

## Article

# SAFS: Object Tracking Algorithm Based on Self-Adaptive Feature Selection

Wenhua Guo <sup>1</sup>, Jiabao Gao <sup>2,\*</sup>, Yanbin Tian <sup>2</sup>, Fan Yu <sup>2</sup> and Zuren Feng <sup>1</sup>

<sup>1</sup> State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an 710049, China; markguo@xjtu.edu.cn (W.G.); fzr9910@xjtu.edu.cn (Z.F.)

<sup>2</sup> School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China; tianyanbin@stu.xjtu.edu.cn (Y.T.); fanyu.cs@stu.xjtu.edu.cn (F.Y.)

\* Correspondence: gaojiabao@stu.xjtu.edu.cn; Tel.: +86-150-3548-7489

**Abstract:** Object tracking is one of the most challenging problems in the field of computer vision. In challenging object tracking scenarios such as illumination variation, occlusion, motion blur and fast motion, existing algorithms can present decreased performances. To make better use of the various features of the image, we propose an object tracking method based on the self-adaptive feature selection (SAFS) algorithm, which can select the most distinguishable feature sub-template to guide the tracking task. The similarity of each feature sub-template can be calculated by the histogram of the features. Then, the distinguishability of the feature sub-template can be measured by their similarity matrix based on the maximum a posteriori (MAP). The selection task of the feature sub-template is transformed into the classification task between feature vectors by the above process and adopt modified Jeffreys' entropy as the discriminant metric for classification, which can complete the update of the sub-template. Experiments with the eight video sequences in the Visual Tracker Benchmark dataset evaluate the comprehensive performance of SAFS and compare them with five baselines. Experimental results demonstrate that SAFS can overcome the difficulties caused by scene changes and achieve robust object tracking.

**Keywords:** object tracking; self-adaptive feature selection; feature sub-template; maximum a posteriori



**Citation:** Guo, W.; Gao, J.; Tian, Y.; Yu, F.; Feng, Z. SAFS: Object Tracking Algorithm Based on Self-Adaptive Feature Selection. *Sensors* **2021**, *21*, 4030. <https://doi.org/10.3390/s21124030>

Academic Editors: Xue-Bo Jin and Yuan Gao

Received: 26 April 2021

Accepted: 7 June 2021

Published: 11 June 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

With the rapid development of hardware facilities and artificial intelligence technology, object tracking has become more important within the domain of computer vision. Object tracking has many applications in real life, such as video surveillance [1], athlete competition analysis, and intelligent human–computer interaction [2].

Traditional tracking algorithms can be divided into two types based on their appearance models: generative and discriminative trackers. Using the object's appearance information, generative trackers create an appearance model and search for the most similar candidate in each frame with the least amount of reconstruction error. Ross [3] demonstrates a monitoring approach that discovers a low-dimensional subspace representation incrementally and digitally updates it to respond to changes in the object's presence. In [4,5], the model of tracking is a sparse approximation problem in which a number of trivial templates are added to deal with difficult problems such as particle occlusion, scale variance, etc. Riahi D et al. [6] propose a novel generative approach for multiple object tracking using a multiple feature framework. Tkach A et al. [7] design a real-time generative object tracking based on an online optimization algorithm that jointly estimates pose and shape in each frame, and determines the uncertainty in such estimates. On the other hand, discriminative trackers locate the object through the learning of a decision that limits the object and the context. Helmut Grabner et al. [8] suggest an approach to pick features for monitoring objects for online boosting. In the CVPR 2013 tracking benchmark [9], Sam Hare et al. [10] use an online standardized output support vector machine (SVM) to map

objects. Kalal et al. [11] add a tracking re-detection module, which can reboot tracking if the object is totally overstepped or does not appear in the scene. The feature selection is a critical task in object tracking, so many researchers are trying to improve the self-adaptability of this module to overcome this challenge. Different correlation filters have recently been proposed to help traditional object tracking algorithms achieve robust performance and better results [12–14]. Furthermore, Zhang J et al. [15] suggest a robust target detection, Multi-task Correlation Particle Filter, for which the strengths of multi-task correlation filter (MCF) and a particle filter are used and improved. Perez-Cham O. E. et al. [16] apply the bio-inspired algorithms to object tracking, which is based on the parallelization of the Honeybee Search Algorithm, and employs the Zero Mean Normalized Cross-Correlation criterion as a fitness function. In addition, after considering the frequent occlusion of target objects in the complex scene, Bae S H et al. [17] propose a robust online multi-object tracking (MOT) approach. Chen Y et al. [18] design a novel Adaptive Focused Discriminative (AFOD) segmentation tracker with the following advanced components. Varfolomeiev A [19] proposes the improvements for visual object trackers based on discriminative correlation filters using channel-independent spatially regularized method for filter calculation. Unfortunately, although these strategies can solve this problem, they are often based on deep learning, which needs large-scale labeled data and computing resources. Tschannen M et al. [20] propose a general framework for self-supervised learning of transferable visual representations based on Video-Induced Visual Invariances (VIVI). Although some workers have been proposed to overcome those challenges, such as self-supervised, transfer learning or semi-supervised learning. However, those methods still require a deep neural network. The ever-increasing requirements of those models may raise the threshold for training them on the edge devices.

In this paper, we propose SAFS, an object tracking method based on a self-adaptive feature selection algorithm. Our method selects the suitable tracking feature by one new strategy. In this manner, we compare the similarity of each feature sub-template, and then measure the separability of different features of the object using the maximum a posteriori (MAP). Lastly, we adopt modified Jeffreys' entropy as the discriminant metric for this task. Through updating the feature sub-pattern, our model can update the target template and reduce the drift of the template. To evaluate the effectiveness of the algorithm, we implement our algorithm using color and texture features and apply it in eight different video sequences. Our proposed model can obtain better success rates and reduce the center location errors. Especially in the complex environment (such as basketball), the SAFS has improved above 90% performance than other baselines. In addition, these results demonstrate that our proposed algorithm can achieve higher robustness and faster inference speed.

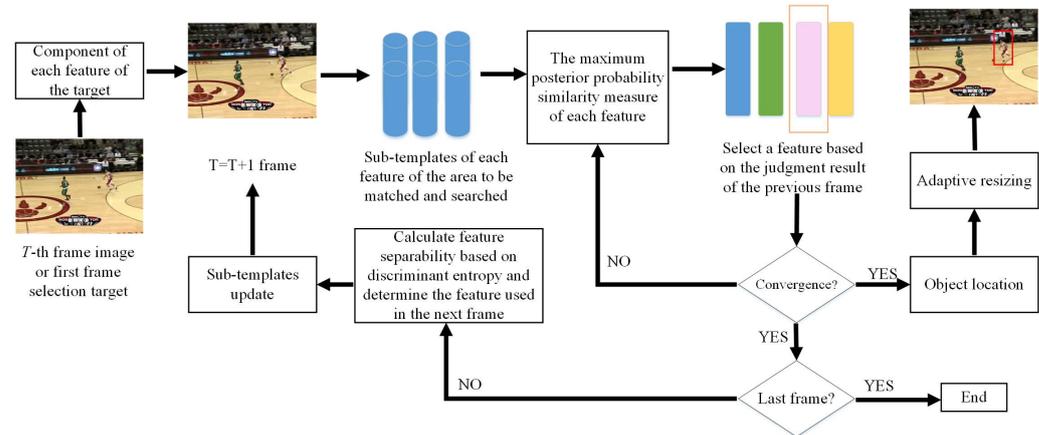
The main contributions of our paper can be described as follows:

- We characterize the sub-template of each feature and calculate the similarity value matrix of each sub-template based on the linear property of the maximum posterior probability [21].
- We optimize the calculation of Jeffreys' entropy [22] and use it as a metric of similarity value, which can measure the difference between categories more efficiently.
- We propose an object tracking method based on a self-adaptive feature selection algorithm, called SAFS, which can obtain better tracking performance without human labor.

## 2. Adaptive Multi-Feature Selection Tracking Algorithm

Feature selection of object tracking is a hard mission. We use different image features to calculate the similarity between the target area and the area to be matched as the criterion to select the most appropriate tracking feature. The adaptive feature selection mechanism can make full use of the adaptability of different features to different scenarios. When the environment changes, select the most distinguishable feature to track the target stably. This method also has good extensibility, in that it can combine various features in an

algorithm to track objects. Based on this idea, we designed the object tracking algorithm based on adaptive feature selection. The framework of the object tracking algorithm adopts the template matching method of generative class. According to the similarity value, it is necessary to measure the ability of each feature to distinguish the foreground and the background, so as to select the appropriate feature in the tracking of the next frame. The framework of the algorithm shows in Figure 1.



**Figure 1.** The framework of object tracking algorithm based on self-adaptive feature selection.

The framework mainly includes three steps:

- (1) Description of object features: calculate the description of each feature of the target area, the area to be matched, and the searching area, respectively. Including the feature description of the target in the first frame and the feature description of the searching area and the area to be matched in each subsequent frame.
- (2) Flexibly select the most proper feature: the optimal feature calculated in the previous frame is selected as the descriptive feature of this frame in the tracking process. Then, the object location is calculated iteratively according to the maximum posterior probability similarity criterion.
- (3) Selection for multi-feature: according to the central position of the object, divide the foreground and background of the searching area, and calculate the similarity of each feature. According to the similarity value map, the  $j$ -divergence entropy [22] is used to compute the ability of different features to distinguish the foreground and background. Select the most distinguishing feature according to the  $j$ -divergence entropy [22], and use this feature in the next frame tracking.

### 2.1. Multi-Feature Description and Similarity Measure of Sub-Template

This algorithm separately calculates the value of each feature operator of the target, constructs the sub-template of each feature of the target, and then combines the sub-templates of each feature to describe the target. In this way, two purposes can be achieved:

1. Decrease the feature vector dimension of multiple features;
2. Flexibly select the most proper feature.

Assume:

The template of target area:  $\{q_u^i\}_{i,\dots,n}$

The template of area to be matched:  $\{p_u^i\}_{i,\dots,n}$

The template of searching area:  $\{s_u^i\}_{i,\dots,n}$

In the above formula,  $n$  represents the feature vector dimension of one sub-template;  $q_u^i$  denotes the sub-template described by the  $i$ -th feature in target area;  $p_u^i$  denotes the sub-template described by the  $i$ -th feature in area to be matched;  $s_u^i$  represents the sub-template described by the  $i$ -th feature in searching area.

The features of each sub-template are described by feature statistical histogram. Using the tracking algorithm based on maximum posterior probability [21] similarity criterion to

calculate each sub-template's similarity contribution value  $g^i(x_j)$  and similarity function value  $\rho_{yj}$ :

$$\begin{aligned}\rho_{yj}^i &= \sum_{x_j \in A_{yj}} g^i(x_j) \\ g^i(x_j) &= \frac{q_u^i(x_j)}{s_u^i(x_j)}\end{aligned}\quad (1)$$

where  $\{x_j\}_{j=1, \dots, m}$  denote positions of each pixel in searching area;  $y_j$  represents the central point of this area;  $g^i(x_j)$  denotes the similarity contribution value of each pixel of the  $i$ -th feature sub-template in searching area;  $A_{yj}$  represents the aggregation of pixels centered in searching area of  $y_j$ ;  $\rho_{yj}^i$  denotes the similarity function value of the  $i$ -th sub-template in searching area.

Based on the sub-templates described by each feature of target area, area to be matched and searching area, we can figure out the feature map corresponding to the similarity contribution value of each sub-template. According to the linear property of the maximum posterior probability [21], the similarity value matrix of each sub-template can be calculated.

Different sub-templates have different similarity value matrices. Some features have a stronger distinguishing ability, so their similarity value matrix has a higher peak in the target area and the value in the edge part far away from the target in the searching area is lower. The closer the matrix is to the target center, the higher the similarity value. Some features are weak in distinguishing ability, so the values of the edge parts away from target have only a small difference between the target area and searching area of their similarity value matrix. Once we design one criterion to measure the difference between target position and edge part of searching area in the similarity value matrix of each sub-template, we can describe the distinguishing ability of sub-template in this frame, and use it to instruct the tracking of the next frame.

## 2.2. Description of Feature Selection

Obtaining the prior information of the scene is a hard mission; even if the prior information of the scene can be obtained by the machine learning method, the information is still uncertain as the environment changes. Therefore, if the tracking algorithm relies on a feature for long-term continuous tracking, it will lose the object easily. One way to solve this problem is to determine the distinguishing ability of each feature of target in current scene in real-time. The discriminant tracking algorithm is to deal with the tracking task along this way: regard tracking as a two-category task of foreground and background, then design a classifier and find the optimal classification result to obtain the object. We can design a new tracking algorithm based on this idea. Assuming that a feature takes into account both the accuracy of the feature's description and the separability of foreground and background, this feature will show good tracking results during the object tracking process. Therefore, we can determine the tracking features of the current frame by designing an algorithm to compute the separability of different features of the object in the scene.

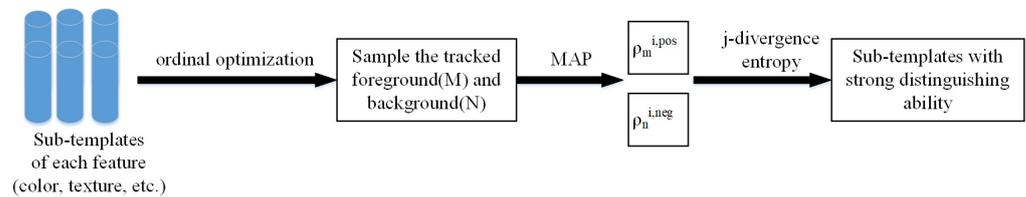
By sampling the tracked foreground and background separately, and then using mathematical tools to calculate the distance between the two classes, the distinguishability of different features can be measured. The sampling process can uniformly sample  $M$  regions with the same target size as positive examples in the target center according to the principle of the ordinal optimization, and uniformly sample  $N$  regions with the same target size as negative examples around the background of the target. The bigger the difference is between positive examples and negative examples, the better the distinguishing ability is of the sub-template. If more positive examples and negative examples are chosen, the separability measure of the judging feature will be more accurate. However, if too many examples are chosen, it will bring the problem of increasing computation complexity. In this paper, the maximum posterior probability similarity criterion is used to measure the similarity between the target and the positive example and the similarity between

the target and the negative example. Therefore, the linear computation property of the maximum posterior probability can be used to quickly calculate the similarity value of the feature.

Assume the similarity between the target and positive example:  $\rho_m^{i,pos}$ , where  $i$  denotes the  $i$ -th sub-template and  $m$  denotes the similarity of the  $m$ -th positive example.

Assume the similarity between the target and negative example:  $\rho_n^{i,neg}$ , where  $i$  represents the  $i$ -th sub-template,  $n$  represents the similarity of the  $n$ -th negative example.

Now, the selection task of sub-templates is turned into a classification task of two sets of feature vectors: a set of similarity values  $\rho_m^{i,pos}$  between positive examples and the target, and a set of similarity values  $\rho_n^{i,neg}$  between negative examples and the target. There are many criteria for discriminating metrics: distance metric, fisher discriminant criteria [23], etc. We select the relative entropy [24] and j-divergence entropy [22] as the discriminating metric of classification. The selection process of feature sub-template of next frame is shown in Figure 2.



**Figure 2.** The selection process of feature sub-template.

### 2.3. Introduction of Relative Entropy and j-Divergence Entropy

Relative entropy [24] is a measure of the asymmetry of the difference between two probability distributions  $P$  and  $Q$ . The physical meaning of relative entropy is the metric of the number of extra bits required to encode samples from  $P$  using  $Q$ -based coding. Normally,  $P$  represents the actual distribution of the data,  $Q$  represents the theoretical distribution of the data, the template distribution, or the approximate distribution of  $P$ . It has four characteristics:

1. It can be used to measure the similarity between two functions whose values are positive.
2. For two functions that are entirely the same, their relative entropy is zero. The bigger the difference is, the bigger the relative entropy is.
3. If the values of two randomly distributed probability density functions are greater than zero, the relative entropy can measure the difference between the two random distributions.
4. Relative entropy is not symmetric, and commutative law does not work.

We consider two situations in our paper. Assume that two kinds of feature vector are  $P = \{p_i\}_{i=1}^n$  and  $Q = \{q_i\}_{i=1}^n$ , and  $\sum p_i = \sum q_i = 1$  is satisfied. The relative entropy  $D(P, Q)$  of these two feature vectors is defined as:

$$D(P, Q) = \sum_{i=1}^n p_i \log \frac{p_i}{q_i} \quad (2)$$

where  $n$  is the dimension of the feature.

Obviously,  $D(P, Q) \geq 0$ .  $D(P, Q)$  is equal to 0 if and only if  $P$  equals to  $Q$ . In other words, the bigger the difference is between these two kinds of random distribution, the bigger the relative entropy is between them. When relative entropy equals zero, they have the same distribution.

From the formula of relative entropy, we can know relative entropy is asymmetric. To solve the asymmetry, the discriminant entropy  $J(P, Q)$  (also called Jeffreys' entropy [22])

of two types of probability distributions can be defined to characterize the difference between the two types of distributions.

$$J(P, Q) = D(P, Q) + D(Q, P) \quad (3)$$

Because the calculation of the logarithm is complicated, for the convenience of calculation, the following Formula (4) is used as the approximate measure of  $J(P, Q)$ .

$$U(P, Q) = \|P - Q\|^2 = \sum_{i=1}^n (p_i - q_i)^2 \quad (4)$$

$$\begin{cases} p_i = \frac{1}{N_1} \sum_{k=1}^{N_1} (s_{k_i}^{(1)})^2 \\ q_i = \frac{1}{N_2} \sum_{k=1}^{N_2} (s_{k_i}^{(2)})^2 \end{cases}$$

where  $N_1$  and  $N_2$  denote the number of samples of class 1 and class 2, respectively;  $i$  stands for feature number;  $s_{k_i}^{(1)}$  denotes the  $i$ -th feature of  $k$ -th sample in class 1;  $s_{k_i}^{(2)}$  represents the  $i$ -th feature of  $k$ -th sample in class 2. The above features must be normalized.

#### 2.4. The Discriminant Algorithm for Distinguishing Ability of Multi-Features

Assume that  $\{q_u^i\}_{i,\dots,n}$  is a sub-template described by the  $i$ -th feature in the target area. The similarity value between the positive example and the target is  $\rho_m^{i,pos}$ , which represents the similarity value of the  $m$ -th positive example of the  $i$ -th feature point template, where  $m \in M$ . The similarity value between the negative example and the target is  $\rho_n^{i,neg}$ , which denotes the similarity value of the  $m$ -th negative example of the  $i$ -th feature point template, where  $n \in N$ . According to the problem description and the calculation method of discriminant entropy described above, the separability metric  $J^i$  of each feature sub-template after normalization can be designed as:

$$J^i = \sum_{i=1}^k (\bar{\rho}^{i,pos} - \bar{\rho}^{i,neg})^2 \quad (5)$$

$$\begin{cases} \bar{\rho}^{i,pos} = \frac{1}{M} \sum_{k=1}^M (\rho_k^{i,pos})^2 \\ \bar{\rho}^{i,neg} = \frac{1}{N} \sum_{k=1}^N (\rho_k^{i,neg})^2 \end{cases}$$

where  $k$  denotes feature vector dimension of a sub-template;  $\rho_k^{i,pos}$  denotes the similarity between the  $k$ -th positive example of the  $i$ -th feature sub-template and the target;  $\rho_k^{i,neg}$  represents the similarity between the  $k$ -th negative example of the  $i$ -th feature sub-template and the target;  $\bar{\rho}^{i,pos}$  denotes the mean squared of similarity of the positive example of the  $i$ -th feature sub-template;  $\bar{\rho}^{i,neg}$  denotes the mean squared of similarity of the negative example of the  $i$ -th feature sub-template. All features are normalized.

According to Formula (5),  $J^i$  measures the separability ability of the  $i$ -th feature sub-template. When  $J^i$  is larger, the separability of the feature template is better. Therefore, we can find the best sub-template by calculating  $\max(J^i)$ . According to the Winner-Takes-All criterion, when the sub-template of a feature has the best distinguishing ability, the next frame will use this feature to track. By calculating the value of this judge criterion, it can adaptively select the best feature, thereby achieving robust tracking in transform scenarios.

#### 2.5. Selectively Update for Sub-Template

In the object tracking process, especially in the long-term object tracking, the update of template is a necessary step. Through the update of template, the tracking algorithm can better adapt to the changes of the environment and the object itself. Through the target update, the target template can be adjusted to achieve long-term tracking. However, there is a problem in the current target update: the drift of the template. In the paper [25], the target template uses the following two methods:

1. Adaptively update target size based on the characteristics of the maximum posterior probability similarity criterion;
2. Two thresholds are set to control the number and degree of template updating.

The above methods can be applied to a single feature target template or a multi-feature template in a high dimensional space. For the tracking algorithm based on adaptive feature selection in this paper, the complementarity between target feature sub-templates can be used to further reduce template drift. For example, tracking objects with color and texture features. If the object enters a bright environment from a dark environment, the color feature will change significantly due to illumination. The nature of the texture feature determines that it will not be affected by illumination. At this time, under the tracking framework based on adaptive feature selection, the texture feature will be used for image tracking during the environment switching, and the color feature will be used for tracking in the stable lighting environment.

Therefore, this paper proposes a method for selective updating of sub-template: in each frame, the tracking stable features are not updated, only the sub-templates with poor separability are updated. With this updated strategy, we can prevent the target template from introducing background information due to incorrect updates, thereby gaining adaptability to changes in the scene or object. The algorithm first sorts the feature sub-templates according to the discriminant entropy value; secondly, sets an update threshold  $\lambda$ , the feature sub-templates larger than this threshold are not updated, the feature sub-templates smaller than the threshold are updated.

In the specific algorithm, the two target update methods in paper [25] can be used to enhance the validity of the template update. In the tracking algorithm, the first method described above is used to adaptively adjust the target size. When determining the updating feature sub-template, a second method is used to control the speed of the update and the degree of the update. Through the above three methods, the algorithm can be adaptive to the target change.

### 2.6. Steps of Adaptive Feature Selection Algorithm

The similarity of the positive example, negative example, and target area of each feature sub-template can be calculated by simple linear summation. In this way, more positive and negative examples can be selected to better determine the distinguishing ability of the feature sub-template.

The process of adaptive selection algorithm based on color and texture features is as follows. First, initialize the various parameters of the algorithm. In the first frame, the color feature of target is described as  $q_u^1$ , and the texture feature is described as  $q_u^2$ .  $y_i$  represents the target center of the initial position.  $\{x_i\}_{i=1,\dots,m}$  denote the positions of each pixel in searching area in the first frame. Since there is no prior information in the first frame, either color or texture feature can be selected to start tracking. We select the color feature to start the tracking of the first frame.

1. Set the target position  $y_i$  of the last frame as initial position. Calculate the color feature description  $s_u^1$  and texture feature description  $s_u^2$  in searching area centered on the initial position;
2. According to the formula  $g(x_i) = \frac{q_u(x_i)}{s_u(x_i)}$ , calculate the similarity contribution  $g^1(x_i)$  and  $g^2(x_i)$  of each pixel in the searching area in current frame. According to the separability result of the previous frame, the similarity contribution value of the corresponding feature is selected for tracking;
3. Initialize  $k = 0$  which represents the number of initial iterations;
4. Calculate the position of the center to be matched of the next iteration by Formula (6). At the same time, set  $k = k + 1$ .

$$y_{i+1} = \frac{\sum_{x_i \in A_{y_i}} x_i g(x_i)}{\sum_{x_i \in A_{y_i}} g(x_i)} \quad (6)$$

where  $\{x_i\}_{i=1, \dots, m}$  is the position of each pixel in the searching area;  $y_i$  is the center of the area;  $g(x_i)$  is the similarity contribution value of each pixel in the searching area;  $A_{y_i}$  is the set of pixels representing the search area centered at  $y_i$ ;  $y_{i+1}$  is the target position of the next frame.

5. Repeat step 4 until  $\|y_{i+1} - y_i\| < \varepsilon$  or  $k \geq N$ ;
6. Calculate the distinguishing abilities of color and texture features in current scene by Formula (5);
7. According to Formula (7) [25], calculate whether the target size changes adaptively;

$$\omega(k+1) = \begin{cases} \omega(k+2) + 2, & \text{if } \bar{\phi}_{-1} > 0.8 \text{ and } \bar{\phi}_0 > 0.75 \text{ and } \bar{\phi}_1 > 0.6 \\ \omega(k-2) - 2, & \text{if } \bar{\phi}_0 < 0.6 \text{ and } \bar{\phi}_1 < 0.3 \\ \omega(k), & \text{otherwise} \end{cases} \quad (7)$$

where  $k$  is the number of frames of the image;  $\omega(k)$  represents the size of the target template in the  $k$ -th frame;  $\bar{\phi}_i (i = 1, \dots, a)$  is the average of the posterior probability measure (PPM) of each pixel;  $a$  is the comparison step of scale adaptation and is set to 1 without losing the generality.

8. According to Formula (8) [25], determine whether to update the target.

$$\bar{\phi}(p, q) \geq \delta \quad (8)$$

where  $p$  represents the current frame template;  $q$  denotes the target template;  $\bar{\phi}(p, q)$  is the similarity of the posterior probability measure for the current frame and the target template;  $\delta$  is the threshold parameter. If the sub-template needs to be updated, update the vector of the target feature sub-template according to Formula (9) [25]:

$$q' = \gamma p + (1 - \gamma)q \quad (9)$$

where  $\gamma$  is the updated weighted value;  $p$  is the target feature vector of the current frame;  $q$  is the feature vector of the target template;  $q'$  is the updated target template feature vector;

9. Import the next picture in the sequence and jump to step 1.

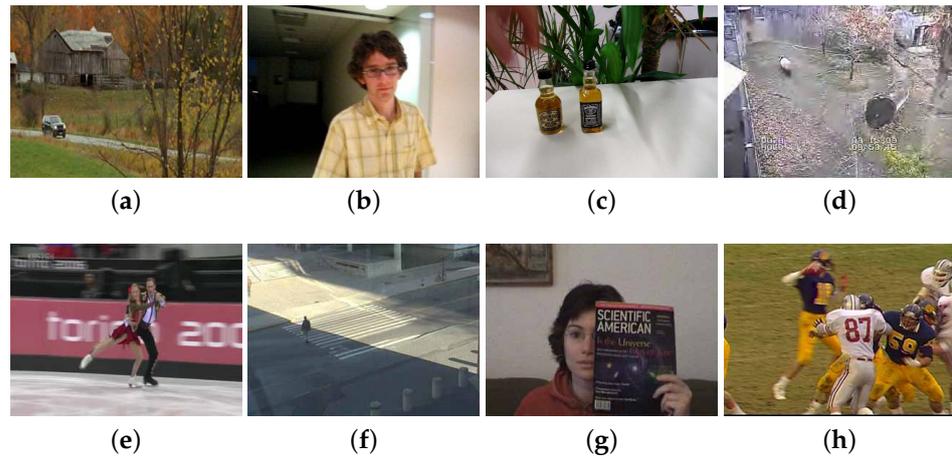
During the iterative process of the entire algorithm, the characterization of the search area is calculated only once. In the actual algorithm process, only step 4 and step 5 are iterated.

### 3. Experiments

In this section, we first introduce the experimental datasets and evaluation metrics. Then, we describe the process and details of the experiment. The results of ablation and contrast experiment are shown in the third part. Finally, we analyze the process and speed of the experiment.

#### 3.1. Experimental Dataset and Evaluation Metrics

In the experiment, we select eight video sequences from the Visual Tracker Benchmark (VTB) dataset [9] ([http://cvlab.hanyang.ac.kr/tracker\\_benchmark/datasets.html](http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html), accessed on 7 June 2021), which is shown in Figure 3. These sequences have attributes such as scale variation, occlusion, deformation, and out-of-plane rotation, which are all representative in object tracking tasks.



**Figure 3.** Eight test sequences used in experiment. (a) CarScale; (b) David; (c) Liquor; (d) Panda; (e) Skating2; (f) Crossing; (g) FaceOcc1; (h) Football1. These eight sequences possess the nine attributes described in the VTB dataset (SV, OCC, FM, etc.). For visual effects, some sequences have been stretched to achieve the same size.

To illustrate the effectiveness of this algorithm, we adopt two evaluation metrics to measure the results of the experiment in this paper. The former is the center location error (CLE) [26,27], which calculates the Euclidean distance of the center of the tracking box. It is defined as:

$$\text{CLE} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (10)$$

where  $(x_1, y_1)$  represents the center position of the ground truth label in a certain frame.  $(x_2, y_2)$  represents the center position of the tracking result label in a certain frame.

Another evaluation metric is the success rate (SR), which involves a concept commonly used in object tracking—Intersection over Union (*IoU*). It calculates the overlap ratio between the generated candidate box and the ground truth box. The formula of *IoU* is as follows:

$$\text{IoU} = \frac{\text{area}(C) \cap \text{area}(G)}{\text{area}(C) \cup \text{area}(G)} \quad (11)$$

where  $\text{area}(C)$  represents the area of the generated candidate box,  $\text{area}(G)$  denotes the area of the ground truth box. The ideal situation is complete overlap, which means *IoU* equals 1. The threshold is set to 0.5. It is considered to have successfully tracked the object when the *IoU* of a frame reaches 0.5 or more. The SR is equal to the number of frames successfully tracked divided by the number of all frames.

### 3.2. Implementation Description

Our proposed SAFS algorithm can add multiple features for tracking. The color and texture features of the target are used to design the corresponding tracking algorithm. Readers can also add features such as gradients and edges to the algorithm.

Color information is an important global feature that is robust to target rotation, partial occlusion, and non-rigid transformation when color feature histograms are used to represent color information. In this algorithm, the feature histogram of the RGB color space is used to make the sub-template of the target color feature, and the RGB feature can be quantized into a feature vector of  $8 \times 8 \times 8 = 512$  dimensions.

In this paper, texture feature is selected as another feature sub-template, which is also a global feature. The original LBP feature operator [25] is chosen to describe the shape information of the object. For the original LBP feature operator with the radius of 1 and sampling points of 8, the 58-dimensional feature vector is obtained by rotation and equivalent transformation. The statistical histogram of LBP feature is also used to describe

the texture feature sub-template. In the experiment, the tracking feature of the first frame selects the color feature. Table 1 lists the environment information used in our experiment.

**Table 1.** The experimental environment information.

| System    | CPU                         | Frequency | RAM  | Software      |
|-----------|-----------------------------|-----------|------|---------------|
| Windows10 | Intel (R) Core (TM) i7-7700 | 3.60 GHz  | 16.0 | MATLAB R2019b |

### 3.3. Comparison Experiment

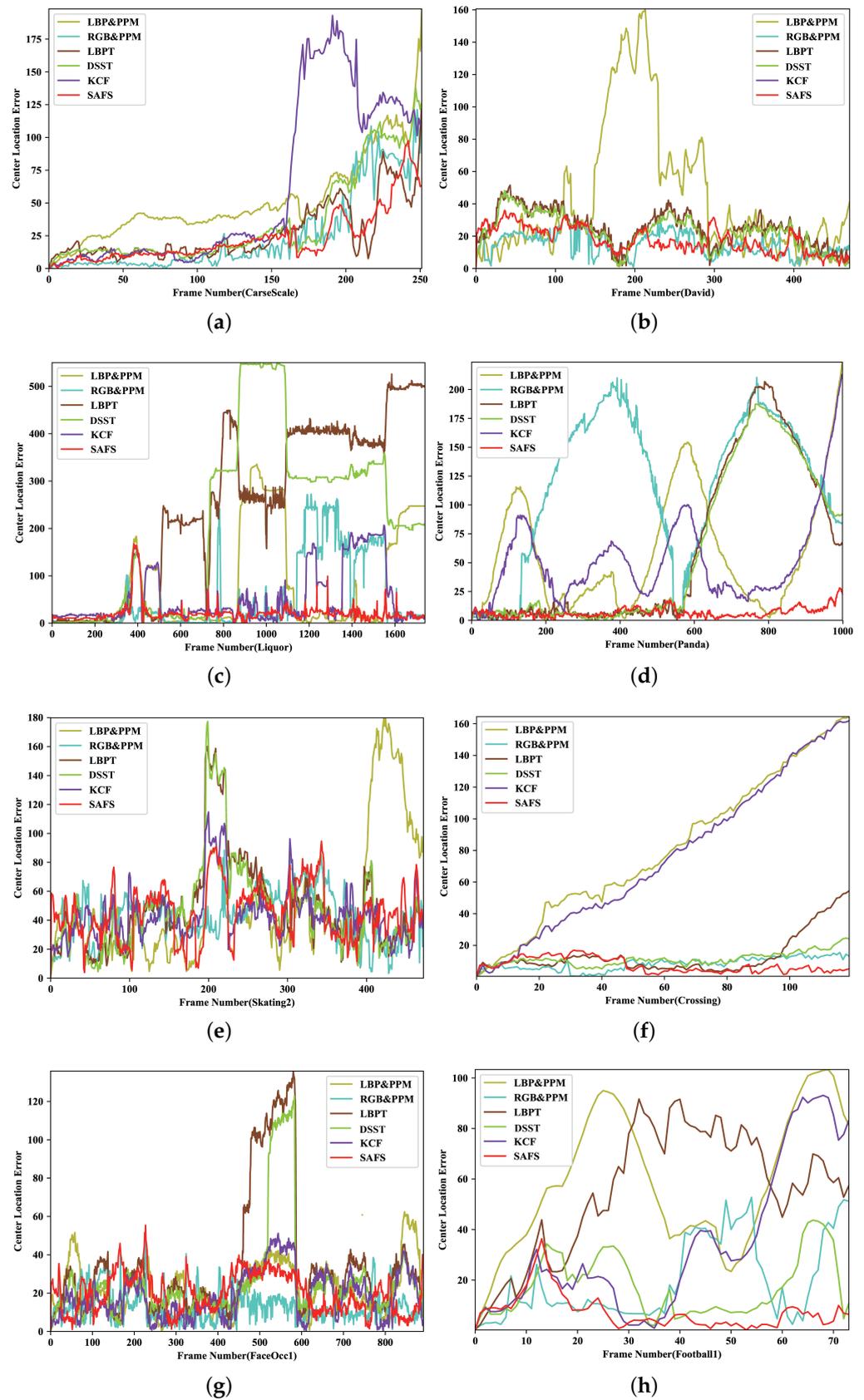
Our algorithm can adaptively select features between color and texture. In order to show the advantages of our self-adaptive algorithm, we select two algorithms that track only one feature and use them as baselines for comparison. We compare SAFS algorithm with the maximum posterior probability iterative tracking algorithm using texture feature alone (LBP&PPM) [21] and color feature alone (RGB&PPM) [21]. In addition, considering that SAFS is not based on deep learning, we should compare it with some classic algorithms. Therefore, we add three state-of-the-art traditional algorithms in the comparison, which are the mean-shift algorithm using the joint color-texture histogram (LBPT) [28], discriminative scale space tracking (DSST) [13] and high-speed tracking with kernelized correlation filters (KCF) [29]. All the algorithms are designed to track one object in our experiments.

From Table 2, we can easily see that our SAFS has the highest average SR. The DSST [13] gets 100 in “David” sequence, the RGB&PPM [21] gets 95 in “Skating2” sequence and 100 in “Crossing” sequences, which obtain better SR than SAFS. Meanwhile, SAFS achieves the best results in five sequences.

**Table 2.** The SR (%) of the SAFS method compared with other five methods.

| Video Sequence | LBP&PPM [21] | RGB&PPM [21] | LBPT [28] | DSST [13] | KCF [29] | SAFS |
|----------------|--------------|--------------|-----------|-----------|----------|------|
| CarScale       | 89           | 100          | 100       | 85        | 88       | 100  |
| David          | 71           | 99           | 96        | 100       | 99       | 99   |
| Liquor         | 68           | 75           | 28        | 39        | 75       | 98   |
| Panda          | 27           | 16           | 58        | 57        | 18       | 100  |
| Skating2       | 71           | 95           | 90        | 83        | 92       | 91   |
| Crossing       | 11           | 100          | 80        | 97        | 12       | 96   |
| FaceOcc1       | 100          | 100          | 96        | 91        | 100      | 100  |
| Football1      | 24           | 76           | 24        | 90        | 61       | 95   |
| Average SR     | 57.6         | 82.6         | 71.5      | 80.3      | 92.5     | 97.3 |

Table 3 lists the center location errors of 8 sequences using 6 methods, respectively. Our SAFS algorithm achieves the lowest average center location error which is 18 pixels. In addition, we display the results of the center-pixel location error frame by frame for eight sequences, as shown in Figure 4. The red line representing SAFS is always at the bottom of the figure, which means that its average CLE is minimal. Experimental results show that the algorithm can achieve robust tracking of the object when the object or scene changes. It should be pointed out that in the process of updating the target template, template drift still exists, which affects the tracking effect to a certain extent.



**Figure 4.** The experimental results of comparing the CLE (in pixel) frame by frame in eight sequences. Our SAFS algorithm can track the object robustly and accurately. (a) CarScale; (b) David; (c) Liquor; (d) Panda; (e) Skating2; (f) Crossing; (g) FaceOcc1; (h) Football1.

**Table 3.** The CLE (pixels) of the SAFS method compared with other five methods.

| Video Sequence | LBP&PPM [21] | RGB&PPM [21] | LBPT [28] | DSST [13] | KCF [29] | SAFS |
|----------------|--------------|--------------|-----------|-----------|----------|------|
| CarScale       | 51           | 25           | 27        | 33        | 59       | 23   |
| David          | 58           | 15           | 25        | 21        | 18       | 7    |
| Liquor         | 76           | 54           | 251       | 205       | 53       | 21   |
| Panda          | 27           | 117          | 60        | 57        | 52       | 7    |
| Skating2       | 52           | 42           | 48        | 45        | 43       | 45   |
| Crossing       | 81           | 8            | 14        | 9         | 76       | 8    |
| FaceOcc1       | 22           | 13           | 34        | 24        | 19       | 20   |
| Football1      | 59           | 19           | 55        | 17        | 31       | 8    |
| Average CLE    | 53           | 37           | 64        | 53        | 44       | 18   |

### 3.4. Tracking Process Analysis

To illustrate the difference of the algorithm, we analyze the tracking process of four sequences. Figure 5 lists the tracking conditions of the four fundamental frames in each sequence. “CarScale” is a sequence with scale variation, occlusion, and fast motion. “David” is a sequence with illumination variation, deformation, and motion blur. “Panda” is a sequence with scale variation, occlusion, and low resolution. “FaceOcc1” is a sequence with occlusion. The process analysis of the experimental results is as follows:

- (1) CarScale: The most significant attribute of this sequence is scale variation. The scale of the car is increasing as it comes fast from afar. In the first 155 frames, all algorithms can track the object correctly. Starting from frame 165, KCF gradually loses the target and re-tracks the target at frame 201 the end because of the occlusion of the tree during these frames, which means KCF cannot handle occlusion issues well. From the overall point of view, LBPT and SAFS show better performance. Although most of the algorithms can track the target well, the tracking boxes cannot increase with the scale of the target. This is a point that our algorithm needs to pay attention to and improve.
- (2) David: At first, the environment is dark. LBPT, DSST, KCF and SAFS get better results. From frame 162, LBP&PPM loses the object because the light is too dim, resulting in sparse texture feature. RGB&PPM loses the target between frame 451 and frame 536 because of the confusing effect of the poster on the wall. During the tracking process, although the LBPT has tracked the object, some offset has occurred. The reason for the offset is that when the color sub-template was updated in the previous tracking process, a template drift occurred, which resulted in a matching shift. The performance of LBPT shows that the combined feature of color and texture can simultaneously show the characteristics of different features in the tracking, but this type of feature will interfere with each other, resulting in inaccurate positioning of the target. DSST, KCF and SAFS can always track the object well.
- (3) Panda: This is a long sequence with 1000 frames. LBP&PPM loses the target since frame 51 and RGP&PPM loses the target since frame 138. This indicates that a single feature cannot track the target well. LBPT and DSST lose the target when the panda passes the brand for the second time. KCF losses the target several times due to the occlusion of trees or interference of similar objects. Our SAFS algorithm is very robust, so it can track the target very well throughout the process.
- (4) FaceOcc1: This is a relatively simple target tracking sequence because it only has one attribute—occlusion. From frame 523, LBPT and DSST lose the tracking of the face, which indicate that they cannot deal with the object occlusion well. Other algorithms show good performance in this sequence.



**Figure 5.** Experiment results of some fundamental frames of six algorithms on four sequences (CarScale, David, Panda, FaccOcc1).

### 3.5. Speed Analysis

Table 4 lists the tracking frames per second (fps) and average fps of the six algorithms in this experiment. LBP&PPM [21] and RGB&PPM [21] both get high average fps in that they all use a single feature. The speeds of LBPT [28] and KCF [29] are 425 fps and 314 fps, respectively, which are slower than ours. DSST [13] is a low-speed algorithm, which gets the slowest average speed 85. Our SAFS gets a medium speed (467). Its main time-consuming steps are in the description of features and the selection of sub-templates. Therefore, we can consider appropriately reducing the calculation of positive and negative examples and narrowing the search area to increase the tracking speed. Furthermore, if more features are introduced for object tracking, the speed of the algorithm will be closely related to the types of features and the dimensions of feature vectors.

**Table 4.** Speed comparison of six algorithms.

| Sequence    | LBP&PPM [21] | RGB&PPM [21] | LBPT [28] | DSST [13] | KCF [29] | SAFS |
|-------------|--------------|--------------|-----------|-----------|----------|------|
| CarScale    | 1139         | 2357         | 628       | 132       | 464      | 732  |
| David       | 297          | 734          | 123       | 27        | 124      | 259  |
| Liquor      | 120          | 234          | 97        | 89        | 312      | 60   |
| Panda       | 1702         | 3220         | 1294      | 203       | 721      | 1061 |
| Skating2    | 99           | 290          | 48        | 11        | 41       | 63   |
| Crossing    | 1251         | 1561         | 793       | 128       | 523      | 1029 |
| FaceOcc1    | 136          | 197          | 94        | 9         | 37       | 50   |
| Football1   | 897          | 1147         | 324       | 78        | 287      | 486  |
| Average FPS | 705          | 1217         | 425       | 85        | 314      | 467  |

#### 4. Conclusions

In this paper, we propose an object tracking method based on a new self-adaptive feature selection algorithm, called SAFS. This method can overcome the challenge of robustness by selecting the most suitable feature, which is a significant problem in the image domain. Furthermore, our proposed method can achieve faster inference speed, which is the inevitable factor in edge application.

To evaluate the performance of the SAFS, we test it on eight different video sequences from comprehensive perspectives. Our proposed method can obtain a higher success rate and shrink the center location errors in various degrees, and even outperform 90% compared to other baselines in some specific scenes. Future work will be dedicated to decreasing the complexity of the algorithm and further improving the performance.

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

**Funding:** This research was funded by Natural Science Foundation of Shaanxi Province grant number 2019JQ-601.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

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

#### References

- Mangawati, A.; Leesan, M.; Aradhya, H.V.R. Object Tracking Algorithms for video surveillance applications. In Proceedings of the International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 3–5 April 2018; pp. 667–671.
- Verma, R. A review of object detection and tracking methods. *Int. J. Adv. Eng. Res. Dev.* **2017**, *4*, 569–578.
- Ross, D.A.; Lim, J.; Lin, R.S.; Yang, M.H. Incremental Learning for Robust Visual Tracking. *Int. J. Comput. Vis.* **2008**, *77*, 125–141. [[CrossRef](#)]
- Mei, X.; Ling, H. Robust visual tracking using L1 minimization. In Proceedings of the IEEE International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 1436–1443.
- Yu, K.; Lin, Y.; Lafferty, J. Learning image representations from the pixel level via hierarchical sparse coding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011; pp. 1713–1720.
- Riahi, D.; Bilodeau, G.A. Multiple object tracking based on sparse generative appearance modeling. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 27–30 September 2015; pp. 4017–4021.
- Tkach, A.; Tagliasacchi, A.; Remelli, E.; Pauly, M.; Fitzgibbon, A. Online generative model personalization for hand tracking. *ACM Trans. Graph. (ToG)* **2017**, *36*, 1–11. [[CrossRef](#)]
- Grabner, H.; Grabner, M.; Bischof, H. Real-Time Tracking via On-line Boosting. *Br. Mach. Vis. Conf.* **2006**, *1*, 47–56.
- Wu, Y.; Lim, J.; Yang, M.H. Online object tracking. A benchmark. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 2411–2418.
- Hare, S.; Golodetz, S.; Saffari, A.; Vineet, V.; Cheng, M.M.; Hicks, S.L.; Torr, P.H. Struck: Structured output tracking with kernels. In Proceedings of the IEEE International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 263–270.
- Kalal, Z.; Matas, J.; Mikolajczyk, K. Pn learning: Boot-strapping binary classifiers by structural constraints. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 49–56.
- Bolme, D.S.; Beveridge, J.R.; Draper, B.A.; Lui, Y.M. Visual object tracking using adaptive correlation filters. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 2544–2550.
- Danelljan, M.; Häger, G.; Khan, F.; Felsberg, M. Accurate scale estimation for robust visual tracking. In Proceedings of the British Machine Vision Conference, Nottingham, UK, 1–5 September 2014.
- Zhang, J.; Jin, X.; Sun, J.; Wang, J.; Sangaiah, A.K. Spatial and semantic convolutional features for robust visual object tracking. *Multimed. Tools Appl.* **2020**, *79*, 15095–15115. [[CrossRef](#)]
- Zhang, T.; Xu, C.; Yang, M.H. Multi-task correlation particle filter for robust object tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4335–4343.
- Perez-Cham, O.E.; Puente, C.; Soubervielle-Montalvo, C.; Olague, G.; Aguirre-Salado, C.A.; Nuñez-Varela, A.S. Parallelization of the honeybee search algorithm for object tracking. *Appl. Sci.* **2020**, *10*, 2122. [[CrossRef](#)]

17. Bae, S.H.; Yoon, K.J. Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 595–610. [[CrossRef](#)] [[PubMed](#)]
18. Chen, Y.; Xu, J.; Yu, J.; Wang, Q.; Yoo, B.; Han, J.J. AFOD: Adaptive Focused Discriminative Segmentation Tracker. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2020; pp. 666–682.
19. Varfolomeiev, A. Channel-independent spatially regularized discriminative correlation filter for visual object tracking. *J. Real-Time Image Process.* **2021**, *18*, 233–243. [[CrossRef](#)]
20. Tschannen, M.; Djolonga, J.; Ritter, M.; Mahendran, A.; Houlsby, N.; Gelly, S.; Lucic, M. Self-supervised learning of video-induced visual invariances. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 13806–13815.
21. Feng, Z.; Lu, N.; Jiang, P. Posterior probability measure for image matching. *Pattern Recognit.* **2008**, *41*, 2422–2433. [[CrossRef](#)]
22. Clarke, B.S.; Barron, A.R. Jeffreys' prior is asymptotically least favorable under entropy risk. *J. Stat. Plan. Inference* **1994**, *41*, 37–60. [[CrossRef](#)]
23. Mika, S.; Ratsch, G.; Weston, J.; Scholkopf, B.; Mullers, K.R. Fisher discriminant analysis with kernels. In Proceedings of the Neural Networks for Signal Processing IX: 1999 IEEE Signal Processing Society Workshop, Madison, WI, USA, 25 August 1999; pp. 41–48.
24. Joe, H. Relative entropy measures of multivariate dependence. *J. Am. Stat. Assoc.* **1989**, *84*, 157–164. [[CrossRef](#)]
25. Guo, W.; Feng, Z.; Ren, X. Object tracking using local multiple features and a posterior probability measure. *Sensors* **2017**, *17*, 739. [[CrossRef](#)] [[PubMed](#)]
26. Kim, H.U.; Lee, D.Y.; Sim, J.Y.; Kim, C.S. Sowp: Spatially ordered and weighted patch descriptor for visual tracking. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 3011–3019.
27. Joukhadar, A.; Scheuer, A.; Laugier, C. Fast contact detection between moving deformable polyhedra. In Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyongju, Korea, 17–21 October 1999; pp. 1810–1815.
28. Ning, J.; Zhang, L.; Zhang, D.; Wu, C. Robust object tracking using joint color-texture histogram. *Int. J. Pattern Recognit. Artif. Intell.* **2009**, *23*, 1245–1263. [[CrossRef](#)]
29. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *37*, 583–596. [[CrossRef](#)] [[PubMed](#)]