An Object Detection and Tracking Algorithm Combined with Semantic Information

Qingbo Ji[1,2], Hang Liu[1], Changbo Hou[1,2(✉)], Qiang Zhang[1], and Hongwei Mo[3]

[1] College of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, HLJ, China
houchangbo@hrbeu.edu.cn

[2] Key Laboratory of Advanced Marine Communication and Information Technology, Ministry of Industry and Information Technology, Harbin Engineering University, Harbin 150001, HLJ, China

[3] College of Intelligence Systems Science and Engineering, Harbin Engineering University, Harbin 150001, HLJ, China

Abstract. This paper proposes a novel algorithm for object detection and tracking combined with attribute recognition that can be used in embedded systems. The algorithm, which is based on a single-shot multibox detector (SSD) and kernelized correlation filter (KCF), can distinguish other objects similar to the tracking object, thereby solving the problem of confusion between similar objects. Different classification tasks in the multi-attribute recognition algorithm share the feature extraction module, which uses depthwise separable convolution and global pooling instead of standard convolution and fully connected (FC) layers, thereby improving the overall recognition accuracy and computational efficiency. Additionally, an attribute weight fine-tuning mechanism is added to improve the overall precision and ensure that different tasks are fully learned according to the degree of difficulty. Moreover, this algorithm reduces the size of the model without decreasing the accuracy, making it possible to be run on an embedded device. The results of experiments performed on OTB-100 demonstrate that a superior accuracy of 82.85% is achieved, and the precision and F1 indicator values reach 69.84% and 70.85%, respectively. The precision rate (PR) and success rate (SR) of the overall algorithm respectively reach 85.34% and 80.88%, which are higher than those achieved by the SSD algorithm. However, the size of the proposed attribute recognition algorithm is only 3.05 MB, which is about 1% of the size of other algorithms, indicating that the proposed algorithm not only improves the overall recognition accuracy, but also effectively reduces the model size.

This work is supported by the National Key Research and Development Program of China under Grant 2018AAA0102702; Natural Science Foundation of Heilongjiang Province (JJ2019LH2398); Fundamental Research Funds for the Central Universities (3072020CFT0801, 3072019CF0801 and 3072019CFM0802).

J. Xiong et al. (Eds.): MobiMedia 2021, LNICST 394, pp. 837–854, 2021.
https://doi.org/10.1007/978-3-030-89814-4_62

Keywords: Object detection and tracking · Attribute recognition · Similar object confusion · Shared feature extraction module

1 Introduction

As the development of artificial intelligence (AI) and embedded devices has become increasingly smarter and autonomous, object detection and tracking have become the key components of real applications, such as autopilot programs, intelligent monitoring, and smart robots. Embedded devices dominated by AI have consistently attracted the attention of researchers. However, limited by the computing and storage capacity of embedded hardware, it is difficult to directly deploy high-performance convolutional neural networks (CNNs) to devices. Therefore, when designing the algorithm network structure, the attribute recognition accuracy, the calculation amount, and the size of the network should all be thoroughly considered. Although the related research on tracking tasks has made great progress, the existing tracking algorithms still face difficulties in overcoming dense objects [1], cross-motion [2], and confusion between similar objects [3]. Moreover, the further technical development of computer vision has resulted in the desire to extract high-level semantic information, i.e., to identify the refined attributes of the target while tracking it. In existing research, detection tracking and attribute recognition algorithms have usually been studied separately, though the achievement of fine-grained attribute recognition results to make up for the defects of the tracking algorithm has not been considered. For example, considering frame sequences with multiple pedestrians, if the trajectories of the people in the image are crossed, a tracking algorithm usually uses the apparent characteristics of other pedestrians to update the model due to the confusion of other pedestrian targets, which eventually leads to tracking the wrong pedestrian. Thus, improving the performance of object detection and tracking via the use of semantic information is one of the goals of detection and tracking algorithms.

The contributions of the present work are summarized as follows:

A multi-attribute object recognition method is integrated into a detection and tracking algorithm. When multiple similar targets are detected, the attribute recognition results are used to screen specific tracking targets;

A detection method based on CNN has been introduced into the tracking model to extract the semantic and spatial features. Thus, the performance of the algorithm has been improved in complex background such as motion occlusion, scale variation;

To reduce the amount of calculation and ensure classification accuracy, the standard convolution in the feature extraction module is replaced with depthwise separable convolution, and global pooling or convolutional layers are used in the classifier instead of fully connected (FC) layers, which can avoid the geometric distortion caused by unifying the size of the input image when using FC layers.

2 Related Work

2.1 Attribute Recognition

Attribute recognition (AR) and classification have been widely studied in the field of computer vision to obtain more detailed information about an object. Related methods primarily depend on contextual information or side information [4] and can be applied to the recognition and classification of faces and pedestrian attributes [5], which plays an important role in the video surveillance field. In contrast to some low-level features, like the histogram of oriented gradients (HOG) [6] and the local binary pattern (LBP) [7], attributes can be considered as high-level semantic information [8]. Therefore, training a model by using attributes can improve its robustness.

There are two ways to achieve pedestrian attribute recognition (PAR), namely via traditional and deep learning methods. Traditionally, more convincing features could be obtained by manually labeling the features, training more effective classifiers, and associating the features with attributes. Compared with deep learning methods, traditional methods like HOG, scale-invariant feature transform (SIFT)[9], and the support vector machine (SVM) [10] are not very robust, because they all use manually-designed, low-level features.

Deep learning methods can be classified as either multi-label or multi-task learning methods, and have achieved appreciable results in the attribute recognition field.

Multi-label learning treats all classifications as one task, and uses an FC layer to classify attributes. The prediction results are not in a single category, but are a list of attributes. Zhu et al. [11] proposed a multi-label convolutional neural network (MLCNN), which predicts multiple attributes in the same framework; the authors divided a person in an image into several parts, which were then input into MLCNN via the detection of body parts. Finally, the softmax function was used to classify the attributes.

Regarding multi-task learning, the front of the neural network shares parameters, and the multiple FC layers in the back of the network are juxtaposed. Abdulnabi et al. [12] proposed a joint multi-task attribute classification algorithm based on CNN called multi-task CNN (MTCNN). The model takes the entire image as the input, shares features among different attributes, and reduces the loss by end-to-end training. As compared with traditional feature extraction methods, multi-task-based AR algorithms can extract deeper features and are more robust. In addition, the correlations between different attributes are used to share features, which improves the calculation efficiency and overall accuracy.

2.2 Neural Network Detection Framework

Object detection has been the basis of computer vision tasks, the goal of which is to extract the region of interest (ROI) from images or videos [13]. The existing object detection algorithms can be classified into two categories, namely

region proposal-based two-stage detection and one-stage detection, based on the regression method used to predict the object location and category.

Two-stage algorithms mainly use a combination of region proposal with a CNN to locate and classify the objects. Girshick et al. have successfully proposed R-CNN [14], fast R-CNN [15], and faster R-CNN [16]. R-CNN uses a selective search to select the region, and its feature vector is extracted by a CNN. It then conducts classification via an SVM and finally uses a regressor for box regression. However, disadvantages of R-CNN are that the CNN and SVM are trained separately and each region proposal extracts feature vectors, which causes poor performance in real-time scenarios. Moreover, it takes too long to process each image, which necessitates more space and time while training.

Fast R-CNN and faster R-CNN have been proposed to overcome the shortcomings of R-CNN. Fast R-CNN extracts features of the entire image instead of an ROI. In contrast, faster R-CNN uses a region proposal network (RPN) instead of a selective search to select the candidate region, which improves the running speed of the algorithm. However, these algorithms still cannot meet real-time requirements.

Although one-stage methods do not perform as well as two-stage methods in terms of accuracy, they are superior in speed. Many representative algorithms have been proposed, including you only look once(YOLO) [17], the DenseBox and the single-shot multi-box detector (SSD) [18]. The DenseBox [19] network based on VGG-19 first utilizes an image pyramid and pays more attention to small and severely obscured targets. YOLO, an evolution of GoogleNet, is a single CNN that can predict multiple bounding boxes and classification possibilities. However, its shortcomings are obvious; compared with SSD, YOLO has a lower accuracy rate and larger errors for the recognition of small objects.

2.3 Object Tracking Methods

Recently, with the improvement of computing power and the expansion of labeled datasets, many object tracking methods like multi-domain network(MDNet) [20] and tree structure based CNN(TCNN) [21] have made great progress via deep learning techniques. However, deep learning algorithms are expensive in terms of computation and use online fine-tuning, and it is difficult for them to meet real-time performance requirements. Thus, researchers have made improvements to the tracking speed, such as with the Staple [22], generic object tracking using regression network(GOTURN) [23], and large margin object tracking with circulant feature maps(LMCF) [24] algorithms, among which GOTURN can achieve a tracking speed of 100 FPS. The existing object tracking methods can be classified into two categories, namely discriminative tracking algorithms and generative tracking algorithms.

Generative tracking algorithms [25] model the target area of the current frame and determine a similar area in the next frame as the prediction area. Representative methods include the mean-shift method [26], Kalman filter tracking, and the particle filter tracking algorithm [27]. Researchers have also proposed an adaptive object tracking method based on the mean-shift method, which solves

the problem of tracking failure caused by constant changes in the target scale in videos. Moreover, the video processing speed can reach 125 FPS. However, because they do not consider the background information, these methods have low accuracy despite their fast processing speeds.

In contrast, discriminative object tracking algorithms borrow ideas from machine learning, and treat the tracking problem as two classification problems. In the current frame, the target and background areas are treated as two types of training samples, and the optimal discriminant function is trained via the machine learning method. In the process of tracking subsequent frames, the area with the smallest optimal discriminative function is the target area. The circulant structure of tracking-by-detection with kernels (CSK) algorithm [28] utilizes the cycle sample matrix as the dataset of the classifier, which greatly reduces the amount of calculation and radically increases the tracking speed. However, the CSK tracking algorithm is performed in the gray space, which lacks color information. Moreover, it can only track objects with a fixed size, and cannot robustly track objects in the case of changing scales. The kernel correlation filter (KCF) [29] has been proposed to improve the accuracy while ensuring speed, and uses the fast HOG instead of single-channel gray features. Compared with generative tracking algorithms, discriminative tracking algorithms perform better in terms of speed and accuracy. Moreover, the proposal of the KCF has solved the problem of the large amount of calculation; thus, it was chosen as the object tracking algorithm used in the present study.

3 Proposed Method

This paper proposes an object detection and tracking algorithm combined with attribute recognition that can be applied in embedded systems. Figure 1 presents the flow chart of the proposed method. We use the SSD to detect the target position in the first frame as the KCF initialization coordinates and use the attribute recognition network to identify and record the attribute characteristics of the target. To reduce the error accumulation of the KCF, SSD is utilized to obtain the location information of the object every N frames, and the obtained object area is used as the initial value of the KCF tracking algorithm to initial the model. Since the SSD target detection model may detect multiple targets of the same category or even multiple similar targets, multiple attributes of the target are obtained through the attribute recognition algorithm, and multiple targets in the detection results are compared with the attributes of the previously tracked targets to filter out the attributes. We filter out one or several targets with the closest attributes, and select the target with the closest Hamming distance between these targets and the tracking result of the KCF algorithm as the final tracking target and initialize the KCF algorithm. Both the template and the tracking result of KCF are encoded into binary codes and then calculated the distance to assist in identifying whether the target is well tracked.

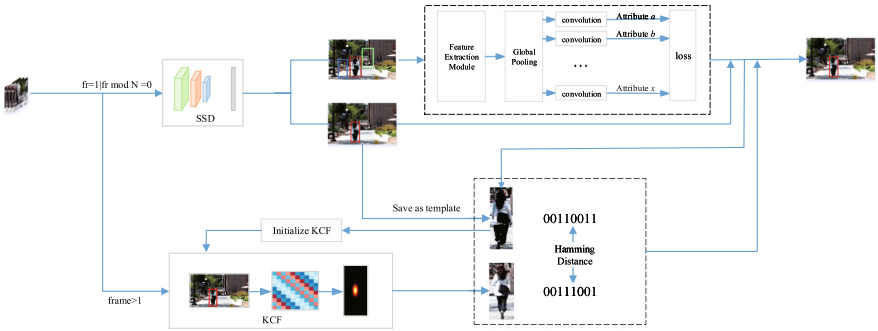


Fig. 1. Flowchart of the proposed object detection and tracking algorithm with attribute recognition.

3.1 Network Architecture

Because the fine-grained multi-task attribute algorithm places high requirements on the computing power and storage sources of the device, the application scenarios of the proposed method are mostly mobile devices with limited computing and storage capabilities, such as self-driving cars and unmanned aerial vehicles, which makes it difficult to deploy high-performance CNNs directly into the devices [30]. Therefore, when designing the algorithm network structure, not only the attribute recognition accuracy of the network, but also the calculation amount and size of the network, must be thoroughly considered.

Attribute Recognition Network Architecture. In natural scenarios, the attribute categories of an object are diverse. For example, considering attribute recognition for pedestrians, there are different attributes, such as gender, clothing style, hair length, etc. Regarding the task of car attribute recognition, the attributes include the color, make, model, and type. When using the deep learning method to solve different tasks, a one-to-one single-task learning mechanism is usually adopted, for which a network is designed separately for each task and different networks solve different tasks. Although the single-task learning mechanism is highly targeted, it ignores the internal connections between different tasks and lacks diversified information that can be borrowed from other attributes, thereby reducing the overall recognition accuracy. With the increase of the number of attributes, the calculation amount of the single-task learning mechanism increases linearly, and the overall calculation efficiency is extremely low. Because different attributes are related to each other, it is possible that the potential information of one task will help improve the recognition accuracy of the model on another task. For example, pedestrians with a male gender usually have short hair, and those with short skirt attributes in their dress style are usually female. Therefore, a multi-task-based recognition algorithm can identify the attributes of the target, the connections between the tasks can be used, and

the shared feature representations between different tasks can be fully mined, thereby allowing the model to better summarize the overall attributes.

Multi-task-based attribute recognition can be classified as either hard-sharing or soft-sharing [31], and Fig. 2 presents the structures of the two types of sharing. In soft-sharing, each task has its own model parameters, and regularization is introduced to make the parameters between different tasks as similar as possible. In contrast, hard-sharing fully considers the sharing of the information source among different parallel networks. The common attributes of multi-tasks are learned while sharing the hidden layer, and the classification problems of different tasks are then completed via the specific task layer. If the number of tasks is large, the shared hidden layer of the model needs to extract as many common effective features as possible to meet the needs of different tasks. Thus, the hard-sharing method can also reduce the risk of model overfitting.

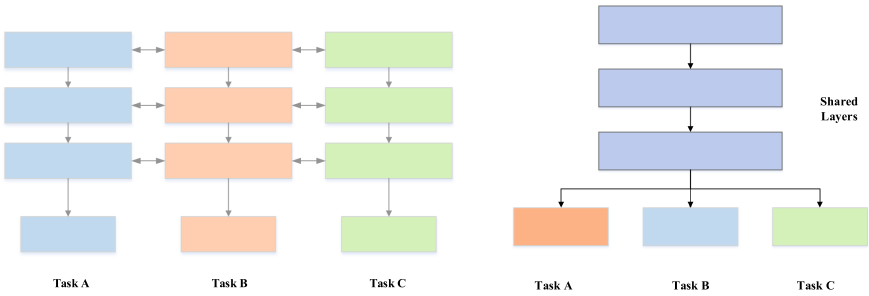


Fig. 2. Parameter sharing of multi-task-based attribute recognition. The left represents soft-sharing and the right represents hard-sharing module.

Based on the preceding analysis, the hard-sharing mode of multi-task recognition is used to design the attribute recognition network, as presented in Fig. 3. In the network, different attributes share a feature extraction module. A global pooling layer is used to reduce the number of parameters and plays a role in the FC layer to obtain global features. A convolutional layer is then used as a classifier for different attributes, and the loss of each classifier is weighted as the overall loss value.

Network of the Feature Extraction Module. In the proposed method, traditional convolution is replaced with a depthwise separable convolutional network to compress the storage and computation on the algorithm level, which significantly reduces the computing requirements and storage overhead on embedded systems. Usually, several FC layers are connected in the feature extraction module in the CNN, and the feature map is mapped into a fixed-length feature vector. In other words, each node of the FC layer is connected to all the nodes of the previous layer, and is used to synthesize the acquired features and acts as a classifier in the CNN. However, there is a flaw in the FC layer, namely the huge

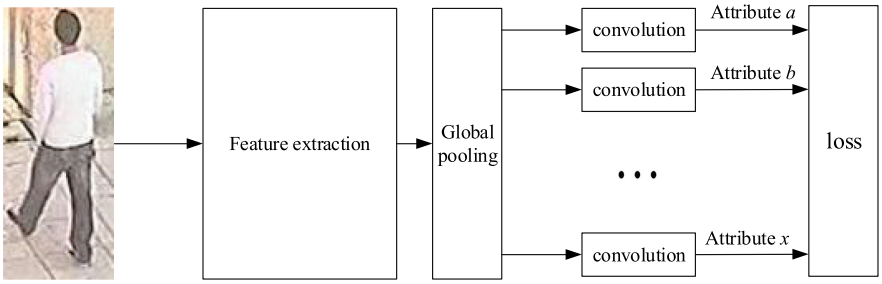


Fig. 3. The attribute recognition network.

number of parameters. Taking VGGNet [32] for example, the size of the feature map that the feature extraction module outputs to the FC layer 1 is 7×7×512, the FC layer 1 has 4096 neurons, and it has 102,70,468 parameters, accounting for 74% of the total number of model parameters of 138M. Behind the FC layer, there are two other FC layers with 16,777,216 and 4,096,000 parameters, respectively. The three FC layers account for about 90% of the total number of parameters. This brings about two problems, as follows.

(1) A large number of parameters results in the substantial consumption of memory and power, and also limits the operating speed. In embedded systems, not only does a large number of parameters occupy the limited space, but, more importantly, there are some differences between the on-chip memory and the off-chip memory (double-rate synchronous dynamic RAM for weights). The RAM of the embedded systems takes up valuable chip area, so the capacity is usually small (the capacity of the Xilinx Zynq UltraScale + MP Soc is only 32.4 MB), while the capacity of off-chip memory can be very large (the memory capacity of the ZCU developed board is 4 GB) with a low reading speed and high power consumption. To improve the operating speed and reduce the power consumption, access to the off-chip memory must be reduced, which means that the size of the network must be reduced as much as possible.

(2) A large number of parameters brings about overfitting and reduces the generalization performance. Due to the sampling error included in the training set, too many parameters will also fit the sampling error to the model while parameter fitting. The intuitive performance results in the model having a small loss in the training set, but a large loss in the test set.

Global pooling and a convolutional layer can be used instead of FC layers to solve these problems. Figure 4 presents the structures of the FC layer and global pooling layer. The FC layer expands the feature map into a vector and then conducts classification. For example, in VGGNet, the feature extraction module first obtains a 7×7×512 feature map and converts it into a 1×1×512 map through the FC layer, and then connects 1000 neurons to output a fractional vector. The global pooling layer converts the 7×7×512 map into a 1×1×512 map without any parameters. Max pooling or average pooling is then used to

transform the feature map into a 1000-dimensional score vector, and the required parameters are $512 \times 1000 = 0.5$M. It is evident that the number of parameters for global pooling is much less than that of the FC layer. Because the number of global pooling parameters is greatly reduced, overfitting can be avoided, and there is almost no effect on the performance of the global pooling and FC layers. Additionally, if there exists an FC layer in the CNN, the input image must be converted into the same size, which will cause geometric distortion and other factors that will affect the accuracy. Regarding the convolutional layer, the parameters are shared in each channel, and the sizes of the kernel and input image are not related; therefore, the convolutional layer can accept images of different sizes.

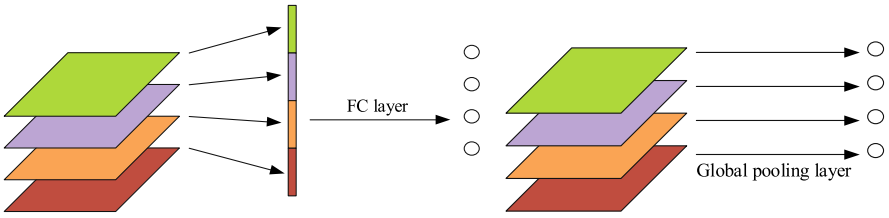


Fig. 4. The structures of the FC layer and global pooling layer.

3.2 Loss Function

The loss function is the end of, and the key to, the learning quality of a CNN, and is widely used to describe the differences between the predicted and true values. Usually, when processing a single-labeled multi-class classification problem, a CNN uses the softmax function as the activation function of the output layer. For example, the output layer of a CNN has k nodes, and the activate function is defined as follows:

$$f(z_j) = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{1}$$

where e^{z_j} is the probability for the j-th class, $f(z_j)$is the probability distribution, and z_j refers to the j-th neuron.

Via the softmax function, the probability value in the output vector is mapped to the interval $(0, 1)$, and each element in the vector is regarded as the confidence of the corresponding category. Considering the implementation of the CNN, the multi-attribute recognition of objects is a multi-label classification task for which the softmax activation function is not suitable. Regarding multi-label classification, the softmax function can be replaced with the sigmoid function to compute the loss. For the i-th sample $X_i \in \mathbb{R}$ in the sample space, and it passes the CNN to obtain an L-dimensional attribute predict vector $\hat{y}_i = [\hat{y}_{i1}, \hat{y}_{i2}, ..., \hat{y}_{iL}] \in \mathbb{R}^L$. It then predicts every element $\hat{y}_{im}$ in the vector $\hat{y}_i$, where $m \in [1, L]$ indicates the confidence of the corresponding attributre.

The true label of the sample $X_i \in \mathbb{R}$ is $y_i = [y_{i1}, y_{i2}, ..., y_{iL}] \in \mathbb{R}^L$. Each value of element y_{im} in the true label vector is shown in Eq. (2), where $m \in [1, L]$ indicates whether the attribute corresponding to the sample exists.

$$y_{im} = \begin{cases} 1, & \textit{if this attribute exists} \\ 0, & \textit{otherwise} \end{cases} \tag{2}$$

The DeepMAR algorithm trains the input image and multi-attribute labels, and uses the sigmoid cross-entropy loss to jointly consider multiple attributes. Aiming at the problem of the uneven distribution of multi-attribute label loss values, the cross-entropy loss function is improved as follows:

$$\begin{cases} L_{wce} = -\dfrac{1}{N}\displaystyle\sum_{i=1}^{N}\sum_{l=1}^{L} w_l(y_{il}log(\hat{p}_{il}) + (1 - y_{il})log(1 - \hat{p}_{il})) \\ w_l = e^{-\frac{p_l}{\sigma^2}} \end{cases} \tag{3}$$

where y_{il} refers to the true label, $\hat{p_{il}}$ refers to the predicted score of the l-th attribute for sample X_i, w_i refers to the weight of the l-th attribute, p_l refers to the propotion of positive samples of the l-th attribute in the training set, and σ refers to a hyperparameter. By improving the sigmoid cross-entropy loss function, attributes with a small proportion of positive samples can be given larger weight values; therefore, when the attribute is incorrectly classified as a negative sample by the CNN, a larger "penalty" will be given to improve the ability to learn when the samples are unbalanced.

However, using the same network structure to classify different attributes results in different classification difficulties. It is obviously inappropriate to determine the weight of loss via the sample ratio. Therefore, the weight penalty mechanism is introduced in consideration of the difficulty of classification. In other words, the network determines the weights of attributes according to the classification difficulties of different attributes, and sets a high loss weight for more difficult classification tasks. The difficulty of classifying different attributes is measured by the loss corresponding to each attribute in the training process.

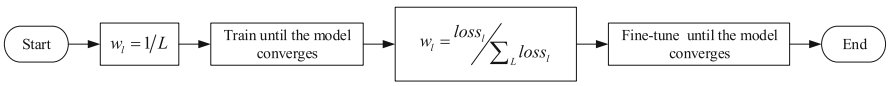


Fig. 5. The process of attribute weighting in the loss function.

In this process, all the attributes are first treated with equal degrees of importance (weight $w_l = \frac{1}{L}$). Next, the model is trained until the loss function remains stable, and the loss $loss_i$ corresponding to each attribute is recorded. The value w_l is then taken as the proportion of the overall loss. Finally, the model is fine-tuned until the loss function remains stable.

4 Experimental Results

4.1 Training Process

The PEdesTrian Attribute (PETA) dataset was used for the training and testing of the attribute recognition network. Each image in the dataset is labeled with 61 binary attributes and 4 multi-valued attributes. In this research, 15 attributes were selected for training and testing: "upper Plaid," "upper ThinStripes," "upper Black," "upper Brown," "upper Green," "upper Gray," "upper Red," "upper White," "upper Yellow," "lower Black," "lower Brown," "lower Gray," "lower White," "lower HotPants," and "lower Long Skirt." For the experiment, 1,000 images were randomly chosen for testing, and the others were used for training.

In the training process, different weights must be assigned for attribute recognition tasks of different difficulty levels (as shown in Fig. 1). Figure 6 presents the trend comparison of the loss functions of the weighted and unweighted tasks when the total number of training iterations was 221.4×10^3. First, the weights of all tasks were made equal, and training was stopped at the 100×10^3-th iteration. The weights were then saved in a file named "caffemodel," based on which a comparative experiment of continuing training and fine-tuning was conducted. In the continuing training process, all the task weights were kept equal, which means no weight was assigned. However, in the fine-tuning process, weights were respectively assigned to different tasks according to the degree of difficulty at the 100×10^3-th and 200×10^3-th iterations.

Figure 6 reveals that, due to the introduction of the weight penalty mechanism, larger loss weights were assigned to the more difficult identification tasks; therefore, the loss of the weighted tasks in the later stage of training became gradually less than that of the unweighted tasks. This proves that the proposed penalty mechanism has a better multi-task learning ability.

4.2 Experimental Results and Analysis

Attribute Recognition Experiment Results and Analysis. To quantitatively evaluate the effectiveness of the proposed algorithm structure and loss function design, different indexes in the PETA dataset were tested, and the performance was compared with those of four existing algorithms that exhibit good performance in multi-attribute recognition tasks. The ELF-em algorithm is a traditional discriminant method that uses integrated features based on color and texture features. The other three comparison algorithms are all based on a CNN. In addition, the size of the model was also taken into consideration to facilitate the implementation of the algorithm on embedded systems.

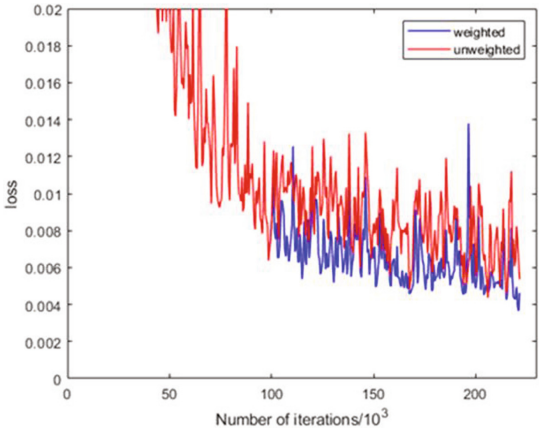


Fig. 6. Comparison between the loss values of the assigned and unassigned weights in the training phase.

Table 1. Evaluation results of attribute recognition by different methods. The comparison algorithms are all target attribute recognition algorithms, and target detection algorithms are not included.

Method	mAP	Accuracy	Precision	Recall	F1	Size of model
ELF-mm	75.21%	43.68%	49.45%	74.24%	59.36%	–
CNN-mm	76.65%	45.41%	51.33%	75.14%	61.00%	217.45
ACN	69.66%	62.61%	80.12%	72.26%	75.98%	232.57
DeepMAR	73.79%	82.60%	74.92%	76.21%	75.56%	232.57
Proposed	70.13%	82.85%	69.84%	71.91%	70.85%	3.05

As presented in Table 1, the accuracy of the proposed method was significantly higher than the accuracies of the other algorithms, which demonstrates that the loss function design of the algorithm and the weighting operation of different attributes in the training process significantly improved the accuracy, especially for the more difficult attributes. However, because different attributes are weighted according to the classification difficulty, the proposed algorithm did not perform as well under the condition of uneven sample data (the PETA dataset has fewer positive samples and more negative samples), which resulted in a poor performance in terms of the mAP, precision, recall, and F1 indicators.

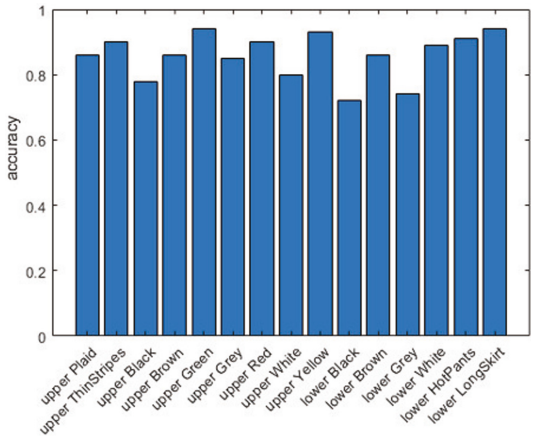


Fig. 7. Accuracy distribution map of each attribute.

In the training process of the model, weights for different tasks were redistributed according to the proportion of classification difficulty at the 100×10^3-th and 200×10^3-th iterations so that the classification tasks of different attributes were fully learned during the training process. In the final result, the accuracy of each attribute was more even and without obvious polarization, as shown in Fig. 7. The accuracies for "lower Black" and "lower Grey" were respectively 0.72 and 0.74, and slightly lower than the accuracies for the other attributes. This is because the two colors of black and grey present large changes in the images under different lighting and angles, which may easily cause misdetection. However, this did not severely affect the overall distribution due to the large loss weights assigned to these two attributes.

Regarding the size of the model, CNN-mm uses AlexNet as its backbone, while ACN and DeepMAR adopt CaffeNet. Many standard convolutions are used in these two backbones, and the size of the convolutional kernel is relatively large. Although the performances of various indicators are improved to a certain extent, the size of the model usually exceeds 200 MB. In the proposed method, the standard convolution is replaced with depthwise separable convolution, which effectively reduces the model size while ensuring greater accuracy; the final size of the model is only 3.05 MB.

Overall Algorithm Experiment Results and Analysis. There are many tracking algorithms that exhibit good real-time performance and robustness, including KCF, Struck, CSK, and Staple. To evaluate the performance of the proposed algorithm, these methods were tested with the SSD detection model on the OTB-100 and LaSOT datasets. To ensure the objectivity and fairness of the experimental results, the public datasets VOC2007 and VOC2012 were adopted for the SSD model training dataset of the proposed algorithm. Because these three datasets are quite different from the OTB-100 and LaSOT test sets,

13 video sequences in the OTB-100 test set ("BlurBody," "Couple," "Girl," "Gym," "Crows," "Human2," "Human6," "Human7," "Human8," "Human9," "Man," "Trellis", and "Woman") and 33 video sequences with a person category (20 test video sequences) in LaSOT were used for testing.

Table 2. Precision rate (PR) and success rate (SR) of different object detection and tracking algorithms. The best results are presented in bold.

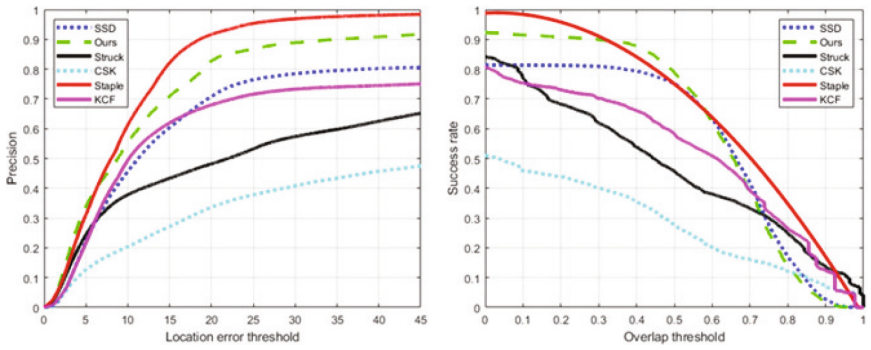
	Proposed	SSD	Struck	CSK	Staple	KCF
PR(%)	**85.34**	75.28	48.98	37.51	93.10	70.51
SR(%)	**80.88**	74.75	46.53	27.49	73.17	60.23

The one-pass evaluation (OPE) method was chosen to test the KCF, Struck, CSK, and Staple tracking algorithms by manually setting the target position in the first frame. The SSD object detection and tracking algorithm, as well as the proposed algorithm, detect the object automatically without providing the initial object position. Table. 2 presents the comparison results.

As presented in Fig. 8, the precision and success results were plotted to verify the feasibility and effectiveness of the proposed algorithm. Compared with the SSD object detection model, the success rate was slightly improved and the precision was increased by nearly 10%. Moreover, compared with the Struck, CSK, and KCF algorithms, the precision and success rates of the proposed algorithm were greatly improved.

Intuitive comparison has also been made to test our detection and tracking method in some sorted video sequence. As Fig. 9 show, the yellow box represents the tracking result of proposed method, and the blue, green, red, purple boxes represent the KCF, CSK, Staple, Struck, respectively.

However, the precision and success rates of the proposed algorithm were not higher than those of the Staple algorithm. This is because the proposed algorithm requires the SSD model to detect the object position as the training base sample of the KCF in the initial frame and in tracking failure situations. Moreover, the datasets VOC2007 and VOC2012 used for training the SSD model are quite different from the test video. In other words, the "person" category in the video sequence of the test set mostly includes images from the perspective of a road monitor, whereas the "person" images in the training set are all images from the horizontal perspective. Moreover, the video sequence of the "bottle" category in the test set is longer than that in the training set.



(a)Precision (b)Success

Fig. 8. Curves of OPE test results.

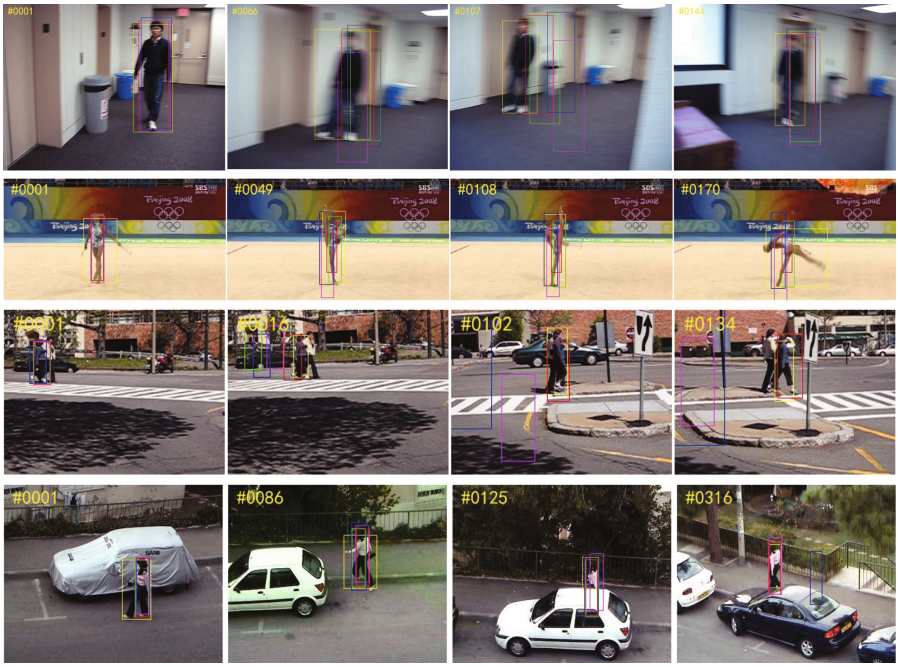


Fig. 9. Intuitive comparison of tracking results in different video sequences.

Due to the large difference between the training and test sets, the performance of SSD was not very good (the precision and success rates were 74.51% and 75.12%, respectively), which reduced the accuracy of the proposed algorithm. To verify this point of view, another set of experiments was conducted by adding approximately 200 pedestrian images from the perspective of a road monitor, which were not included in the test set, to the training set. The test results

after retraining the SSD model are reported in Table 3, which demonstrates that the proposed method exhibited a great improvement, while the performance of SSD was slightly improved with the increase in the number of images in the training set. The precision rate of the proposed method was basically the same as that of the Staple algorithm, while the success rate was 13% higher.

Table 3. Test results after adding images into the training set.

Method	SSD		Proposed	
	Test results	Increase	Test results	Increase
PR%	78.32	3.04	92.29	6.95
SR%	75.83	1.08	86.55	5.67

5 Conclusion

In this work, an object detection and tracking method based on attribute recognition was proposed. This method utilizes the attribute difference of the object to distinguish multiple similar targets to avoid confusion with other objects during tracking. Additionally, to increase the overall recognition accuracy and computing efficiency, the performance is improved via the combine of SSD detection method and KCF tracking algorithm, assisted by shared feature extraction, and global pooling. Experimental results indicate that the recognition accuracy of the proposed method is higher than the accuracies of other algorithms, and the size of the model is significantly reduced. The motion blur, confusion between similar objects and scale variation are well overcome. In follow-up research, an attempt will be made to automatically adjust the weights in the weight penalty mechanism, instead of manually setting fixed parameters in various stages.

References

1. Wang, X., Xiao, T., Jiang, Y., Shao, S., Sun, J., Shen, C.: Repulsion loss: detecting pedestrians in a crowd. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
2. Feichtenhofer, C., Pinz, A., Zisserman, A.: Detect to track and track to detect (2017)
3. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: Proceedings of the 12th European conference on Computer Vision - Volume Part III (2012)
4. Mao, L., Yan, Y., Xue, J.H., Wang, H.: Deep multi-task multi-label CNN for effective facial attribute classification. In: Proceedings of International Conference on Computer Vision and Pattern Recognition (2020)
5. Sarafianos, N., Xu, X., Kakadiaris, I.A.: Deep imbalanced attribute classification using visual attention aggregation (2018)

6. Zhou, W., Gao, S., Zhang, L., Lou, X.: Histogram of oriented gradients feature extraction from raw Bayer pattern images. IEEE Trans. Circuits Syst. II: Express Briefs (99), 1 (2020)
7. Zhao, G., Wang, X., Cheng, Y.: Hyperspectral image classification based on local binary pattern and broad learning system. Int. J. Remote Sens. **41**(24), 9393–9417 (2020)
8. Wang, X., Zheng, S., Yang, R., Luo, B., Tang, J.: Pedestrian attribute recognition: A survey (2019)
9. Song, Z., Zhang, J.: Image registration approach with scale-invariant feature transform algorithm and tangent-crossing-point feature. J. Electron. Imaging **29**(2), 1 (2020)
10. Yu, X., Wang, H.: Support vector machine classification model for color fastness to ironing of vat dyes. Textile Res. J. 004051752199236 (2021)
11. Zhu, J., Liao, S., Lei, Z., Li, S.Z.: Multi-label convolutional neural network based pedestrian attribute classification. Image Vis. Comput. **58**, 224–229 (2017)
12. Abdulnabi, A.H., Wang, G., Lu, J., Jia, K.: Multi-task CNN model for attribute prediction. IEEE Tran. Multimedia (2015)
13. Guan, H., Cheng, B.: Taking full advantage of convolutional network for robust visual tracking. Multimedia Tools Appl. **78**(8), 11011–11025 (2018)
14. Olmez, E., Akdogan, V., Korkmaz, M., Er, O.: Automatic segmentation of meniscus in multispectral MRI using regions with convolutional neural network (R-CNN). J. Digital Imaging **33**(30) (2020)
15. Girshick, R.: Fast R-CNN. Computer Science (2015)
16. 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. **39**(6), 1137–1149 (2017)
17. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: Unified, real-time object detection, You only look once (2015)
18. Chen, X., Yu, J., Wu, Z.: Temporally identity-aware SSD with attentional LSTM. IEEE Transactions on Cybernetics (2018)
19. Shan, Y.: ADAS and video surveillance analytics system using deep learning algorithms on FPGA. In: 2018 28th International Conference on Field Programmable Logic and Applications (FPL), pp. 465–466 (2018)
20. Wang, J., Li, A., Pang, Y.: Improved multi-domain convolutional neural networks method for vehicle tracking. Int. J. Artif. Intell. Tools **29**(07n08), 2040022 (2020)
21. Lee, J., Kim, S., Ko, B.C.: Online multiple object tracking using rule distillated siamese random forest. IEEE Access **8** (2020)
22. He, W., Li, H., Liu, W., Li, C., Guo, B.: rstaple: a robust complementary learning method for real-time object tracking. Appl. Sci. **10**(9), 3021 (2020)
23. Ahmad, M., Ahmed, I., Khan, F.A., Qayum, F., Aljuaid, H.: Convolutional neural network based person tracking using overhead views. Int. J. Distribut. Sensor Netw. **16** (2020)
24. Gao, L., Li, Y., Ning, J.: Maximum margin object tracking with weighted circulant feature maps. IET Comput. Vis. **13**(1), 71–78 (2019)
25. Chen, Y., Sheng, R.: Single-object tracking algorithm based on two-step spatiotemporal deep feature fusion in a complex surveillance scenario. Math. Probl. Eng. (2021)
26. Xiang, Z., Tao, T., Song, L., Dong, Z., Wang, H.: Object tracking algorithm for unmanned surface vehicle based on improved mean-shift method. Int. J. Adv. Rob. Syst. **17**(3), 172988142092529 (2020)

27. Tang, Y., Liu, Y., Huang, H., Liu, J., Xie, W.: A scale-adaptive particle filter tracking algorithm based on offline trained multi-domain deep network. IEEE Access (99), 1 (2020)
28. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels (2012)
29. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Trans. Pattern Anal. Mach. Intell. **37**(3), 583–596 (2014)
30. Zhang, W., Zhang, Z., Zeadally, S., Chao, H.C., Leung, V.C.: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence. IEEE Trans. Ind. Inform. (99), 1 (2019)
31. Sun, T., et al.: Learning sparse sharing architectures for multiple tasks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 8936–8943 (2020)
32. Zamani, N.S.M., Zaki, W.M.D.W., Huddin, A.B., Hussain, A., Mutalib, H.A.: Automated pterygium detection using deep neural network. IEEE Access **8**, 191659–191672 (2020)