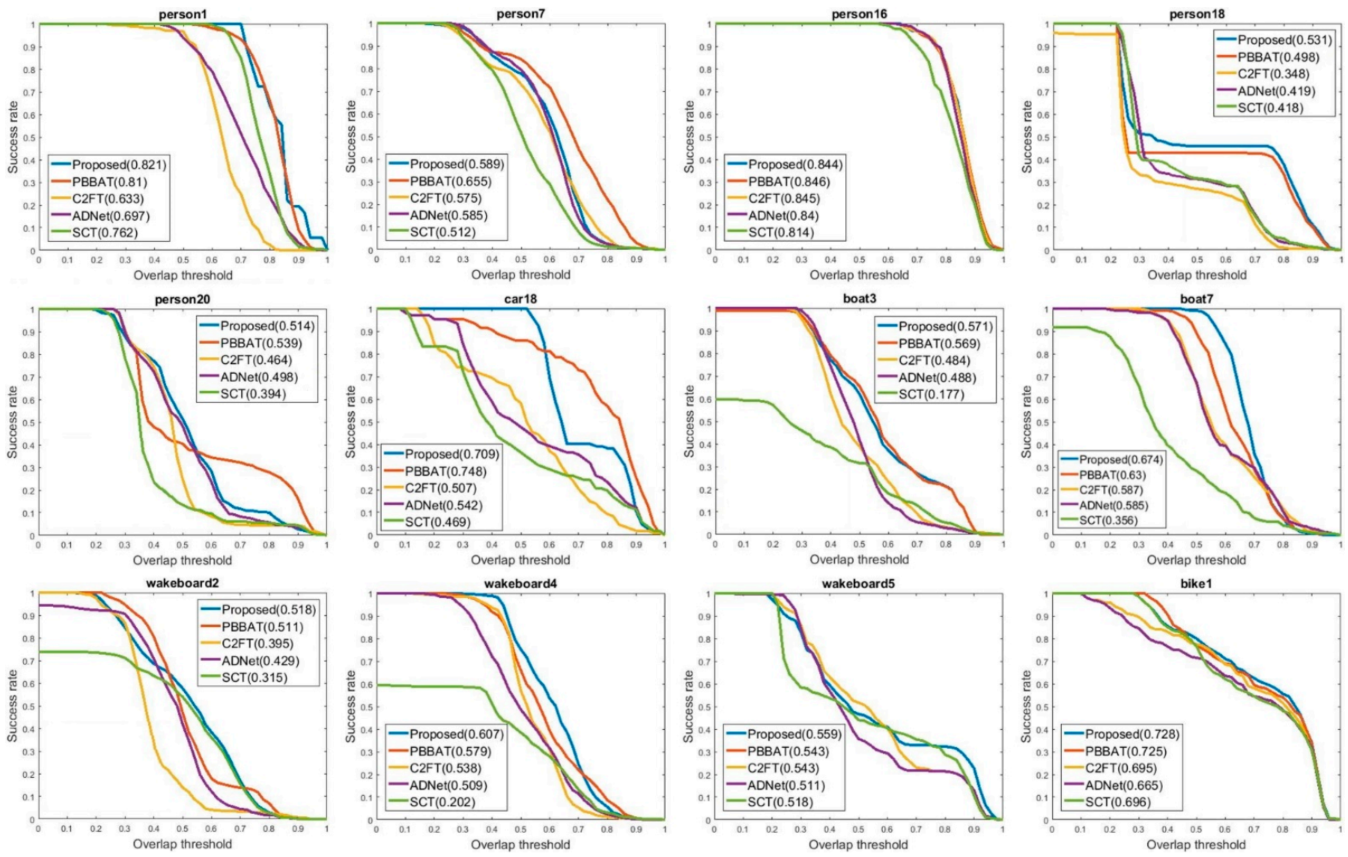


Figure 18. The success plot of OPE on UAV123 [35].

For the overall evaluation of the average precision result and the average AUC score with the threshold set to 20, our algorithm reveals a great performance on object tracking. The result indicates that our approach achieved a very near result to PBBAT [17], which is a DL method, and its computational cost is more complex than our proposed approach.

4.3. Algorithm Speed Assessment

The suggested method was tested on the i5-6400 CPU architecture, which has a 14 nm lithography process, four cores and four threads, a maximum processing speed of 3.3 GHz, and a TDP of 65 W. In terms of performance, the PCGF is implemented using C++ and the Open CV library, and it achieves 32.5 FPS on the CPU platform. This work may also be carried out on the Raspberry Pi 4, which has a low-power BCM2711 SoC processor with an ARM-based cortex A72 4 core. On the high-configurational GPU platform, the ADNet [20] model obtained greater FPS, whereas the PBBAT [17] model only reached 0.87 FPS, as shown in Table 3. On the other hand, the proposed PCGF algorithm achieved 32.5 FPS on the CPU platform and 17 FPS when executed on Raspberry Pi 4. To compare with the power consumption, the algorithms in [17,20,29] require a GPU accelerator for achieving the proposed, which takes 65 W (GTX 650) extra power consumption compared with the CPU-only platform. In addition, the Raspberry Pi 4 only requires 6.4 W on stress benchmark testing, which saves 97% of the power consumption of desktop computers with a 300 W power supply. It is the better solution for achieving the PCGF on the UAV onboard platform for discriminative tracking.

Table 3. Hardware specifications and FPS comparison.

Algorithm	**Processor**		**RAM**	**FPS**
	CPU	**GPU**		
PBBAT [17]	i7-8700K	Quadro P2000	48 GB	0.87
ADNet [20]	i7-4790K	GTX TITAN X	32 GB	37.8
SCT [29]	i7-2600	GTX 650	16 GB	15.0
Proposed	i5-6400	N/A	16 GB	32.5
	Embedded CPU BCM2711	N/A	4 GB	17

4.4. Discussion

In terms of precision comparability, the PCGF average result is displayed in Table 1. Achieving 100 percent precision is difficult because the UAV123 [35] dataset has some limitations corresponding to object occlusion, boundary, visibilities, and so on. In addition, for the field which may request higher precision such as defense, the PCGF can further combine with the re-identification or recognition function to reach the higher standard. However, there is still the opportunity to continue further research on the following areas: (1) To reduce the impact of light effects on PCGF, an additional process needs to convert the pixels into HSV color gamut. (2) When the object color and its neighboring area are similar colors, then the object feature value similar characteristics. (3) The proposed technique incorporates several conditional clauses that cause the parallel process to accelerate at a slower rate. (4) When the UAV is close to the object, the number of PCGF increases, affecting the FPS result due to computational costs.

5. Conclusions

In conclusion, in the general method, a novel algorithm for the discriminative tracking system has been developed. The target object features have been separated by the PCGFs which represent the object and its location. The object location is predicted from the searching window and calculating the PCGF matching score. Thereby, object pixels and the background pixels distinguished successfully with the strong resistance of object deformation, rotation, distortion. The proposed technique considerably reduced the computational complexity and increased the performance which achieved real-time proceeding speed in the CPU platform without the GPU accelerator. Lastly, we implemented the proposed technique on Raspberry Pi 4 as a discriminative tracking system for UAV applications.

Author Contributions: Conceptualization, M.-H.S. and Y.-F.H.; methodology, M.-H.S., Y.-S.J., and Y.-F.H.; software, Y.-S.J. and Y.-F.H.; validation, Y.-S.J. and C.-C.S.; formal analysis, S.-C.L. and C.-C.S.; investigation, M.-H.S., S.M.S.M., C.-C.S. and S.-C.L.; resources, M.-H.S. and Y.-S.J.; data curation, Y.-S.J. and S.M.S.M.; writing—original draft preparation, Y.-S.J., S.M.S.M. and Y.-F.H.; writing—review and editing, M.-H.S., Y.-S.J. and S.M.S.M.; visualization: C.-C.S. and S.-C.L.; supervision: M.-H.S.; project administration, M.-H.S.; funding acquisition, M.-H.S. and S.-C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Ministry of Science & Technology, Taiwan under contract No. 109-2221-E-224-050.

Data Availability Statement: Publicly available datasets-UAV123 [35] were analyzed in this study. This data can be found here: https://cemse.kaust.edu.sa/ivul/uav123 (accessed on 8 July 2021).

Conflicts of Interest: The authors declare that no conflict of interest.

References

1. Barmpounakis, E.N.; Vlahogianni, E.I.; Golias, J.C. Unmanned Aerial Aircraft Systems for transportation engineering: Current practice and future challenges. *Int. J. Transp. Sci. Technol.* **2016**, *5*, 111–122. [CrossRef]
2. Chen, S.; Laefer, D.F.; Mangina, E.; Zolanvari, S.M.I.; Byrne, J. UAV bridge inspection through evaluated 3D reconstructions. *J. Bridge Eng.* **2019**, *24*, 5019001. [CrossRef]
3. Khan, M.A.; Ectors, W.; Bellemans, T.; Janssens, D.; Wets, G. Unmanned Aerial Vehicle-Based Traffic Analysis: A Case Study for Shockwave Identification and Flow Parameters Estimation at Signalized Intersections. *Remote Sens.* **2018**, *10*, 458. [CrossRef]
4. Meng, F.; Li, M.; Wang, J.; Zhang, L.; Zhong, T.; Cong, Q.; An, Y. The Research of Oil Pipeline Patrol by UAV in the First Sub-Factory of PCOC. In Proceedings of the 7th International Conference on Education, Management, Information and Mechanical Engineering (EMIM 2017), Shenyang, China, 28–30 April 2017; pp. 610–614.
5. Wang, W.; Huang, Y.; Wang, Y.; Wang, L. Generalized autoencoder: A neural network framework for dimensionality reduction. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 496–503.
6. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, New York, NY, USA, 5–9 July 2008; pp. 1096–1103.
7. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
8. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; Volume 4, pp. 3104–3112.
9. Luz, E.J.d.S.; Schwartz, W.R.; Cámara-Chávez, G.; Menotti, D. ECG-based heartbeat classification for arrhythmia detection: A survey. *Comput. Methods Programs Biomed.* **2016**, *127*, 144–164. [CrossRef] [PubMed]
10. Gulli, A.; Pal, S. *Deep Learning with Keras*; Packt Publishing Ltd: Birmingham, UK, 2017; ISBN 1787129039.
11. Wang, L.; Ouyang, W.; Wang, X.; Lu, H. Visual Tracking with Fully Convolutional Networks. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015; pp. 3119–3127.
12. Ma, C.; Huang, J.; Yang, X.; Yang, M. Hierarchical Convolutional Features for Visual Tracking. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015; pp. 3074–3082.
13. Zhang, K.; Liu, Q.; Wu, Y.; Yang, M. Robust Visual Tracking via Convolutional Networks Without Training. *IEEE Trans. Image Process.* **2016**, *25*, 1779–1792. [CrossRef] [PubMed]
14. Zhang, W.; Song, K.; Rong, X.; Li, Y. Coarse-to-Fine UAV Target Tracking With Deep Reinforcement Learning. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1522–1530. [CrossRef]
15. Zha, Y.; Wu, M.; Qiu, Z.; Sun, J.; Zhang, P.; Huang, W. Online semantic subspace learning with Siamese network for UAV tracking. *Remote Sens.* **2020**, *12*, 325. [CrossRef]
16. Qi, Y.; Zhang, S.; Qin, L.; Yao, H.; Huang, Q.; Lim, J.; Yang, M.H. Hedged Deep Tracking. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4303–4311.
17. Fu, C.; Zhang, Y.; Huang, Z.; Duan, R.; Xie, Z. Part-Based Background-Aware Tracking for UAV with Convolutional Features. *IEEE Access* **2019**, *7*, 79997–80010. [CrossRef]
18. Cui, Z.; Xiao, S.; Feng, J.; Yan, S. Recurrently target-attending tracking. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1449–1458.
19. Dai, K.; Zhang, Y.; Wang, D.; Li, J.; Lu, H.; Yang, X. High-Performance Long-Term Tracking with Meta-Updater. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 6297–6306.
20. Yun, S.; Choi, J.; Yoo, Y.; Yun, K.; Choi, J.Y. Action-decision networks for visual tracking with deep reinforcement learning. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1349–1358.
21. Teng, Z.; Xing, J.; Wang, Q.; Zhang, B.; Fan, J. Deep spatial and temporal network for robust visual object tracking. *IEEE Trans. Image Process.* **2020**, *29*, 1762–1775. [CrossRef] [PubMed]
22. Hong, S.; You, T.; Kwak, S.; Han, B. Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network. In Proceedings of the 32nd International Conference on International Conference on Machine, Lille, France, 7–9 July 2015; pp. 597–606.
23. Bolme, D.S.; Beveridge, J.R.; Draper, B.A.; Lui, Y.M. Visual object tracking using adaptive correlation filters. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 2544–2550.
24. Danelljan, M.; Häger, G.; Khan, F.S.; Felsberg, M. Learning Spatially Regularized Correlation Filters for Visual Tracking. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015; pp. 4310–4318.
25. Fu, C.; Lin, F.; Li, Y.; Chen, G. Correlation filter-based visual tracking for UAV with Online multi-feature learning. *Remote Sens.* **2019**, *11*, 549. [CrossRef]
26. Shi, W.; Wang, Y.; Wu, S. Robust UAV-Based tracking using hybrid classifiers. In Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017; pp. 2129–2137.

27. Silva, A.S.; Severgnini, F.M.Q.; Oliveira, M.L.; Mendes, V.M.S.; Peixoto, Z.M.A. Object Tracking by Color and Active Contour Models Segmentation. *IEEE Latin Am. Trans.* **2016**, *14*, 1488–1493. [CrossRef]
28. Bharati, S.P.; Wu, Y.; Sui, Y.; Padgett, C.; Wang, G. Real-Time Obstacle Detection and Tracking for Sense-and-Avoid Mechanism in UAVs. *IEEE Trans. Intell. Veh.* **2018**, *3*, 185–197. [CrossRef]
29. Choi, J.; Chang, H.J.; Jeong, J.; Demiris, Y.; Choi, J.Y. Visual Tracking Using Attention-Modulated Disintegration and Integration. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4321–4330.
30. Fu, C.; Xu, J.; Lin, F.; Guo, F.; Liu, T.; Zhang, Z. Object Saliency-Aware Dual Regularized Correlation Filter for Real-Time Aerial Tracking. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 8940–8951. [CrossRef]
31. Li, F.; Fu, C.; Lin, F.; Li, Y.; Lu, P. Training-Set Distillation for Real-Time UAV Object Tracking. In Proceedings of the IEEE International Conference on Robotics and Automation, Paris, France, 31 May–31 August 2020; pp. 9715–9721.
32. Yu, M.; Zhang, Y.; Li, Y.; Lin, Z.L.; Li, J.; Wang, C. Saliency guided visual tracking via correlation filter with log-gabor filter. *IEEE Access* **2020**, *8*, 158184–158196. [CrossRef]
33. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Stat. Soc. Ser. B* **1977**, *39*, 1–38.
34. Lin, H.H.; Chuang, J.H.; Liu, T.L. Regularized background adaptation: A novel learning rate control scheme for gaussian mixture modeling. *IEEE Trans. Image Process.* **2011**, *20*, 822–836. [PubMed]
35. Mueller, M.; Smith, N.; Ghanem, B. A Benchmark and Simulator for UAV Tracking. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 8–16 October 2016; pp. 445–461.
36. Wu, Y.; Lim, J.; Yang, M.H. Online object tracking: A benchmark. In Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2013; pp. 2411–2418.