AI City Challenge: M6 Report

Mert Yazan, Berkay Arpacı, Marcos Conde
UPC - UAB
{1619479, 1628973, 1619163}@uab.cat

Abstract

Multi-target multi-camera tracking (MTMC) is implemented for large scale road traffic monitoring. Traffic monitoring is useful for traffic management or a part of an intelligent city system. The main problems are computation cost, occlusions, camera configuration, viewing angle and illumination differences. Several state of the art algorithms are used for MTSC and MTMC. mAP and IDF1 are used on NVIDIA AI City Challenge dataset for comparison. Finally, possible improvement methods are discussed.

1. Introduction

Traffic monitoring enables a better understanding of the traffic dynamics and allows designing more optimized transportation systems and road networks. For large cities, using multi-target multi-camera (MTMC) tracking, it is possible to determine the characteristics of road traffic and manage it to an extent without significant human intervention [3, 10, 14]. The increased number of cameras along the roads can be used to monitor the traffic.

MTMC tracking is the most comprehensive form of object tracking where each detected object is tracked simultaneously through multiple cameras until it leaves every camera frame or becomes undetectable. It is used for player tracking, anomaly detection, surveillance, crowd behavior analysis and suspicious activity detection. As most of the public places such as airports install cameras for security nowadays, the objects of interest can be tracked for wide areas automatically.

There are many problems to tackle for MTMC. First, many video streams must be processed in real time which requires a significant amount of resource. Second, the overlapping portions of most camera systems are kept as small as possible to use fewer cameras for the area. This also causes large discrepancies of viewing angles, illumination conditions and color. Occlusions are another problem for tracking tasks as it may cause a trajectory to switch to another object. Finally, the number and characteristics of the objects are uncertain.

There are several steps used for an MTMC algorithm. Firstly, the objects must be detected from each camera view. This is also called multi-target single-camera tracking which can be done using object detection [3, 10, 14]. Once the tracklets of each object is obtained, if the object enters another camera view before leaving the current camera view, the tracklets must be matched to extend the range of tracking. For this part of the task several matching algorithms and tracklet feature extractors are used.

2. Related work

2.1. Object Detection

This task has been an important problem for computer vision researchers and companies due to its wide range of applications. Before ResNet [5, 6], classical methods comprising of feature extractors and machine learning classifier combinations. SIFT and HOG [2, 9] were the most popular feature extractors and random forest and support vector machines are some of the classifiers. Complementary techniques such as background subtraction were also used.

After the success of CNN, increasingly faster and more accurate techniques are developed by utilizing region proposals. Overfeat and R-CNN were the first of such networks. They are succeeded by Fast and Faster RCNN [12] later. The accuracy is then increased with the introduction of RetinaNet where the loss function becomes capable of analyze many regions at once while retaining an acceptable inference speed.

This problem is overcome by single-shot detectors such as you look only once (YOLO) [11] while sacrificing some of the accuracy as only a single pass is used. The speed-up is achieved by the removal of proposals and using grids instead. Many improvements are developed for YOLO [11]. There are also several other successful single-shot detectors.

2.2. Object Tracking

This task can be considered as the extension of the previous one to video processing [15]. As such in general, tracks are obtained by matching detections from a frame sequence using similarity and spatio-temporal continuity. Matching

performance can be improved with the addition of motion models. Both learned and hand-crafted features are used. Additionally, offline tracking can be achieved by utilizing optimization using hierarchical methods and graphs.

3. Dataset

Dataset used throughout the project is the one explained in the paper *CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification*. [10, 14]. Dataset is composed of different sequences of cameras and contains up to 3 hours of video. There are 5 sequences and 40 cameras originally but we used sequences 1,3 and 4 and their corresponding cameras.

City Flow dataset is specifically created to tackle the problems of MTMC and MTSC. It contains a groundtruth for each camera, which is designed in the MTMC format. It means that only cars that appear at least in 2 cameras are tracked.

Each camera also contains detection results from YOLO3 [1], Mask-RCNN [4] and SSD512 [8] networks. On top of that, there are also single camera tracking results for the given network predictions using different algorithms like MOANA [13] or DeepSORT [15].

4. Multi-Target-Single-Camera-Tracking

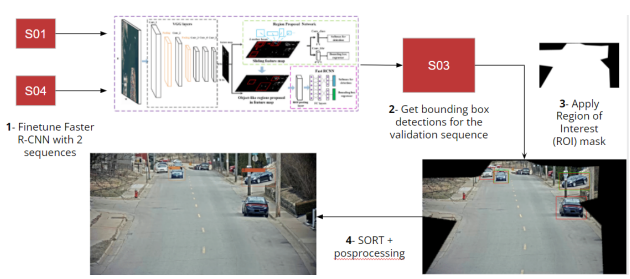

Figure 1. Pipeline used for MTSC.

First problem we tried to solve is MTSC. A general overview of the pipeline used can be seen in Figure 1. First step is to finetune an object detector to our dataset. We chose a Faster R-CNN [12] because experiments in the last week's works showed that it is able learn the problem successfully. To make bounding box detections for a sequence, the other 2 is chosen to finetune the network. This has been done 3 times, using each sequence as validation and the other 2 as training.

We used Detectron2 [16] for the finetuning process. Library has the Faster R-CNN [12] in its model zoo and it allows you to choose different versions of the network. We chose the one which has Resnet-50+FPN [5] as the backbone. We used 1000 iteration steps for finetuning. Experiments show that having more iteration steps lead to overfitting. COCO [7] evaluation metrics implemented in Detectron2 are used for evaluation. Default learning rate and augmentation strategies for Faster R-CNN in Detectron2 aren't changed.

After getting the bounding box predictions for each sequence, SORT [15] is applied to them to get the tracking results.

4.1. Post-Processing

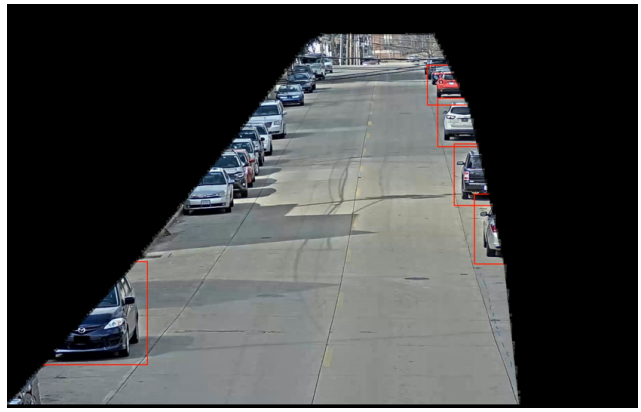

Figure 2. Boxes that are partly in the ROI.

Even though they are created for the MTMC task, we used the given camera ground truth bounding boxes for finetuning. As a rule, only cars that pass through at least 2 cameras are labeled in ground truth files and the parked cars aren't labeled. In practice, the finetuned network shouldn't take parked cars into account but we saw that it can still detect some parked cars. This affects the performance especially in some camera settings where there are parked cars through the video. Tracking them leads to poor performance compared with the ground truth.

To overcome this problem, 2 steps are implemented:

1. Look only into the Region of Interest (ROI) of cameras
2. Try to detect parked cars while tracking and remove their track

The dataset [10, 14] provides a ROI mask for each camera. We see that most of the parked cars are at least partially not in the ROI. Removing cars that aren't 100% in the ROI greatly improves the performance. So for each bounding box prediction we calculate its overlap with the ROI and remove it if it is not fully in the ROI Figure 2

After applying the ROI mask, we still see that there are some parked cars that are completely in the ROI Figure 3. A simple algorithm is implemented to remove the tracks which contain parked cars. To detect if a car is parked or not, we calculate the initial centroid of a car. After a car is started to get tracked, we calculate the new centroids in each

frame and calculate the euclidean distance with the initial centroid. If it is above a certain distance, we increase the parked flag by 1. Distance threshold is determined as 10 for our case. If parked flag reaches 10, meaning if the distance is above 10 for 10 frames, car is determined as parked and the track is removed.

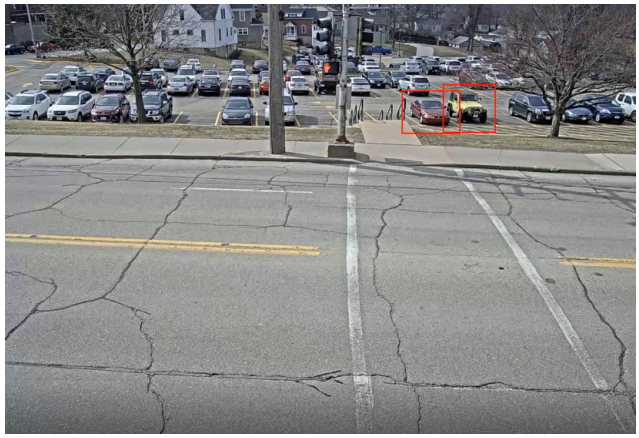

Figure 3. Parked cars that are inside ROI.

5. Multi-Target-Multi-Camera-Tracking

For MTMC, we built our pipeline from the MTSC results. For each camera, we have the bounding box predictions but for a multi camera setting we need to match the cars across different cameras. There can be different approaches for matching, but we choose to use a histogram based approach.

Figure 4. Tracking of a car in multiple cameras.

A 3D color histogram is calculated for each car. We concatenated the channels and normalized them. There are different parameters to take into account when using histograms, for example the color space and the bin sizes. Our experiments show that RGB images with a bin size of 32 gives the best results for our problem. Results for different color spaces are given in the next section. For the distance, we choose Hellinger. Compared to cosine and euclidean, Hellinger performed much better.

Sequence	IDF1 With Parked Cars	IDF1 Without Parked Cars	Change in IDF1
S01	0.7977	0.8177	+%2,5
S03	0.5204	0.6133	+%17
S04	0.6209	0.7211	+%16

Table 1. MTSC - Average IDF1 results for each sequence

5.1. Limitations of Color Histograms

Looking at the results Table 3, we see that they aren't very satisfactory compared to the MTSC task. We analyzed our approach and visualized matched cars. There are some common mistakes that appear many times, like not matching a car when its position changes. The problem is that colors in the frame can change with different position of the car and histograms don't catch this.

Another very important problem is that there is a background in the bounding box of the car. In some situations, completely unrelated objects like traffic signs can appear in the cropped box and this leads the histogram to not match the cars.

Figure 5. Cars that have unrelated backgrounds for histograms.

6. Results

Analyzing the MTSC results, we see that removing parked cars is helping a lot in IDF1, especially in sequences 3 and 4 Table 1. Some cameras in Sequence 4 are short and they contain very few cars. Removing even 1 parked contributes tremendously to the final IDF1 results. Sequence 1 benefits the least from the increase, probably due to the fact that it is composed mostly about cameras that are recording the highways.

Apart from the average results, we also analyzed IDF1 scores camera by camera and the increase is tremendous in some cases. For example in the 31st camera of Sequence 4, IDF1 increases from %16 to %84.

For the MTMC, we did a grid search of parameters between RGB, HSV, YCRCB color spaces and histogram bin

Seq-Cam	IDF1 With Parked Cars	IDF1 Without Parked Cars	Change in IDF1
S03-C12	0.036	0.3186	+%700
S04-C31	0.1681	0.8348	+%388
S04-C23	0.2825	0.7233	+%157

Table 2. MTSC - Top 3 cameras with the most increase in IDF1

Seq.	IDF1	IDP	IDR	Precision	Recall
S01	0.1904	0.1577	0.4357	0.2687	0.7077
S03	0.2126	0.2443	0.2977	0.3858	0.4205
S04	0.2377	0.2345	0.3829	0.3269	0.5838
Avg.	0.2135	0.2121	0.3721	0.3271	0.5706

Table 3. MTMC Results

Color Space	IDF1	IDP	IDR	Prec.	Recall
RGB	0.2135	0.2121	0.3721	0.3271	0.5706
HSV	0.2091	0.2109	0.2987	0.2864	0.4788
YCRCB	0.1296	0.1324	0.2625	0.2784	0.4003

Table 4. MTMC - Average result of sequences in different color spaces

sizes of [8, 16, 32, 64].

7. Conclusions

For MTSC, results aren't very bad considering that ground truth values aren't suited to track each car in a Single Camera setting. For most of the cameras, at least an IDF1 of %70 can be achieved. With the exception of a very few cameras, our method performed each detection provided by the dataset in every sequence and camera.

In some camera settings, there are very few cars and even one mistake can lower the results. For example in Figure 6, a car is leaving the camera but a very similar looking car enters 3-4 seconds later. According to the ground truth file, these are 2 different cars but they are tracked as the same car. Even one mistake in this setting leads to a poor IDF1 result.

For MTMC, Vehicles are often missed due to distortion or lighting conditions. In the scenario of MOT, this is often eased by better detectors, better data association strategies or even single object tracking.

We see that these methods rely on heavy training data or external models. Vehicles share similar appearances and thus pure re-identification models fail. The same vehicle instance may suffer from great appearance changes. Because of these reasons, MTMC is a much harder task compared to MTSC.

Figure 6. Example of a failed tracking.

And as final conclusion, we see that color histograms aren't a very reliable method to match cars. They are easy to implement but prone to fail when there is even a slight change in occlusion, car position or background. Looking at the work of different groups from previous years, we see that a metric learning approach can be better for multi camera settings. But regardless of the method used, it should be robust because the same car can be as very different thanks to the setting of different cameras.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 2

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. 1

[3] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 2

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc. 1

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016. 2

[9] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. 1

[10] Milind Naphade, Shuo Wang, David C Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E Lopez, et al. The 5th ai city challenge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4263–4273, 2021. 1, 2

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. 1

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 1, 2

[13] Zheng Tang and Jenq-Neng Hwang. MOANA: An online learned adaptive appearance model for robust multiple object tracking in 3d. *IEEE Access*, 7:31934–31945, 2019. 2

[14] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. 2019. 1, 2

[15] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017. 1, 2

[16] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 2

This work is licensed under a Creative Commons "Attribution-NonCommercial-ShareAlike 3.0 Unported" license.