The Journal of Open Source Software

OfflineMOT: A Python Package for multiple objects detection and tracking from bird view stationary drone videos

Yasin Maan Yousif[1], **Awad Mukbil**[1], **and Jörg Müller**[1]

1 Institut für Informatik, Technische Universität Clausthal 38678, Clausthal-Zellerfeld, Germany

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Hugo Ledoux

Reviewers:

- @Shiming94
- @lucaferranti

Submitted: 20 December 2021
Published: unpublished

Summary

The topic of multi objects tracking (MOT) is still considered an open research area (Dendorfer et al., 2020). Among the many available methods for this problem, it is worth mentioning *Deep Sort* (Wojke et al., 2017), where a detection and tracking steps are done in real time and for different types of scenes and areas (for example for pedestrians movement or for vehicles tracking). Another state-of-the-art method is *Tracktor* (Bergmann et al., 2019), where tracking is done by repetitive detections on all the frames in the video.

The importance of the problem comes from its many applications, for example self-driving cars software, traffic analysis, or general surveillance applications.
Unfortunately, due to the variety of scenes and contexts and due to the time constrains that are needed for some applications, there is no one general solution capable of working perfectly for all cases.

For example, between the two cases of a moving camera recording side view road traffic, and a drone camera recording from above, there are many different challanges that should be addressed for each case. Developing one method for both cases, will make this method less effective for addressing each case problems alone.

`OfflineMOT`, the package introduced here, tries to provide a solution to a more restricted problem, namely bird's eye stationary videos without real-time condition. It applies mainly three techniques for detection and tracking on three different priority levels. Below they are listed from lowest to highest.

- The first level is **background subtraction**. It is a fast method to find what pixels have changed in the image (the forground), and this is possible because of the stationary condition. Otherwise, if the drone's camera is moving freely, then the background will be less learnable. Another problem here is the subtle movement of the drone due to wind and control noise. To solve this, a program for fixing the view is implemented. It is based on matching the background features with every next frame, and then transforming the frame if a big movement is detected.
- The second level is **multi-objects tracking** methods such as *kernelized correlation filters* (KCF) (Henriques et al., 2014). This method takes the output of the detection and the next frame and it gives the next postions of the objects. It can also return the failure or success states of tracking.
- The third level is the **deep learning-based detection and classification** of the objects. Here, *Yolo-v4* (Alexey Bochkovskiy, 2020) is used as a model structure where training is done separately. The used code to implement, load and train Yolo structure is taken from (Tianxiaomo, 2020)

Finally, all these parts are implemented separately in the code, making it easy to enable and disable any parts of them, with many tunable parameters. This is done on purpose to facilitate the processing on any new video with different settings by changing only a few parameters.

The following pesudo code Figure 1 illustrates the main workflow implemented in this project.

Algorithm 1: OfflineMOT workflow

```
Input: The top view video
Output: The tracked objects trajectory data in a text file
Initialization of parameters;
for every frame do
    Fix the frame in relation to a reference frame;
    Do background subtraction ;
    Track all the objects ;
    for All the objects do
        if the tracking fails then
            Try to track with a detection step
        end
        track with background subtraction step
    end
    Add the new objects from the background subtraction to the candidates list;
    if Frames count is divisible by N then
        Do detection with Yolo;
        for all the objects and candidates do
            match the object with the detection;
            transfer the candidate to confirmed objects if detected;
        end
    end
    for all the objects do
        if an object is tracked under a percentage from its history then
            Remove Object;
        end
        if an object is overlapping with another above a threshold then
            Remove Object;
        end
    end
    Draw all the objects on the current frame and show it;
end
Save all the successfully tracked objects to a text file;
```

Figure 1: The general workflow of the method.

Statement of need

The specific case for extracting trajectories from top view, stationary traffic videos (for pedestrians, cyclists and vehicles) lacks targeted open source solutions in the literature. Therefore, the development of this package is directed towards helping researchers in the field of traffic analysis or any other fields where trajectories of moving objects are needed.

With the help of this package, the extraction of trajectories from a cyclists' behavior dataset in TU Clausthal will be done. The package has proved its ability of producing very accurate results for different scenes and conditions in these videos. The dataset itself will be published later.

Parameters Tuning Procedure

In order to run the program on a new video, optimally all the parameters should be tuned for all the tracking and detection modules, starting from the basic operations of general settings and background subtraction and ending with detection and post processing operations.

All these parameters can be changed, and saved through the command line. If a suitable set of parameters are found, then it can be saved for later usage to a `.ini` file.

```
cfg = offlinemot.config.configs() # load previous set by passing its file.
cfg['detect_every_N'] = 5
```

In the following table, the most important parameters are listed along with how to tune them for a new video.

Tracking and Detection Module	The Parameter	The Function
General Settings	`detect_every_N`	The frame rate in which the detection is performed. Higher values make the processing faster but mistakes become more probable. For optimal accuracy: 1
	`missing_thresh`	The threshold for deleting a tracked object if its fail-to-success tracking ratio is lower than this number. If the objects are deleted fast consider decreasing this number
	`colors_map`	It is a list of RGB colors to annotate tracked objects based on class. Define your object classes ids and set a color for each of them there starting from object: 1
Background Subtraction	`bgs_shadows`	Set to `True` in case your video has strong shadows.
	`bgs_min_area`	Set to 90% of the smallest object area to be tracked.
View Fixing	`do_fix`	If your video has some fast moves that wil disturb the tracking, then set to True

Detection	`model_name`
	`model_config`
	`classes_file_name`
		These three parameters are to be loaded when you train Yolo on your own dataset.
		They are respectively for the model file path, configuration file path and classes names file path.

	`detection_thresh`	It is the minimum threshold for the detection probability. If you want more detections but with higher false positive rate, you may set it to lower values
	`dist_thresh`	The distance threshold to assign a detection to a tracked object
	`size_thresh`	The minimum threshold of size difference between a detection and assigned tracked object
Filtering and Smoothing	`overlap_thresh`	The maximum intersection to object's area ratio between two objects to delete the first object
	`do_smooth`	A Boolean to determine whether to smooth trajectories or not
	`save_out_video`	Whether to save the output annotated video or not

Figure 2: Important parameters to tune in config.py

The Journal of Open Source Software

Scope

The scope of the problems that can be handled by this package is defined by the following conditions:

1. *The video is stationary*
2. *The real time performance is not a priority*
3. *The view direction of the video is from bird's eye view*
4. *A pretrained detection model for the objects of interest is avaliable*

Regarding the last point, the model provided with the package is trained on random images of cyclists, cars and pedestrians from bird's eye view. This can be enough if the problem is the same, i.e. tracking traffic entities. Otherwise, this model could be a good starting point to train for other kinds of objects if these objects are similar and Yolo v4 is used as a model structure.

Failure Cases

If the video is too noisy, has low resoluation, or the training dataset detection is very different from the video background and objects, then errors in tracking can happen.

As an example, the sample video has some problems with one moving object, because of the different background and the new scene of the video. This can be avoided by retraining the detection part (Yolo network) on similar examples. Additionally, a thorough tunning step for the parameters with the `configs` class should be done to eliminate possible errors in the result.

Acknowledgment

This work was supported by the German Academic Exchange Service (DAAD) under the Graduate School Scholarship Programme (GSSP).The training of Yolo network and labeling the datasets was done by Merlin Korth and Sakif Hossain.

References

Alexey Bochkovskiy, H.-Y. M. L., Chien-Yao Wang. (2020). YOLOv4: YOLOv4: Optimal speed and accuracy of object detection. *arXiv*.

Bergmann, P., Meinhardt, T., & Leal-Taixe, L. (2019). Tracking without bells and whistles. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 941–951. https://doi.org/10.1109/iccv.2019.00103

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., & Leal-Taixé, L. (2020). MOT20: A benchmark for multi object tracking in crowded scenes. *arXiv:2003.09003[cs]*. http://arxiv.org/abs/1906.04567

Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(3), 583–596.

Tianxiaomo. (2020). Pytorch yolo v4. In *GitHub repository*. GitHub. https://github.com/Tianxiaomo/pytorch-YOLOv4

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. *2017 IEEE International Conference on Image Processing (ICIP)*, 3645–3649. https://doi.org/10.1109/icip.2017.8296962