Multitarget Tracking Using Siamese Neural Networks

NA AN and WEI QI YAN, Auckland University of Technology

In this article, we detect and track visual objects by using Siamese network or twin neural network. The Siamese network is constructed to classify moving objects based on the associations of object detection network and object tracking network, which are thought of as the two branches of the twin neural network. The proposed tracking method was designed for single-target tracking, which implements multitarget tracking by using deep neural networks and object detection. The contributions of this article are stated as follows. First, we implement the proposed method for visual object tracking based on multiclass classification using deep neural networks. Then, we attain multitarget tracking by combining the object detection network and the single-target tracking network. Next, we uplift the tracking performance by fusing the outcomes of the object detection network and object tracking network. Finally, we speculate on the object occlusion problem based on IoU and similarity score, which effectively diminish the influence of this issue in multitarget tracking.

CCS Concepts: • **Computing methodologies;**

Additional Key Words and Phrases: SSD, SiamRPN, SiamFC, ResNet50, AlexNet

ACM Reference format:
Na An and Wei Qi Yan. 2021. Multitarget Tracking Using Siamese Neural Networks. *ACM Trans. Multimedia Comput. Commun. Appl.* 17, 2s, Article 75 (May 2021), 16 pages.
https://doi.org/10.1145/3441656

1 INTRODUCTION

Visual object tracking is a critical issue in the fields of computer vision and deep learning, which have a rich assortment of applications such as intelligent surveillance, human action or behavior analysis, and automated driving. Visual object tracking is comprised of single-target tracking and multitarget tracking. The earliest single-target tracking algorithm was based on MOSSE (i.e., minimum output sum of squared error) [1], which is measured by using the affinity between two groups of visual signals through cross correlation. The classical work based on correlation filtering was derived from MOSSE, which encapsulates KCF (i.e., kernelized correlation filters), DSST (i.e., discriminative scale space tracking), C-COT (i.e., continuous convolution operator tracker), and ECO (i.e., efficient convolution operators) [2, 3, 8, 10]. These tracking algorithms need to accommodate CN (i.e., color names), HOG (i.e., histogram of oriented gradients), and visual features extracted by using CNN or ConvNet (i.e., convolutional neural network) for model training. With the development of deep learning methods, the single-target algorithms based on the Siamese neural network are evaluated by using the affinity between visual objects.

Authors' address: N. An and W. Q. Yan, Auckland University of Technology, No. 2-14, Wakefield street, CBD Auckland, 1010 New Zealand; emails: jmj8905@autuni.ac.nz, wyan@aut.ac.nz.

1551-6857/2021/05-ART75 $15.00
https://doi.org/10.1145/3441656

Although both SOT (i.e., single object tracking) and MOT (i.e., multiple object tracking) are successful in object tracking, the algorithms have distinctions. SOT tracking algorithms cater to the single class in Pattern classification, whereas MOT tracking methods appeal specifically to tracking visual objects that have multiple classes simultaneously. In terms of video duration, the former copes with short time frame sequences, whereas the latter generally deals with longer videos, which encompass the appearance, occlusion, and disappearance of multiple objects.

Generally, the steps of the MOT algorithm consist of the follows: (1) identify visual objects based on an input video; (2) with regard to each detected object, compute the visual and motion features; (3) calculate the probability that two given objects belong to the same target; and finally, (4) assign an ID to each target, which is the key step of affinity calculation and data association [5, 6]. SORT (i.e., simple online and real-time tracking) [9] algorithms tackle the correlation in a frame-by-frame way by using a Kalman filter through motion analysis, whereas the Hungarian algorithm was employed to obtain the correlation according to the positions of the center point and the moving speeds of a visual object. In 2017, SORT algorithms were updated by using Mahalanobis distance as the affinity metric and cosine distance as the metric from inner product of two visual features. The overall affinity is acquired by using weighted average of these two metrics [33, 35].

Since deep learning has been employed to solve the MOT problem, a method using a recurrent neural network was proffered for target state prediction and data association, but its tracking performance was curbed by exploiting advanced methods [51]. In 2018, a few new algorithms were applied to deep neural networks to solve multitarget tracking, such as DMAN (i.e., dual matching attention networks) [14], TBA (i.e., tracking by animation) [52], and TAMA (i.e., deep temporal appearance matching association) [53]. However, there remains a big gap between these methods and the traditional ones.

Generally speaking, SOT algorithms based on deep learning are comparable to the traditional ones. Therefore, we utilize a SOT algorithm combined with an object detection algorithm to achieve multitarget tracking in this work. Intuitively, object detection and tracking are two complementary tasks, perfect tracking is able to upgrade missed detection, and excellent detection can curtail the tracking errors from drifting. We thus offer SSD (i.e., single shot multibox detector) as the object detection algorithm, which is one of the state-of-the-art methods in deep learning and has been applied to real-time object recognition [4]. However, SiamRPN (i.e., Siamese-RPN) [5, 47, 48], as a tracker, has been trained offline and achieved exceptional performance over the most advanced correlation tracking algorithms.

Overall, we propose the SSD detector in deep learning to detect visual objects. We successfully implement the recognition of five classes of visual objects, namely cars, buses, pedestrians, bicycles, and scooters. We calculate the affinity between visual objects by using Siamese networks. Finally, the Hungarian algorithm is employed to find the best visual object matches.

Pertaining to the remaining parts of this article, we delineate our related work in Section 2, our methods are elucidated in Section 3, our experimental results are demonstrated in Section 4, our resultant analysis is presented in Section 5, and our conclusion is drawn in Section 6.

2 RELATED WORK

In object tracking, SOT algorithms based on the structure of the Siamese network have the capabilities of object detection and recognition; MOT algorithms that use an SOT structure instead of motion models or spatial models only have emerged one after another. For example, by exploiting Bi-LSTM (i.e., bi-directional long-short term memory), DMAN implements feature extraction and compares historical features of the observation frames and the target trajectory. Regarding data association, both spatial and temporal attention mechanisms are carved out to handle the problems of object detection and object occlusion. Another multitarget tracking algorithm is STAM

(i.e., spatial-temporal attention mechanism) [15], which takes advantage of CNN-based trackers to extract motion features and spatial features, using a spatiotemporal attention mechanism to highlight foreground targets so as to resolve occlusion problems by using data association. In contrast, another algorithm [18] was designed for dissolving the occlusion problem, and the basic tracker is based on SiamRPN. A high-quality neural network was designed based on ResNet-18 (i.e., residual neural network), which selects k features related to the target trajectory, and a classifier was trained by using a regularized Newton boosting decision tree to achieve the data association.

FAMNet [62] is another Siamese network–based MOT algorithm. The appearance and location information of visual objects were extracted by using the Siamese network, and the visual features were compared based on the response graph. An R1TA power iteration layer was employed to reduce the complexity of continuous multiframe association and enhance the continuous multiframe tracking. In addition, UMA (i.e., unified object motion and affinity model) [60], which is akin to the DMAN algorithm, utilizes SOT to achieve the acquisition of apparent features and motion information, and attain online matching and association. UMA uses SiamFC as the tracker and AlexNet for feature extraction. For data association, the occlusion is estimated through the changes of affinity similarity and IoU. Moreover, the KCF [61] algorithm encapsulates the instance-aware SOT tracker, which uses multicut to correlate the SOT results. To solve the occlusion problem, KCF makes use of time distance, position shape, IoU, histogram, and so forth as visual features for classification through SVM to achieve cross-frame matching.

However, a Siamese symmetric network [11] was proffered to learn the affinity of epigenetic features, and the affinity between motion features and epigenetic features was attained based on a gradient descent algorithm. A fully connected network was designed, and its output is based on spatial features and 6D motion contextual features associated with relative changes of object size, position, and speed. Throughout the gradient descent algorithm, multitarget tracking results are exported by using linear programming optimization.

In addition, historical trajectory information is crucial to judge the target state in multitarget tracking. It is feasible to design a network structure that is able to store the historical information and match the similarity based on historical information [58]. A feature fusion algorithm was proposed based on the **long short-term memory (LSTM)** network, which calculates the matching similarity between trajectory history information and current detection [59]. LSTM is treated as the historical information model.

The deep learning model in the proposed algorithm was originally from a convolutional network [64, 65]. Since object tracking has applied historical trajectory information to determine a new state of the object, the model thus is able to store the historical information and find the matches by using the affinity of historical information. A fusion algorithm was designed based on an LSTM model of the recurrent neural network to learn the affinity between historical and current trajectory information by calculating three features: Apparent feature, motion feature, and interactive mode feature. All three features are facilitated based on the LSTM. The final classification score is treated as the affinity [12, 54]. A multiview hash model was set forth [55] to enrich the multiview information by using deep neural networks, utilizing view stability to actively explore the relationships among multiple views.

3 OUR METHODS

3.1 Our Algorithms

The proposed model consists of SSD and a modified SiamRPN neural network. The raw frames from a video were imported into SSD so as to detect visual objects. With regard to the template

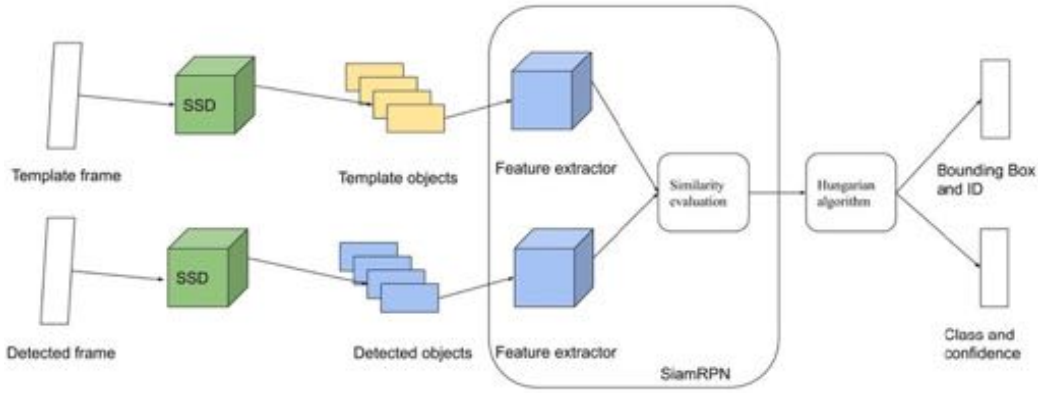


Fig. 1. The architecture of the proposed model.

frame, the detected objects were cropped from the input image and fed into SiamRPN for feature extraction and similarity evaluation. Regarding the detected frame, the whole image is fed into SiamRPN for feature extraction and affinity calculation between the template objects in the template frame and the predicted objects in the detected frame. The affinity score from SiamRPN is input into the Hungarian algorithm so as to acquire the best match. If the affinity score is higher than a given threshold, the tracking is successful. The architecture of this model is shown in Figure 1.

SiamRPN consists of a Siamese subnetwork for feature extraction and a **region proposal network (RPN)** for classification [17, 18]. The RPN was proposed as a part of Faster R-CNN [7]. RPN generates region proposals for visual object detection, which has a classification branch for object detection and a regression branch for location prediction.

In this work, the proposed Siamese network as a classifier predicts the probability of a region proposal for target tracking. Suppose that there are k anchors in total, and each anchor needs to be classified into foreground or background; thus, the number of classes is $2k$. Each anchor is given by using four parameters: x and y coordinates of the location (x, y), width w, and height h. For any given images, a scale is related to the size of the given image, and an aspect ratio is the proportion between the corresponding width and height of this image. If we choose three scales and three aspect ratios in total, then nine proposals are generated. For the whole image, the number of anchors is $w \times h \times k$.

We assume that the i-th frame has j objects and the $(i+1)$-th frame has l objects. $F[i][j]$ denotes the j-th object of the i-th frame, which corresponds to the template frame in the original RPN; therefore, $F[i+1][l]$ is the l-th object of the $(i+1)$-th frame in RPN. The RPN network for object classification is shown in Figure 2.

The RPN classification network is based on binary classification. Firstly, we segment the input image into $w \times h \times k$ regions, where k denotes the number of anchors, h is the height of a feature map, and w is the width of the feature map. We need to determine whether the anchor is foreground or background by comparing the overlapping area between this anchor and the associated ground truth, then assign each anchor with a label of foreground or background.

The proposed method is to treat the correlation layer as a convolutional layer so that the number of features is $2k$ by adjusting the shape of the convolution kernel. For a template frame, the convolutional layer is responsible for the dimension of feature maps and the number of features is increased to $2562k$. To maintain alignment, the detected frame has an additional convolutional layer, as shown in Figure 3.

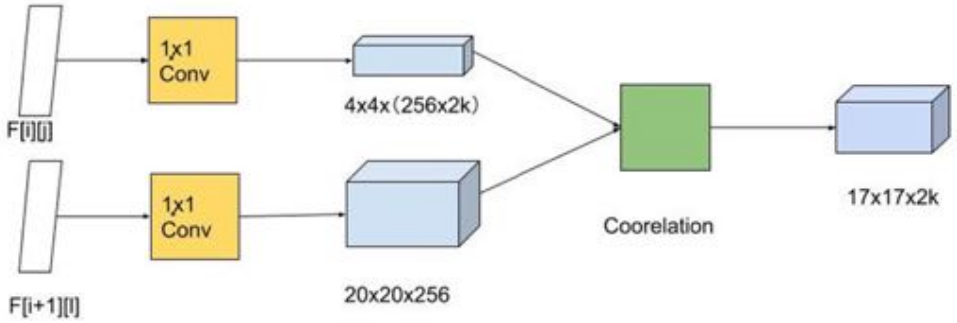


Fig. 2. The RPN network for object classification.

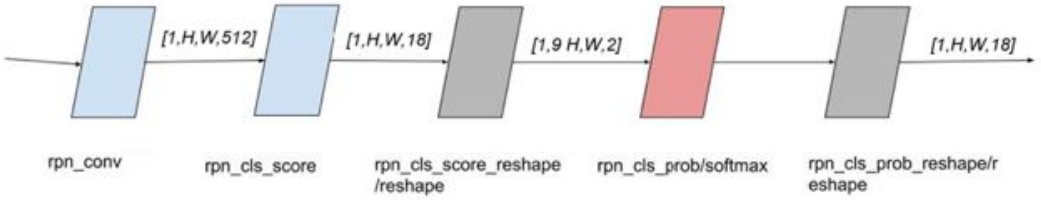


Fig. 3. The process of object classification.

The correlation operation is conducted to get the final classification [37]. The correlation is operated by calculating the affinity of each position in the form of sliding windows. The affinity is defined as

$$f(z,x) = \emptyset(z) * \emptyset(x) + b, \tag{1}$$

where z is the template frame, x is the detected frame, and b represents the score of each position. Equation (1) treats $\emptyset(z)$ as a convolution kernel that conducts convolutions with $\emptyset(x)$.

Amid tracking, the search image centered on the previous frame of target position is accommodated to calculate the corresponding score. While the window is sliding, the position with the largest score is collaborated with the step length to determine the target position. Thus, RPN is trained to have the ability to classify any inputs as one of the classes, namely foreground or background [36].

An image with the resolution $M \times N$ is fed into the deep learning model, and this image is resized as $(M/16) \times (N/16)$ before imported into the RPN network. We define $w = M/16$ and $h = N/16$. The image size $w \times h$ is imported for 1×1 convolution. Then, a reshape layer is followed by the 1×1 convolution; we see that the output is $9 \times w \times h \times 3$, which corresponds to the possibilities of $9 \times h \times w$ anchors being classified as the class foreground or background. The outputs are the labels of anchor classes, and we compare this with the ground truth to get the classified loss. A feature map has $9 \times w \times h$ anchors, and each corresponds to nine anchors. These nine anchors have three aspect ratios—1:1, 1:2, and 2:1—and thus each feature map has three sizes. The loss function is defined as

$$\mathrm{L}(\{p_i\},\{t_i\}) = \left(\frac{1}{N_{cls}}\right) \cdot \sum L_{cls}(p_i, p_i^*) + \left(\frac{\lambda}{N_{reg}}\right) \cdot \sum p_i^* \cdot L_{reg}(t_i \cdot t_i^*), \tag{2}$$

where i is the index of anchors, p_i is probability of the i-th anchor that contains an object, t_i is the vector of predicted bounding box, p_i^* represents the ground truth of the i-th anchor, t_i^* is the vector of the bounding box of the ground truth, and $L_{cls}(\cdot)$ and $L_{reg}(\cdot)$ stand for logarithmic loss function over two classes. p_i^* in Equation (2) ensures that if a visual object is identified, only the

regression loss will be counted; otherwise, $p_i^* = 0$, and the regression item will become zero in the cost function. N_{cls} and N_{reg} are normalization constants, and $\lambda = 10$ by default is assigned to scale classifier and regressor.

In the RPN network, throughout the classification branch for object detection, we get the affinity score of the detected object, and the location prediction of the detected object is obtained by using the regression branch of the RPN network. Thus, we merge the results of object detection based on SSD to improve the tracking accuracy. If the affinity score is greater than a given threshold, and the IoU between the tracking bounding box and the detection bounding box is greater than the given threshold, the matching is considered successful. We take the union of the two bounding boxes as the bounding box of the visual object, and the feature information of the template object is thus updated.

In MOT, the tracking branch of the Siamese network needs to match the tracking targets within the given frames, whereas SSD does not provide any matching targets in a given frame. If a target is successfully matched in the historical frames, then we assign an object ID to the target and update the feature map of the template object. In the case of occlusion, we observe the IoUs and affinity scores. If the two indicators continuously are rising and the tracking target cannot be found within the video frames, it is considered that the occlusion occurs. At this time, though the target matching is successful, the feature map of the template object is not updated.

ALGORITHM: Multitarget tracking algorithm based on the Siamese network

```
Initialization: templist = extend2dimension(ssdResults)
For each image in the list, do
        If the image is the first frame, do
       Init each bounding box with box ID and count
        Else do
              for each object in object list of pre-frame, do
                 feature(object) = GetFeature(object)
                 score(object) = GetScore(feature(object))
                 if score(object) < threshold
              templist.append(object)
       end
       for each template object in templist, do
          feature(object) = GetFeature(object)
                 score(object) = GetScore(feature(object))
                 scorelist.append(score(object))
                 if score(object) < threshold
                    template object count + 1
       end
       for each template object in templist, do
          if template object count > threshold
             remove object from template list
       end
       trackers = MatchID(scoreMatrix)
       for each tracker in trackers, do
          tracker.updateID()
          tracker.updatePos()
       end
end
```

4 RESULTS

4.1 Dataset

We took a total of 1-hour video footage and chose 3,000 frames as our own dataset—approximately 600 images for each class. Additionally, for the tracking, we took advantage of our own dataset and the MOT16 dataset [56]. The MOT16 dataset was employed to evaluate the multitarget tracking method of the MOT Challenge Series proposed in 2016. The dataset is comprised of 14 videos, and the videos were selected to extract features from natural scenarios without editing or postprocessing, with a quality that is the same as that of a handheld camera. The selected video frames of our training dataset are shown in Figure 4.

Figure 5 illustrates the tracking results using our test videos. During object tracking, we assign an ID to the same target in different frames. The labels on the top of the bounding boxes are the class tags of the target, and the labels at the bottom of the bounding box are the target IDs. We assess our model based on the MOT16 challenging benchmark [19]. The MOT16 dataset was presented in 2016 to measure multitarget trackers that are specifically used for pedestrian tracking. In the MOT16 dataset, seven videos were employed for training and the others for test; all of them are with static or moving cameras in an unconstrained environment.

4.2 Tracking Accuracy of Siamese Networks

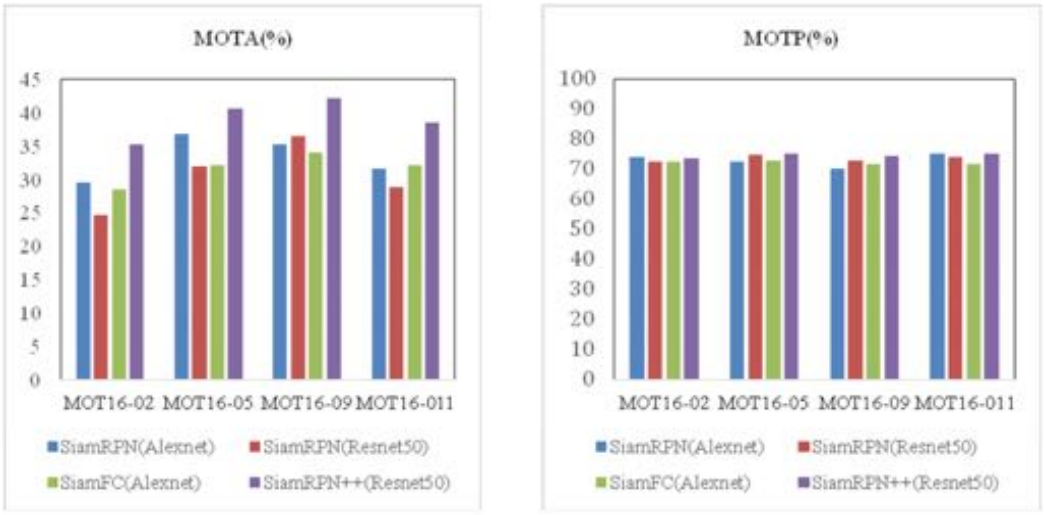
In our experiment, we used the AlexNet-based SiamRPN as the single object tracking network. The average MOTA is 33.2%, and the MOTP is up to 72.9%. Compared with the benchmark of the MOT challenge [46], this result is acceptable. We assume that the deeper the neural network is, the higher the accuracy of object tracking will be, because deeper neural networks extract visual features effectively. We therefore accommodate ResNet-50 as the backbone to observe the tracking accuracy based on various CNN models. The average MOTA based on the MOT dataset is 31.3% and MOTP is up to 73.4%, which is akin to the performance of the AlexNet-based tracking network. The object tracking could not benefit from the deeper neural network, and therefore we substitute SiamRPN with SiamFC to observe the tracking results. Meanwhile, the outcomes from the SiamFC-based tracking model resemble the AlexNet-based tracking network. The average MOTA of the SiamFC-based tracking network is 31.6%, whereas it is up to 33.2% using the AlexNet-based network. The average accuracy rates of the two deep learning models are 72.0% and 72.9%, respectively. In Figure 6, we compare the tracking performance of various Siamese networks.

Because deeper neural networks do not attain higher tracking accuracy, our focus is on other aspects of the network rather than simply raising the network depth. SiamRPN calculates affinity based on the correlation, which has two constraints. First, the neural network needs to satisfy strict translation equivalence, which means that when the target moves within the video frames, the tracking position will also shift accordingly. Second, the neural network needs to have symmetry, namely the calculated affinity should be unaltered, because the affinity has the symmetric attribute [24, 26].

If we deepen the structure of SiamRPN, it may lead to the following problems. First, using ResNet as a backbone network will inevitably introduce padding. The padding damages the translation equivalence of the neural network because the response at the margin of a video frame will be shifted, which results in dissatisfaction of translation equivalence. Second, SiamRPN will be imposed to the regression of position offsets and classification scores. The calculation of these two variables is not a symmetric process. Therefore, the neural network will lose symmetry [27].

SiamRPN++ was proposed to solve the problems by using the optimized deep learning model ResNet-50. Since the padding in ResNet has triggered the deep learning model to lose translation

Fig. 4. The selected frames from our dataset (a) and MOT16 (b).



(a) MOTA with multiple Siamese networks (b)MOTP with multiple Siamese networks

Fig. 5. The comparations of object tracking algorithms by using multiple Siamese networks.

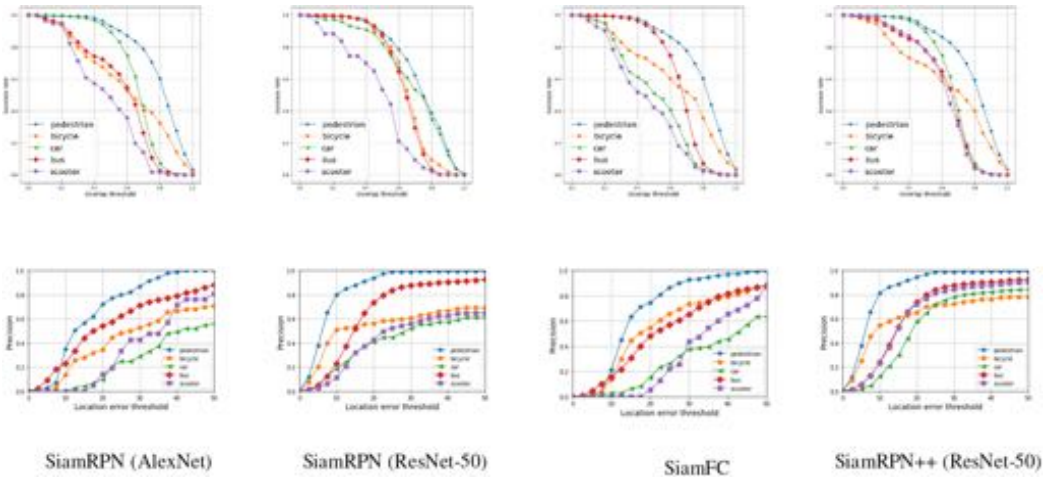


Fig. 6. The success plot and the precision plot of multiple Siamese networks.

equivalence, the model will learn position preference amid model training [25]. Therefore, if the padding is enabled in a deep neural network, the positive samples are located in the center of the training image, which means that the network has the location preference due to the distribution of the training samples. With regard to the prediction preference, no matter where the target moves in the image, the network will only predict locations of the object around the central area. This is the why tracking performance could not be updated after deepening the network with ResNet.

SiamRPN++ allows the network to learn the position preference from a training dataset by using the positive samples uniformly distributed in various positions within the image so that the tracking performance of the network will not be abated in the deepened neural network.

Additionally, SiamRPN++ makes use of depth-wise cross correlation to solve the asymmetric problems. Throughout convolutions, the feature map is not channel upscaled but is fine tuned for the two groups of feature maps. Furthermore, to integrate the feature maps with multiscale, semantics, and resolutions, SiamRPN++ enables a multilevel cascading method and fuses the feature maps after multiple convolutions so as to expand the receptive field of the feature maps. The stride is also shrunk to gain an accurate positioning prediction. In this article, we replace SiamRPN with SiamRPN++. Compared with SiamRPN, the MOTP of SiamRPN++ is not significantly improved, but the average MOTA has grown 8%, which means that SiamRPN++ finds more matched objects.

4.3 Tracking Accuracy of Multiclass Visual Objects

The proposed model is able to detect and track visual objects from five classes. To verify performance of the deep learning models in practical applications, we utilize the precision plot and the success plot to evaluate the models based on our own dataset. In the plots [20], the success plot reflects the successful rate, whereas the **overlap score (OS)** is calculated by using the bounding box that is acquired by using the tracking algorithm and another bounding box that is provided by the ground truth.

With regard to the single-target tracking algorithm, if the OS of a frame is greater than a predefined threshold (usually 0.5), the tracking is considered successful. In our experiment, the successful object detection is also defined if the OS of this object is greater than the given threshold. The successful rate of object detection is defined as the ratio between the number of visual objects that are successfully detected and the total number of the objects.

The precision plot reflects the deviation of object positioning. Location error is the distance between the center point of the bounding box estimated by using the tracking algorithm and the location of ground truth. In the single-target tracking algorithm, the successful rate is the percentage of video frames, whose location errors are less than an assigned threshold to the total number of video frames [34]. In our experiment, the percentage represents the ratio of the number of visual objects to the number of total objects, whose location errors are less than a given threshold.

The success plot and the precision plot for each class are shown in Figure 6. Pedestrian tracking has the highest success rate among the five classes in four Siamese networks that achieve an average rate of 0.75, whereas the scooter tracking has the lowest one 0.53. The success rates of other three classes are acceptable. The location error of pedestrian tracking is the lowest one among the five classes. Conversely, the car tracking has the highest location error. The location errors of bus tracking and bicycle tracking are similar. The scooter tracking has the lowest success rate, but the position deviation is acceptable. However, though the success rates of car tracking and bus tracking are closer, the deviation of car positions is much higher than that of buses.

In our experiments, the performance of SiamRPN (AlexNet), SiamRPN (ResNet-50), and SiamFC networks has not been changed too much. This is inconsistent with our hypothesis that the deeper the neural network, the better the tracking performance. However, the average tracking accuracy and location error of SiamRPN ++ have been uplifted a lot. By using the MOT16 dataset as the benchmark, the tracking performance of our model is acceptable. In our dataset, the tracking result is significantly upgraded, as we have mingled the results of the detection network and tracking network together, and the detection network provides a more accurate position in object tracking. In addition, we retain historical trajectory information of the tracking target. The tracking network corrects the missed instances from the detection network by using the historical trajectory information. Among the five classes, namely pedestrian, car, bus, bicycle, and scooter, pedestrian has the best tracking result, whereas scooter has the worst result. The reason is that we have many pedestrian samples and few scooter samples in our dataset.

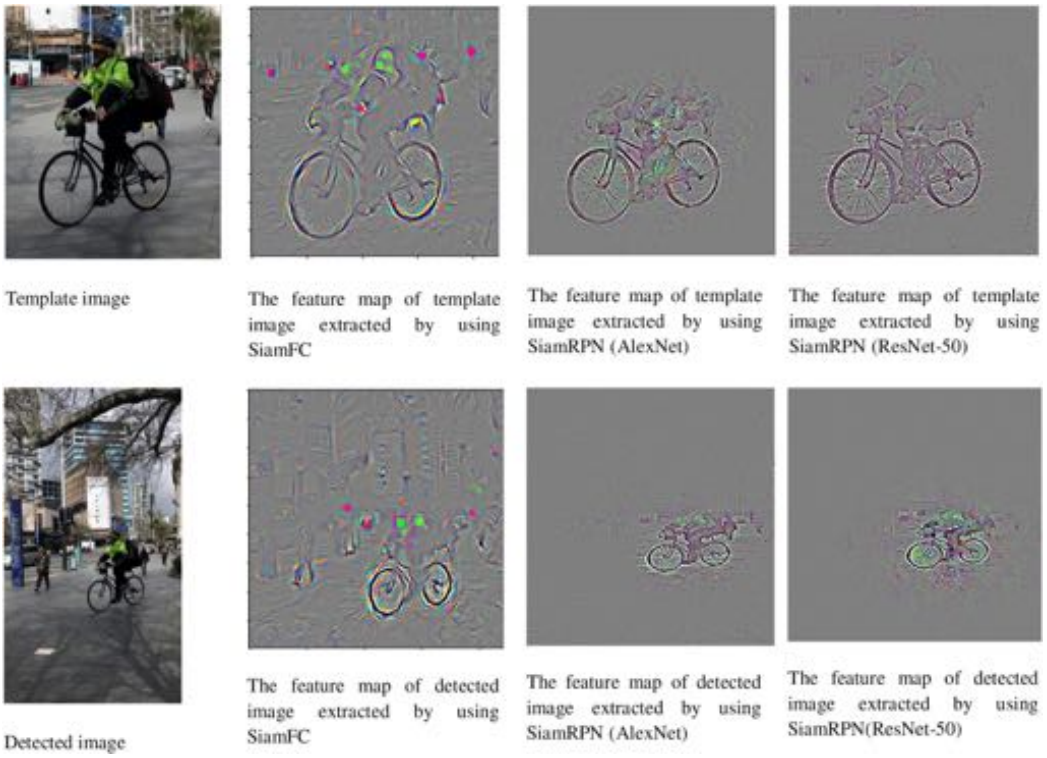


Fig. 7. The feature maps of multiple Siamese networks.

5 ANALYSIS

5.1 Feature Extraction

In our experiment, we take advantage of deep neural networks as the backbone of SiamRPN and SimaRPN++ for feature extraction. The performance of the SiamRPN-based tracking network has not significantly increased, whereas the SiamRPN++ related network successfully ameliorates the performance by overcoming the barrier that SiamRPN loses translation equivalence in ResNet due to padding.

In Figure 7, we compare the Gradient-weighted Class Activation Mapping (Grad-CAM) [41, 42] of SiamFC, SiamRPN (AlexNet), and SiamRPN (ResNet50) for feature extraction. Figure 8 shows the "important" regions after feature extraction. The three networks clearly mark off the shapes of bicycle wheels. The distinction is that the feature maps extracted by using SiamFC are coarse. Compared with SiamRPN (AlexNet), SiamRPN (ResNet-50) has a larger region of feature maps.

5.2 Affinity Measurement and ID Assignment

In this article, we treat multitarget tracking problem as an issue of affinity measure between visual objects. The Siamese neural network has two subnetworks or channels that share weights. The similarity between two input vectors are measured using cross correlation. Therefore, the affinity measures of SiamFC, SiamRPN, and SiamRPN++ are all based on cross correlation [43].

SiamFC takes advantage of cross correlation as the affinity measure to calculate the similarity at each position of two feature vectors so as to acquire a score map [28]. We assume that the

Fig. 8. The affinity measures between template frames and detected frames.

feature vectors of the template image and the detected image are $\mathbf{x}$ and $\mathbf{y}$, respectively. The cross correlation is calculated using Equation (3).

$$\mathrm{S} = \frac{cov(\mathbf{x}, \mathbf{y})}{\sqrt{var(\mathbf{x})var(\mathbf{y})}}, \tag{3}$$

$$\mathrm{cov}(\mathbf{x}, \mathbf{y}) = E((\mathbf{x} - \mu)(\mathbf{y} - \gamma)) = \mathrm{E}(\mathbf{xy}) - \mu\gamma, \tag{4}$$

where $\mathrm{cov}(\mathbf{x}, \mathbf{y})$ is covariance of $\mathbf{x}$ and $\mathbf{y}$; $var(\mathbf{x})$ and $var(\mathbf{y})$ are variances of vectors $\mathbf{x}$ and $\mathbf{y}$, respectively; and $E(\bullet)$ is mathematical expectation. A sliding window is employed to calculate the affinity measure between the template image and the detected image. Then, a 17×17 score map is generated as the output [29, 30]. We assume that feature vector $\mathbf{x}$ of the training object and feature vector $\mathbf{y}$ of the test object have the same dimension n, and we apply the cosine function or inner product to calculate the affinity measure of two vectors as Equation (4).

$$\cos(\theta) = \frac{||x||||y||}{||x||||y||} = \frac{\sum_{i=1}^{n} \mathbf{xy}}{\sqrt{\sum_{i=1}^{n} \boldsymbol{x}x^2}\sqrt{\sum_{i=1}^{n} \boldsymbol{y}^2}}, \tag{5}$$

where $y = \cos(\theta)$ is a monotonic function within the interval [0, 1.0]. The larger the $\cos(\theta)$, the smaller the angle between the two vectors. If the two vectors are parallel, $\cos(0) = 1.0$, which means that the two objects are the same; if the directions of the two vectors are perpendicular, $\cos(90^\circ) = 0$, which indicates that the two objects are completely different.

SiamFC applies the cross correlation to obtain a single-channel response map, whereas SiamRPN extends the cross correlation to a higher level by adding a convolutional layer to scale the channels. However, the up-channel module makes the parameters very unbalanced [31]. The dimension of feature maps of the template image is $2k$ times or $4k$ times the dimension of feature maps of the detected image, which makes SiamRPN training very difficult.

Depth-wise correlation was proposed in SiamRPN++ to reduce computational cost and memory. The feature maps of the template image and detected image, in the regression branch and the classification branch, are firstly fed into a convolutional layer to obtain the same spatial resolution and channel dimension. The convolutional layer of SiamRPN++ is different from the layers in SiamRPN. SiamRPN++ does not increase the dimension of feature maps but fine tunes the two feature maps to make them symmetrical. After passing through the convolutional layer, depth-wise correlation based on the feature maps of the template image and detected image takes effect on the components of each channel and outputs the same number of score maps. Finally, for the score maps of the same resolution and channel, output by different network branches, a 1×1 convolution

Fig. 9. Comparisons of the matched results of various affinity measures.

kernel is used for dimension adjustment to obtain the dimensions of output. The output of the classification branch has $2k$ times the size of the channel dimension, and the regression branch correspondingly outputs $4k$ times the size of the channel dimension.

Each feature point on the feature maps corresponds to k anchors in a receptive field of the original image. For the regression branch, each channel is given as the quadruple (x, y, w, h) for k anchors. For the classification branch, the output value of each channel is the classification score of the k anchors. In this way, the number of parameters of template object tends to balance, resulting in a very stable training process. In Figure 8, we compare the tracking results based on cross correlation and depth-wise correlation, which are thought of as the affinity measures.

The rows in Figure 9 show the results of cross correlation, and we see that the targets were identified but not matched. At the same time, there is also a target that switches its ID when re-identified [40]. In contrast, the tracking result based on depth-wise correlation for similarity measurement is better. Only one object in these seven frames is recognized but not matched, and there is not an ID switch that occurs when this object is re-recognized.

Because the template images are like the detected images in multitarget tracking, we feed the affinity measure into the Hungarian algorithm to obtain the best matching result. The Hungarian method is a combinatorial optimization algorithm that solves the assignment problem. For example, in two consecutive video frames, there are many-to-many matches if we set the threshold to 0.6. We get the affinity measure between the template frame and detected frame through the Siamese network as follows.

$$\begin{bmatrix} 0.921 & 0.003 & 0.001 & 0.001 & 0.001 & 0.001 \\ 0.001 & 0.890 & 0.001 & 0.001 & 0.001 & 0.001 \\ 0.001 & 0.001 & 0.934 & 0.563 & 0.001 & 0.001 \\ 0.001 & 0.001 & 0.761 & 0.864 & 0.001 & 0.642 \\ 0.001 & 0.001 & 0.001 & 0.001 & 0.957 & 0.001 \\ 0.001 & 0.001 & 0.637 & 0.609 & 0.001 & 0.827 \end{bmatrix} \tag{6}$$

We import the score matrix as the affinity measure into the Hungarian algorithm [44] and find the maximum match. Given a nonnegative matrix, we have to find an assignment of the targets to the objects such that each object is assigned to one target and each target is assigned one object. The key steps of the Hungarian algorithm are as follows. In step 1, each value is subtracted with the minimum value in the row; this procedure is repeated for all rows and all columns. In step 2, by searching for the matches, only the elements with a value of zero are matchable. In step 3, if the maximum number of matches is n, the current match is the best one; otherwise, go to next step. In step 4, we replace all zero values in the matrix with the smallest one in the row and the

column; we select the smallest element of the nonzero submatrix or block matrix; we subtract the smallest element from all elements of the new block matrix; and then we return to step 2 [32]. After completing all iterations, we will get the best matches.

6 CONCLUSION

Data association is a key step in the pipeline of multitarget tracking. Most of the existing MOT methods are based on traditional machine learning algorithms [57]. In this article, we employ a new SOT algorithm in which SiamRPN is combined with SSD to solve the issue of data association. As a single-target tracking algorithm, SiamRPN cannot discriminate multiple objects with similar characteristics in the same frame. Therefore, the Hungarian algorithm is utilized after the Siamese network to get the best matches between visual objects and tracking targets [63].

For the occlusion problem, we speculate on the occurrence of occlusions based on IoU and similarity scores of historical trajectory information, and we retain the original features if an occlusion occurs to lessen the impact. Overall, the main contributions of this article are as follows:

- Different from single-target tracking methods that can only track one class of targets, our proposed model achieves multiclass and multitarget tracking simultaneously.
- In this work, we proposed a novel method that achieves multitarget tracking by combining a SOT algorithm with an object detection algorithm.
- The proposed model resolves the problems of disappearance of tracking targets and appearance of new ones by mingling SSD and the Siamese network together. In addition, throughout fusion of the results generated from the detection network and the tracking network, the accuracy of object tracking is uplifted.
- We analyze the IoU and similarity scores of the detected objects and the template objects to determine the occlusion and disappearance issues of tracking targets.

REFERENCES

[1] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. 2010. Visual object tracking using adaptive correlation filters. In *Proceedings of IEEE CVPR*. 2544–2550.
[2] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. 2014. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 3 (2014), 583–596.
[3] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. 2015. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of IEEE ICCV*. 4310–4318.
[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. 2016. SSD: Single shot multibox detector. In *Proceedings of ECCV*. 21–37.
[5] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. 2018. High performance visual tracking with Siamese region proposal network. In *Proceedings of IEEE CVPR*. 8971–8980.
[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of IEEE CVPR*. 580–587.
[7] R. Girshick. 2015. Fast R-CNN. In *Proceedings of IEEE ICCV*. 1440–1448.
[8] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. 2016. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *Proceedings of ECCV*. 472–488.
[9] N. Wojke, A. Bewley, and D. Paulus. 2017. Simple online and realtime tracking with a deep association metric. In *Proceedings of IEEE ICIP*. 3645–3649.
[10] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. 2017. ECO: Efficient convolution operators for tracking. In *Proceedings of IEEE CVPR*. 6638–6646.
[11] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. 2016. Fully-convolutional Siamese networks for object tracking. In *Proceedings of ECCV*. 850–865.
[12] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[13] M. C. Lee and W. G. Chen. 1999. *U.S. Patent No. 5,970,173*. Washington, DC: U.S. Patent and Trademark Office.
[14] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M. H. Yang. 2018. Online multi-object tracking with dual matching attention networks. In *Proceedings of ECCV*. 366–382.

[15] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu. 2017. Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism. In *Proceedings of IEEE ICCV*. 4836–4845.
[16] Z. Huang, J. Zhan, H. Zhao, K. Lin, P. Zheng, and J. Lv. 2019. Real-time visual tracking base on SiamRPN with generalized intersection over union. In *Proceedings of BICS*. 96–105.
[17] S. Cui, S. Tian, and X. Yin. 2019. Combined correlation filters with Siamese region proposal network for visual tracking. In *Proceedings of ICONIP*. 128–138.
[18] W. Feng, Z. Hu, W. Wu, J. Yan, and W. Ouyang. 2019. Multi-object tracking with multiple cues and switcher-aware classification. arXiv:1901.06129
[19] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. 2016. MOT16: A benchmark for multi object tracking. arXiv:1603.00831
[20] L. Wen, D. Du, Z. Cai, Z. Lei, M. C. Chang, H. Qi, and S. Lyu. 2015. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. arXiv:1511.04136
[21] S. S. Deutsch. 2019. *Siamese Networks for Visual Object Tracking*. Ph.D. Dissertation. Universitat Politècnica de Catalunya, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Spain.
[22] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, and V. K. Asari. 2018. The history began from AlexNet: A comprehensive survey on deep learning approaches. arXiv:1803.01164
[23] Z. Huang, J. Zhan, H. Zhao, K. Lin, P. Zheng, and J. Lv. 2019. Real-time visual tracking base on SiamRPN with generalized intersection over union. In *Proceedings of BICS*. 96–105.
[24] Z. Zhang and H. Peng. 2019. Deeper and wider Siamese networks for real-time visual tracking. In *Proceedings of IEEE CVPR*. 4591–4600.
[25] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. 2019. SiamRPN++: Evolution of Siamese visual tracking with very deep networks. In *Proceedings of IEEE CVPR*. 4282–4291.
[26] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. 2018. High performance visual tracking with Siamese region proposal network. In *Proceedings of IEEE CVPR*. 8971–8980.
[27] D. Li, X. Wang, and Y. Yu. 2019. Siamese visual tracking with deep features and robust feature fusion. In *Proceedings of IEEE ICCE-Asia*. 16–34.
[28] L. Zheng, M. Tang, Y. Chen, J. Wang, and H. Lu. 2020. Siamese deformable cross-correlation network for real-time visual tracking. *Neurocomputing* 401 (2020), 36–47.
[29] R. D. Keane and R. J. Adrian. 1992. Theory of cross-correlation analysis of PIV images. *Applied Scientific Research* 49, 3 (1992), 191–215.
[30] N. Dehak, R. Dehak, J. R. Glass, D. A. Reynolds, and P. Kenny. 2010. Cosine similarity scoring without score normalization techniques. In *Proceedings of Odyssey*. 15.
[31] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. 2019. SiamRPN++: Evolution of Siamese visual tracking with very deep networks. In *Proceedings of IEEE CVPR*. 4282–4291.
[32] L. I. Kuncheva. 2010. Full-class set classification using the Hungarian algorithm. *International Journal of Machine Learning and Cybernetics* 1, 1-4 (2010), 53–61.
[33] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, and L. Wixson. 2000. *A System for Video Surveillance and Monitoring*. Final Report. VSAM.
[34] F. Bashir and F. Porikli. 2006. Performance evaluation of object detection and tracking systems. In *Proceedings of IEEE PETS*. 7–14.
[35] A. S. Abdel-Aziz, A. E. Hassanien, A. T. Azar, and S. E. O. Hanafi. 2013. Machine learning techniques for anomalies detection and classification. In *Proceedings of SecNet*. 219–229.
[36] E. Bochinski, T. Senst, and T. Sikora. 2018. Extending IoU based multi-object tracking by visual information. In *Proceedings of IEEE AVSS*. 1–6.
[37] G. Chandan, A. Jain, and H. Jain. 2018. Real time object detection and tracking using deep learning and OpenCV. In *Proceedings of ICIRCA*. 1305–1308.
[38] W. Lotter, G. Kreiman, and D. Cox. 2015. Unsupervised learning of visual structure using predictive generative networks. arXiv:1511.06380
[39] M. J. Shafiee, B. Chywl, F. Li, and A. Wong. 2017. Fast YOLO: A fast you only look once system for real-time embedded object detection in video. arXiv:1709.05943
[40] R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang. 2016. A Siamese long short-term memory architecture for human re-identification. In *Proceedings of ECCV*. 135–153.
[41] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. 2017. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of IEEE ICCV*. 618–626.
[42] M. D. Zeiler and R. Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of ECCV*. 818–833.

[43] L. Lin, G. Wang, W. Zuo, X. Feng, and L. Zhang. 2016. Cross-domain visual matching via generalized similarity measure and feature learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 6 (2016), 1089–1102.
[44] R. Jonker and T. Volgenant. 1986. Improving the Hungarian assignment algorithm. *Operations Research Letters* 5, 4 (1986), 171–175.
[45] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell. 2016. Understanding data augmentation for classification: When to warp? In *Proceedings of DICTA*. 1–6.
[46] Y. Wu, J. Lim, and M. H. Yang. 2015. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 9 (2015), 1834–1848.
[47] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille. 2018. Single-shot object detection with enriched semantics. In *Proceedings of IEEE CVPR*. 5813–5821.
[48] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M. H. Yang. 2018. Online multi-object tracking with dual matching attention networks. In *Proceedings of ECCV*. 366–382.
[49] S. Tang, M. Andriluka, B. Andres, and B. Schiele. 2017. Multiple people tracking by lifted multicut and person re-identification. In *Proceedings of IEEE CVPR*. 3539–3548.
[50] C. Shen, Z. Jin, Y. Zhao, Z. Fu, R. Jiang, Y. Chen, and X. S. Hua. 2017. Deep Siamese network with multi-level similarity perception for person re-identification. In *Proceedings of ACM MM*. 1942–1950.
[51] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. 2017. Online multi-target tracking using recurrent neural networks. In *Proceedings of AAAI*. 4225—4232.
[52] Z. He, J. Li, D. Liu, H. He, and D. Barber. 2019. Tracking by animation: Unsupervised learning of multi-object attentive trackers. In *Proceedings of IEEE CVPR*. 1318–1327.
[53] Y. C. Yoon, D. Y. Kim, K. Yoon, Y. M. Song, and M. Jeon. 2019. Online multiple pedestrian tracking using deep temporal appearance matching association. arXiv:1907.00831
[54] W. Feng, Z. Hu, W. Wu, J. Yan, and W. Ouyang. 2019. Multi-object tracking with multiple cues and switcher-aware classification. arXiv:1901.06129
[55] C. Yan, B. Gong, Y. Wei, and Y. Gao. 2020. Deep multi-view enhancement hashing for image retrieval. arXiv:2002.00169
[56] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. 2016. MOT16: A benchmark for multi-object tracking. arXiv:1603.00831
[57] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T. K. Kim. 2014. Multiple object tracking: A literature review. arXiv:1409.7618
[58] Y. Zhang, D. Wang, L. Wang, J. Qi, and H. Lu. 2018. Learning regression and verification networks for long-term visual tracking. arXiv:1809.04320
[59] A. Sadeghian, A. Alahi, and S. Savarese. 2017. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *Proceedings of ICCV*. 300–311.
[60] J. Yin, W. Wang, Q. Meng, R. Yang, and J. Shen. 2020. A unified object motion and affinity model for online multi-object tracking. In *Proceedings of CVPR*. 6768–6777.
[61] P. Chu, H. Fan, C. C. Tan, and H. Ling. 2019. Online multi-object tracking with instance-aware tracker and dynamic model refreshment. In *Proceedings of IEEE WACV*. 161–170.
[62] P. Chu and H. Ling. 2019. FAMNet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking. In *Proceedings of ICCV*. 6172–6181.
[63] N. An. 2020. *Anomalies Detection and Tracking Using Siamese Neural Networks*. Master's Thesis. Auckland University of Technology, New Zealand.
[64] W. Yan. 2020. *Computational Methods for Deep Learning*. Springer.
[65] W. Yan. 2019. *Introduction to Intelligent Surveillance—Data Capture, Transmission, and Analytics* (3rd ed.). Springer.

Received July 2020; revised December 2020; accepted December 2020