Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 6 (2023)

Multiple Object Tracking using Yolov7 Based Object Detection and Norfair Tracking

Rubeena Banu, M. H. Sidram, Subhash Rao, Vishwanath K.R. , Ravikumar V.

Sri Jayachamarajendra College of Engineering, Visvesvaraya Technological University, Department of Electrical and Electronics Engineering, Karnataka, India – 570006

Vidyavardhaka College of Engineering, Department of Computer Science and Engineering, Karnataka, India – 570002

Abstract

The computer vision techniques have advanced video processing and intelligence generation for a variety of difficult dynamic circumstances. Tracking moving objects is incredibly useful in many computer vision applications. This work provides a method for tracking multiple objects in a video. It entails, first detecting the object using YOLOv7 model and then giving detection results to Norfair framework. Norfair is a customizable, lightweight Python framework for monitoring 2D objects in real time. The Manhattan distance function is employed to assess the distance between recently observed items and the objects it is already monitoring. From the result it has been analyzed that the proposed model considerably enhances tracking performance while handling occlusion, appearance changes and fast moving object.

1. Introduction

Visual object tracking plays a vital role in numerous practical applications. The task of object tracking can be categorized into Single Object Tracking (SOT) and Multiple Object Tracking (MOT), depending on the number of objects being tracked [1]. Object detection is a fundamental task that involves identifying and localizing objects within images or videos. Multiple object tracking is a field which is rapidly evolving within computer vision that has significantly been acknowledged in recent years. The ability to precisely track objects in real-time has a wide range of practical applications, from traffic monitoring, surveillance to security systems, and human activity recognition [2-5].

The primary goal of MOT is to track the trajectories of a set of objects, such as pedestrians, while maintaining their individual identities as they move through a video sequence. Traditionally, MOT approaches follow a two-step tracking-by-detection paradigm. Firstly, objects are detected in individual video frames, and then associations are made between detections across frames to create object tracks over time. This is typically achieved through temporally sparse or dense graph optimization or by using convolutional neural networks to predict matching scores between detections. Multiple object tracking is a crucial task in computer vision that has numerous real-world applications. It typically involves the use of deep learning models, such as Convolutional neural networks (CNN), to detect and track objects of interest across multiple frames in a video sequence. It is used to learn features from image sequence and predict the location and trajectory of objects. It has the potential to transform many industries and improves daily lives [6].

The drawbacks of relying on a linear motion model in video-based multi-object tracking are acknowledged, and it is suggested that overcoming these limitations can enhance performance without the need for additional cues like visual appearance. There are several challenges in MOT, one of the novel approaches is to address the challenge in MOT by leveraging spatial-temporal graph Transformers and a cascade association framework. This

helps in improving robustness, accuracy, and computational efficiency. The primary challenge that will be encountered in multiple object tracking is effectively managing prolonged occlusions, as well as addressing issues related to changes in object appearances, varying speeds, and complex motion patterns. Enhancing the tracking system's robustness in such intricate environments is crucial. The other challenge in this will be dealing with variations in object appearances, speed, and motion. For instance, objects may change their orientation, size, or color over time, or they may move at different speeds or change their direction suddenly. These variations can cause the tracker to lose track of the object or confuse it with other objects in the scene. To address this challenge Norfair algorithm has been used, which is a popular method for multiple object tracking that uses a combination of deep learning and traditional computer vision techniques [7]. The work contributions are summarized below,

- In order to track several objects in videos, the Norfair tracking algorithm was integrated with YOLOv7.
- Various parameters within the Norfair algorithm were adjusted in order to attain the highest level of accuracy in object tracking.
- The Manhattan distance function is employed to measure the distance between recently detected objects and the objects that the algorithm is already monitoring.

2. Multiple Object Tacking (MOT) Dataset

Recent years have seen the computer vision field rely on a number of centralized benchmarks for performance evaluation of a variety of tasks, which includes identifying objects, pedestrian, 3D reconstruction, optical flow, single-object temporary tracking, and stereo estimation. The use of such standards has proven to be quite beneficial in advancing the current state of the art in the relevant research domains, notwithstanding any potential drawbacks. The lack of progress toward standardizing multiple target tracking analyses is interesting. There are several difficulties using subsets of data for certain objectives like 3D tracking, surveillance, and sports analysis. MOT17 and MOT20 [8-9] dataset have been considered in this research work. The example images of MOT benchmark dataset are shown in Figure 1 and 2.

Description of MOT17: Set of three public detections used for MOT17. All three detection sets had to be used by participants to evaluate their trackers, and the findings were then averaged to determine the final score. The major goal of this new procedure was to determine how well trackers would perform when fed to different types of detections. In addition, a second subset known as MOT17Det was made accessible.

Description of MOT20: A dataset for tracking numerous objects is called MOT20. The dataset includes eight difficult video sequences-four for training and four for testing-in unrestricted settings, including busy locations like train stations, town squares, and sports stadiums. In image coordinates, tracking as well as assessment are performed. A stringent methodology with great precision, all sequences have been annotated.

Figure 1: MOT17 Benchmark Dataset

Figure 2: MOT20 Benchmark Dataset

3. Proposed Methodology for Multiple Object Tracking:

A method for tracking multiple objects in a video. It entails, first detecting the object using YOLOv7 model and then giving detection results to Norfair framework. Norfair is a customizable, lightweight Python framework for monitoring 2D objects in real time. Figure 3 shows the Block diagram of multiple object tracking using YOLOv7 and Norfair.

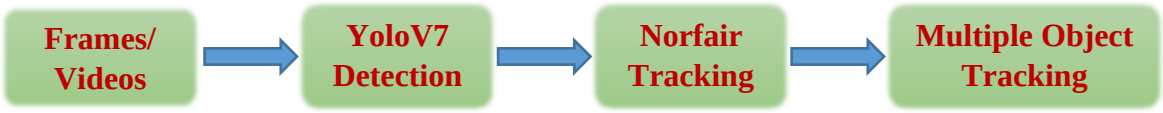

Figure 3: Block Diagram of Multiple Object Teracking Uisng YOLOv7 & Norfair

3.1 YOLOv7

YOLO is a popular real-time object detection algorithm that can be used to detect objects in images and videos. YOLOv7 is one of the variations of the YOLO algorithm. Here are some key points related to using YOLOv7 [10] for object detection

- **Accuracy**: YOLOv7 is known for its good balance between accuracy and speed, making it suitable for real-time applications.
- **Speed**: YOLOv7 is designed for efficiency, which is important for real-time applications where you need to process a video stream quickly.
- **Detection Classes**: YOLOv7 can detect and classify multiple object classes, making it suitable for tracking multiple object types simultaneously.

The YOLO framework consists of three primary parts:

a. Detection Backbone:

The detection backbone is the core neural network architecture responsible for performing object detection in an image or video frame. YOLO uses convolutional neural networks (CNNs) as the backbone. The backbone is responsible for extracting features from the input image and generating a set of feature maps that are used for object detection. In YOLOv1 through YOLOv4, architectures like Darknet-19, Darknet-53, and CSP Darknet were used as detection backbones. In YOLOv5 and YOLOv6, the architecture is further optimized for efficiency and accuracy.

b. Detection Head:

The detection head is the part of the YOLO framework responsible for generating bounding box predictions and class predictions based on the features extracted by the detection backbone. YOLO divides the input image into a grid of cells and predicts bounding boxes for objects within each cell. Each bounding box consists of coordinates (x, y) of the box's center, width (w), and height (h), as well as a confidence score indicating the likelihood of an object being present in that box. Additionally, YOLO predicts class probabilities for each object category for each bounding box. The detection head is where these predictions are generated.

c. Post-processing and NMS (Non-Maximum Suppression):

After obtaining bounding box predictions from the detection head, post-processing steps are applied to refine the results. Non-Maximum Suppression (NMS) is a crucial part of this process. NMS is used to eliminate redundant or overlapping bounding boxes and select the most confident ones. It helps ensure that each detected object corresponds to a single bounding box, preventing multiple detections of the same object. The final output of the YOLO framework is a list of bounding boxes, each associated with a class label and a confidence score.

These three components work together to enable real-time object detection in images and videos, making YOLO a popular choice for various computer vision applications, including autonomous vehicles, surveillance, and object tracking. The architecture of YOLOv7 is given in figure 4.

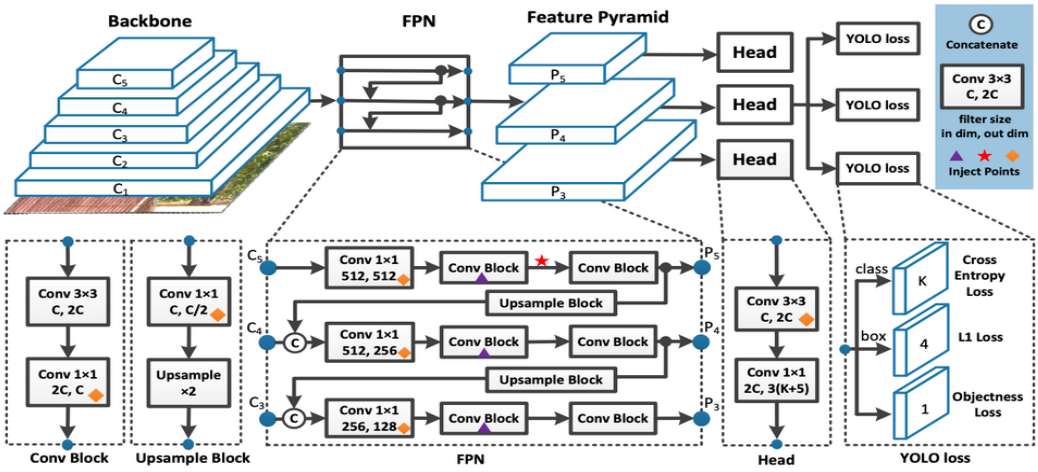

Figure 4: YOLOv7 Architecture

3.2 Norfair Algorithm

Norfair is a customizable, lightweight Python framework for monitoring 2D objects in real time. With just a few lines of code, Norfair may be used to provide any object detection model tracking capabilities. In this work YOLOv7 is used as the detector, which first detects objects in the images or videos before sending those detections to Norfair every frame for object tracking. The detailed explanation of Norfair tracking library is discussed in the following section [11].

3.2.1 Norfair Tracking

Norfair works by forecasting the future positions of each point based on its current positions. It then makes an effort to line up these roughly determined positions with the detector's fresh data points. Any distance function specified by a library user may be used by Norfair to complete this matching. Thus, the level of complexity for each item tracker might vary depending on the situation. A few crucial Norfair algorithmic parameters are discussed in the following section:

a. Distance function:

The distance function is the method used to determine how far newly seen items are from the things it is already tracking. These two inputs should be accepted by function: a detection of type Detection and a tracked object of type Tracked Object, and then calculate the distance and return a float. In this work, a Manhattan distance function is utilized [12].

b. Distance_threshold:

The maximum distance at which a match will occur is called the distance threshold. Beyond this distance, the tracker cannot accommodate detections or things that are being monitored.

c. Hit_inertia_min:

Each tracked item has an internal hit inertia counter that counts the number of times it matches a detection; when it does, the counter rises; when it doesn't, it falls. If the object does not discover a match for a certain amount of frames and then drops below the value determined by this claim, it is destroyed. Its default value is 10.

d. Hit_inertia_max:

Each tracked object keeps an internal hit inertia counter that counts how often it matches a detection; when it does, the counter rises; when it doesn't, it falls. This argument details the size on how long a thing can persist before being detected depends on how inert it can become. 25 is the default value.

e. Initialization_delay:

Delay in initialization. Each tracked object must wait until its internal hit inertia counter exceeds hit inertia min before the Tracker will consider it a possible object to be returned to the user. The object's hit inertia counter must exceed hit inertia min by the amount indicated in the argument initialization delay in order for it to be considered initialized and returned to the user as a real entity. This is a crucial parameter since it explains how the tracker acts when giving each object a distinct ID in video with various frame rates.

3.2.2 Norfair Features

- Any detector that expresses its detections as a sequence of (x, y) coordinates can be utilised with Norfair. Included are detectors that carry out instance segmentation, pose estimation, and object identification.

- The tracker is highly flexible because the user defines the function that determines the separation between monitored objects and detections.

- Any additional information, such as appearance embeddings, may be used by this feature to considerably improve tracking performance.

- Tracking can be easily integrated into complex video processing pipelines to add tracking to existing projects. Simultaneously, a video inference loop can be quickly constructed from scratch using only Norfair and a detector. The only thing limiting inference speed will be the detection network that feeds Norfair's detections.

3.2.3 Manhattan Distance

The Manhattan distance between two vectors (city blocks) is equal to the one-norm of the distance between the vectors. The distance function (also called a “metric”) involved is also called the “taxi cab” metric.

The Manhattan distance as the sum of absolute differences are given in Equation 1

Manhattan Distance [{i, j, k}, {p, q, r}]
Abs [i − p] + Abs [j − q] + Abs [k − r] **1**

The one-norm as Manhattan distance between two city blocks are given in Equation 2

block1 = {1, 2, 3, 4}; block2 = {5, 6, 7, 8}
Norm[block1 - block2, 1] **2**

Equation 3 shows the Manhattan length of two blocks

block1 = {5, 2, -3, 4}; block2 = {1, 6, -7, 8}
{Norm [block1, 1], Norm[block2, 1]} {14, 22} **3**

4. Results And Discussion

In this work the trackers performance is evaluated using the MOT17 and MOT20 benchmark (section 5.3). The YOLOv7 model is merged with Norfair for object tracking. The algorithm utilizes the Manhattan distance function to assess the proximity of newly identified items to those it is currently tracking. The initialization delay used here is 14, as the initialization delay is reduced, the MOTA value decrease. The performance metrics are used to evaluate the model are given in Equation 4 – 9.

$$\text{MOTA} = 1 - \frac{|\text{FN}|+|\text{FP}|+|\text{IDS}|}{|\text{gt}|} \quad \textbf{4}$$

$$\text{IDF1} = \frac{|\text{IDTP}|}{|\text{IDTP}|+0.5|\text{IDFN}|+0.5|\text{IDFP}|} \quad \textbf{5}$$

$$\text{ID-Recall} = \frac{|\text{IDTP}|}{|\text{IDTP}|+|\text{IDFN}|} \quad \textbf{6}$$

$$\text{ID} - \text{Precision} = \frac{|\text{IDTP}|}{|\text{IDTP}|+|\text{IDFP}|} \quad \textbf{7}$$

$$\text{Recall} = \frac{|\text{TP}|}{|\text{TP}|+|\text{FN}|} \quad \textbf{8}$$

$$\text{Precision} = \frac{|TP|}{|TP|+|FP|} \quad (9)$$

Here IDF1 is calculated as the fraction of accurately identified detections divided by the average count of both ground-truth and computed detections. MOTA is Multiple object tracking accuracy. Precision is determined by dividing the number of true positives by the total of true positives and false positives. Recall is calculated by dividing the count of true positives by the sum of true positives and false negatives. IDP and IDR are ID-Recall and ID-Precision respectively. Table 1 and Table 2 present the performance of Norfair on the MOT17 and MOT20 dataset, respectively. The results are compared with the YOLOX-based object detection approach, and the analysis indicates that the proposed model demonstrates promising outcomes. Figure 5 displays the MOT17 tracking result (left: original image, right: multiple object tracked result using YOLOv7 and Norfair) and also shows the situations in which occlusion is effectively handled.

Table 1: Multiple Object Tracking Result using MOT17 Dataset

Methods Using MOT17	IDF1	IDP	IDR	Recall	Precision	MOTA
YOLOX-Norfair	72.6%	73.6%	71.9%	83.7%	86.1%	79.8%
YOLOv7- Norfair	73.8%	72.9%	72.3%	84.6%	86.7%	80.2%

Table 2: Multiple Object Tracking Result using MOT20 Dataset

Methods Using MOT20	IDF1	IDP	IDR	Recall	Precision	MOTA
YOLOX-Norfair	71.3%	72.6%	69.2%	82.7%	84.6%	78.6%
YOLOv7- Norfair	74.1%	75.8%	73.9%	84.9%	85.1%	79.7%

Figure 5: Qualitative evaluation on video sequences (left: original image, right: multiple object tracked result using YOLOv7 and Norfair)

Conclusion

The work outlines a model for enhancing multiple object tracking efficiency by integrating the YOLOv7 detection model with Norfair tracking. In this study, YOLOv7 serves as the detector, initially identifying objects in images or videos and subsequently relaying these detections to Norfair in every frame for tracking purposes. Norfair can be employed to enhance tracking capabilities for any object detection model. The algorithm employs the Manhattan distance function to gauge the proximity of newly detected objects to those it is currently tracking. An initialization delay of 14 is utilized here, and it's noted that as the initialization delay is reduced, the MOTA value decreases. The model was run on the benchmark MOT17 and MOT20 dataset and the results demonstrate that, in terms of MOTA (Multiple Object Tracking Accuracy), Precision, Recall, ID-Recall (IDR) and ID-Precision (IDP). Experimental findings and discussions presented at the end of the paper indicate that the proposed approach significantly enhances tracking performance, even in scenarios involving occlusion, changes in camera speed, and alterations in object appearance.

References:

[1] Kermani, E & Asemani, D, 'A robust adaptive algorithm of moving object detection for video surveillance', Eurasip Journal on Image and Video Processing, vol.1, pp. 27, 2014.

[2] Cannons, Kevin. "A review of visual tracking." *Dept. Comput. Sci. Eng., York Univ., Toronto, Canada, Tech. Rep. CSE-2008-07*, 242, 2008.

[3] Yilmaz, A., Javed, O. and Shah, M., Object Tracking: A Survey. ACM Computing Surveys, Vol. 38, Issue 4, pp.1-4, 2006.

[4] Thenmozhi, T., and A. M. Kalpana. "Adaptive motion estimation and sequential outline separation based moving object detection in video surveillance system. *Microprocessors and Microsystems* 76: 103084, 2020.

[5] Senthil Murugan, A., K. Suganya Devi, A. Sivaranjani, and P. Srinivasan. "A study on various methods used for video summarization and moving object detection for video surveillance applications." *Multimedia Tools and Applications* 77, no. 18: 23273-23290, 2018.

[6] Sun, Peize, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. "Transtrack: Multiple object tracking with transformer." arXiv preprint arXiv:2012.15460, 2020.

[7] Chu, Peng, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. "Transmot: Spatial-temporal graph transformer for multiple object tracking." In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 4870-4880. 2023.

[8] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," arXiv preprint arXiv: 1603.00831, 2016.

[9] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," arXiv preprint arXiv:2003.09003, 2020.

[10] Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7464-7475. 2023.

[11] Kandimalla, Vishnu Vardhan. "Deep learning approaches to classify and track at-risk fish species." (2021).

[12] Sinwar, Deepak, and Rahul Kaushik. "Study of Euclidean and Manhattan distance metrics using simple k-means clustering." Int. J. Res. Appl. Sci. Eng. Technol 2, no. 5, pp.270-274, 2014.