


Article

Fast Face Tracking-by-Detection Algorithm for Secure Monitoring

Jia Su ¹, Lihui Gao ¹ , Wei Li ¹, Yu Xia ¹, Ning Cao ^{2,3,*} and Ruichao Wang ⁴

¹ School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang 050018, China

² School of Internet of Things and Software Technology, Wuxi Vocational College of Science and Technology, Wuxi 214028, China

³ School of Computer and Information Engineering, Chuzhou University, Chuzhou 239000, China

⁴ College of Social Sciences and Law, University College Dublin, Dublin Dublin 4, Ireland

* Correspondence: 2010005@wxsc.edu.cn; Tel.: +86-1860-326-3678

Received: 30 June 2019; Accepted: 4 September 2019; Published: 9 September 2019



Abstract: This work proposes a fast face tracking-by-detection (FFTD) algorithm that can perform tracking, face detection and discrimination tasks. On the basis of using the kernelized correlation filter (KCF) as the basic tracker, multitask cascade convolutional neural networks (CNNs) are used to detect the face, and a new tracking update strategy is designed. The update strategy uses the tracking result modified by detector to update the filter model. When the tracker drifts or fails, the discriminator module starts the detector to correct the tracking results, which ensures the out-of-view object can be tracked. Through extensive experiments, the proposed FFTD algorithm is shown to have good robustness and real-time performance for video monitoring scenes.

Keywords: Internet of Things; secure monitoring; face tracking; tracking-by-detection; correlation filter; convolution neural network

1. Introduction

In recent years, the Internet of Things (IoT) and big data have grown substantially, and secure monitoring is one of the most challenging tasks [1–3]. According to data from HIS Markit, in 2017, 98 million new network monitoring cameras and 29 million high-definition (HD) closed-circuit television (CCTV) monitoring cameras were shipped globally through professional sales channels [4]. According to the “In-Depth Research Report on Digital Surveillance Cameras in the Global and Chinese Markets in 2018” published by Hangzhou Primus, the total sales volume of the global digital surveillance cameras market is expected to reach 19,312.19 million USD by 2022 [5]. With the increase of monitoring cameras, a large quantity of video data will be generated every day, which makes it impossible to manually process them. Thus, an automated and intelligent video processing system is urgently needed.

Secure monitoring describes, interprets and predicts the behavior of moving targets by combining methods from computer vision [6,7], artificial intelligence, image processing and pattern recognition. As an emerging technology, intelligent video monitoring faces many challenges due to rapid development and increasingly mature theoretical technologies. Video face tracking is an important technology for intelligent video monitoring. However, because of factors such as illumination variation, deformation, and occlusion, long-term tracking of arbitrary objects with high robustness and high real-time performance remains a challenging problem.

In recent years, correlation filters and deep learning have been used for tracking and have achieved good results [8,9]. The kernelized correlation filter [10] adopts a cyclic matrix to sample map ridge

regression in linear space to a multidimensional nonlinear space through the kernel function method and converts a large number of intermatrix operations into calculations in the Fourier domain, thus increasing the operation speed. Deep learning has achieved great success in detection and recognition. Some convolutional neural network (CNN)-based correlation filters tracking methods [11,12] and deep learning detection methods [13,14] also have high accuracy in tracking. However, a large number of samples with various types are required for training deep learning methods. Because of lacking data and the complexity of deep algorithms, deep learning is limited in tracking tasks.

The main contributions of the paper are summarized as follows:

1. A real-time face tracking framework that uses a tracking-by-detection method to combine tracking and detection is proposed.
2. The kernelized correlation filter is used to quickly track faces. A multitask cascade CNN, as a detector, is introduced to detect faces and modify the tracker's result. The cascade CNN detector ensures accuracy, and improves the speed by not performing frame-by-frame updating.
3. An update strategy is designed, in which detection result is used to correct the inaccurate tracking result, and the modified result is used to update the tracking filter model. Thus the proposed method can robustly track drastic occlusion and out-of-view objects.

The experimental results show that the fast face tracking-by-detection (FFTD) algorithm has high robustness and real-time operation; therefore, high performance and long-term tracking secure monitoring can be achieved.

The rest of this paper is organized as follows. Section 2 introduces related work on object tracking, face detection and tracking-by-detection methods. The pipeline of the proposed face tracking algorithm is introduced in Section 3. In Section 4, the experimental results of the proposed method are reported and discussed. Finally, Section 5 concludes the paper and presents future work.

2. Related Work

Numerous object detection and tracking algorithms have been proposed over recent decades. In this section, we discuss related work on object tracking, face detection and the tracking-by-detection method.

We can divide object tracking methods into two types: generation methods and discrimination methods. Generation approach is template matching, which searches the most similar candidate region with the target as the tracking result [15–17]. Mean-shift algorithm [15] describes the object by color feature and searches the location with the largest color probability distribution through iteration. The algorithm is simple, but it does not perform well in fast motion. Particle filter algorithm [16] replaces the posterior probability distribution of state with random particles. Kalman filter algorithm [17] performs prediction and correction modules for steady tracking, which models an object with motion. Thus, the object motion is subject to linear Gaussian distribution. Discrimination method trains classifiers to distinguish between target and background [18–23]. Compressive tracking (CT) algorithm [18] extracts features by compressed sensing, in which the naive Bayesian classifier is used to predict the target location. The support vector machine (SVM) is used to classify in the Struck algorithm [19]. Context tracker (CXT) [20] uses the random forest classifier. Tracking-learning-detection (TLD) algorithm [21] combines tracking, detection and learning, which is worthy of reference. But, the speed is slow.

Compared with the traditional generation methods, the discriminant methods are more adaptable to environmental changes and long-term tracking, showing high robustness and high real-time applicability. In addition, in recent years, correlation filters and methods based on deep learning have far surpassed other tracking algorithms and achieved excellent results in various competitions [24,25].

2.1. Correlation Filter Tracking

Correlation filter has been used for tracking with excellent tracking accuracy and speed. In recent years, it has become the mainstream object tracking method. The method trains the correlation filter

according to the information of the current frame and the previous frame. The introduction of the Fourier transform speeds up the computation, so the tracking speed of single-target tracking can reach hundreds of frames.

The Minimum Output Sum of Squared Error (MOSSE) [26] filter, which uses adaptive training strategy, was the first to employ the correlation filter method for tracking. Because of MOSSE's robust and fast results, many MOSSE-based algorithms have been proposed. For instance, in the Circulant Structure with Kernel (CSK) algorithm [27], target samples generated by a circulant structure are used to train classifiers. Meanwhile, calculating in the Fourier transform domain reduces the complexity and greatly improves the speed. The KCF [10] algorithm adopts multichannel features and uses kernel correlation filtering. However, this algorithm is insensitive to scale changes. When the target is removed from view or blocked for a long time, the correlation filter often performs poorly. On the other hand, combining deep learning and correlation filtering, Martin proposed the C-COT [12] and ECO [11] algorithms, which achieve high accuracy. Among them, C-COT won the VOT2016 competition. However, the C-COT and ECO cannot be real-time, which is an important aspect in target tracking. Deep learning also imposes high requirements on the platform, which often requires a highly configured personal computer (PC) for operation.

2.2. Face Detection

Viola and Jones proposed the method of combining the AdaBoost algorithm and Harr features for face detection [28], which greatly improved the detection speed and accuracy. After that, many methods for improving the algorithm were proposed [29,30]. In recent years, computational power has greatly improved, and CNNs have been applied for face detection. For example, regional CNN (R_CNN) [31], Fast R_CNN [32], Faster R_CNN [33] and others [34] have adopted two subnetworks to extract and classify candidate boxes; these methods achieve high performance but are slow. You only look once (YOLO) [35], single-shot detector (SSD) [36], focal loss [37] and other algorithms [38] use a single network, thus they are faster but less accurate than the aforementioned algorithms. In addition, cascade CNN [39], multitask CNN (MTCNN) [40] and others [41] use multiple CNNs for face detection, further improving the performance.

2.3. Tracking-by-Detection

When the target is obscured or removed from sight, algorithms only based on tracking algorithm or detection tend to fail. The TLD algorithm, which combines detection and tracking technology, adds an online learning process. If the target is removed from the field of vision and then reappears, the detector can restart the tracker after detecting the target so that the tracker can continue to track. Inspired by this idea, tracking-by-detection has been adopted by many algorithms and has achieved good results [42–44].

Different from existing correlation filter trackers, we introduce the CNN face detection to the KCF algorithm, following the tracking-by-detection framework. Under the supervision and correction of face detection module, the proposed method can improve the face tracker's precision with fast speed.

3. Proposed Algorithm

In this section, we elaborate the FFTD method. First, we overview the FFTD method and give the whole framework. Moreover, we review the KCF algorithm which is the basic tracker of FFTD algorithm, and then a tracking update strategy is introduced. Finally, we construct the multitask CNN method for the face detection. In addition, we design a detection strategy to obtain the exact object.

To obtain a robust, fast face tracking algorithm in secure monitoring, the FFTD tracking algorithm framework is proposed in this paper. FFTD consists of three parts, i.e., tracking, face detection and discrimination, as shown in Figure 1.

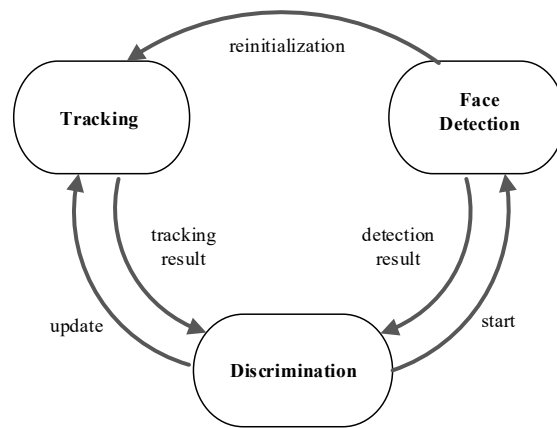


Figure 1. The fast face tracking-by-detection (FFTD) framework. A tracker based on a correlation filter finds the best position of the target in each frame. The face detector, which can correct and restart the tracker, detects and locates the faces in images. The discrimination determines when to perform the detector and update the tracking model.

3.1. Tracking

FFTD uses the KCF algorithm for tracking. Figure 2 shows the tracking module flow and update strategy in a block diagram. If the object is removed from the camera view, the KCF tracker usually fails and never recovers. To overcome the challenge, the detector is introduced. The detection result is used to correct the tracking result of the previous frame, and then the corrected result is used to update the filter model with a certain learning rate in current frame.

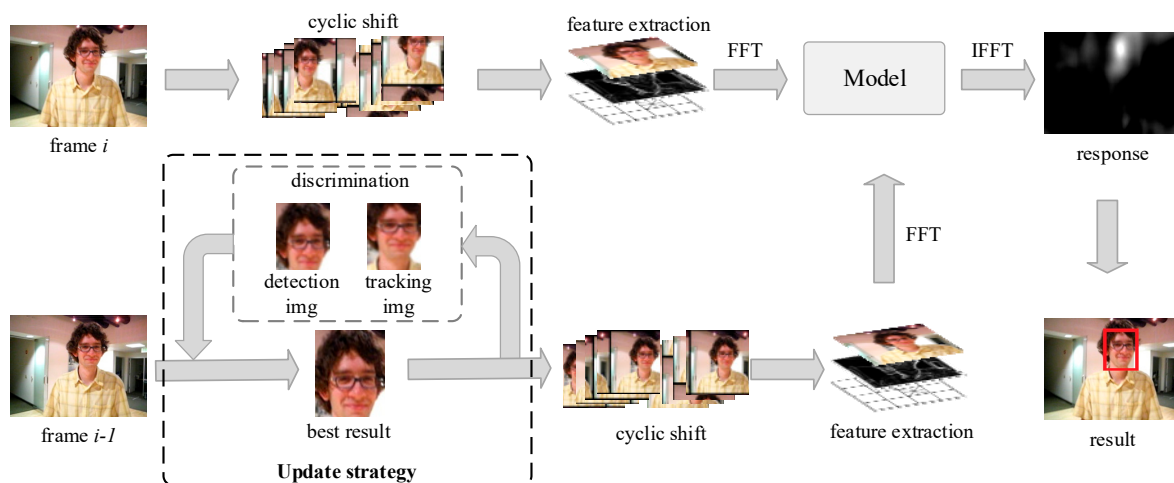


Figure 2. Tracking module flowchart.

In the first frame, to get the training samples, KCF uses the window (2.5 times larger than the given object) to cycle shift, and saves the samples into the filter model after extracting. Then, the training samples are labeled with Gaussian distribution. The regression coefficient of each sample is generated and stored through ridge regression.

In later frames, the windows (2.5 times larger than the best result) are cyclic-shifted to obtain the samples. We calculate the similarity between the samples and the target in the filter model. The current result is the largest response sample. Then, the filter model is updated by the result.

Next, the ridge regression method and update strategy are described in detail.

3.1.1. KCF Tracking Principle

The KCF algorithm adopts the ridge regression method to train the detector. Ridge regression is an improvement on least squares regression, and it is more reliable and practical. Assume the training sample set is $(x_i, y_i), i = 1, 2, \dots, n$, where x_i is the training samples, y_i is the corresponding sample labels and y_i obeys the Gaussian distribution.

Under the linear condition, the regression function is $f(x_i) = \omega^T x_i$, and ω is the weight function of the column vectors, which can be solved by the least squares method:

$$\min_{\omega} \sum_{i=0}^n [f(x_i) - y_i]^2 + \lambda \|\omega\|^2 \tag{1}$$

where λ is the regularization parameter and the embodiment of ridge regression's supplement to least squares regression. The process of training the classifier is to find the optimal ω to minimize the residual error of the regression function.

We set the derivative of (1) to be zero, solving ω , and rewriting it into frequency domain form:

$$\omega = (X^H X + \lambda I)^{-1} X^H y \tag{2}$$

where X^H is the conjugate transpose of X .

Training samples x_i in the KCF algorithm are obtained by the cyclic shift of target sample x . According to the properties of the cyclic matrix and Fourier transform, the formula for calculating the final linear regression coefficient is obtained:

$$\hat{\omega} = \frac{\hat{x} \odot \hat{y}}{\hat{x} \odot \hat{x}^* + \lambda} \tag{3}$$

where \hat{x} is the result of the fast Fourier transform (FFT) of x , and \odot is the operation of multiplying the corresponding elements of the matrix.

In practice, most problems are nonlinear, and the conversion of nonlinear problems into linear solution methods is expressed as:

$$\omega = \sum_i \alpha_i \varphi(x_i). \tag{4}$$

So in the nonlinear regression, solving the coefficient ω turns to solving α_i and the mapping relationship $\varphi(x_i)$. k is a Gaussian kernel function, and the nuclear correlation of two samples can be expressed as:

$$\begin{aligned} K_{ij} &= \varphi^T(x_i) \varphi(x_j) = k(x_i, x_j) \\ &= \exp\left(-\frac{1}{\sigma^2} [\|x_i\|^2 + \|x_j\|^2 - 2F^{-1}(\hat{x}_i^* \odot \hat{x}_j)]\right) \end{aligned} \tag{5}$$

Obtain the solution of the ridge regression of the kernel space:

$$\alpha = (K + \lambda I)^{-1} y. \tag{6}$$

Using the same method to calculate the nonlinear regression coefficient $\hat{\alpha}$,

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda}. \tag{7}$$

Finally, a nonlinear regression function is obtained as follows:

$$f(z) = \omega^T z = F^{-1}(\hat{k}^{xz} \odot \hat{\alpha}). \tag{8}$$

The maximum value of $f(z)$ is the maximum response value and the position of the predicted target.

3.1.2. Update Strategy

In tracking, the filter model should not be constant. It needs to be updated when the target is scale variation or deformation. The samples of object x_i and coefficient α_i should be mainly updated:

$$\begin{cases} x_i = (1 - \theta)x_{i-1} + \theta x_i \\ \alpha_i = (1 - \theta)\alpha_{i-1} + \theta \alpha_i \end{cases} \quad (9)$$

where x_i and α_i represent the samples and coefficient in frame i , x_{i-1} and α_{i-1} are the samples model and coefficient model in frame $i - 1$, θ is the learning rate of the update model.

To improve this defect, the update strategy is summarized as follows:

1. When the face detector and tracker are working, t_{S^c} (the similarity between the tracking result image and the initial object image) and d_{S^c} (the similarity between the detection result image and the initial object image) are calculated. When $d_{S^c} > t_{S^c}$, we set the detection result as the final output. When $d_{S^c} \leq t_{S^c}$, we set the tracking result as the output. The output is used to update the tracking model with a certain learning rate using Equation (9).
2. When the target is subjected to long-term or large-scale occlusion without tracking results, the results of the face detector are directly assigned to the tracker.

3.2. Face Detection

A multitask cascade CNN (MTCNN) [40] is used in the face detection module to accurately detect faces and return the rectangular boxes. Face detection is binary classification, which does not need many filters in each layer. MTCNN provides a lightweight CNN model for face detection and location, which has less filters per layer but more depth of the whole network. Compared with other cascaded CNN detection algorithms [45], the 3×3 filters are introduced to MTCNN, which is faster in calculation and has good real-time performance.

MTCNN processes tasks from coarse to fine through a third-stage cascaded framework. The block diagram is shown in Figure 3. We build an image pyramid by resizing the test image to different scales and then input the different-scale images to the third-stage cascaded CNN.

3.2.1. The Coarse-to-Fine CNN

The coarse-to-fine CNN includes three networks: P-Net, R-Net and O-Net. The CNN architectures are shown in Figure 4.

1. P-Net

P-Net, a fully convolutional network, is the first layer, which roughly filters out nonface regions. The main task of P-Net is to obtain the bounding box regression vectors, calibrate the position of the face candidate regions, and then merge the overlapping face candidate regions with the nonmaximum suppression (NMS) method.

2. R-Net

The results of P-Net are resized to 24×24 , which are the inputs of R-Net. R-Net is not a fully convolutional network, whose final layer has two full connection layers. Compared with P-Net, R-Net outputs more precise face classification and bounding box regression.

3. O-Net

O-Net will further screen candidate regions to obtain more accurate results. The outputs of R-Net are inputted to the final stage with the size of 48×48 . O-Net processes images more accurately because it has an extra convolutional layer. O-Net obtains the final face position.

From P-Net to O-Net, the network has increasingly larger input image sizes and deeper structures, and the features extracted from the network become more expressive.

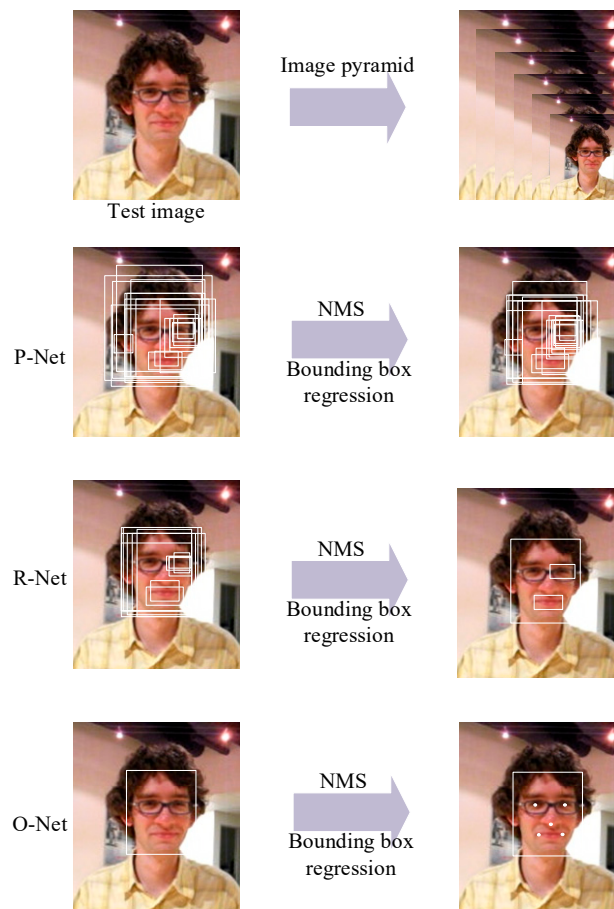


Figure 3. The multitask convolutional neural network (MTCNN) framework.

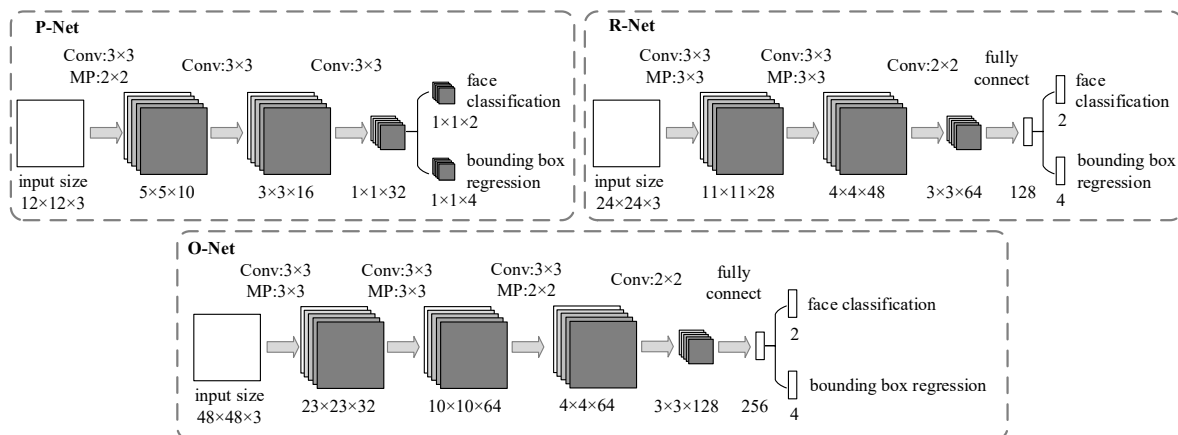


Figure 4. The coarse-to-fine convolutional neural network (CNN) structure, including P-Net, R-Net and O-Net.

3.2.2. Face Detection Strategy

When MTCNN processes multiple faces, we select the candidate box that is the most similar object to the initial object as the result using the match template method, as shown in Figure 5. The initial object box given in the first frame and multiple MTCNN results are passed into the template matching module in subsequent frames. The normalization cross-correlation (NCC) method is introduced in the template matching.

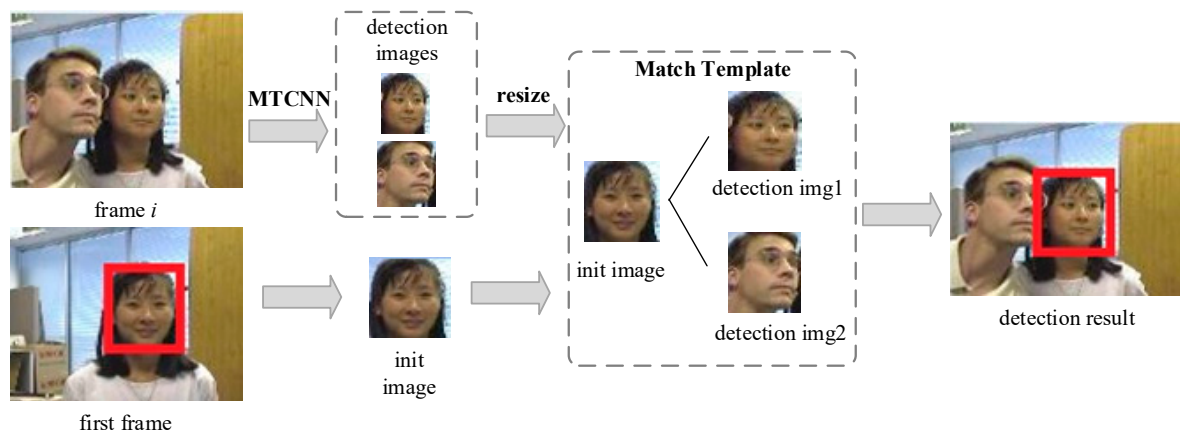


Figure 5. Face detection strategy.

3.3. Main Loop

In Algorithm 1 the main loop of the proposed method is given. When the initial object is selected, the KCF tracking model M is initialized. KCF tracker runs in all frames of the sequence. When tracking result is none or with large error, the MTCNN detector is started in the next frame. Discrimination fuses the tracking result Tr and detecting result Dr into a final result R .

The main works of discrimination as follows: (1) When $Tr = 0$ and $Dr \neq 0$, we set $R = Dr$; (2) when $Tr \neq 0$ and $Dr \neq 0$, R equals to the maximum one of two similarities (t_{S^c} and d_{S^c}), then R is used to update M ; (3) when $Tr = 0$ and $Dr = 0$, the tracker is failure, we set $R(i) = R(i - 1)$.

At this stage, we get the result in frame i . After printing the result, the next image is processed.

In frame $i + 1$, whether the detector is performed is decided by the parameter of *EnableDetect* in frame i . If the *EnableDetect* is set as True in frame i , we start to detect. In two cases the *EnableDetect* is True: (1) no tracking result; (2) $RNet.score(R, I(i)) < 0.99$. To avoid tracking other objects that are not faces, we test the result every k frames (we used three in the experiment). The result is sent to R-Net to calculate the score of face classification. If the score is less than 0.99, we set *EnableDetect* to True. Compared to the complex O-Net, we chose R-Net which can achieve good accuracy and faster speed.

Then, we proceed to the next frame until the last one. In a word, the whole tracking process is completed.

Algorithm 1 Overall Process of the Proposed Algorithm.

Input: image I , $EnableTrack = false$, $EnableDetect = True$, the frequency of judge whether tracker's result is correct k .

Output: the final result R .

```

1  for  $i = 1:n$  do
2    if  $EnableDetect \parallel EnableTrack$  then
3      Detecting;
4       $EnableDetect = False$ ;
5    end if
6    if  $i = 1$  then
7      given the initial object:  $initobj$ ;
8      initialize tracking filter model  $M$ ;
9    else
10   if  $(EnableTrack \parallel Dr(i) \neq 0)$  then
11     Tracking;
12     if  $Tr(i) \neq 0$  then
13        $EnableTrack = True$ ;
14        $R(i) = Tr(i)$ ;
15       if  $Dr(i) \neq 0$  then
16          $R(i) = \max(\text{matchTemplate}(initobj, Tr(i)), \text{matchTemplate}(initobj, Dr(i)))$ ;
17          $M = \text{update}(I(i), R(i))$ ;
18       end if
19     else if  $Dr(i) \neq 0$  then
20        $R(i) = Dr(i)$ ;
21        $M = \text{update}(I(i), R(i))$ ;
22        $EnableDetect = True$ ;
23     else
24        $EnableTrack = False$ ;  $EnableDetect = True$ ;
25        $R(i) = R(i-1)$ ;
26     end if
27     if  $k > 3 \ \&\& \ RNet.score(R, I(i)) < 0.99$  then
28        $EnableDetect = True$ ;
29        $k = 1$ ;
30     end if
31      $k++$ ;
32   end if
33 end if
34 show( $R(i)$ );
35 end for

```

4. Experimental Results

In this section, we introduce the details and methods of the experiments. To verify the validity of the algorithm, we selected 14 groups of face test videos from OTB2015 (object tracking benchmark) [46] and nine videos from the YouTube Face Database [47]. Qualitative and quantitative evaluation are introduced to discuss the results.

4.1. Parameters and Details

The experimental platforms used to verify algorithm performance were Visual Studio 2015, OpenCV3.1.0 and MATLAB 2016a, and the hardware environment was an Intel i5-6500U (Intel Corporation, Santa Clara, CA, USA) 3.20-GHz CPU with 8 GB RAM.

The KCF and FSTD algorithms adopted the histogram of oriented gradients (HOG) feature; the cell size was 4×4 using a Gaussian kernel in the experiment. MTCNN used CelebA [48] and WIDER

FACE [49] to train the CNNs. The WIDER FACE dataset was mainly used to train the detection model, and the CelebA dataset was used to train regression facial landmarks.

4.2. Experimental Metrics

In the experiment, the two basic parameters used to measure target tracking accuracy were the precision and success rate, which were calculated from the center location error (CLE) and overlap score. The CLE is the average Euclidean distance between the center locations of the tracking bounding box and the ground truth. The unit of CLE is pixels, and 20 pixels is usually used in experiments as a threshold. The overlap rate is defined as the ratio of the intersection to the union between the estimated bounding box and given ground truth box. The threshold for the overlap rate varies between 0 and 1; 0.5 is often used for performance evaluation. For the OTB2015 dataset, we used one-pass evaluation (OPE) to test the precision and success rate of all trackers.

4.3. Qualitative Evaluation

4.3.1. Evaluation on OTB2015

In this work, we would like to present the results of 33 trackers in 14 group sequences, in order to clearly visualize the results, besides those of the FFTD and KCF [10] algorithms; we selected the top four trackers (context tracker (CXT) [20], collaborative correlation tracker (CCT) [50], Struck [19], TLD [21]) from the whole 31.

Figure 6 shows the results of the FFTD and KCF algorithms and the other four trackers. We present the main results for four challenging sequences from the OTB2015 dataset.

1. Occlusion and out-of-view. In the dragonBaby sequence, when the object moves out of camera view, the KCF and CXT trackers failed. In the girl sequence, the girl was completely occluded by the man. The KCF method did not perform well.
2. Out-of-plane rotation. In frame 119 of the girl sequence, the girl turned her head such that we could not see her face clearly; TLD and Struck did not perform well.
3. Illumination. In the trellis sequence, TLD and CXT drifted off the object, while FFTD achieved the highest precision.
4. Fast motion and motion blur. In dragonBaby, only FFTD could track the fast-moving face. In the jumping sequence, KCF could not follow a blurred face.

Overall, the proposed FFTD algorithm performed well in most sequences.

4.3.2. Evaluation on YouTube

We also evaluated FFTD, KCF and TLD on the YouTube Face dataset. The dataset, downloaded from YouTube, has 3425 videos of 1595 people. Figure 7 shows some sequences of the testing sequences. In fact, most faces in the YouTube dataset had distinct features and little movement, which reduces the difficulty of tracking tasks. Although three trackers performed well in most videos, we can observe that FFTD exhibits higher precision.

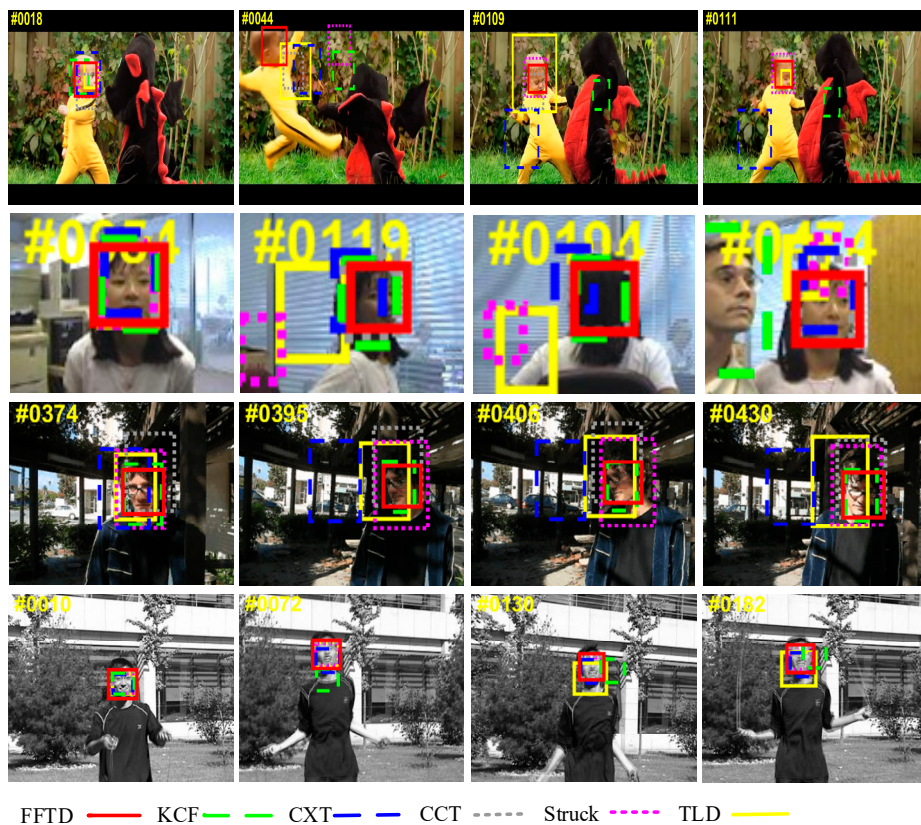


Figure 6. Tracking results of the six trackers on four sequences from the OTB2015 dataset (from top to bottom are dragonBaby, girl, trellis and jumping).

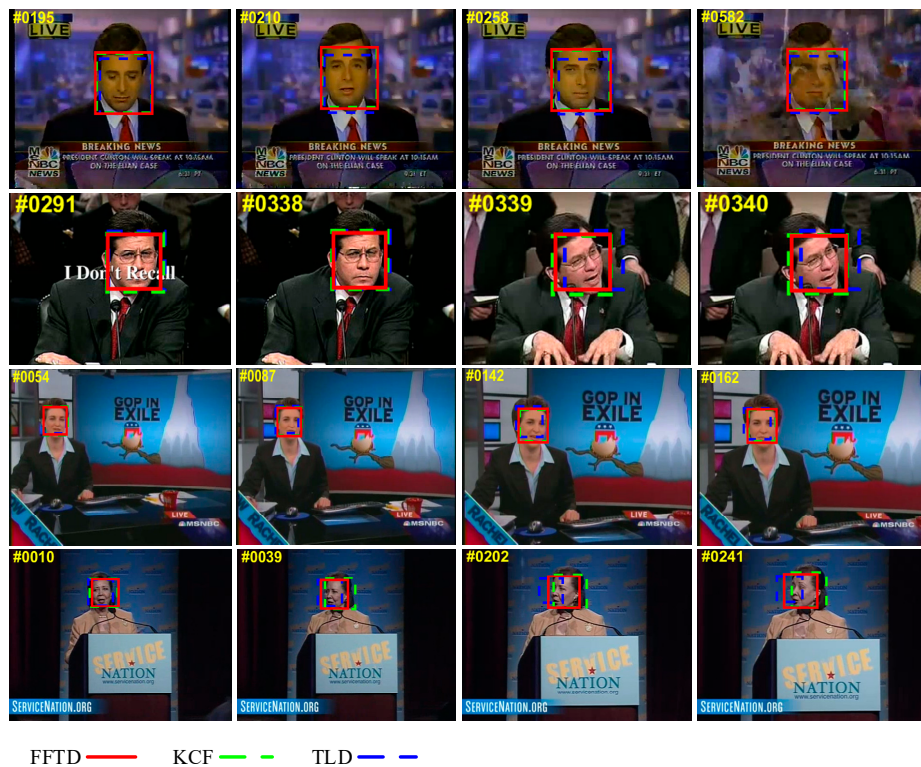


Figure 7. Tracking results of the three trackers on four sequences from the YouTube Face dataset (from top to bottom are Alex_Penelas0, Alberto_Gonzales3, Alexandra_Pelosi0, Alma_Powell4).

4.4. Quantitative Evaluation

This work evaluates the FFTD and KCF algorithms with 31 trackers using OTB2015. Figure 8 shows the one-pass evaluation results in terms of the success rate and precision. The proposed FFTD performed well, with a precision of 88.1% and an overlap success rate of 86.2%. The CCT and Struck were the top two methods among the 31 trackers [43], with 84.6%/83.1% and 77.6%/78.2% for the precision score (PS)/success rate (SR) metrics. Compared with other trackers, FFTD was an improvement.

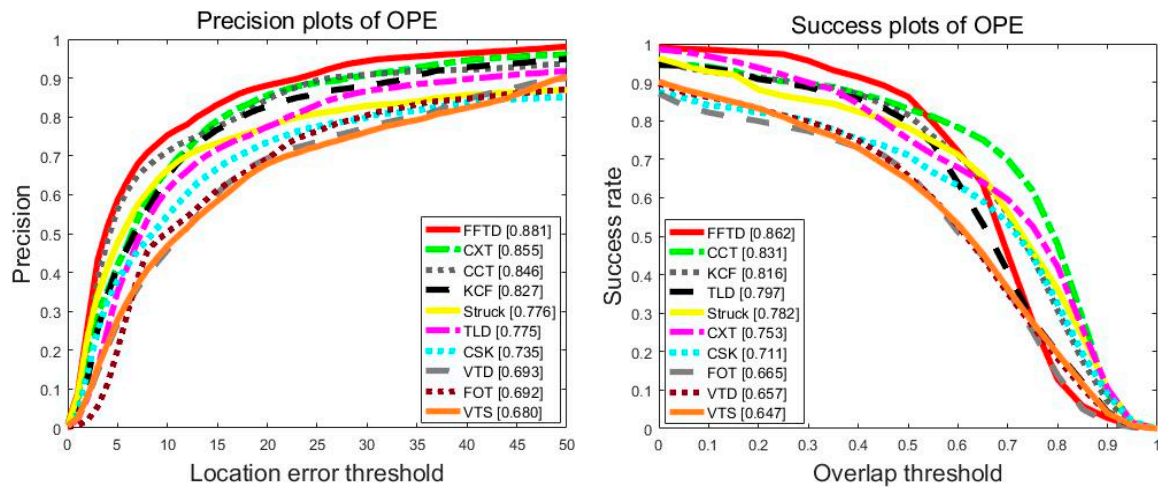


Figure 8. Precision and success plots over the 14 sequences using one-pass evaluation on the OTB2015 dataset. The top 10 trackers are shown in the figure. The legend shows the success rate with an overlap threshold of 0.5 and precision score for a threshold of 20 pixels for each tracker.

The KCF method performed well, with a precision of 82.7% and an overlap success rate of 81.6%. However, it usually failed when an object moves out of view, which is a fatal problem. Figure 9 shows the tracking performance in the out-of-view tracking problem. Compared with KCF, the proposed FFTD method performed better by 20.9/20.5 in terms of PS/SR. The face detection module and new tracking strategy of the FFTD algorithm solved this problem. Overall, the FFTD algorithm performed well on the OTB2015 dataset.

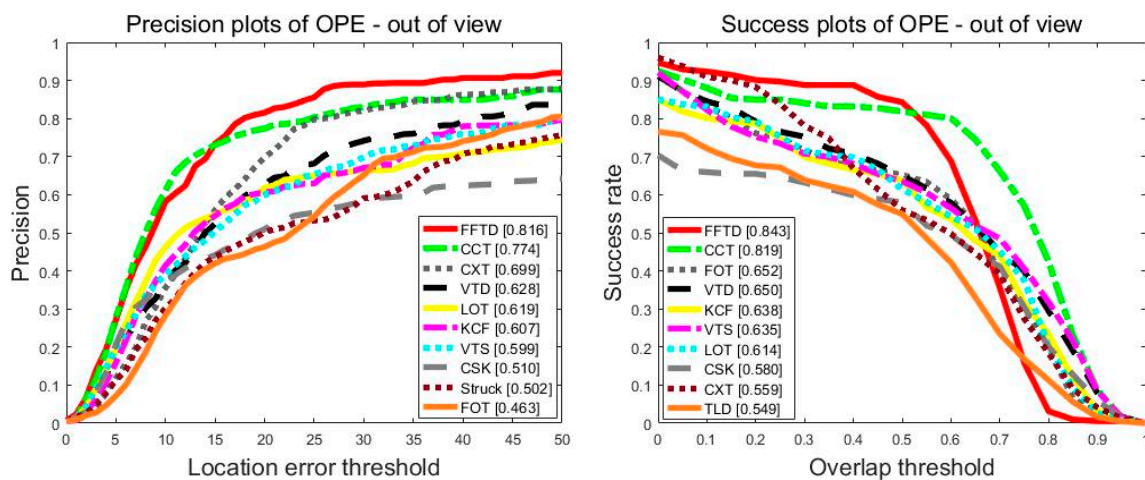


Figure 9. Tracking performance in the out-of-view challenge over the 14 sequences from the OTB2015 dataset.

1. Attribute-Based Analysis

Table 1 reports the success rate and precision data of the FFTD and KCF algorithms and the other four trackers (CXT, CCT, Struck, TLD) for some challenging tracking attributes. The attributes were scale variation (SV), out-of-view (OV), out-of-plane rotation (OPR), low resolution (LR), in-plane rotation (IPR), illumination (IV), motion blur (MB), background clutter (BC), occlusion (OCC), deformation (DEF), and fast motion (FM). They are helpful to test the performance of the algorithms in terms of different aspects.

Table 1. The precision and success rate of six algorithms on eight attributes.

	OPR	SV	OCC	MB	FM	IPR	OV	BC
FFTD	0.849 ¹	0.784	0.8	0.765	0.863	0.848	0.816	0.971
	0.838 ²	0.772	0.791	0.712	0.838	0.816	0.843	0.994
KCF	0.851	0.791	0.796	0.689	0.669	0.851	0.607	0.959
	0.814	0.734	0.773	0.644	0.703	0.814	0.638	0.939
CXT	0.864	0.806	0.751	0.847	0.817	0.864	0.699	0.931
	0.814	0.65	0.719	0.577	0.592	0.749	0.559	0.911
CCT	0.817	0.777	0.722	0.874	0.859	0.817	0.774	0.935
	0.771	0.712	0.74	0.866	0.873	0.771	0.819	0.864
Struck	0.785	0.693	0.651	0.585	0.68	0.728	0.502	0.925
	0.771	0.673	0.714	0.533	0.657	0.715	0.534	0.921
TLD	0.764	0.687	0.638	0.794	0.727	0.766	0.427	0.323
	0.748	0.687	0.749	0.763	0.741	0.759	0.549	0.736

¹ For each attribute of the algorithm, the top is precision and the bottom is success rate. ² Bold represents the best.

The FFTD algorithm performed well in terms of the attributes of scale variation, out-of-view, out-of-plane rotation, in-plane rotation, background clutter, occlusion and fast motion. According to the experimental results, the FFTD algorithm did not perform well in motion blur. In motion blur, the target region is blurred due to the motion of target or camera. The fuzzy face features lead to the inaccurate detection result. In the future work, we will improve the performance in the challenging attributes. The CXT method had the highest success rate in terms of scale variation, out-of-plane rotation and in-plane rotation. However, in most cases, FFTD algorithm performed better than other algorithms.

2. Real-Time Evaluation

We also evaluated frame rates of the six algorithms. The frame rate FR is defined as, $FR = N/T$, where N is the total frames in the sequences, T is the time of a tracker to pass through the sequences.

Table 2 reports the frame per second (fps) rate of the six trackers over 14 sequences in the OTB2015 dataset. The average frame rate of KCF was the highest, reaching 92.2 fps. The FFTD algorithm was next, at 59.4 fps. The TLD algorithm obtained 19.5 fps. CCT had the worst real-time performance. The speed of FFTD is more than double that of the classic tracking-by-detection algorithm TLD. Compared with KCF, FFTD had reduced speed, but it still met the requirements of real-time applications.

Table 2. Frame rates (fps) of the six algorithms.

	KCF	FFTD	TLD	CXT	Struck	CCT
Blurface (493 frames) ¹	88.9	62.4	13.0	10.9	14.2	6.2
Boy (602 frames)	121.3	50.4	13.4	5.1	13.1	7.2
David (770 frames)	55.7	52.8	17.2	9.5	6.1	7.4
david2 (537 frames)	137.5	61.4	26.7	21.3	6.9	8.3
Dudek (1145 frames)	43.6	64.6	7.1	5.7	10.5	4.6
dragonBaby (113 frames)	87.2	40.8	18.5	8.6	8.3	7.5
faceocc1 (892 frames)	75.5	64.0	17.9	8.5	9.7	6.0
faceocc2 (812 frames)	35.8	60.8	21.9	6.9	7.3	7.0
Fleetface (707 frames)	67.0	42.1	7.9	6.8	10.2	5.7
Girl (500 frames)	139.2	64.9	20.1	33.5	9.0	7.8
Man (134 frames)	176.0	80.1	52.9	32.0	14.8	7.6
Mhyang (1490 frames)	62.3	80.3	17.1	10.5	6.8	7.0
Jumping (313 frames)	153.6	42.5	15.0	7.3	10.7	7.5
Trellis (569 frames)	46.7	63.7	24.8	12.4	6.3	6.7
average	92.2	59.4	19.5	12.4	9.6	6.9

¹ Frame length of sequence.

5. Conclusions

To improve the robustness and real-time performance of the tracking algorithm in intelligent video monitoring, this work proposes the FFTD algorithm, which is based on the tracking-by-detection framework. First, the KCF algorithm is adopted in FFTD, with an updating strategy that incorporates the tracking result modified by the detector into the filter model. In addition, the MTCNN face detector enhances the overall robustness performance of the algorithm. The experimental results show that the FFTD algorithm has good precision and accuracy, even under challenging conditions. In conclusion, the FFTD algorithm satisfies the requirements of fast and long-term secure monitoring.

In future work, we will improve the performance at tracking challenges involving motion blur, scale variation and rotation. Moreover, this work will be extended to track other objects, requiring us to design more-robust neural networks.

Author Contributions: Conceptualization, J.S. and L.G.; Methodology, J.S. and L.G.; Software, L.G. and W.L.; Validation, L.G. and Y.X.; Formal analysis, J.S.; Investigation, W.L. and Y.X.; Resources, J.S.; Data curation, J.S.; Writing—original draft preparation, J.S., L.G. and W.L.; Writing—review and editing, L.G., S.G., and N.C.; Supervision, J.S., N.C. and R.W.; Project administration, J.S., N.C. and R.W.; Funding acquisition, J.S.

Funding: This work was supported by key laboratory of equipment pre-research (No. 6142A010301), science and technology support plan of Hebei Province (No. 18210803D).

Acknowledgments: We would like to acknowledge the collaboration of the Fujian Province University Key Lab for Industry Big Data Analysis and Application.

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

References

1. Kaur, H.; Sahambi, J.S. Vehicle tracking using fractional order Kalman filter for non-linear system. In Proceedings of the International Conference on Computer Communication and Automation, Noida, India, 15–16 May 2015; pp. 474–479.
2. Huang, K.; Chen, X.; Kang, Y.; Tan, T. Intelligent Visual Surveillance: A Review. *Chin. J. Comput.* **2015**, *6*, 1093–1118.
3. Xu, J.; Wei, L.; Zhang, Y.; Wang, A.; Zhou, F.; Gao, C.Z. Dynamic Fully Homomorphic Encryption-based Merkle Tree for Lightweight Streaming Authenticated Data Structures. *J. Netw. Comput. Appl.* **2018**, *107*, 113–124. [CrossRef]
4. Top Video Surveillance Trends for 2017. Available online: <http://cdn.ihs.com/www/pdf/TEC-Video-Surveillance-Trends.pdf> (accessed on 19 May 2019).

5. Global Digital Surveillance Camera Market. Available online: <http://www.zk71.com/qyresearch/product/138125842.html> (accessed on 19 May 2019).
6. Smeulders, A.W.M.; Chu, D.M.; Cucchiara, R.; Calderara, S.; Dehghan, S.; Shah, M. Visual tracking: An experimental survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 1442–1468. [[PubMed](#)]
7. Barrow, H.G.; Tenenbaum, J.M. Computational vision. *Proc. IEEE* **1981**, *69*, 572–595. [[CrossRef](#)]
8. Danelljan, M.; Hager, G.; Khan, F.S.; Felsberg, M. Accurate Scale Estimation for Robust Visual Tracking. In Proceedings of the British Machine Vision Conference, Nottingham, UK, 1–5 September 2014; pp. 1–11.
9. Nam, H.; Han, B. Learning multi-domain convolutional neural networks for visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 4293–4302.
10. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 583–596. [[CrossRef](#)] [[PubMed](#)]
11. Danelljan, M.; Bhat, G.; Khan, F.S.; Felsberg, M. ECO: Efficient convolution operators for tracking. In Proceedings of the IEEE Conference Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 6931–6939.
12. Danelljan, M.; Robinson, A.; Khan, F.S.; Felsberg, M. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 472–488.
13. Bertinetto, L.; Valmadri, J.; Henriques, J.F.; Vedaldi, A.; Philii, H.S. Fully-convolutional Siamese networks for object tracking. In Proceedings of the European conference on computer vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 850–865.
14. Held, D.; Thrun, S.; Savarese, S. Learning to Track at 100 FPS with Deep Regression Networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 125–133.
15. Comaniciu, D.; Ramesh, V.; Meer, P. Real-time tracking of non-rigid objects using mean shift. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, SC, USA, 15 June 2000; pp. 142–149.
16. Nummiaro, K.; Koller-Meier, E.; Van Gool, L. An adaptive colorbased particle filter. *Image Vis. Comput.* **2003**, *21*, 99–110. [[CrossRef](#)]
17. An Introduction to the Kalman Filter. Available online: <http://www.cs.unc.edu/~jwelch/kalman/kalmanIntro.html> (accessed on 27 June 2019).
18. Zhang, K.; Zhang, L.; Yang, M.H. Real-time Compressive Tracking. In Proceedings of the European Conference on Computer Vision, Florence, Italy, 7–13 October 2012; pp. 864–877.
19. Hare, S.; Golodetz, S.; Saffari, A.; Philip, H.S. Struck: Structured output tracking with kernels. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *23*, 263–270.
20. Dinh, T.B.; Vo, N.; Medioni, G. Context tracker: Exploring supporters and distracters in unconstrained environments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011; pp. 1177–1184.
21. Kalal, Z.; Mikolajczyk, K.; Matas, J. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 1409–1422. [[CrossRef](#)] [[PubMed](#)]
22. Danelljan, M.; Hager, G.; Khan, F.S. Convolutional features for correlation filter based visual tracking. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Santiago, Chile, 11–18 December 2015; pp. 621–629.
23. Liu, Q.; Lu, X.H.; He, Z.Y.; Zhang, C.K.; Chen, W.H. Deep convolutional neural networks for thermal infrared object tracking. *Knowl.-Based Syst.* **2017**, *134*, 189–198. [[CrossRef](#)]
24. Kristan, M.; Matas, J.; Ales, L.; Felsberg, M.; Cehovin, L.; Fernandez, G.; Vojir, T.; Hager, G.; Nebel, G.; Pflugfelder, R.; et al. The Visual Object Tracking VOT2015 challenge results. In Proceedings of the IEEE International Conference on Computer Vision Workshop (ICCVW), Santiago, Chile, 7–13 December 2015; pp. 564–586.
25. Chanho, K.; Fuxin, L.; Arridhana, C.; James, M.R. Multiple Hypothesis Tracking Revisited. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 11–18 December 2015; pp. 4696–4704.

26. Bolme, D.S.; Beveridge, J.R.; Draper, B.A.; Lui, Y.M. Visual object tracking using adaptive correlation filters. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 2544–2550.
27. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. Exploiting the circulant structure of tracking-by-detection with kernels. In Proceedings of the European Conference on Computer Vision, Florence, Italy, 7–13 October 2012.
28. Viola, P.; Jones, M.J. Robust real-time face detection. *Int. J. Comput. Vis.* **2004**, *57*, 137–154. [[CrossRef](#)]
29. Zakaria, Z.; Suandi, S.A. Face Detection Using Combination of Neural Network and Adaboost. In Proceedings of the TENCON 2011–2011 IEEE Region 10 Conference, Bali, Indonesia, 21–24 November 2011; pp. 335–338.
30. Mahmood, A.; Khan, S. Early terminating algorithms for AdaBoost based detectors. In Proceeding of the 16th IEEE International Conference on Image Processing, Cairo, Egypt, 7–10 November 2009; pp. 1209–1212.
31. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
32. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 11–18 December 2015; pp. 1440–1448.
33. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137. [[CrossRef](#)] [[PubMed](#)]
34. Dai, J.F.; Li, Y.; He, K.M.; Sun, J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In Proceedings of the Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 379–387.
35. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
36. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
37. Lin, T.Y.; Goyal, P.; Girshick, R. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *99*, 2999–3007.
38. Liu, S.; Huang, D.; Wang, Y. Receptive Field Block Net for Accurate and Fast Object Detection. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 404–419.
39. Zhao, C.; Nuno, V. Cascade R-CNN: Delving into High Quality Object Detection. *arXiv* **2018**, arXiv:1712.00726.
40. Zhang, K.; Zhang, Z.; Li, Z.; Qiao, Y.; Rong, G. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Process. Lett.* **2016**, *23*, 1499–1503. [[CrossRef](#)]
41. Zhu, C.; Zheng, Y.; Luu, K.; Savvides, M. CMS-RCNN: Contextual multi-scale region-based CNN for unconstrained face detection. *arXiv* **2016**, arXiv:1606.05413.
42. Danelljan, M.; Hager, G.; Khan, F. Learning spatially regularized correlation filters for visual tracking. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 11–18 December 2015; pp. 3074–3082.
43. Li, Y.; Zhu, J. A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 256–265.
44. Alan, L.; Tomas, V.; Luka, C. Discriminative Correlation Filter with Channel and Spatial Reliability. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4847–4856.
45. Li, H.; Lin, Z.; Shen, X.; Brandt, J.; Hua, G. A convolutional neural network cascade for face detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 8–10 June 2015; pp. 5325–5334.
46. Wu, Y.; Lim, J.; Yang, M. Object tracking benchmark. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*, 1834–1848. [[CrossRef](#)] [[PubMed](#)]
47. Wolf, L.; Hassner, T.; Maoz, I. Face Recognition in Unconstrained Videos with Matched Background Similarity. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, SC, USA, 20–25 June 2011; pp. 529–534.

48. Liu, Z.; Luo, P.; Wang, X. Deep learning face attributes in the wild. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 11–18 December 2015; pp. 3730–3738.
49. Yang, S.; Luo, P.; Loy, C.C.; Tang, X. Wider face: A face detection benchmark. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 5525–5533.
50. Zhu, G.; Wang, J.; Wu, Y.; Lu, H. Collaborative Correlation Tracking. In Proceedings of the British Machine Vision Conference, Swansea, UK, 7–10 September 2015; pp. 184.1–184.12.



© 2019 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 (<http://creativecommons.org/licenses/by/4.0/>).