Vehicle Detection and Tracking using YOLO, SORT, and MDP

Jason Kurohara
Stanford University
Stanford, CA
jkuro@stanford.edu

Allen Zhu
Stanford University
Stanford, CA
allenzhu@stanford.edu

Abstract

The ability to identify and track vehicles is critical across a wide array of sectors, from urban planning to surveillance to public safety. In order to achieve the goals in these sectors, users and stakeholders need to obtain reliable information on vehicles in real time. Additionally, the rapid growth and commercialization of drone technology has led to an abundance of video and images of traffic scenes from an aerial view that fuel research and numerous applications in computer vision. This paper lies at the intersection of drone technology and computer vision; we use a convolutional neural network to identify and different multi-object trackers to track cars in sequential images trained on a variety of scenes captured by an Unmanned Aerial Vehicle (UAV). We compare our approach to baseline results produced by a Faster Region Convolutional Network (Faster-RCNN) architecture for detection followed by Markov Decision Processes (MDP) for tracking. Our approach uses the You-Only-Look-Once (YOLOv3) real-time object detection system and Simple Online and Real-time Tracking (SORT) algorithm to improve object detection and tracking of vehicles from an aerial view. We also compare the efficacy of Markov Decision Processing (MDP) as an object tracker over SORT. In this paper, we show that our network outperforms state-of-the-art baseline models in many metrics such as average precision, recall, F1, MOTA scores, and more.

1. Introduction

Developments in drone technology and artificial intelligence intersect at the demand for intelligent unmanned aerial systems. In the global drone industry, UAVs have been applied in many areas such as security and surveillance, agriculture, and 3D mapping. In comparison to other types of camera platforms, UAV cameras come with their own separate set of challenges: high density as a result of wide angle views, small object detection due to the high cruising altitude of drones, and camera motion. Convolutional neural networks and object trackers are used in tandem to tackle the high level semantical tasks in computer vision such as object recognition and tracking of people, vehicles, buildings, and monuments.

Unmanned Aerial Vehicles (UAVs) are excellent mobile platforms for collecting high resolution images and videos. Satellite images do not provide the same quality of temporal data but still offer high density, small object images.

This paper describes our approach to accurately identify and track cars on roads in comparison to other state-of-the-art models. The goal is to be able to recognize the same car in multiple frames of a video. This will allow us to identify moving vehicles and their trajectory.

We create a neural network to detect all visible vehicles (cars, trucks, buses, etc.) and create bounding boxes to segment an image. Additionally, we track the detected vehicles through videos in a variety of scenes (weather conditions, camera view, flying altitude, etc).

We test two approaches to achieve this goal. Our first model uses the YOLOv3 architecture (a fast and reliable object detector) which then feeds a bounding box and class for each detected objects to the Simple Online Realtime Tracking (SORT) algorithm that matches detected objects across frames. Our second model uses the YOLOv3 architecture which then feeds a bounding box and class for each detected object to a MDP (Markov Decision Process) tracker.

The workflow for the first approach is described as follows: we start by feeding a 1024 by 540 pixel image into the YOLOv3 model [10]. Our YOLOv3 network is only trained to detect three classes (truck, motorbike, car). A vehicle is considered as classified correctly and passed on to the object tracker if it passes a confidence threshold of 0.5.

After our trained YOLOv3 network outputs a prediction for the bounding box and class, we feed this output into the SORT algorithm [1]. This is a simple online tracking framework that focuses on frame-to-frame prediction and association to track the detected vehicles through time. SORT tracking performance is evaluated using Multi-Object Tracking (MOT) benchmark. [6].

Our second approach differs when the output of the YOLOv3 network is fed into an MDP tracker [13]. MDP

trackers differ from SORT in that MDP trackers are trained using reinforcement learning that treats the appearance and reappearance of targets as state transitions in the tracker. The SORT and MDP trackers output an object ID and its bounding box coordinates and sizes through the sequential images. Both these techniques are discussed in more detail later in the **Methods** section of this paper.

We evaluate the performance of our models to benchmark results produced by Du et al [2].

2. Related Work

We divide the task into two parts: object detection and object tracking. There are many models for these tasks as described by the Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking by Du et al [2], which include Faster-RCNN, R-FCN, RON, and SSD. As mentioned earlier in this paper, YOLOv3 is an object detection system targeted for real time processing. YOLOv3 has 53 convolutional layers for performing feature extraction and uses binary cross-entropy loss to output the coordinates and size of a bounding box and label of a detected object. In theory, YOLOv3's Average Precision (AP) metric is similar but 3 times faster in comparison to other state of the art object detectors. We decided to use this framework for its simple implementation and speed. For more information on its technical aspects, see the YOLOv3 paper here [10].

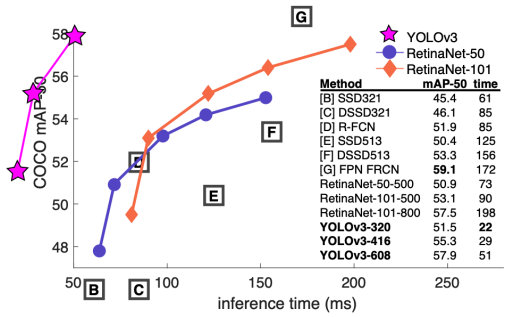

Figure 1. YOLOv3 performance compared to other architectures.

Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking [2] provides benchmark metrics on state-of-the-art models performing object detection and tracking on our dataset. The best reported model for tracking is an Faster-RCNN as the object detector and an MDP tracker. The best reported model for detection is an R-FCN detector. We compare our two models to these results and compute other useful metrics.

3. Methods

In this section, we discuss the YOLOv3 architecture, MDP, and SORT algorithms in more detail.

3.1. YOLO

The first step of the task is to perform object detection in images. Multiple objects are identified in the image, classified under 80 class labels, and a location is determined by a bounding box on the image. Since the Unmanned Manned Aerial Vehicle Benchmark: Object Detection and Tracking [2] paper did not attempt to use the YOLO architecture, we wanted to see if this state-of-the-art, fast, real-time object dectection algorithm would outperform the architectures proposed in the paper.

We trained the YOLOv3 architecture on the UAVDT data set. This dataset is described in greater detail in section 4 of this paper. A vehicle is classified correctly if its intersection over union threshold (the ratio of area of overlap of the predicted and ground truth labels to total area of the predicted and ground truth boxes) is greater than 0.5.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2. Darknet-53

YOLO is one of the fastest algorithms for object detection because it looks at the entire image at test time and its predictions are informed by the global context of the image. Additionally, YOLO makes bounding box predictions with a single network evaluation unlike R-CNN and Faster R-CNN which require thousands of evaluations for a single image.

For feature extractions, the YOLOv3 network uses the Darknet-53 architecture shown above of 53 layers. 53 more layers for detection are added on to this detector so that the entire network has 106 layers.

Our YOLOv3 model operates on 416 x 416 images and generates detections of shape 1 x 1 x (B x (5 + C)). Each of

these detections contains a x, y, w, h and confidence score, which is where the 5 comes from. C is the conditional class probabilities as $Pr(Class_i|Object)$, which are probabilities conditioned on the grid cell containing an object. Our YOLOv3 based on YOLOv3's trained on the COCO data set had 80 class probabilities. B is the number of detections per cell, which in our YOLOv3 model is 3. With these numbers, the detection is of size 255.

There were 9 different bounding box sizes for our COCO-based YOLOv3 model. The sizes of the bounding boxes had widths and heights of (1013), (1630), (3323), (3061), (6245), (59 119), (116 90), (156 198), and (373 326). These 9 box sizes were grouped into 3 different groups according to their scale. Each group was assigned to a specific feature map in detecting objects.

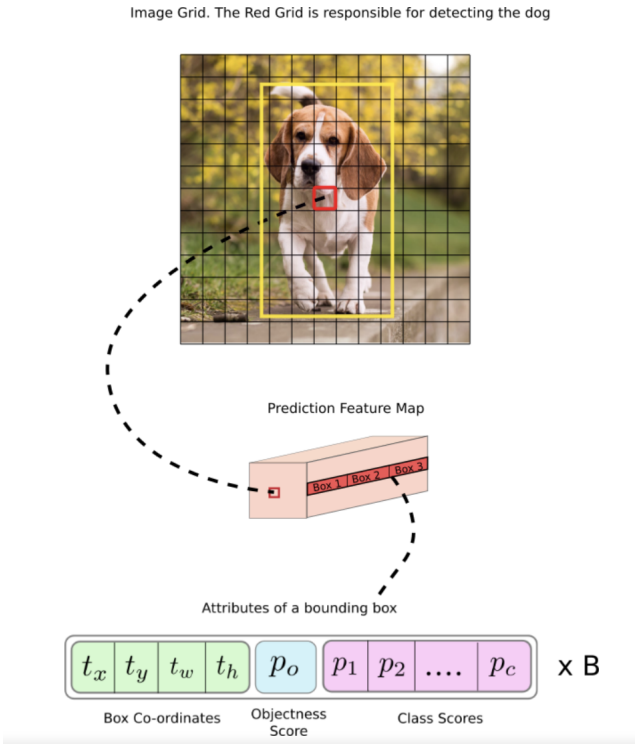

Figure 3. The image is divided into smaller equally sized grids. The red box contains the center of the ground truth object and is "responsible" for predicting the object.

YOLOv3 predicts boxes at 3 different scales. The first detection is made by the 82nd layer of the network. For the first 81 layers, the image is down sampled by the network, so that at this point feature map is of size 13 x 13. One detection is made here using the detection kernel, giving us a detection feature map of 13 x 13 x 255.

Afterwards, the feature map from layer 79 is passed through convolutional layers and up sampled to dimensions of 26 x 26. This feature map is combined with other layers and passed through a few 1x1 convolutional layers, to ultimaely yield a detection feature map of 26 x 26 x 255.

For the last stage, a similar procedure is performed on the feature map from layer 91 as it is subjected to few convolutional layers before being combined with other layers and passed through a few 1x1 convolutional layers. The final feature map here is of size 52 x 52 x 255.

The 13 x 13 layer is mainly in charge of detecting large objects, while the 52 x 52 layer detects for the smallest objects, and the 26 x 26 layer detets medium-sized objects.

During training, we optimize the multi-class loss function as displayed below. $\vec{1}_i^{obj}$ denotes if the object appears in cell i and $\vec{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i is "responsible" for the class prediction. λ_{coord} and $\lambda noobj$ are parameters that increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects, respectively. So λ_{coord} = 5 and $\lambda noobj$ = 0.5.

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

Figure 4. Multi-class loss function for YOLOv2

The last three terms of the YOLOv2 loss function in Figure 4 are squared errors. However, our YOLOv3 model replaces them with cross-entropy error terms. The object confidence and class predictions in YOLOv3 are instead calculated using logistic regression.

For more information, see the original YOLOv3 paper here [9]. The full network architecture is visualized below.

3.2. Markov Decision Processes (MDPs)

Markov Decision Processes or MDPs use reinforcement learning to track objects through time. The lifetime of a target is modeled as a MDP that consists of a target state $s \in S$ which is the status of the target, an action $a \in A$ that is performed on the object, a state transition function $T : S$ x $A \rightarrow S$ that describes the effect of each action on each state, and a reward function $R : S$ x $A \rightarrow R$ after executing action a to s.

Each object is in state $S_{active}, S_{tracked}, S_{Lost}, S_{inactive}$ which are subspaces of S. The seven transition functions

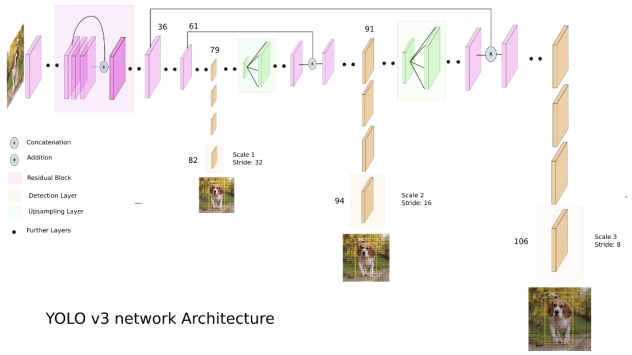

Figure 5. YOLOv3 network Architecture. Image credits to Ayoosh Kathuria [4]

model the actions that can be performed on any given state. For instance, a_4 on a tracked target transitions the object into S_{Lost}.

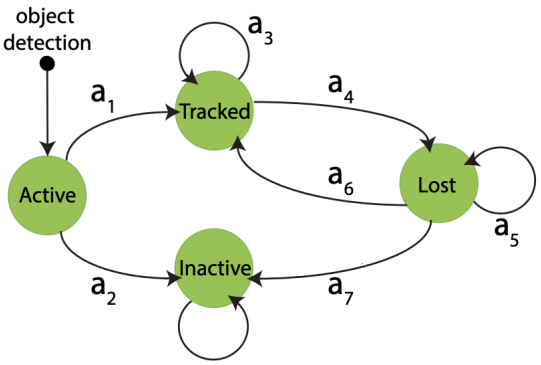

Figure 6. Visual representation of a MDP for one object [13].

While the technical details of the Markov Decision Process are out of scope of this paper, after learning the policy/reward function of the MDP using reinforcement learning, we dedicate a MDP for each object. Thus, given sequential video frames, targets are tracked in states. Additionally, we initialize a MDP for each object detection which is not covered by any tracked targets to account for multiple objects. For more information, see Yu Xiang's paper on Multi-Object Tracking by Decision Making [13].

3.3. Single Online Real-time Tracking (SORT)

SORT is a multi-object tracker suggested by Alex Bewley that utilizes a combination of a Kalman Filter and Hungarian algorithm [5] implementation as motion prediction and data association components. In SORT, appearance features beyond the detection component are ignored, and only the bounding box position and size are used for both motion estimation and data association. The state of each target is modeled as:

$$\vec{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s},]$$

Here, u and v represent the horizontal and vertical pixel location of the target center, $\dot{u}$,$\dot{v}$ represent the target's horizontal and vertical velocities, and s and r represent the area scale and the aspect ratio of the targets bounding box. Upon association of a detection with a target, the detected bounding box is used to update the target state in which the velocities are solved with a Kalman filter [12]. If there is no association the linear velocity model is used to update the target state. To assign detections to targets, an assignment cost matrix is computed as the IOU distance between each detection and all predicted bounding boxes from the Kalman filter. The Hungarian algorithm is used to optimally solve this assignment. SORT has a MOTA score of 33.4% when trained on the MOT benchmark sequences [6].

3.4. Metrics

First to evaluate our detection model, we calculate the precision, recall, F1 score and average precision on the test set. Precision, recall, and F1 score are the usual derived measures, $P = TP/(TP + FP)$, $R = TP/(TP + FN)$, and $F1 = 2PR/(P + R)$. Average precision is the Area Under the Curve (AUC) average precision and is computed by

$$AP = \sum((r_{n+1} - r_n)p_{itp}(r_{n+1})) \quad (1)$$

$$p_{itp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r}) \quad (2)$$

where r_n and r_{n+1} are adjacent sampled recall value points, and $p_{\tilde{r}}$ is the sampled precision at recall $\tilde{r}$ [3].

In order to evaluate our model on multi-object tracking, we utilize the standard MOT Benchmark evaluation metrics [6][11]. These include identification precision (IDP), identification recall (IDR), and the corresponding F1 score IDF1 (the ratio of correctly identified detections over the average number of ground-truth and computed detections), Multiple Object Tracking Accuracy (MOTA), and Multiple Object Tracking Precision (MOTP). For each tracked target, we also classify them as Mostly Tracked targets (MT, the number of ground truth trajectories that are covered by a track hypothesis for at least 80 percent of their lifespan), Partially Tracked targets (PT, the number of ground truth trajectories that are covered by a track hypothesis for at least 20 percent of their lifespan), and Mostly Lost targets (ML, the number of ground truth trajectories that are covered by a track hypothesis for less than 20 percent of their lifespan). For each sequence we track we also compute the total number of False Positives (FP), the total number of False Negatives (FN), the total number of ID Switches (IDS) between frames, and the total number of times a trajectory is Fragmented (FM).

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \quad (3)$$

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \quad (4)$$

$$IDP = \frac{\sum_t TP_t}{\sum_t (TP_t + FP_t)} \quad (5)$$

$$IDR = \frac{\sum_t TP_t}{\sum_t (TP_t + FN_t)} \quad (6)$$

$$IDF1 = \frac{2\sum_t TP_t}{\sum_t (2TP_t + FP_t + FN_t)} \quad (7)$$

In addition to the aforementioned variable explanations, TP refers to true positives, t is the frame index, GT is the number of ground truth objects, c_t denotes the number of target matches in frame t and $d_{t,i}$ is the bounding box overlap of target i with its ground truth object.

4. Dataset and Features

4.1. Training and Validation Dataset

To train and validate our model, we utilize the UAVDT benchmark provided by Du et al [2]. The UAVDT benchmark consists of 100 video sequences, which are selected from over 10 hours of videos taken with a UAV platform at a number of locations in urban areas, representing various common scenes including squares, arterial streets, toll stations, highways, crossings and T-junctions. The videos are recorded at 30 frames per seconds (fps), with the JPEG image resolution of 1080 x 540 pixels. About 80, 000 frames in the UAVDT benchmark data set are annotated with over 2,700 vehicles with 0.84 million bounding boxes. Regions that cover too small vehicles are ignored in each frame due to low resolution. We split the data into training, validation, and testing with an 80-10-10 split.

Figure 7. This is an image from our dataset. The UAVDT dataset contains scenes from a variety of angles, weather, and altitude

Each video sequence is annotated with information about each vehicle in each frame of the video. For a certain car of interest in a frame these annotations document the frame number, car id number, and bounding box information for each car. The car id is used to track cars from one frame to subsequent frames. The same car will have the same car id throughout the video sequence.

After processing and formatting the data, a sample processed picture with a bounding box from the ground truth annotation labels and an example of an annotation label for which object detection are shown below.

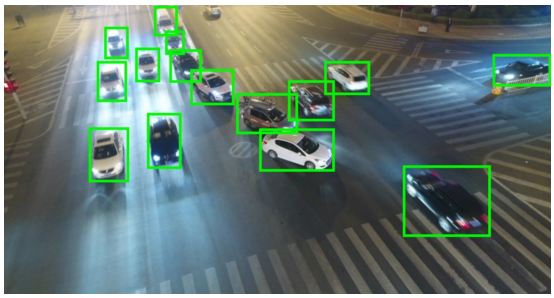

Figure 8. Our dataset is labels a ground truth for the bounding box coordinates and size. In this image, we visualize the bounding box around each vehicle.

In comparison to other datasets, the UAVDT dataset provides higher object density with moving vehicles (as opposed to parked vehicles) and is captured from various weather conditions, flying attitudes, and camera views.

5. Experiments/Results/Discussion

We evaluate our model based on its ability to detect objects and also track them.

5.1. Benchmark

The benchmark model we use comes from the UAVDT paper [13]. The model that received the best MOTA score of 43.0% is an Faster-RCNN as the object detector and MDP as the object tracker. Although this model received the highest MOTA score, the Faster-RCNN only had an average precision of 22.32%. The Faster-RCNN performed worse than the R-FCN object tracker that had an average precision of 34.35%. Despite the R-FCN's higher average precision, its combination with the various object trackers tested in the UAVDT paper yield a lower MOTA score than the Faster-RCNN and MDP model. Therefore, our benchmark model is the Faster-RCNN and MDP.

5.2. Object Detection Results

For training, we began with untrained weights of the YOLOv3 network configured detect the 80 classes from the COCO dataset [7]. Figure 9 shows the loss over 63,000 images.

Before feeding the images into YOLOv3, the images are normalized and resized to a 416 x 416 image. We use a

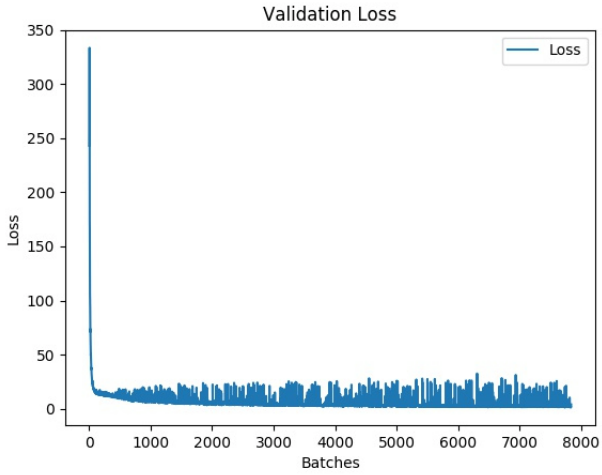

Figure 9. We graph the loss over around 63,000 images (8 batches). The training set contains 40 scenes that vary in illumination, altitude, angle, etc.

batch size of 8 images, momentum value of 0.9, and a decay rate of 0.0005.

Our model is configured to detect 80 different classes. However, changing the number of classes to only detect cars, trucks and motorbikes (the only classes in our dataset) did not significantly improve the evaluation metrics. So, we trained the 80 class object detector on the dataset with only the three types of objects.

Figure 10. Our dataset only labels cars, trucks, and motorbikes. After training for 7 epochs, we see that our model detected most of the vehicles in the image along with a truck and motorbike.

The evaluation metrics on the validation set after training for 7 epochs are listed in Figure 11.

After training our model to find the best weights, we evaluate the model on 4 different test scenes called M1303, M1304, M1305, and M1306. Each scene is recorded over 40 seconds.

The YOLOv3 model performs detections better than all of the object detectors suggested in the UAVDT paper, with higher Average Precision scores for all of the tested scenes by at least 10% and at most 26.45% compared to the UAVDT paper's R-FCN, the best detector mentioned in the paper. Other state-of-the-art detectors include Single Shot MultiDetector (SSD), Faster R-CNN, and Reverse Object Connection with Prior Networks (RON) which had AP scores of 33.62%, 22.32%, and 21.59%, respectively. YOLOv3 outperforms all these models.

Epoch	Average Precision	F1	Precision	Recall
1	0.40015	0.52680	0.54149	0.51288
2	0.32969	0.51344	0.63695	0.43004
3	0.35594	0.56401	0.65146	0.49726
4	0.41825	0.57822	0.57495	0.58153
5	0.39798	0.56930	0.57155	0.56706
6	0.5116	0.605775	0.5790678	0.63505890
7	0.35731	0.53782	0.68978	0.44072

Figure 11. After each epoch, we test on a validation set to generate the evaluation metrics. We used the best weights from epoch 6 to test our model.

Test Scene	Average Precision	F1	Precision	Recall	Runtime
M1303 (T-intersection)	0.4617430219	0.6356060	0.6971405	0.58405	15.82 fps
M1304 (T-intersection)	0.60866821	0.668687	0.589402	0.7726192	15.23 fps
M1305 (Highway)	0.508115334551	0.6173818628	0.654620082	0.58415222	16.57 fps
M1306 (T-intersection)	0.5716698922	0.648893220	0.70771333	0.58153	16.599

Figure 12. The T-intersection and Highway are the type of scenes that are captured by the drone. The runtime is the number of frames evaluated over one second.

We also evaluate our model's run-time performance. Using 1 vCPU with 6 GB memory and a NVIDIA Tesla K80 GPU, the speed of the you-only-look-once algorithm truly shines. R-FCN and other architectures from the UAVDT benchmark ran at less than 5 fps, which is far too slow for practical applications. Our model, on the other hand, achieves an average run-time of 16.05 fps, which is 2x faster than the benchmark R-FCN and usable on a real-time video feed. This is a significant improvement from the benchmark model.

To see evaluation metrics from the UAVDT paper in more detail, see [13].

5.3. Object Tracking Results

The next part of this paper discusses the evaluations on object tracking. After the YOLOv3 network outputs the bounding box coordinates and confidence score, the results are fed into a multi-object tracker. We utilized both the SORT and MDP trackers. We compare our results using the SORT and MDP trackers to the benchmark Faster-RCNN and SORT model and the benchmark Faster-RCNN and MDP model.

In order to train the MDP tracker, we pick 4 random training scenes from our training set. We used few training scenes due to constrained resources for this project. Despite using limited training data, our results using the MDP tracker are significantly better than our results from the SORT tracker.

Metrics	M1303	M1304	M1305	M1306
MOTA	56.2	31.9	34.1	40.0
MOTP	76.6	75.1	73.4	76.5
IDF1	52.3	41.0	46.0	38.0
IDP	59.3	39.0	55.3	45.1
IDR	46.8	43.2	39.3	32.8
GT	60	81	63	74
MT	26	36	12	11
PT	26	27	40	58
ML	8	18	11	5
FP	1164	9670	2027	3331
FN	3459	6968	5257	9203
IDs	153	271	93	386
FM	209	359	166	504

Figure 13. MOT Metrics computed from feeding the YOLOv3 detections into the SORT tracker.

Metrics	M1303	M1304	M1305	M1306
MOTA	64.1	41.3	36.0	50.7
MOTP	76.5	75.1	73.0	76.9
IDF1	68.9	68.8	59.0	60.0
IDP	72.9	64.8	67.0	68.1
IDR	65.3	73.5	52.7	53.5
GT	60	81	63	74
MT	34	42	14	19
PT	18	24	41	52
ML	8	15	8	3
FP	1352	8922	2370	2932
FN	2499	5576	4760	7548
IDs	52	72	31	148
FM	110	193	104	255

Figure 14. MOT Metrics computed from feeding the YOLOv3 detections into the MDP tracker.

From these results it is evident that compared to the benchmark model, in addition to generating better detections, YOLOv3 in combination with both the SORT and MDP trackers is also better at multi-object tracking. The best detector with SORT that was tested in the UAVDT paper was Faster-RCNN with SORT, which achieved a 39.0% MOTA score and a 43.7% IDF1 score. Averaged over our four test scenes, YOLOv3 with SORT achieves a 40.6% MOTA score and a 44.3% IDF1 score. Compared to Faster-RCNN with SORT, YOLOv3 with SORT is more accurate with detections running 3 times faster. YOLOv3 with SORT outperforms all of the combinations of detectors with SORT tested in the paper. In using the MDP tracker, the benchmark set by the UAVDT paper, Faster-RCNN with MDP, achieves a MOTA score of 43.0% and 61.5% IDF1 score. Averaged over our four test scenes, YOLOv3 with MDP achieves a 48.0% MOTA score and a 64.2% IDF1 score. The YOLOv3 with MDP combinations outperforms all of the detector-tracker combinations in the UAVDT paper.

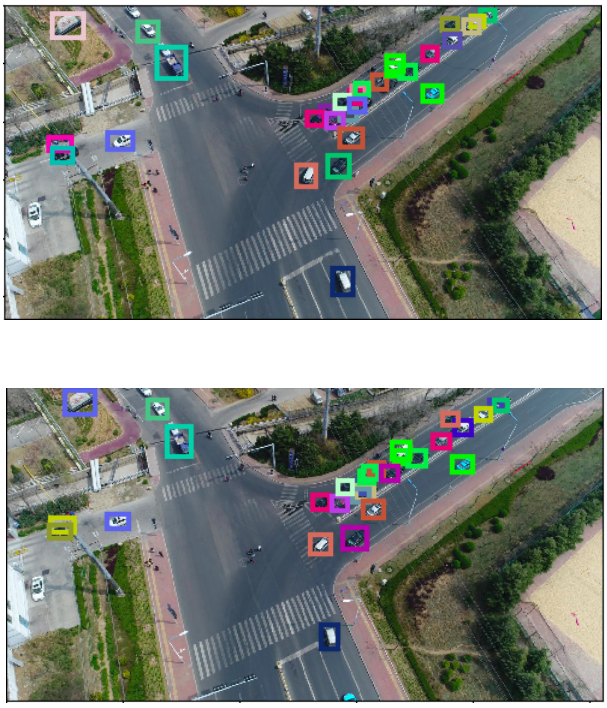

Figure 15. Comparison of the detection associations computed by SORT from the bounding boxes input to it by YOLOv3. The top shows frame 139 from the M1201 sequence, while the bottom shows frame 159 also from the M1201 sequence. Boxes of the same color are associations of the same target through the frames.

6. Conclusion/Future Work

This paper presents a fast, accurate, and reliable model to detect and track cars from an aerial view. The applications to this technology are ubiquitous because there are many use cases in the areas of surveillance, military, commercial, and urban planning.

The YOLOv3 network outperforms other state-of-the-art models like R-FCN, Faster R-CNN, and SSD in both runtime and accuracy/precision. YOLOv3 runs 3x faster than other models at 16.05 fps, making it useful for real-time applications on aerial platforms and other embedded systems. For object tracking, having the fastest detector is critical in order to pass detections to the tracker. With object tracking, the MDP tracker outperforms SORT, with a higher MOTA score of over ten percent in some scenes. However, in order to perform real-time multi-object tracking, the MDP tracker is too slow to be of use. On the M1303 test sequence for example using 1 CPU, SORT processed 109.5 frames per second while MDP only processed 0.51 frames per second. Further, MDP must first be trained, while SORT does not. We recommend using YOLOv3 and SORT to perform real-

time object tracking for its combination of accuracy and speed.

In the future, we would like our model to run at least 30 fps in order to be used on UAVs. Another useful improvement is to use a smaller network that achieves similar scores as YOLOv3 and MDP, since these models are quite large and bulky on mobile platforms. Additionally, we would like to perform object detection and tracking on objects other than trucks, motorbikes, and cars to increase the practicality of our model.

7. Appendix

Below is the equation for binary cross entropy loss, used in the last 3 terms of the YOLOv3 loss function.

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

Figure 16. Binary cross entropy loss

8. Contributions and Acknowledgments

J.K. adapted the YOLOv3 network, trained the model, tuned hyper-parameters, and processed image and label datasets. A.Z. adapted the SORT and MDP trackers to perform multi-object tracking, trained the MDP tracker, and evaluated MOT metrics. J.K. and A.Z. wrote the paper. This project has not been submitted to a peer-reviewed conference or journal. A.Z. and J.K. would like to thank the CS231N staff for a fantastic quarter of artificial intelligence and computer vision.

8.1. Starter Code

The YOLOv3 algorithm was implemented by Erik Linder-Norn [10]. We made changes to these files in order to fit the model into our pipeline and datasets and run object tracking on the model outputs. We did start by using pretrained weights and trained the model on the UAVDT dataset [13]. We also did not write the source code for the SORT algorithm. The implementation can be found on Alex Bewley's github [1]. MOT metrics are implemented using the python MOT metrics library. The MDP source code was implemented by Yu Xiang. Yu Xiang's implementation can be found here [13]. The YOLOv3 model is implemented in PyTorch [8].

References

[1] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, Sep. 2016.

[2] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian. The unmanned aerial vehicle benchmark: Object detection and tracking. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[4] A. Kathuria. Whats new in yolo v3?, April 2018. towards-datascience.com [Online; posted 23-April-2018].

[5] H. W. Kuhn and B. Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[6] L. Leal-Taixé, A. Milan, I. D. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *CoRR*, abs/1504.01942, 2015.

[7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[8] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[9] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[10] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[11] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. *CoRR*, abs/1609.01775, 2016.

[12] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[13] Y. Xiang, A. Alahi, and S. Savarese. Learning to track: Online multi-object tracking by decision making. In *International Conference on Computer Vision*, 2015.