sensors

MDPI

Article

Real-Time Multiobject Tracking Based on Multiway Concurrency

Xuan Gong [1,2], Zichun Le [2,*], Yukun Wu [1,3] and Hui Wang [1]

[1] College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China; 1111712011@zjut.edu.cn (X.G.); 1121712004@zjut.edu.cn (Y.W.); 1111712012@zjut.edu.cn (H.W.)
[2] College of Science, Zhejiang University of Technology, Hangzhou 310023, China
[3] School of Artificial Intelligence, Zhejiang Post and Telecommunication College, Shaoxing 312366, China
* Correspondence: lzc@zjut.edu.cn; Tel.: +86-130-8281-9650

Abstract: This paper explored a pragmatic approach to research the real-time performance of a multiway concurrent multiobject tracking (MOT) system. At present, most research has focused on the tracking of single-image sequences, but in practical applications, multiway video streams need to be processed in parallel by MOT systems. There have been few studies on the real-time performance of multiway concurrent MOT systems. In this paper, we proposed a new MOT framework to solve multiway concurrency scenario based on a tracking-by-detection (TBD) model. The new framework mainly focuses on concurrency and real-time based on limited computing and storage resources, while considering the algorithm performance. For the former, three aspects were studied: (1) Expanded width and depth of tracking-by-detection model. In terms of width, the MOT system can support the process of multiway video sequence at the same time; in terms of depth, image collectors and bounding box collectors were introduced to support batch processing. (2) Considering the real-time performance and multiway concurrency ability, we proposed one kind of real-time MOT algorithm based on directly driven detection. (3) Optimization of system level—we also utilized the inference optimization features of NVIDIA TensorRT to accelerate the deep neural network (DNN) in the tracking algorithm. To trade off the performance of the algorithm, a negative sample (false detection sample) filter was designed to ensure tracking accuracy. Meanwhile, the factors that affect the system real-time performance and concurrency were studied. The experiment results showed that our method has a good performance in processing multiple concurrent real-time video streams.

Keywords: single-object tracking; multiobject tracking; tracking-by-detection; real-time; multiway; concurrency

Citation: Gong, X.; Le, Z.; Wu, Y.; Wang, H. Real-Time Multiobject Tracking Based on Multiway Concurrency. *Sensors* **2021**, *21*, 685. https://doi.org/10.3390/s21030685

Academic Editor: SangMin Yoon
Received: 30 December 2020
Accepted: 18 January 2021
Published: 20 January 2021

1. Introduction

Tracking moving targets involves processing and analyzing the video (sequence) images captured by photoelectric sensors and making full use of the information to locate and track the target. Target tracking is also the foundation of computer vision. It has important applications in the field of intelligent monitoring [1], pose estimation [2], motion recognition [3], behavioral analysis [4], automatic driving. For example, in an automatic driving system, target-tracking algorithms need to track cars, pedestrians and animals and predict their positions, speeds, and in the future, perhaps other information.

Target tracking is usually classified into single-object tracking (SOT) and multiobject tracking (MOT) from the perspective of the number of targets tracked. Usually, MOT or multiple target tracking (MTT) is more complex. The main task of MOT is to locate multiple targets of interest simultaneously in a given video sequence and maintain their identifications and record their trajectories [5]. Simultaneously, MOT needs to solve several key problems: (1) determine the number of targets (usually changing with time) and maintain their respective identification, (2) frequent occlusion, (3) initialization and termination of a track, (4) similar appearance, (5) conflict between multiple objectives [1].

In practical applications, such as intelligent video application systems and intelligent safety monitoring systems, target tracking often plays an important role. The result of tracking is often used as the input to a correlation processing module, an intelligent video application system that integrates a target-tracking algorithm. However, a real scene of target tracking is often very complex. The numbers and categories of tracked objects are usually not unitary, for example, multiple object categories include pedestrians, vehicles, and nonmotor vehicles. Usually, tracking algorithms solve some common performance problems, such as occlusion or deformation; the complexity of the algorithms also increases linearly. However, in some practical applications, we must face two basic problems: (1) How to process multiple concurrent real-time video streams effectively with MOT under the condition of limited computing resources and storage resources. For example, one GPU (graphics processing unit) card may be used to process more than twenty video streams with MOT. In this case, if the processing speed of the tracking is too low, the "frame loss" phenomenon will occur frequently, resulting in tracking abnormalities. (2) How to balance real-time tracking and algorithm performance. That is, how to effectively improve real-time MOT, keeping system performance within an acceptable range—this is the emphasis in practice. In actuality, other computer vision algorithms will face the same problem also, such as object recognition [6].

Aims of this paper were as follows: (1) Propose one kind of new MOT framework that supports processing of multiway real-time image sequences. Before that, there was no relevant research. (2) Study how to trade off MOT real-time and algorithm performance and design an MOT algorithm based on detection directly. (3) Open new horizons for researchers in the field of high concurrency and real-time processing for MOT. In addition, we also utilized the inference optimization features of NVIDIA TensorRT to accelerate the deep neural network (DNN) processing in the tracking algorithm.

2. Related Works

The real-time performance of a tracking algorithm is usually very important in practice. For example, if processing speed is 30 fps in an MOT system, then only one video sequence with 25 fps frame rate can be supported, which is not sufficient for satisfying large-scale video access in a practical application.

In recent years, there has been research on MOT real-time. The research has mainly two aspects: tracking framework research and optimization research based on specific hardware and the toolkit environment.

MOT based on tracking by detection (TBD) has become a topic of research in recent years. Alex Bewley has proposed SORT (simple online and real time tracking) [7] tracking technology; this effectively implements faster RCNN (Region-based Convolution Neural Networks)-based detection and tracking using Kalman filtering [8] and the Hungarian algorithm [9] and improves the tracking speed and accuracy. Its frame rate can reach 260 fps. Nicolai Wojke et al. proposed DeepSort (simple online and real time tracking with a deep association metric) [10] based on SORT; this uses deep features learned in the task of person reidentification [11], a greatly reduced ID switch, and solved the problem of short-term target occlusion. Due to the introduction of deep appearance features for a similarity metric, the speed of tracking is approximately 30% slower than the SORT algorithm. In addition, multiple hypothesis tracking (MHT) [12] facilitates precise target detection by maintaining a small list of assumptions, and it adopts an online training appearance model for each trace hypothesis and branch. These need only some extra operations to efficiently learn to the appearance of the model with the regular least squares method framework. However, the complexity of the algorithm will rise as the hypothesis branch increases, especially in scenes with more tracking targets. The tracking speed is only 0.8 fps. In [13], a CNN-based framework was proposed to apply simple single-target tracking to MOT. In terms of computing efficiency, features of CNN are shared, and individual information of each target is obtained by ROI (Region Of Interest) pooling. Concurrently, a spatiotemporal attention mechanism (STAM) is introduced to control the

drift problem caused by the interaction between occlusion and the target. The complexity of the algorithm is relatively high, and the tracking speed is only 0.5 fps with a certain CPU power. In [14], a joint detection and embedding (joint detection and embedding, JDE) mode was proposed that uses a single network to output the detection results and the corresponding appearance embedding together. In contrast, the SDE (separate detection and embedding) [10] method and the two-stage method take resampled pixels (bounding box) and a feature map as features, respectively. The MOTA (Multiple Object Tracking Accuracy) of the method was 64.4% on the MOT16 [15] test sequence, obtaining a running time of 18.8 fps. K. Fang et al. [16] proposed a novel recurrent autoregressive network (RAN), which couples an external memory and an internal memory to capture the history and the characteristics of an object during tracking. The external memory explicitly stores previous inputs of each trajectory in a time window, while the internal memory learns to summarize long-term tracking history and associate detections by processing the external memory. The algorithm has good performance on MOT15 [17] and MOT16 [15] datasets and in highly crowded and occluded scenes. Due to the fact that each object needs a RAN, the system needs more space and time consumption, which is not suitable for high concurrency video processing. In [18], authors solved the problem of motion noise and ambiguities between long ranged objects based on a unified framework. This approach was based on the strategy to join the association in both structural and temporal domains. In terms of processing efficiency, the framework was decomposed into a three-stage scheme within an alternative optimization fashion and the processing time was decoupled from the tracklet length via handling association for tracklets-to-tracks based on motion patterns. The average frame rate can reach 10 fps. In [19], the author combined Siamese structure and random forest (RF) and improved the matching performance and solved existing slow CNN-based tracking issues for MOT via a shared-rule based Siamese structure. The algorithm considers both performance and real-time, the processing speed was 12.4 fps. In [20], a new model TubeTK of end-to-end one-step training was proposed for MOT tasks. It utilizes tubes to encode target's temporal-spatial position and local moving trail. The algorithm achieves the new state-of-the-art performances compared with other online models. In [21], the author focused on solving the problem of tracking trajectory integrity and introduced an iterative clustering method that generates more tracklets while maintaining high confidence. In terms of motion and appearance, two evaluation networks of motion evaluation network and appearance evaluation network were constructed to learn long-term features of tracklets for association. The whole algorithm mainly focuses on performance improvement, not real-time performance.

Generally, real-time optimization at the tracking-algorithm level (including the tracking framework) is usually limited, so optimizing acceleration based on a specific hardware and toolkit environment has become a research focus. Some companies, such as NVIDIA and Intel, have developed customized accelerators, such as TensorRT (https://developer.nvidia.com/tensorrt) or a neural-computing SDK. They use images created by TensorFlow or Caffe as input and convert them into an accelerated format of inference hardware [22]. Other scholars have also researched DNN acceleration with a GPU. Paras Jain et al. [23] improved the utilization rate of a GPU in a deep-learning inference load by studying time–space multiplexing technology; their experiments showed that the GPU utilization rate increased by a factor of five with such a strategy. In [24], inference, 16-bit quantization, and a CPU multithread based on TensorRT were used to accelerate image processing and good results were achieved. In the field of embedded navigation, Anish Singhani et al. [25] used NVIDIA TensorRT to accelerate the inferencing of a neural network in mobile robot navigation to improve the navigation performance; TensorRT improved inference performance by a factor of nearly three.

As yet, most research on the real-time performance of MOT is mainly based on one-way video stream or image sequence. These cannot satisfy practical applications. In this paper, we mainly focused on the MOT efficient processing of multiple real-time video

streams (or image sequences). The experiment results show that our method has a good performance in processing multiple concurrent real-time video streams.

3. Methodology

This section mainly focuses on the study of new MOT framework to support concurrent process of multiway real-time video stream. The new MOT framework was expanded in width and depth based on a tracking-by-detection model. In terms of width, the framework can support the process of multiway video sequences at the same time by introducing batch processing mechanism and multiple tracker instance; in terms of depth, image collectors, dispatchers, and filters of negative samples were introduced to meet system performance requirement. In addition, one kind of real-time MOT algorithm based on directly driven detection was introduced to meet the real-time performance and multiway concurrency ability.

3.1. MOT Framework Of Multiway Concurrency

In this section, we propose an MOT framework based on a multiway concurrency scenario. The framework mainly consists of four parts: image collector, target detector, task dispatcher, and target tracking, as shown in Figure 1:

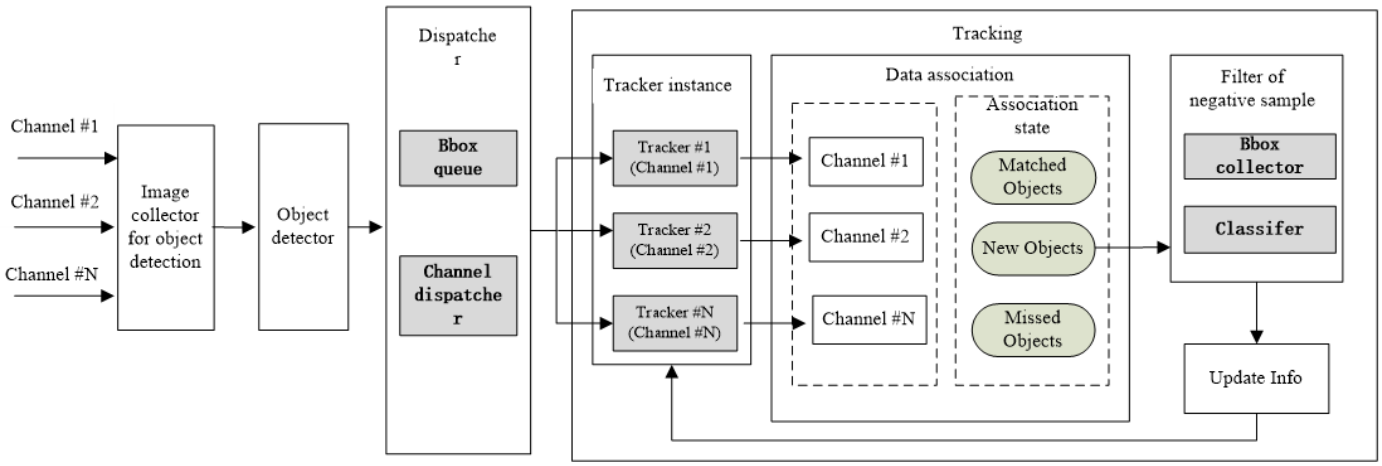


Figure 1. The multiobject tracking (MOT) framework based on multiway concurrency.

In operation, the image collector mainly collects multiway frame sequences and packs each sequence into one image bundle to be used as input to the detector. The bundle consists of a multidimensional vector:

$$\mathbf{B} = [\mathbf{b}_1, \mathbf{b_2}, \ldots \mathbf{b_n}] \quad n \in \max_{channel} . \tag{1}$$

Each element of the bundle consists of a four-dimensional vector as shown in Equation (2):

$$\mathbf{b} = \left[d_{frame}, s_{width}, s_{height}, c_{index} \right]. \tag{2}$$

d_{frame}, s_{width}, s_{height}, and c_{index} denote frame data, frame width, frame height, and channel information, respectively. Multidimensional vector **B** contains the input parameters for the detector inference. The output state of each detector target is modeled as shown in Equation (3):

$$\mathbf{x} = [x_{topleft}, y_{topleft}, x_{bottomright}, y_{bottomright}, l, s]. \tag{3}$$

The first four parameters are, respectively, the upper left and lower right corner coordinates of the target bounding box, and the last two parameters are label value and score, respectively. Each target datum (bounding box) will be packaged into a bundle with the appropriate channel identification, that is the channel id: $channel_i \in \{0, 1, \ldots N\}$,

$bundle = \{\mathbf{x}, channel_i\}^{i \in K}$; K is a configurable parameter that determines the size of a bundle. The packaged data will be put into the bounding box queue.

Considering the unbalanced processing speeds of the detector and tracker, we designed one task dispatcher module. The module is mainly responsible for extracting the target bbox (Bounding Box) from the bbox queue and dispatching the corresponding target bounding box to the tracker according to the channel id. Each image sequence corresponds to one tracker (one tracker instance will be generated for one new image sequence). The operation of the queue is executed in asynchronous mode.

The tracking part is the focus of this paper. We propose a tracking algorithm based on detection to satisfy real-time performance requirements; one classification filter is designed to filter negative samples, which will improve tracker performance. The tracking part will be introduced in detail in Section 3.3.

3.2. Multitarget Detection

At present, in research on multitarget tracking, the framework based on TBD is still dominant. One of the main reasons can be attributed to the application of deep learning in target recognition. The performance of the detector plays an important role in tracking. Mainstream target detection algorithms can be divided into two categories: two-stage, e.g., single shot multiBox detector (SSD) [26], you only look once (Yolo) [27,28] and one-stage, e.g., RCNN [29], faster-CNN [30–33]. The two categories of detection algorithms have their own advantages and disadvantages. Two-stage detection algorithms divide the detection task into two stages, namely, detection and classification. This kind of algorithm has high accuracy, but its detection speed is slow, so it is not suitable for scenes with strict real-time requirements; one-stage algorithms integrate the two stages into one stage. The real-time performance of this kind of algorithm is better, but its accuracy is slightly worse. Currently, Yolo has been developed to Yolov3 [34], improving the prediction accuracy, especially for small-object recognition on the premise of maintaining the speed advantage. Therefore, Yolov3 was chosen as the detection algorithm in this paper.

Yolov3 adopts an architecture similar to that of feature pyramid networks (FPN) for object detection to realize multiscale prediction (Yolov3 predicts three different scale boxes). In basic image feature extraction, Yolo3 uses 3×3 and 1×1 convolution to design a network of 53 convolution layers, darknet-53. At the same time, it uses the residual network method to set up shortcut connections between some layers. In addition, due to the application of the multiscale prediction method, the performance of Yolov3 in the detection of small targets is significantly improved. Although the network is larger, the speed is still very fast, and the accuracy is considerable.

To study the MOT framework of multiway concurrency, we first obtained the Yolov3 model with the COCO (Common Objects In Context) database based on the TensorFlow deep-learning framework and evaluated the time consumption of the detector with different batch sizes as in Figure 2.

Figure 2 shows that time consumption increases with the increase of batch size. So, it is important to choose the appropriate batch size. Of course, this depends on the overall resources and performance requirement of the system.

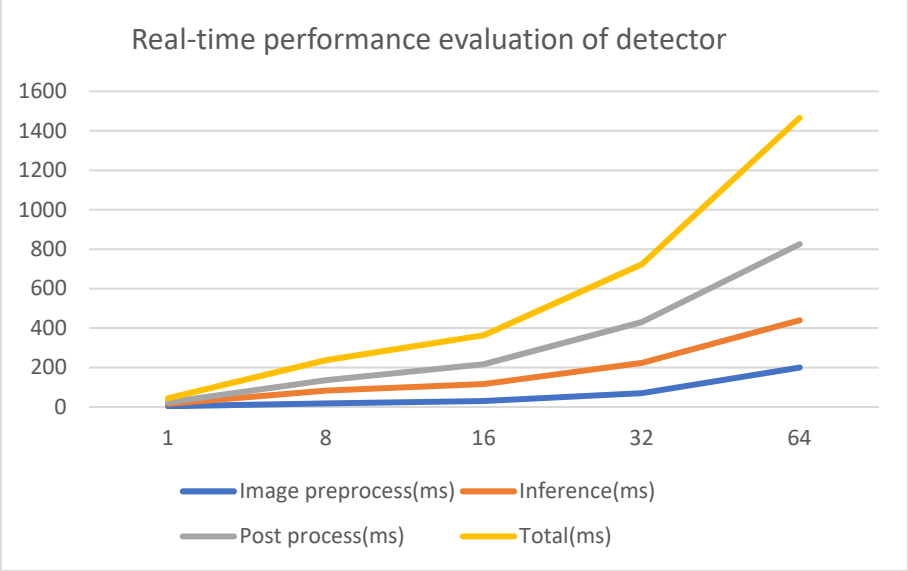


Figure 2. The detector with different batch size (X-axis represents batch size, Y-axis represents time consumption). Blue line indicates image preprocessing, it is related to the operation of the original frame resize; brown line indicates model inferencing for image data; gray line indicates processing of model inference results, including NMSs (nonmaximum suppression); yellow line indicates time consumption of detector.

3.3. MOT Based on Detection Driven

Target-state prediction is a common method for target tracking. Typical target-state prediction algorithms include the Kalman filter and the particle filter [35], which predict the current state and then update it according to the actual state obtained by detection. Typical algorithms include SORT [7] and DeepSort [10]. Data association is an important stage in the process of multitarget tracking. Some reference elaborated the purpose and significance of data association from different point of view [36,37]. In [38], the author showed the role of data association in the usual MOT system in an intuitive graphical way. Data association is closely related to the similarity measurement of targets. Currently, some MOT systems adopt a deep model in the detection and similarity measurement stages [39–42].

In this paper, we propose one kind of real-time detection-driven MOT algorithm. The algorithm does not adopt a traditional target-prediction method, but associates tracking results and detection results directly (the common method is to associate the prediction results with the detection results) in real-time. To trade off algorithm performance and real-time, one filter of a negative sample based on deep CNN is introduced as well.

3.3.1. Data Association

As mentioned above, the first problem to be solved in multitarget tracking is how to associate the detection results of the current frame and the trajectory set of the tracking target. This is different from single-target tracking, such as MOSSE (Minimum Output Sum Of Squared Error Filter) [43], KCF (Kernel Correlation Filter) [44,45], and DSST (Discriminative Scale Space Tracking) [46,47] (single-target tracking is not related to data association).

In Figure 3, the tracking targets at time t are A, B, and C. At time $t+1$, the association to the tracking targets need to establish (here only the association of the target A is drawn). A′ is the position of target A at time $t+1$ (the detection result at time $t+1$). A′ needs to match with A at time t, and simultaneously we also need to match other targets to determine if the results of detection and tracking identified the same target. The intersection-over-union (IOU) is usually used to measure the overlap area of two targets. We define a set of tracked targets $T_m = \{t_1, t_2, \ldots t_m\}^{m \in N}$, t_m to represent each tracked target. The detection target set in the current frame is expressed as $D_n = \{d_1, d_2, \ldots d_n\}^{n \in N}$. d_n represents the detected

target object. We use the Jaccard coefficient to calculate the degree of matching between the detection result D_n of the current frame and the tracking target set T_m. Jaccard coefficient is used to compare the similarity and difference between limited sample sets. Given two sets D and T, Jaccard coefficient is defined as the ratio of the size of the intersection of D and T to the size of the union of D and T (Intersection over Union, IOU). The definition is as follows:

$$J(D,T) = \frac{|D \cap T|}{|D \cup T|} = \frac{|D \cap T|}{|D| + |T| - |D \cap T|}. \tag{4}$$

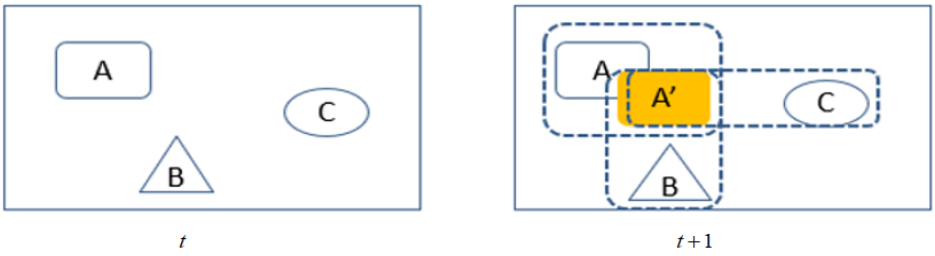


Figure 3. Target matching.

These values will be used to build a cost matrix, as shown in Equation (5):

$$C_{m\times n} = \begin{pmatrix} j_{d_1,t_1} & j_{d_1,t_2} & \cdots & j_{d_1,t_n} \\ j_{d_2,t_1} & j_{d_2,t_2} & \cdots & j_{d_2,t_n} \\ \vdots & \vdots & \ddots & \vdots \\ j_{d_m,t_1} & j_{d_m,t_2} & \cdots & j_{d_m,t_n} \end{pmatrix} \quad m,n \in \{1,2,\ldots,N\} \ . \tag{5}$$

j_{d_m,t_n} represents the Jaccard coefficient between the detected target d_m and the tracking target t_n in the current frame. The data association is generated by the Hungarian algorithm as shown in Equation (6):

$$\mathbf{L}_{(d_{index},t\ _{index})} = H(C_{m\times n}) \quad m,\ n \in \{1,2,\ldots,N\}. \tag{6}$$

The function H is a linear assignment function that returns the matrix $\mathbf{L}_{(d_{index},t\ _{index})}$. Each row of the matrix establishes a matching relationship between the detection target and the tracking target.

The target-tracking process may be affected by some unknown external factors that may lead to abnormal tracking, such as illumination, occlusion, and data association problems caused by missing detection of the detector. Figure 4 shows missing detection and new target appearing scenarios in multitarget tracking.

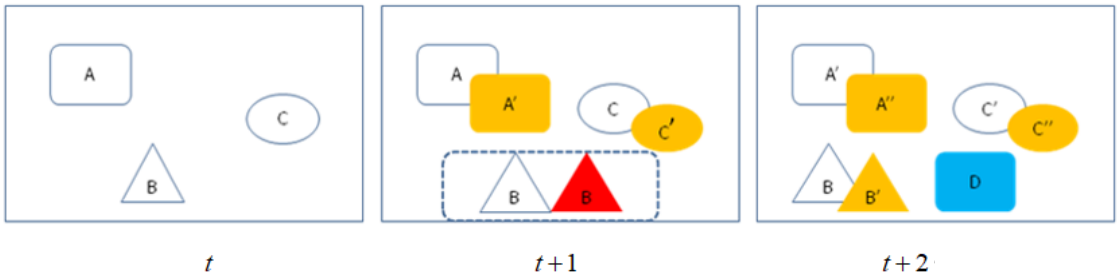


Figure 4. Data correlation scenario in multitarget tracking (the yellow represents target tracked correctly; red represents target of last frame was not detected at current frame; blue represents new target). At time t, tracklet includes three targets; At time $t+1$, target B wasn't detected; at time $t+2$, new target D was detected.

As shown in Figure 4, the three objects A, B, and C can be considered as tracking targets at time t. At time $t+1$, A and C are detected correctly and correspond to A′ and C′ respectively, but object B in the current frame is undetected; this means that the established association relationship failed with tracking object B at time t. We can obtain the target of missing detection D_{miss} via Equation (7):

$$D_{miss} = \left\{ d_{object} \middle| d_{object} \in T_m, d_{object} \notin L_{t_{index}} \right\}^{m,index \in N}. \quad (7)$$

At time $t+2$, the three tracking objects, A, B, and C, were detected and associated correctly, but object D was a new target. The new emerging target D_{new} can be obtained by Equation (8):

$$D_{new} = \left\{ d_{newobject} \middle| d_{newobject} \in D_n, d_{newobject} \notin L_{d_{index}} \right\}^{n,index \in N}. \quad (8)$$

3.3.2. Object Creation and Disappearance

In the process of multitarget tracking, it is necessary to solve the matching problem between the detection result of the target and the trajectory of the target; this can also be regarded as target reidentification. Therefore, how is a tracked target trajectory maintained? The usual TBD method needs to adjust the result of a prediction algorithm by detecting the result, and then update the result to the nearest target trajectory. Meanwhile, some targets will appear, and some will disappear. The life cycle of an object begins upon entering the lens and ends on exiting the lens. During this period, the object may be occluded or missed. However, maintenance of the target trajectory is crucial in the whole tracking process. In this paper, the detection results of the first frame (for example, at time t) were taken as the basic target set, and the detection results of the next frame (for example, at time $t+1$) were associated with the target track set. A new target was assigned a new target id, and its motion information was initialized. In addition, if the number of missed detection targets exceeded a threshold value A_{loss} (this threshold value can be set based on a priori knowledge, such as 10 or 15), then we decided the target had disappeared, and the target trajectory set was updated at the same time.

3.3.3. Filter of Negative Sample

In the MOT framework of multiway concurrency, we introduced one filter of a negative sample to improve tracking accuracy. Next, we will describe the function and real-time performance in detail.

Usually, the quality of the detector has a direct impact on tracking accuracy based on the tracking-by-detection framework, for example, false detection or missed detection. Missed detection does not lead to time consumption of the tracking algorithm, but false detection not only causes time consumption but also affects the accuracy of tracking. However, each target may be related to false detection problems; for example, it is possible that garbage or some road signs, and even a roadside tree are detected as pedestrians as shown in Figure 5 (orange rectangle, raw image was from MOT challenge dataset [15]). Id 28 was detected mistakenly as pedestrians. The false detections cause unnecessary consumption of computing and storage resources and impact tracking performance.

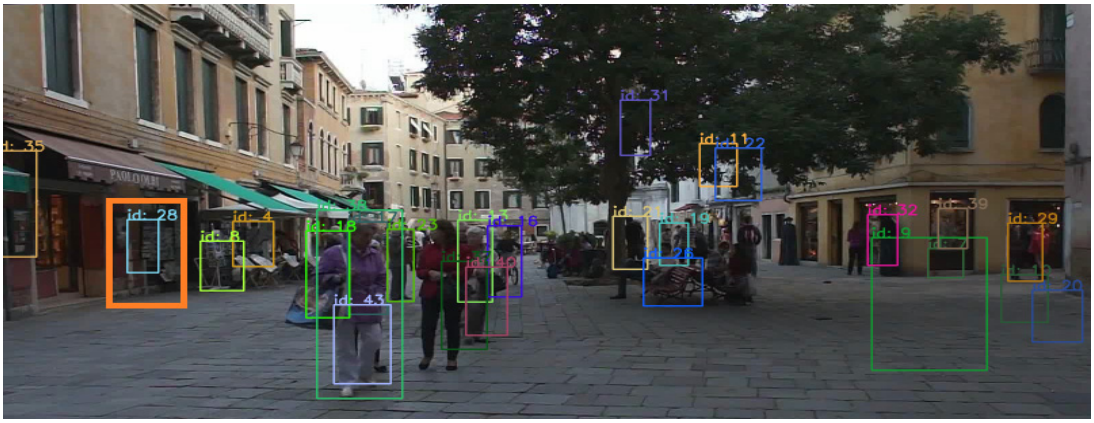


Figure 5. False target detection.

To mitigate false detection, we designed a deep CNN filter. In the process of tracking, if there is a new target, the filter is used to check if the target meets the requirement. Currently, the filter supports two categories, vehicles and pedestrians. The structure of the deep network is shown in Table 1:

Table 1. Network structure of the filter.

The Name	The Kernel Size/Stride	Num Output
1 Conv	$7 \times 7/2$	32
Pool2	$3 \times 3/2$	32
Residual 3	$3 \times 3/1$	32
Residual 4	$3 \times 3/1, 1 \times 1/2$	128
Residual 5	$3 \times 3/1, 1 \times 1/2$	256
Residual 6	$3 \times 3/2, 1 \times 1/2$	128
FC7		2
SoftMax		2

The network architecture of CNN is composed of a convoluton layer, two pooling layers, four residual blocks, a full-connection layer, and a SoftMax layer. The initial input picture dimension is 196×196, and the full connection layer and SoftMax layer are used for category prediction. The residual block structure is as shown in Figure 6.

In a convolutional network, since the number of feature maps of x_l and $\mathrm{B}x_{l+1}$ is different, 1×1 convolution is needed to increase or reduce dimensions. The residual block is expressed as:

$$x_{l+1} = h(x_l) + F(x_l, W_l) \tag{9}$$

where $h(x_l) = W_l' x$, W_l' is the convolutional operation of 1×1; $F(x_l, W_l)$ is residual portion.

From the perspective of real-time performance, we must consider the filter inference speed and the whole system impact. If multiway concurrency occurred and every frame has many detected results to filter, then system real-time will be impacted; the resource consumption of the system will be linear with the number of targets bounding boxes. We solved this problem from three aspects: (1) we designed a bounding-box collector to pack target bounding boxes of every channel into one batch to be input parameters of the filter; (2) if the filter processes all target bounding boxes, then the computing cost is still high, so we allowed the filter to process only new targets in every frame—missed and matching targets will not be processed by the filter; (3) we fully exploited TensorRt's inference and optimization characteristics to accelerate the processing speed of the classifier. Based on the three-aspect improvement, resource consumption will be greatly reduced.

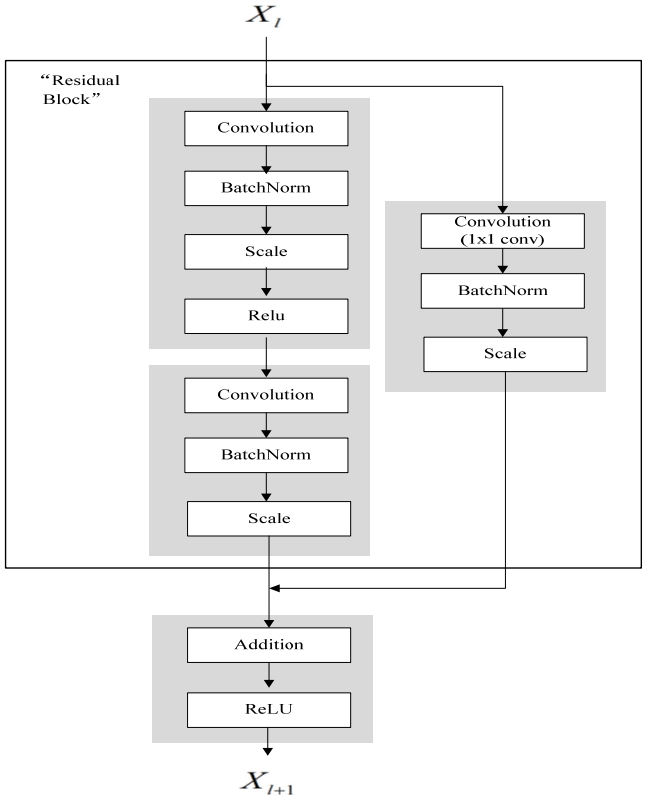


Figure 6. Residual block structure.

In addition, we found that the bbox queue will overflow after a period of time in multiway concurrency and that the main time consumption is the filter; the experimental average time consumption per frame is approximately 9.93 ms, which cannot meet the real-time requirements of the multiway concurrent MOT system very well.

The above problems are mainly caused by frequent image data copying between the CPU and GPU. The decoded images are stored in the DDR (Double Data Rate) memory of the system and processed by the CPU. When the cropped images (bounding box) are transferred to the filter, the bounding box will be copied to the GPU memory by the CUDA (Compute Unified Device Architecture) during preprocessing. Therefore, we attempted to allocate GPU memory for the image decoded by CUDA, with subsequent image operations implemented in the GPU. Experimentally, the time consumption of the filter processing one frame was approximately 5.97 ms, an obvious improvement.

4. Results

In this section, we first introduce the experiment environment, including the hardware and software environments and experiment inputs. Next, in the experiment evaluation part, we mainly compare the resource consumption of the algorithm in the tracking process and compare its performance with similar algorithms. Finally, the experiment results are analyzed and summarized.

4.1. Experiment Environment

We used the MOT challenge dataset MOT16 [15] as the data source. The experiment environment included: GTX 2080TI GPU, CUDA version 10.0, cuDNN version 7.4.2 for both toolkits, and the TensorRT version was 10.0.1. Based on the above environment, three groups of experiments were compared to reach experiment conclusions.

According to the MOT system structure, the detection results were packaged into a bounding box bundle as the input to the tracking algorithm (these bundle is first inserted into bbox queue and then dispatched to corresponding tracker by channel dispatcher.). The

target bounding box in the bundle could come from different image sequences (that is, a different camera) or the same image sequence; every image sequence was identified by a unique channel id.

In this experiment, to verify the correctness of the basic function on MOT system, we simulated processing two channels with identical input image sequences, that is, five frames in a bundle were from channel #1, while the other five frames were from channel #2. Bundle size was set as 10. The tracking algorithm processed these two image sequences in parallel (one image sequence was processed by one tracker instance). If the processing is correct, the output of each channel should be the same; that is, the two channels have the same tracking target information.

We only sampled two tracking targets and visualized them as shown in Figure 7. From the output results, it can be observed that the tracking had the same processing results for two identical image sequences, this shows that the logic of the tracking system is correct, which is the basis of subsequent experiments.

Figure 7. Tracking results of two identical image sequences. MOT16 [15] dataset was used to verify the basic function of tracking system: (**a**) The tracking results of channel #1; (**b**) the tracking results of channel #2.

4.2. Real-Time Evaluation

The detector has a direct impact on the tracking system in terms of real-time and algorithm performance based on the TBD framework, that is the different detector will generate the different impact on MOT system. To objectively evaluate the real-time performance of the tracking part in the paper, we did not consider the detection part.

In the paper, we proposed a MOT framework of multiway concurrency. We also designed a filter of negative sample to improve tracking accuracy. In terms of width of the MOT framework, the number of tracker instance will increase with the increase of concurrency. One tracker instance process one image sequence. One performance issue needed to be considered, that is, does one filter correspond to one tracker instance or multiple tracker instances? Which strategy is the most efficient? According to the different approaches of image data storage and processing in the tracking process, we divided the experiment into the following three groups, in the first two groups' experiments, we adopted one filter correspond to one tracker instance, but in the third group experiment, we adopted one filter to process the multiple tracker instance in parallel.

(1) In group one, all of the image data in the bundle were stored in DDR memory (image data were collected by image collector) and the tracker processed the detection

results in each frame (one tracker is for only one image sequence) and output the processing result of each frame after processing a bundle.

(2) In group two, we allocated GPU memory for each frame by CUDA memory management function, that is, the image data in the bundle were stored in the pre-allocated GPU memory (image data were collected by image collector) and the tracking algorithm processed the detection results using the same processing flow as the first group experiment.

(3) In group three, the frames in a bundle were allocated the same way as in the second group experiment, but differently from the above two group experiments for bundle processing. In other words, after one bundle was processed, all target information was collected by the bounding box collector, and these targets were processed by the classifier at once.

We have statistics of the classifier time consumption in processing each bounding box for experiments group one and group two (the statistics of only 30 bounding boxes was done), as shown in Figure 8. The x-coordinate is the number of bounding boxes and the y-coordinate is the time consumption of processing a bounding box:

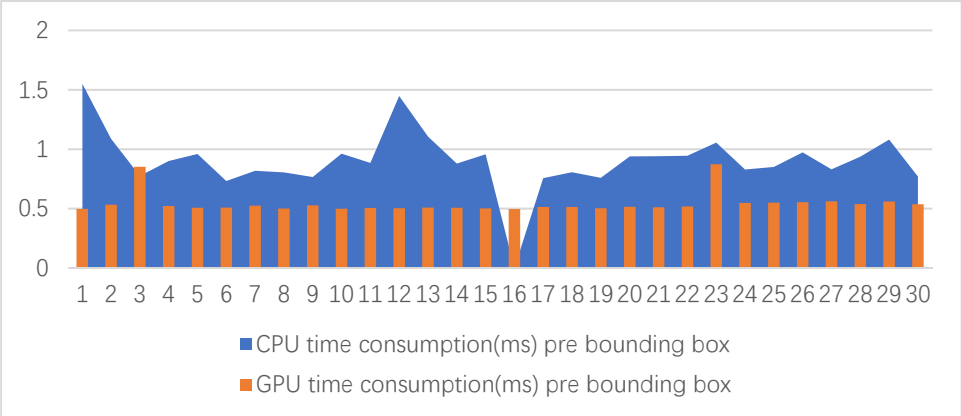


Figure 8. Computational resource consumption of each bounding box.

According to Figure 8, the classifier time consumption in the second group experiment was, on average, much less than the first group for processing a bounding box. The classifier is related to GPU preprocess. In the preprocess, image data are copied from external memory to GPU memory in the first group experiment, which consumed more time. The time consumption of three groups of experiments for processing one bundle were 149.05539 ms, 89.57784 ms and 2.65549 ms, respectively. One bundle included the fifteen-frame information from different image sequences. The statistics show the third group experiment improved the second group experiment efficiency by a factor of 33.

The classifier is responsible for the image processing in the whole tracking algorithm, which is one of the key factors that impact the system performance. So, we evaluated tracking speed in the third group experiment with different images batch sizes (1,5,10,15) separately (the images of different batch size were used as input to classifier), as shown in Figure 9:

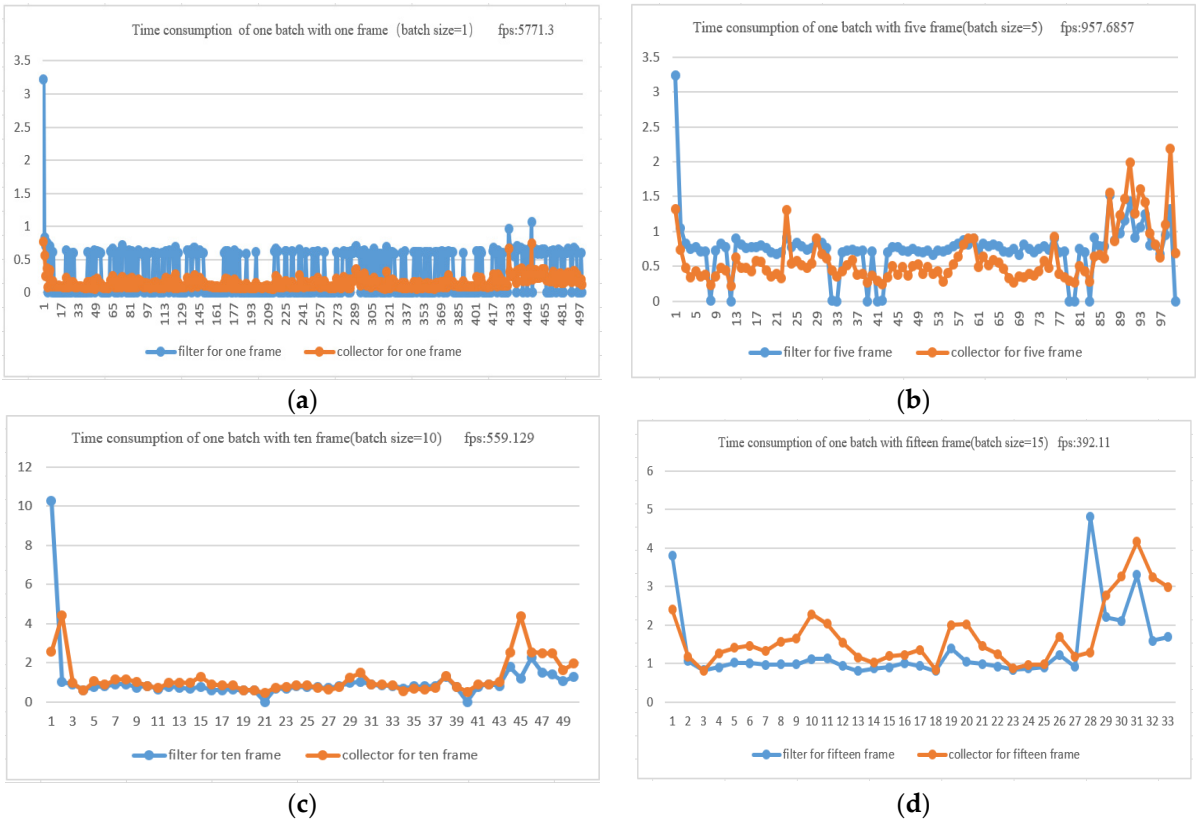


Figure 9. Time consumptions of different batch sizes: (**a**) the batch size was 1; (**b**) the batch size was 5; (**c**) the batch size was 10; (**d**) the batch size was 15.

The experimental results show that the tracking speed decreased with the increasing batch size as shown in Table 2, which was caused by the bounding box collector (the collector will crop images to get a bounding box). We will analyze this more deeply in Section 4.3.

Table 2. The tracking speed with different batch size of bounding box.

Batch Size (Concurrency)	fps
1	5771.3
5	957.6857
10	559.129
15	392.11

4.3. Performance Evaluation

Next, we evaluated the performance of MOT based on MOT challenge dataset MOT16 [15], evaluation indications and definitions as shown in Table 3.

Table 3. MOT evaluation indications and definitions. "↑": higher scores denote better performance, "↓": lower scores denote better performance.

Evaluation Indications	Definitions
MOTA ↑	The multiple objects tracking accuracy [48]
MOTP ↑	The multiple objects tracking precision [48]
MT ↑	Ratio of successful tracking target trajectory to real target trajectory
ML ↓	Proportion of lost target trajectory to real target trajectory
IDsw ↓	The total number of target identity switch during the whole target tracking process
FP ↓	The total number of false positives
FN ↓	The total number false negatives
Frag ↓	The number of times the real trajectory is interrupted

MOTA is a combination of three errors—false positive, missed targets, and identify switch—it is computed as:

$$\text{MOTA}= 1 - \frac{\sum_t (fn_t + fp_t + idsw_t)}{\sum_t g_t} \in (-\infty, 1] \tag{10}$$

where for frame t, fn_t is the number of false negatives (missed targets), fp_t is the number of false positives (false detection), $idsw_t$ is the number of identity switches, and g_t is the number of objects present. The higher MOTA indicates higher tracking accuracy. MOTP (Multiple Object Tracking Precision) measures the error of the predicted bounding boxes and the ground truth. The original formula was:

$$\text{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{11}$$

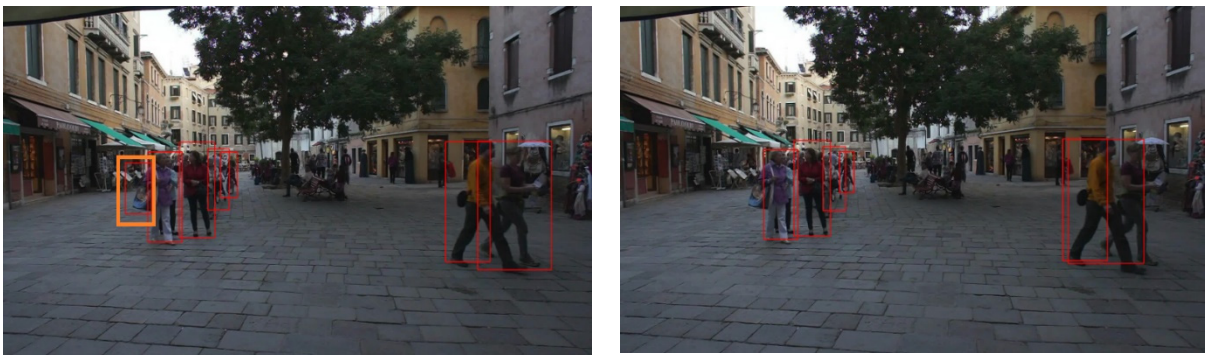
where for frame t, d_t^i is the distance between the ground truth and prediction of object i, and c_t the number of matches found.

To be fair, we compared only the key performance indicators; the real times of the algorithms were not compared, because the run environments of the algorithm were different, as shown in Table 4.

Table 4. Multitarget tracking evaluation list. "↑": higher scores denote better performance, "↓": lower scores denote better performance.

Methods	MOTA↑	MOTP↑	MT↑	ML↓	IDsw↓	FP↓	FN↓	Frag↓
SORT [7]	59.8%	79.6%	25.4%	22.7%	1423	8698	63,245	1835
DeepSort [10]	61.4%	79.1%	32.8%	18.2%	781	12,852	56,668	2008
MHT_DAM [12]	45.8%	76.3%	16.2%	43.2%	590	6412	91,758	781
POI [49]	66.1%	79.5%	34.0%	20.8%	805	5061	55,914	3093
RAN [16]	63%	78.8%	39.9%	22.1%	482	13,663	53,248	1251
SiameseRF+rule distillation [19]	57.2%	79.4%	28.2%	23.5%	2000	7265	68,860	2520
Tracktor++ [50]	54.4%	78.2%	19%	36.9%	682	3280	79,149	1480
TPM [51]	51.3%	75.2%	18.7%	40.8%	569	2701	85,504	707
Ours	52.2%	77.6%	21.3%	31%	1209	8214	41,514	1311

From the above evaluation results, under the premise of high concurrency and real-time, the performance of our method was still acceptable. This is mainly because the filter suppresses the negative samples and ensures the performance of the tracking algorithm. The results after applying the filter are shown in Figure 10 (raw image was from MOT challenge dataset MOT16).



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

Figure 10. Tracing effect after using filter. (**a**) Before using the filter of negative sample; (**b**) after using the filter of negative sample.

In the left picture, the detector mistakenly thought the satchel was a person, and in the right picture, the misidentified target was filtered to improve the tracking performance. In our approach, the ID switch was still a bit high, caused mainly by the overlap of some targets or long-term occlusion in the test set, which has not been solved well in this paper.

4.4. Discussion

According to the above experiments, the time consumption of tracking depends mainly on two factors: (1) Processing of image data between the CPU and GPU. In the GPU preprocess, every bounding box needs to be copied from external memory to GPU memory for the first group experiment, causing time consumption. (2) In the third group of experiments, the detected objects in a bundle were collected by the collector. This process cropped the image (bounding box) and allocated GPU memory for the image cropped by CUDA, then the filter processed them at once. Table 5 presents statistics of the time consumption of the collector and filter with different batch sizes. According to the statistics, the time consumption of both increased with the increase of batch size.

Table 5. Time consumption of collector and filter with different batch sizes.

	Batch Size = 1	Batch Size = 5	Batch Size = 10	Batch Size = 15
Collector (C)	0.147 ms	0.599 ms	1.211 ms	1.694 ms
Filter (F)	0.289 ms	0.744 ms	1.047 ms	1.361 ms

Further analysis showed that the proportion of time consumption of the collector $C/(F+C)$ increased with the batch size, but the proportion of filter time consumption $F/(C+F)$ decreased gradually with the increasing batch size, as shown in Figure 11.

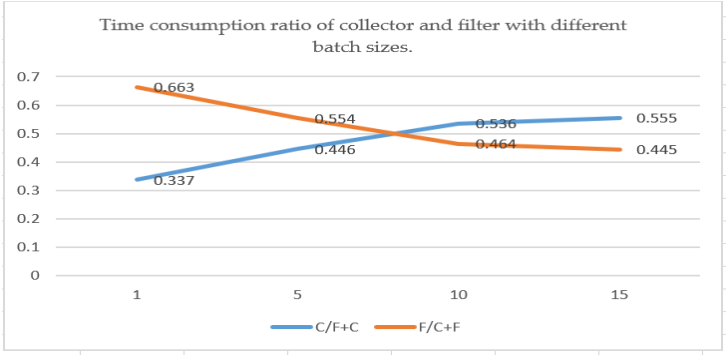


Figure 11. Time consumption ratio of collector and filter with different bundle sizes (X-axis represents batch size, Y-axis represents time consumption ratio).

In the collector process, the detected bounding box needs to be cropped and stored in GPU memory by CUDA; a larger batch size caused more time consumption. For the filter (classifier), a suitable number of input images causes full filter use of GPU computing resources. From Figure 11, when the number of input images was small, the time consumption ratio of the filter was higher than that of the collector, and just near the cross point, the time consumption ratio of the two was close, so a batch size of 10 can be a suitable value.

Of course, from the memory point of view, the results of the last two groups of experiments caused GPU memory to increase significantly, especially when the number of objects processed increased. In the process, GPU memory is frequently allocated and released, this also causes more time consumption.

In addition, the paper adopted a type of detection-based tracking algorithm; that is, the detection result of the current frame was regarded as a query, the tracking object list was regarded as gallery, and the best match was found through the Hungarian algorithm. The algorithm performed well without long-time occlusion. When there is long-time occlusion, lost-object problems may occur. The long-term occlusion issue is complicated, solving such problems usually means consuming more computing and storage resources, which then impacts the system's real-time performance. Considering the high concurrency and real time of processing of the tracking algorithm, we traded off the performance of the algorithm and the processing speed. Before that, there was no relevant research. Actually, the balance between the real-time performance of target tracking and algorithm performance in case of high concurrency is a problem worth studying, even if sometimes they are contradictory. How to balance their relationship? There is no definitive answer or metric. In our opinion, when the algorithm meets the real-time requirements of the system, performance loss may be acceptable under the conditions of practice.

5. Conclusions

The MOT with a multicamera is a kind of general vision application. In this case, usually there is a critical demand for real-time tracking performance and system concurrency capability. In our paper, we proposed a new MOT framework to support the concurrent process of multiway real-time video stream. The new MOT framework was expanded in width and depth based on a tracking-by-detection model. We then conducted the performance research of MOT based on three aspects of the working model: (1) We proposed one type of simple MOT-based direct detection. In the tracking algorithm, a CNN network was designed as the filter (classifier) to reduce false detection. (2) We designed a batching mechanism to speed up the classifier efficiency in the tracking process. The experimental results showed that the new mechanism generated larger gain. (3) We utilized the inference optimization features of NVIDIA TensorRT to accelerate the filter processing in the tracking algorithm.

The final experiment showed that our implementation and optimization of a real-time multiway concurrent multiobject tracking system can meet the practical scene application requirements. Of course, some issues need to be researched further, for example, the long-duration occlusion caused the target to be lost. Although this issue was solved well by DeepSort, it was slower, so further real-time tracking performance will be impacted. In addition, frequent GPU memory allocation and deallocation will incur more time consumption.

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

Funding: This research was funded by the Special Funding of the 'Belt and Road' International Cooperation of Zhejiang Province, grant number 2015C04005 and the Natural National Science Foundation of China (NSFC), grant number 61571399.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw/processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

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

References

1. Wang, X. Intelligent multi-camera video surveillance: A review. *Pattern Recognit. Lett.* **2013**, *34*, 3–19. [CrossRef]
2. Pfister, T.; Charles, J.; Zisserman, A. Flowing convnets for human pose estimation in videos. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1913–1921.
3. Choi, W.; Savarese, S. A unified framework for multi-target tracking and collective activity recognition. In Proceedings of the 12th European Conference on Computer Vision (ECCV), Florence, Italy, 7–13 October 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 215–230.
4. Hu, W.; Tan, T.; Wang, L.; Maybank, S. A survey on visual surveillance of object motion and behaviors. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2004**, *34*, 334–352. [CrossRef]
5. Luo, W.; Xing, J.; Milan, A.; Zhang, X.; Liu, W.; Zhao, X.; Kim, T.K. Multiple object tracking: A literature review. *arXiv* **2014**, arXiv:1409.7618.
6. Manzo, M.; Pellino, S. FastGCN + ARSRGemb: A novel framework for object recognition. *arXiv* **2020**, arXiv:2002.08629.
7. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and Realtime tracking. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468.
8. Kalman, R. A New Approach to Linear Filtering and Prediction Problems. *ASME J. Basic Eng.* **1960**, *82*, 35–45. [CrossRef]
9. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
10. Wojke, N.; Bewley, A.; Paulus, D. Simple online and Realtime tracking with a deep association metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649.
11. Zheng, Z.; Yang, X.; Yu, Z.; Zheng, L.; Yang, Y.; Kautz, J. Joint Discriminative and Generative Learning for Person Re-Identification. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 2133–2142.
12. Kim, C.; Li, F.; Ciptadi, A.; Rehg, J.M. Multiple Hypothesis Tracking Revisited. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4696–4704.
13. Chu, Q.; Ouyang, W.; Li, H.; Wang, X.; Liu, B.; Yu, N. Online multi-object Tracking Using cnn-based Single Object Tracker with spatial-temporal Attention Mechanism. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 4846–4855.
14. Wang, Z.; Zheng, L.; Liu, Y.; Li, Y.; Wang, S. Towards real-time multi-object tracking. *arXiv* **2019**, arXiv:1909.12605.
15. Milan, A.; Leal-Taix'e, L.; Reid, I.; Roth, S.; Schindler, K. Mot16: A benchmark for multi-object tracking. *arXiv* **2016**, arXiv:1603.00831.
16. Fang, K.; Xiang, Y.; Li, X.; Savarese, S. Recurrent Autoregressive Networks for Online Multi-Object Tracking. In Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, USA, 12–15 March 2018; pp. 1–10.
17. Leal-Taix'e, L.; Milan, A.; Reid, I.; Roth, S.; Schindler, K. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv* **2015**, arXiv:1504.01942.
18. Tian, W.; Lauer, M.; Chen, L. Online Multi-Object Tracking Using Joint Domain Information in Traffic Scenarios. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 374–384. [CrossRef]
19. Lee, J.; Kim, S.; Ko, B.C. Online Multiple Object Tracking Using Rule Distillated Siamese Random Forest. *IEEE Access* **2020**, *8*, 182828–182841. [CrossRef]
20. Pang, B.; Li, Y.; Zhang, Y.; Li, M.; Lu, C. TubeTK: Adopting Tubes to Track Multi-Object in a One-Step Training Model. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 6307–6317.
21. Zhang, Y.; Sheng, H.; Wu, Y.; Wan, S.; Lyu, W.; Ke, W.; Xiong, Z. Long-term tracking with deep tracklet association. *IEEE Trans. Image Process.* **2020**, *29*, 6694–6706. [CrossRef]
22. Milioto, A.; Stachniss, C. Bonnet: An open-source Training and Deployment Framework for Semantic Segmentation in Robotics using CNNs. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 7094–7100.
23. Jain, P.; Mo, X.; Jain, A.; Subbaraj, H.; Durrani, R.S.; Tumanov, A.; Stoica, I. Dynamic space-time Scheduling for GPU Inference. *arXiv* **2018**, arXiv:1901.00041.
24. Alyamkin, S.; Ardi, M.; Brighton, A.; Berg, A.C.; Chen, Y.; Cheng, H.P.; Gauen, K. 2018 Low-Power Image Recognition Challenge. *arXiv* **2018**, arXiv:1810.01732.
25. Singhani, A. Real-time Freespace Segmentation on Autonomous Robots for Detection of diction and drop-offs. *arXiv* **2019**, arXiv:1902.00842.

26. Womg, A.; Shafiee, M.J.; Li, F.; Chwyl, B. Tiny SSD: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In Proceedings of the 2018 15th Conference on Computer and Robot Vision (CRV), Toronto, ON, Canada, 8–10 May 2018; pp. 95–101.
27. Lan, W.; Dang, J.; Wang, Y.; Wang, S. Pedestrian detection based on yolo network model. In Proceedings of the 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, China, 5–8 August 2018; pp. 1547–1551.
28. Mane, S.; Mangale, S. Moving object detection and tracking using convolutional neural networks. In Proceedings of the 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 14–15 June 2018; pp. 1809–1813.
29. Zhang, Y.; Huang, Y.; Wang, L. What makes for good multiple object trackers? In Proceedings of the 2016 IEEE International Conference on Digital Signal Processing (DSP), Beijing, China, 16–18 October 2016; pp. 467–471.
30. Abbas, S.M.; Singh, S.N. Region-based object detection and classification using faster R-CNN. In Proceedings of the 2018 4th International Conference on Computational Intelligence & Communication Technology (CICT), Ghaziabad, India, 9–10 February 2018; pp. 1–6.
31. Özer, C.; Gürkan, F.; Günsel, B. Object tracking by deep object detectors and particle filtering. In Proceedings of the 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2–5 May 2018; pp. 1–4.
32. Liu, Y.; Wang, P.; Wang, H. Target tracking algorithm based on deep learning and multi-video monitoring. In Proceedings of the 2018 5th International Conference on Systems and Informatics (ICSAI), Nanjing, China, 10–12 November 2018; pp. 440–444.
33. Zhao, X.; Li, W.; Zhang, Y.; Gulliver, T.A.; Chang, S.; Feng, Z. A faster RCNN-based pedestrian detection system. In Proceedings of the 2016 IEEE 84th Vehicular Technology Conference (VTC-Fall), Montreal, QC, Canada, 18–21 September 2016; pp. 1–5.
34. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
35. Arulampalam, S.M.; Maskell, S.; Gordon, N.; Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.* **2002**, *50*, 174–188. [CrossRef]
36. Li, Y.F.; Zhou, S.R. Survey of online multi-object video tracking algorithms. *Comput. Technol. Autom.* **2018**, *37*, 73–82.
37. Yang, F.D. Summary of data association methods in Multi-target tracking. *Sci. Technol. Vis.* **2016**, *6*, 164–194.
38. Ciaparrone, G.; Luque Sánchez, F.; Tabik, S.; Troiano, L.; Tagliaferri, R.; Herrera, F. Deep learning in video multi-object tracking: A survey. *Neurocomputing* **2020**, *381*, 61–88. [CrossRef]
39. Ran, N.; Kong, L.; Wang, Y.; Liu, Q. A robust multi-athlete tracking algorithm by exploiting discriminant features and long-term dependencies. In Proceedings of the 25th International Conference on MultiMedia Modeling (MMM 2019), Thessaloniki, Greece, 8–11 January 2019; Springer: Cham, Switzerland, 2019; pp. 411–423.
40. Chen, L.T.; Peng, X.J.; Ren, M.W. Recurrent metric networks and batch multiple hypothesis for multi-object tracking. *IEEE Access* **2019**, *7*, 3093–3105. [CrossRef]
41. Xiang, J.; Zhang, G.S.; Hou, J.H. Online multi-object tracking based on feature representation and bayesian filtering within a deep learning architecture. *IEEE Access* **2019**, *7*, 27923–27935. [CrossRef]
42. Kwangjin, Y.; Du, Y.K.; Yoon, Y.-C.; Jeon, M. Data association for multi-object tracking via deep neural networks. *Sensors* **2019**, *19*, 559.
43. Bolme, D.S.; Beveridge, J.R.; Draper, B.A.; Lui, Y.M. Visual Object Tracking using Adaptive Correlation Filters. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 2544–2550.
44. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. Exploiting the circulant structure of tracking-by-detection with kernels. In Proceedings of the 2012 European Conference on Computer Vision (ECCV), Florence, Italy, 7–13 October 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 702–715.
45. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High-speed Tracking with Kernelized Correlation Filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *37*, 583–596. [CrossRef]
46. Danelljan, M.; Häger, G.; Khan, F.; Felsberg, M. Accurate Scale Estimation for Robust Visual Tracking. In Proceedings of the 2014 British Machine Vision Conference (BMVC), Nottingham, UK, 1–5 September 2014.
47. Danelljan, M.; Häger, G.; Khan, F.S.; Felsberg, M. Discriminative Scale Space Tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1561–1575. [CrossRef]
48. Bernardin, K.; Stiefelhagen, R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP J. Image Video Process.* **2008**, 1–10. [CrossRef]
49. Yu, F.; Li, W.; Li, Q.; Liu, Y.; Shi, X.; Yan, J. POI: Multiple Object Tracking with High Performance Detection and Appearance Feature. In Proceedings of the 2016 European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 8–16 October 2016; Springer: Cham, Switzerland, 2016; pp. 36–42.
50. Bergmann, P.; Meinhardt, T.; Leal-Taix'e, L. Tracking without bells and whistles. In Proceedings of the 2019 IEEE International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 941–951.
51. Peng, J.; Wang, T.; Lin, W.; Wang, J.; See, J.; Wen, S.; Ding, E. TPM: Multiple Object Tracking with Tracklet-Plane Matching. *Pattern Recognit.* **2020**, 107480. [CrossRef]