Real-Time* Multiple Object Tracking (MOT) for Autonomous Navigation

Ankush Agarwal[§,1]
ankushag@stanford.edu

Saurabh Suryavanshi[§,2]
saurabhv@stanford.edu

[§]Authors contributed equally for this project.
[1]Google Inc. [2]Stanford University

Abstract

We build a real-time multiple object tracker (MOT) for autonomous navigation using deep convolutional neural networks. To achieve this, we combine state-of-the-art object detection framework, Faster R-CNN, with modified GOTURN (Generic Object Tracking Using Regression Networks) architecture. We freeze the pre-trained weights for the detection network and train the tracking network on the MOT dataset. We show that such end-to-end modular approach for MOT performance is at par with the available computer vision techniques. We also use our model on real-world scenarios to show the generality of our model.

Index Terms - *Computer Vision, Multiple Object Tracking, Object Detection, Deep Convolutional Neural Networks*

1. Introduction

Driverless cars and robots are increasingly relying on optical devices such as a video camera to navigate. To develop a full-proof autonomous navigation system, it is necessary to have a holistic understanding of the surrounding. The state-of-the-art (SOAT) technologies relies on computer vision based methods as well as hardware-based solutions such as LIDAR[1] and SONAR[2]. Such methods have proven to be sub-optimal in extreme cases. For example, hardware based solutions do not work optimally in harsh environment such as rain or storm. While vision based technique are susceptible to variation in optical conditions as well as camera instability [1].

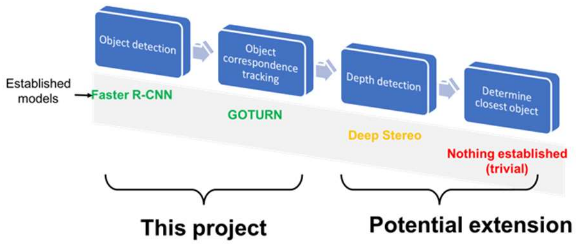

Figure 1: The end-to-end solution to identify the closest object to the camera. In this project, we aim to tackle the first two steps of detecting and tracking multiple objects in the video.

In real life, therefore, we require rich information about the surrounding. We need to understand how the objects are moving with respect to the camera. It would also help to recognize the interaction between objects. For example, in case of the self-driving car the knowledge about the interaction between the pedestrians will help to predict the pedestrian behavior accurately. This prediction will eventually help the self-driving car to make intelligent choices on a crowded road.

We can capture these requirements using a simple model presented in Fig. 1. At the least, we should be able to identify the objects in the video. To get a better understanding, we can track these objects by establishing object correspondence between frames. We can extend this model further and estimate the object depth from the camera. Such rich data can then be processed to provide additional insights such as the closet object to the camera. In this project, we try to solve the most essential part of this problem which is multiple object tracking or MOT as shown in Fig. 1.

In our model, we take a video as an input and track the objects in the video maintaining their identity across frames (Fig. 2).

2. Related Work

2.1 Multiple object tracking as a computer vision problem

* Our model can track single object in real-time. We loop back to track other objects.

[1] LIDAR or Light Detection and Ranging, is a remote sensing method that uses light in the form of a pulsed laser to measure distance between the laser and remote object.

[2] SONAR or a system for the detection of objects by emitting sound pulses and detecting or measuring their return after being reflected.

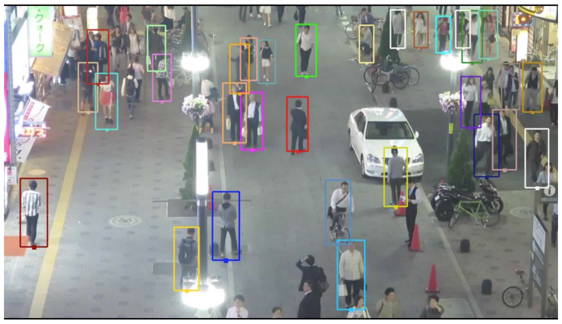
Figure 2: A sample frame from the HCC [2] tracker running on the MOT Dataset. Here we can see that the tracker is able to detect and track most of the pedestrians on the street.

Recently due to the increased availability of the video data, MOT has become an important problem in computer vision [3]. Traditionally, computer vision approaches primarily used Joint Probabilistic Data Association (JPDA) filters [12, 13] and Multiple Hypothesis Tracking (MHT) [11]. Most of the approaches are not suitable for real-time applications such as autonomous navigation as related problems are intractable. These approaches relies on understanding various features or cues from the images. These features include point features, color, intensity, optical flow, gradient, pixel-comparison, region covariance matrix, depth etc. Using these features, the model measures similarity and differences between observations. These approaches are also known as observational models. Other type of models try to build appearances of the objects or a global model. For example, most of these model use momentum as a feature. To establish one-to-one correspondence between objects in different frames, globally optimal solutions such as Hungarian algorithms have been used.

Additional information regarding individual objects can be obtained by incorporating motion models to predict potential position of object in the future frames, interaction model to understand interaction between multiple objects in the frame, and occlusion handling to track objects when they are occluded by other objects in a frame. Eventually, this information is used in dynamic model to investigate object transition across frames. The dynamic model can be probabilistic that uses various features from the observational model to provide a probabilistic distribution for target object. Though we are not interested in traditional computer vision approaches, the information discussed here will help us in getting insights.

2.2 Multiple object tracking using deep learning

Deep learning has only recently made inroads into the field of multiple object tracking [4][5][6]. All the models using neural networks propose a modular approach of dividing the problem into observational and dynamic model as discussed in the previous section. For example, Lee et al. [4] uses VGG-net and ResNet to build the object detector. In addition, they employ Lucas-Kanade Tracker (LKT) algorithm to provide object transitions and motion moves. They then calculate observation likelihood using CNN. Milan et al. [5] employs neural networks to provide end-to-end multi-object tracking. They use RNN to initialize the object (detection) and LSTM to provide object correspondence. Ondruska et al. [6] also focuses on online tracking relying on modular approach and employs RNNs.

3. Our Approach

Taking inspiration from the previous MOT work in computer vision, we divide the problem of MOT into two sub modules: multi-object detection and object correspondence tracking.

For the first part of the problem, we leverage pre-trained Faster-RCNN [7], one of the popular and widely used architecture for multi-object detection. This model combines CNN to propose the region of interest and a region-based (R) CNN module that detects the presence of the object in these regions. As seen from Fig. 3a, the faster R-CNN shares the parameters between two stages allowing efficient detection. This network does not require knowledge of object class to detect the object. As such, the flexible architecture of faster R-CNN allows swapping it's architecture to multiple designs. We can thus use this architecture in the context of a MOT problem.

For the object correspondence tracking, Held et al. recently proposed GOTURN (Generic Object Tracking Using Regression Networks) to track a single object in a video [8]. The GOTURN architecture is shown in Fig. 3b. One of the important advantage of GOTURN is that it is faster than previous approaches and can track the object at 100 fps. GOTURN, trained on offline videos, uses images at time 't' and 't-1' of an online test video, crop them, and feed them individually in different CNNs. The output of these CNNs is then fed to another neural network that tries to establish an object correspondence by trying to look for similar features nearby the original object. A subtle but important assumption is that the object is moving slowly so that the object in frame 't' remains near the bounding box of the object in frame 't-1'. In addition, an important drawback of GOTURN is its single-object tracking. Also, it requires the object detection boundary to be fed as input and does not perform object detection as part of its architecture.

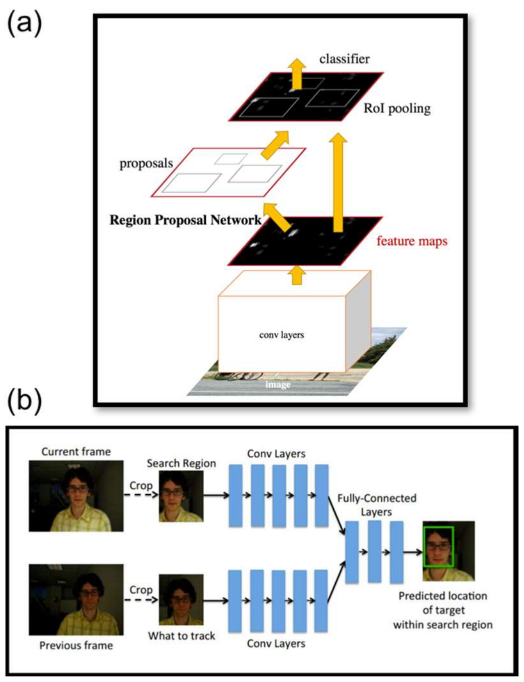

Figure 3: Network architecture for (a) Faster R-CNN [7] and (b) GOTURN [8].

We combine these two architecture to perform real-time MOT as discussed in section 5 of this paper.

4. Dataset and Features

MOT requires labeled video data with bounding boxes and IDs for all objects in the video. For this project, we use the MOT challenge data set [9]. This dataset provides multiple videos with bounding boxes for (almost) all the objects in all frames of the video. This allows us to test and debug our detection and tracking models separately. The dataset contains videos from various scenarios: both static and moving cameras, low and high image resolutions, varying weather conditions and times of the day, viewpoints, pedestrian scale, density, and more. Such wide variety of scenarios helps to train a robust model. In addition, this dataset allows us to benchmark our approach against another proposed model tested on the same dataset [10].

To train our tracking model, we need pairs of subsequent frames with labels. Each video in MOT has over 200 frames and the overall training dataset includes 200k pairs of frames that could be used for training. We note that the MOT dataset is focused exclusively on tracking people. Since our base models for detection as well as tracking can work with objects of other classes, our approach could be easily extended to track objects of any class. For example, in Fig. 4 the output of Faster R-CNN detects a car. We can use our model to easily detect and track this car. However, while evaluating or comparing the model performance, we only track people

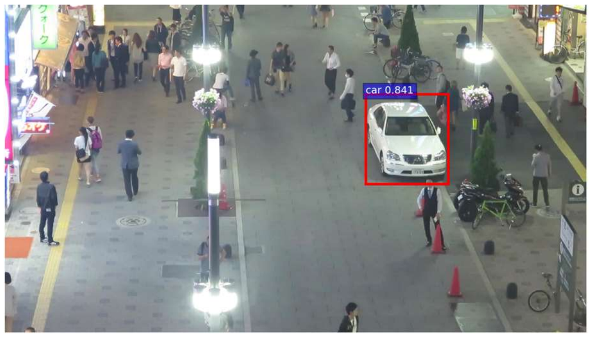

Figure 4: Detecting car using Faster R-CNN. MOT dataset does provide labels for object classes other than people.

as MOT dataset does not have labels for other object classes.

Performance metric

Evaluating performance on MOT is non-trivial. Leal-Taxi et al. [10] have come up with a method to benchmarking MOT model based on precision and accuracy. Precision measures how well the objects are localized i.e. the misalignment between predicted and the ground truth bounding box. While accuracy evaluates how many distinct errors such as missed targets (FN), ghost tracks (FP), or identity switches (IDSW) are made. In ideal world, we expect the precision and accuracy both to be 100 %. This also implies that that the FN, FP, and IDSW are ideally zero. Other important performance metrics for MOT is the maximum number of frame rate the model can handle. Frame rate or the inference time strongly depends on the underlying hardware and less on the architecture. We revisit the performance metric in the results section of this paper.

5. Method

As shown in Fig. 5 and briefly discussed in Sec. 3 of this report, our approach involves strategically stacking two different networks. The pre-trained faster R-CNN network provided very good accuracy when tested on the MOT dataset. We, therefore, decided to freeze the weights in the first part of our network and train only train the tracking network [Code provided in supplementary material].

5.1 Training the tracking network

We use the MOT dataset to train the tracker. The tracker was based on previously available tensor flow code [17]. We train the tracker to provide bounding box of the object in the current frame based on the bounding box of the object in the previous frame. So, our training set includes pairs of consecutive images with bounding box ground truth in both frames. By parsing training videos from MOT dataset, we could generate ~200k such labeled pairs. We first crop the object in the previous

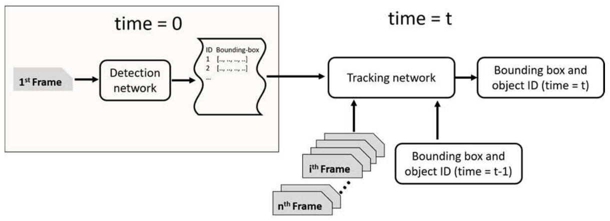

Figure 5: A schematic diagram representing our MOT network. We use the detection network once at the start of the tracking. At t=0, the detection network generates a list of object IDs and corresponding bounding boxes. These bounding boxes are then fed to the tracking network, which tracks each object sequentially frame by frame.

frame along the bounding box (height = *h*, width = *w*) and rescale the cropped image using bilinear interpolation to a predefined dimension ***H***x***W***. ***H*** and ***W*** are hyper parameters but we choose to fix them to 227 as done earlier work [17]. The current frame is also cropped with the same center as the previous bounding box but with twice (another hyper parameter which is fixed and not tuned) the height and width of the previous bounding box (so for the cropped area of the current height = 2*h*, width = 2*w*). We assume that the object in the current frame is near the previous bounding box. The cropped image from the current frame is also scaled using bilinear interpolation to predefined dimension ***H***x***W***.

We train our model using batch size of 100 and Adam Gradient Descent Optimizer with a constant learning rate of 1e-5. Our hardware was NVIDIA Tesla K80 GPU with 12GB Memory. The loss for training was average L1 distance between the predicted bounding box and the ground truth bounding box. We trained our model for 1 epoch and it took about 2 hours for the loss to saturate. The training loss decreases at first but then saturates quickly as observed from Fig 5.

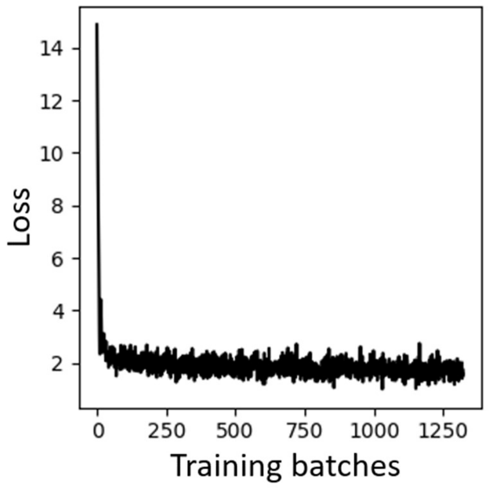

Figure 6: Decrease in training loss with training batches for the tracking network trained over MOT dataset.

5.2 Other Experiments

To improve the tracking further, we performed experiments with look-ahead to see if it helps to overcome the problem of occlusion. In this method, we look-ahead a few frames and compare the confidence of detections of the look-ahead frames. We pick the most confident frame and use linear interpolation to backfill the previous frames which could have occluded objects. To get a sense of confidence, we take the detections of current frame and the detections of a few look-ahead frames and compare the similarity by passing them through a pre-trained VGG16. We use the penultimate layer output vector for feature comparison between the frames. Given our limited time with this project, we did not see any significant improvement over our baseline online-model. However, such model improvements if properly trained should be able to give benefit in accuracy as well as precision. We could potential extend this model to calculate physical object properties such as velocity and acceleration. This additional information in turn will help us in tracking the object.

6. Results

Figure 7 shows a sample output of the detection network. We are only detecting people in this image. A single object tracking using the entire model is shown in Fig. 8. We compare the model output (red) with the ground truth (white). We can use our model sequentially to track multiple objects as shown in Fig. 9.

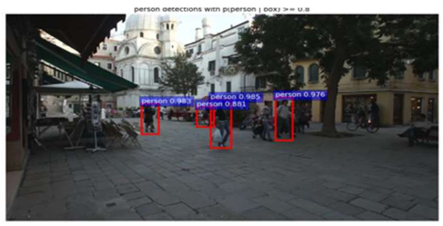

Figure 7: Multiple object detection using Faster -RCNN. Above image is the first frame of a video from the MOT dataset [9].

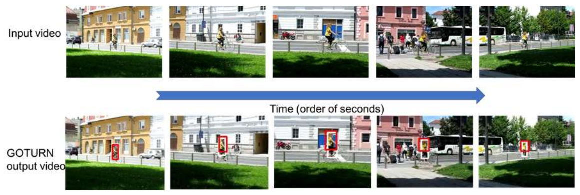

Figure 8: Tracking single object. The model tracks the person on the bike as shown by the box. White box shows the ground truth while the red box shows the model prediction. We have showed few frames of the video as representation of the capability.

Figure 9: The combined model tracking two distinct objects in the crowd from the MOT dataset [9] (The tracked objects are shown in red boxes)

Method	MOTA	MOTP	FAF	MT	ML	FP	FN	ID Sw	Frag
RMOT[14]	18.6	69.6	2.2%	5.3%	53.3%	12473	36835	684	1282
TC_ODAL2[15]	15.1	70.5	2.2%	3.2%	55.8%	12970	38538	637	1716
TDAM3[16]	33.0	72.8	1.7%	13.3%	39.1%	10064	30617	464	1506
Ours	16.2	58.7	3.8%	2.8%	61.2%	10103	35719	812	1362

Table 1: Results of running our tracker. [MOTA: Multi object tracking accuracy, MOTP: Multi object tracking precision, FAF: number of false alarms per frame, MT: number of mostly tracked trajectories, ML: number of mostly lost trajectories, FP: number of false detection, FN: number of missed detection, ID Sw: number of times the ID switches, Frag: total number of times the trajectory is fragmented.

Evaluating the results of a MOT Tracker can be very challenging and no single score can be used to compare various trackers. We evaluate the performance of our tracker primarily using various scenes of the MOT benchmark video database. Table 1 shows the results of our tracker compared to some of the other trackers on MOT16.

Our tracker has good overall precision (MOTP) MOTP stands for Multiple Object Tracking Precision, which includes the misalignment between the annotated and the predicted bounding boxes. This shows that GOTURN is a very good generic tracker and generalizes well to the MOT dataset. The accuracy (MOTA) for our model is mediocre at about 16.2. Current State of the Art trackers have MOTA of about 45.

7. Conclusions and Future Work

We presented a simple real-time multi-object tracking framework. We leveraged two different models, the Faster R-CNN's high detection performance and a state-of-the-art tracking method (GOTURN), to develop an end-to-end system. Our modified model achieved a very good precision (MOTP) score. Although it does not achieve a very high score on MOTA, it demonstrates how we can re-use specialized Convolutional Neural Networks and stack them together to tackle a new problems. We also proposed additional potential improvements such as estimating object velocity and acceleration to improve the tracking precision and accuracy.

References

[1] Eigen, D., Puhrsch, C., and Fergus, R., "Depth map prediction from a single image using a multi-scale deep network", *NIPS*, 2014.
[2] https://motchallenge.net/vis/MOT16-06/HCC
[3] Wenhan Luo, Junliang Xing, Xiaoqin Zhang, Xiaowei Zhao, and Tae-Kyun Kim. "Multiple Object Tracking: A Literature Review", arXiv:1409.7618, 2015
[4] Lee., B, Erdenee, E., Jin, S., and Rhee, P.K. "Multi-Class Multi-Object Tracking Using Changing Point Detection", arXiv:1608.08434, 2016
[5] Milan, A., Rezatofighi, S.H., Dick, A., Reid, I., and Schindler, K. "Online Multi-Target Tracking Using Recurrent Neural Networks", arXiv:1604.03635v2, 2016
[6] Ondruska, P., Dequaire, J., Wang, D.Z., and Posner, I. "End-to-End Tracking and Semantic Segmentation Using Recurrent Neural Networks", arXiv:1604.05091, 2016
[7] Ren, S., He, K., Girshick, R., and Sun, J. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", arXiv:1506.01497, 2016
[8] Held, D., Thrun, S., Savarese, S., "Learning to Track at 100 FPS with Deep Regression Network", arXiv:1604.01802, 2016
[9] Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K., "MOT16: A Benchmark for Multi-Object Tracking", arXiv:1603.00831, 2016
[10] Leal-Taixe´, L., Milan, A., Schindler, K., Cremers, D., Reid, I., Roth, S. "Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking", arXiv:1704.02781, 2017
[11] D. Reid et al., "An Algorithm for Tracking Multiple Targets", Automatic Control, vol. 24, pp. 843–854, 1979.
[12] H. W. Kuhn et al., "The Hungarian method for the assignment problem", Naval Research Logistics Quarterly, vol. 2, pp. 83–97, 1955.
[13] S. H. Rezatofighi, A. Milan, Z. Zhang, A. Dick, Q. Shi, and I. Reid, "Joint Probabilistic Data Association Revisited," in International Conference on Computer Vision, 2015.
[14] J. H. Yoon, M. H. Yang, J. Lim, and K. J. Yoon, "Bayesian Multi-Object Tracking Using Motion Context from Multiple Objects," in Winter Conference on Applications of Computer Vision, 2015
[15] S. H. Bae and K. J. Yoon, "Robust Online Multi-Object Tracking based on Tracklet Confidence and Online Discriminative Appearance Learning," Computer Vision and Pattern Recognition, 2014.
[16] Y. Min and J. Yunde, "Temporal Dynamic Appearance Modeling for Online Multi-Person Tracking,"
[17] https://github.com/tangyuhao/GOTURN-Tensorflow/