Multiple Vehicle Tracking with Enhanced SORT

Jordan Bierbrier
KTH Royal Institute of Technology
Stockholm, Sweden
jordanbi@kth.se

Alessandro Maino
KTH Royal Institute of Technology
Stockholm, Sweden
maino@kth.se

***Abstract*—Efficient multiple vehicle tracking is critical for advancing traffic monitoring, intelligent transportation systems, and collision avoidance technologies. This paper presents enhancements to the Simple Online and Realtime Tracking (SORT) algorithm aimed at improving tracking efficiency, robustness, and reliability. By experimenting with different motion models and varying observation frequencies, we demonstrate improved handling of occlusions and reduced computational requirements. The proposed method is evaluated on multiple datasets, showing practical improvements in tracking accuracy and runtime performance with real-world applicability.**

***Index Terms*—Multiple Vehicle Tracking, SORT, YOLO, Kalman, Computer Vision**

I. Introduction

Vehicles are a nearly ubiquitous mode of transportation, with freeways and highways serving as critical infrastructure for efficient travel. However, this domain presents numerous challenges, including high accident rates, the need for real-time traffic monitoring, and the development of intelligent transportation systems (ITS) capable of autonomous planning and prediction. A common requirement across these challenges is the ability to accurately track multiple vehicles. Effective vehicle tracking supports applications such as automated speed limit enforcement [1], reduces the burden of manual traffic camera monitoring through computer vision techniques, and enables predictive modeling necessary for ITS. Consequently, there is a clear demand for efficient, robust, and reliable multiple vehicle tracking solutions.

Vehicle tracking can be approached in different manners. The most common approach uses a Global Navigation Satellite System [2], [3]. With recent advancements in image object detection and improved computational power, computer vision-based approaches have gained traction. These methods also have various setups. For example, using a single camera on a vehicle's dashboard to track the vehicles in-front [4]. Additionally, a traffic cameras which have a view of the road is another highly used method [1]. The latter approach is studied in this paper as it provides a wide field-of-view and there already exists an abundance of traffic cameras available around the world.

Multiple vehicle tracking can be generalized to the problem of multiple object tracking (MOT) within computer vision. MOT can be categorized into motion-feature-based tracking (MFB-MOT) and detection-based tracking (DB-MOT). MFB-MOT uses motion features of neighbouring pixels to track an objects relative motion. A common technique in this category is the use of optical flow [5]. DB-MOT, on the other hand, finds features which can be associated between frames and follows their trajectory.

To achieve DB-MOT, objects must be detected, then its future position estimated, and associating the saved states with previous detections. This general framework, known as Simple Online and Realtime Tracking (SORT) [6], is highly used within the field of DB-MOT for its high efficiency and simplicity. The estimated positions along with current observations need to be combined to track the object. This requires the use of a filter, such as a Kalman filter [7]. As a result of using a filter like the Kalman filter, a motion model of the object must be derived. This motion model can be used to tackle the challenges with object tracking, which include data association between frames and occlusion [8]. Occlusions occur when the object being tracked is not detected, which can be caused from the object being hidden or not being properly detected by the detection algorithm. These challenges are solved in previous works, like [6] and [8], however, they assign new identification tags if the object is still in the frame after being occluded. This is because of the choice of the motion model used for the position estimated.

The main contributions of this work are:

1) Implementing and analyzing several motion models to overcome long durations of occlusion
2) Increasing time between observations to rely more on model
3) Reducing overall computations

The rest of the paper is organized as follows: Section II introduces some preliminary work; Section III introduces the proposed method; Section IV shows numerous tests and results of the proposed method; and Section V concludes this work and presents future direction.

II. Related Work

Computer vision is a burgeoning research field with many recent relevant works. One field within computer vision which has gained a lot of traction in the past decade recent years includes object detection. Previously, it was common to extract specific features directly from images and use these features to do feature matching or train a simple machine learning algorithm. In 2005, the is exactly what happened when the authors in [9] created employed Histogram of Oriented Gradient (HOG) features with a support vector machine for classification for face detection. Another popular feature extractor

include the Scale Invariant Feature Transform (SIFT), which was created a year before HOG features were introduced, in 2004, by DG Lowe in his pivotal paper [10]. The introduction of these features allowed the research community to employ them to further develop applications which employed these features. For example, Piccinini et al. created a real-time object detection approach for finding duplicate objects in pick-and-place applications, where the underlying approach extracting SIFT keypoints [11].

With further advancements processing power and more research towards deep learning, convolutional neural networks (CNN) made its way to the front of academia for object detection. The first major breakthrough was in 2014 with Region with CNN (RCNN). In Girshick et al. introduction paper on RCNN [12], they introduced object candidate boxes, which are further rescaled to a fix size image and fed into a pre-trained CNN model to extract features and SVM is then used for classification. The next big breakthrough was with achieved when Redmon et al. presented their work on the You Only Look Once CNN (YOLO) [13]. Their work was important as it was a one-pass CNN, meaning it only required the image to be passed through the detector once, which increased computation times to real-time detections [13]. As well, their model architecture provided effective feature extractions which provided accurate detections.

Moving to videos, it is important to follow how objects move from frame to frame. A lot of work has also gone into object tracking. Presently, the most common multiple object tracking (MOT) algorithm is the SORT algorithm [6]. This algorithm begins with object detection through an RCNN, then uses a Kalman filter to propagate the state to the next frame, then it uses the Intersection over Union for data association. The SORT algorithm is more of a framework, as it is generally applied to different fields. For example, the SORT algorithm has been applied to multiple vehicle tracking as in [8], where traffic behaviour is monitored using the SORT algorithm and make an extension of using the YOLO detection algorithm. However, in neither [6] nor [8] do they keep the same unique identification for the same object. These authors are focused on keeping the ensuring the tracking remains between frames, but not over duration of a long period. This is an area which is explored in the paper by varying the motion models to get an more accurate motion of the vehicle over a longer duration.

III. Method

The proposed method strives to achieve online multi-vehicle tracking with using a variation of the SORT algorithm in [6]. The method can be split up into several stages which is shown in Algorithm 1. First, vehicles needs to be detected and extracted from a video frame. Following, the respective vehicles' motion needs to be estimated and projected into the following frame. Using the estimated motion of the vehicles (predicted state) and observations (detected vehicles) in the next frame, the two need to be associated with each other. Lastly, logic (in the form of a state machine) is required when vehicles enter or leave the scene completely.

Algorithm 1 Multiple Vehicle Tracking

1. Vehicle Detection
2. Vehcile State Estimation
3. Data Association Between Estimated State and New Frame
4. Check for New Vehicles or Vehicles Left Frame

A. Object Detection

The first and fundamental step is to detect vehicles from an image. When a vehicle is detected, the location of the vehicle within the image needs to be returned. In this case, we use a bounding box to describe where the vehicle is located.

The algorithm chosen for object detection is the You Only Look Once Version 3 (YOLOv3) algorithm with pre-trained weights, as first introduced in [14]. YOLOv3 is a state-of-the-art real-time object detection algorithm based on a convolutional neural network (CNN). This model is chosen for several reasons. First, YOLOv3 has quick computation speeds [14]. It achieves fast computations since an entire image is processed in a single forward pass of the neural network (known as a one stage detector). Its detection speeds, for example, are eight times faster than the Faster Region-based CNN, which is still widely used, and was used in the original SORT paper [12]. Next, YOLOv3 has demonstrated high accuracy in detections [14]. Additionally, the YOLOv3 algorithm outputs its predicted class with a bounding box, which is useful for this application. In this application, we only consider the vehicle output class.

The third version of YOLO is superficially chosen as its capabilities of multi-scale detection (objects of different size) were improved significantly [15]. This is crucial for this application as vehicles entering or leaving the scene could vary drastically in size.

B. Vehicle State Estimation

After a vehicle is detected, it has to be propagated to the next frame. To achieve this, a motion model of the vehicle's change in position from one frame to the next is required. Before introducing the motion model, a state vector describing the vehicles location within the image must first be defined. The state vector of a detected vehicle is modelled as

$$\mathbf{x} = [u, v, w, h, \dot{u}, \dot{v}]^T, \tag{1}$$

where u and v represent the horizontal and vertical top left corner pixel of the bounding box, respectively. Additionally, w and h are the width and height from the top left corner pixel. Lastly, $\dot{u}$ and $\dot{v}$ represent the horizontal and vertical change in the top left pixel location between frames, respectively. These quantities are sufficient in describing the current state/location of the vehicle in the frame. The velocities are required for the motion model, which is presented below.

A common and easy-to-handle motion model to use is one that is linear. The general setup is shown in (2).

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + \mathbf{w} \tag{2}$$

The linear motion model shown in (2) describes that the change in state $\dot{\mathbf{x}}$ is a linear combination of its current state $\mathbf{x}$ and any inputs $\mathbf{u}$, along with any noise $\mathbf{w}$ which is assumed to be Gaussian. In this application, we will consider $\mathbf{u} = \mathbf{0}$ and the change in state is only dependent on the current state, since there are no known obvious modelled inputs. The general setup for the A matrix is

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

as motivated in [16], where constant motion is approximated since the change in frames represents a small duration of time. However, this assumption only works well for small frames. In our method, we would like to propagate models forward by several frames, which requires an adjustment to (3).

1) Motion Model 1: The first proposed motion model that takes the difference in (x, y) pixel coordinates between two subsequent observations/detections and uses that as the motion to propagate the vehicle forward. Since the observations are not necessarily captured between consecutive frames, the difference in pixel coordinates is divided by the number of frames that have passed by.

2) Motion Model 2: The second proposed motion model extends off the previous model but instead of only using two consecutive frames to calculate the difference, the last n observations (where n is a positive constant) are used to get a longer average of the recent velocity of the vehicle's bounding box.

3) Motion Model 3: The final proposed solution is inspired by many highways having long stretches of a constant direction (no need to turn much). This suggests that we can have a motion model which fixes the initial velocity heading. This is what we propose to implement. For this motion model, we take the first n observations (where n is a positive constant) and fix the heading direction of the bounding box. Next, we only change the magnitude of the velocity by taking the difference between the last two observations. By using the assumption that many highways are straight (for the small field-of-view a camera has) then this will not cause major deviations unless the initial readings are noisy very noisy.

Next, an observation model can also be formulated as

$$\mathbf{z} = C\mathbf{x} + \mathbf{v}, \tag{4}$$

where $\mathbf{z}$ are the observable states, C is the mapping between all states and observable states, and $\mathbf{v}$ is observation noise, which is also modelled as Gaussian. Noise is present in the observations since the detections are not perfect. The observations received are the detections from the YOLOv3 model, which only provide information on the current frame, that is, no information about derivatives/motion. Therefore, the values used for C are

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{5}$$

When a modelled vehicle obtains a (noisy) observation in a subsequent frame, it can be linearly combined with the motion model to improve the overall estimate of the true state of the vehicle. The optimal estimator for this problem is the Kalman filter [7]. The Kalman filter creates a weighted gain based on the covariances of the motion model and observation model. Further defining the motion model and observation model uncertainties as

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0},\, R) \tag{6}$$

and

$$\mathbf{v} \sim \mathcal{N}(\mathbf{0},\, Q), \tag{7}$$

respectively. R and Q represent the covariances of the motion and observation model, respectively. R is chosen as the identity, $I_{6\times 6}$, since there is a general uncertainty in our motion model. As well, Q is selected as

$$Q = (1 - c)I_{4\times 4}, \tag{8}$$

where c is the confidence in the vehicle detection. As a result, when the network is confident in its prediction, Q contains small components, and larger components when the network is less confident.

Using the Kalman filter, the vehicle's state (modelled as Gaussian) can be predicted and updated optimally based on

$$\bar{\mathbf{x}} = A\mathbf{x}, \tag{9}$$

$$\bar{\Sigma} = A\Sigma A^T + R, \tag{10}$$

and

$$\mathbf{x} = \bar{\mathbf{x}} + K(\mathbf{z} - C\bar{\mathbf{x}})\bar{\Sigma}, \tag{11}$$

$$\Sigma = (I - KC)\bar{\Sigma}, \tag{12}$$

respectively. In the above equations, $\bar{\mathbf{x}}$ $\bar{\Sigma}$ and the estimated mean and covariance of the vehicles, respectively, and $\mathbf{x}$ and Σ are the updated mean and covariance once an observation has been included. The Kalman gain, K, is defined as

$$K = \bar{\Sigma}C^T(C\bar{\Sigma}C^T - Q)^{-1}. \tag{13}$$

Additionally, a detection of a vehicle within the frame begins a Kalman filter for that respective vehicle. The initial state covariance is set to $\bar{\Sigma}_0 = Q$ since its state is coming directly from the observation. Additionally, the bounding box from the detection is used as the initial state $\mathbf{x_0}$ with the velocity components, $\dot{u}$ and $\dot{v}$, zero.

Using the Kalman filter ensures that the optimal estimate of the vehicle is obtained [7].

C. Data Association

In the previous section, it was assumed that the detections are assigned to its appropriate vehicle model. The challenge here is to propagate the vehicle states to frame $f+1$ from f (where f denotes an arbitrary fram number), using the motion model, and matching them with the new detections in frame $f+1$. In essence, this is the core part of tracking, which is being able to associate objects between frames.

To solve this problem, we first have to define a metric to evaluate how much overlap two bounding boxes have. A common approach to quantify errors in image segmentation (how far off segmented area is from ground truth) problems is the Intersection over Union (IoU) metric as seen in [17] and [18]. For two bounding boxes A, $B \subseteq \mathbb{S} \in \mathbb{R}^2$, their IoU score is defined as

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{14}$$

When $IoU(A,B) = 1$ then there is a perfect overlap. When $IoU(A,B) = 0$, this can occur because there is no overlap between the bounding boxes. Also, if one bounding box is substantially larger than the other then their union will increase substantially, which results in $IoU(A,B) \to 0$. As a result of these properties, IoU is used as the similarity metric for the bounding box associations.

Next, each propagated forward bounding box in frame $f+1$ has an IoU score with each detection in frame $f+1$, but we are unsure of the matches. This is solved by maximizing the score or minimizing the cost $(1 - IoU)$ by choosing appropriate pairs. This type of problem is known as the linear sum assignment problem and is solved using the Hungarian Algorithm in $O(n^k)$, where k is a positive constant and depends on the size of the problem [19]. In two dimensions, the assignment can be solved by the minimization problem

$$\mathbf{X}^* = \arg\min_{\mathbf{x}} \quad \sum_{i=1}^{N_R}\sum_{j=1}^{N_C} l_{i,j}x_{i,j} \tag{15a}$$

$$\text{s.t.} \quad \sum_{j=1}^{N_C} x_{i,j} = 1 \ \ \forall i \tag{15b}$$

$$\sum_{i=1}^{N_R} x_{i,j} \leq 1 \ \ \forall j \tag{15c}$$

$$x_{i,j} \in \{0,1\} \ \ \forall x_{i,j} \tag{15d}$$

which was presented in [20]. In (15), the cost matrix L is of size $N_R \times N_C$, where the bounding boxes of the propagated states change along the columns and the detected vehicles change along the rows. Additionally, constraint (15a) specifies that every row is assigned to a column of L and (15b) specifies that not every column is assigned to a row of L. An example of how the cost matrix, L, is created is shown in (16), where $IoU(p1, d1)$ refers to the IoU cost of propagated state 1 with vehicle detection 1. Note, that the indices in the rows and columns do not mean that $p1 = d1$!

$$L = \begin{bmatrix} 1 - IoU(p1,d1) & 1 - IoU(p1,d2) & \dots \\ 1 - IoU(p2,d1) & 1 - IoU(p2,d2) & \dots \\ 1 - IoU(p3,d1) & & \\ \vdots & \ddots & \\ & & \end{bmatrix} \tag{16}$$

D. Filter Logic (Creation and Deletion)

This section deals with the logic of the track identities (a filter tracking a possible vehicle). It is the part that decides when a tracker should be added or discarded. A track identity needs to be added when a vehicle enters the frame. Similarly, when a vehicle exits the frame then the associated tracker should be discarded. Beginning with adding new track identity, when there are more detections than identities, then it is known that another track identity needs to be added. Deleting is slightly more difficult because of the possibility of occlusions. If there are more filters than detections, than the vehicle may be occluded or it could have left the scene entirely. To account for this, the track identity that does not have an associated detection (after the assignment step) checks to see if it still lies within the boundaries of the frame, if not, then the vehicles has left the frame/scene, so the tracker should be deleted. However, if the track identity is still in bounds, than a counter called time-to-live (TTL), is decremented and propagated to the next frame. If no detection is found and the track identity is still in-bounds, then the TTL counter is decremented again. If the counter reaches zero then the tracker is discarded, since it is highly uncertain given that it has not seen an observation for many frames. However, if the tracker associates with a detection before the TTL counter reaches zero then the TTL counter is reset. Employing a TTL counter accounts for occlusions while acknowledging that the tracker becomes more uncertain as more frames pass, since a detection would bring down the uncertainty (as opposed to accumulating uncertainty by only propagating the state through the motion model).

IV. Experimental Results

In this section we show results from numerous experiments that test and validate our proposed method. All of the test were conducted on a personal computer with an 11th Gen Intel® Core™ i7-11700 @ 2.50GHz × 16 processor and 32 GB of RAM.

Three separate tests were conducted to analyze different features and metrics. Different data were used for each test so avoid overfitting to a specific scenario. The data comes from two different datasets. One dataset was found online with ground truth positions of the vehicle available, curated by the Rawashdeh Research Group [21]. The second dataset was self-curated and available upon request.

A. Multiple Vehicle Tracking Absolute Error

The first test used the dataset from the Rawashdeh Research Group [21]. A video stream which has vehicles travelling in

different directions are tracked uniquely while they are in the camera frame. Specifically, there are three cars within the video which are moving along the road. Additionally, there are numerous stationary vehicles in the background which are also tracked but remain fixed. The ground truth data points are available for the moving vehicles, so it was possible to directly get the error with our tracker. Figures 1 - 5 cover the test related to absolute error.

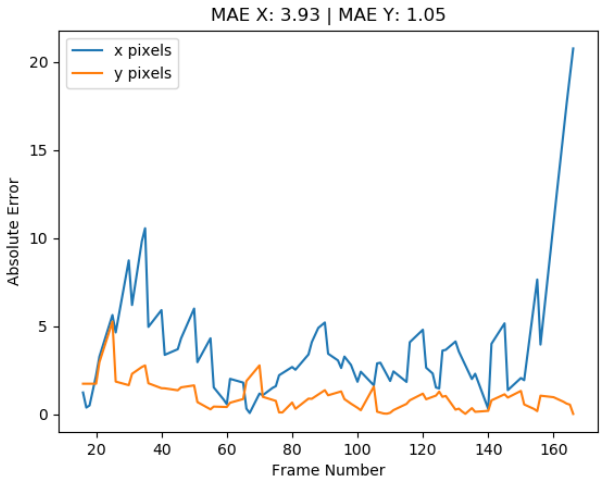

Fig. 1. Absolute Error and Mean Absolute Error (MAE) for Car 1 (ID 9). Using Motion Model 1

The absolute error for car 1 (ID 9) is shown in Figure 1. It can be seen that the tracker does a good job at staying close to the true state of the car, since the absolute error (and mean absolute error) for both the x- and y-components remain low. There is more error (about three times as much) in the x-direction as opposed to the y-direction because most of the motion of the vehicle is in the x-direction. Additionally, there is a larger absolute error in the x-direction near the ending frames as the vehicle leaves the scene but the tracker still models the car inside the frame. Similar conclusions can be drawn about the graphs in Figure 2 and 3. The major difference is that they are uniquely following a different car but exhibit similar behaviour.

Figures 4 and 5 visually confirm that multiple vehicles are being tracked throughout the video sequence. Additionally, the multiple moving objects can be tracked at the same time, which is seen explicitly in Figure 4. Further, analyzing the frame numbers in Figures 1 - 3, it can be seen that there is an overlap in the time that the vehicles are being tracking (specifically Car 1 and Car 2).

This experiment also allows us to directly compare the performance of the three different motion models. The second and third motion model errors are shown in Tables I and II, respectively. The first motion model error is shown in Figures 1 - 3. It can be seen that the third motion model has the lowest overall error. This is because the third motion model saves a unit vector for the heading after five measurements at the beginning, which was favourable in this situation since

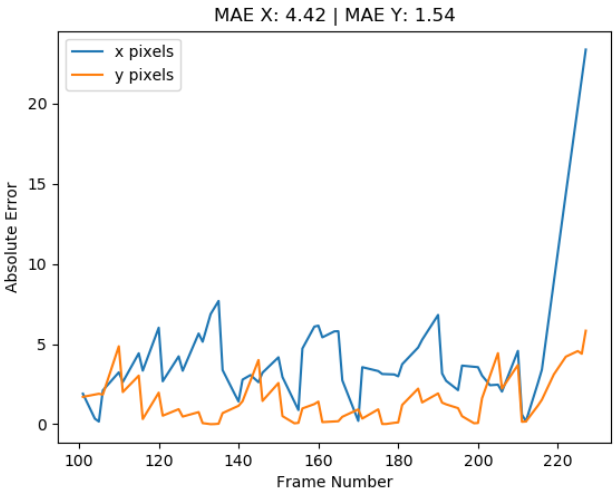

Fig. 2. Absolute Error and Mean Absolute Error (MAE) for Car 2 (ID 14). Using Motion Model 1

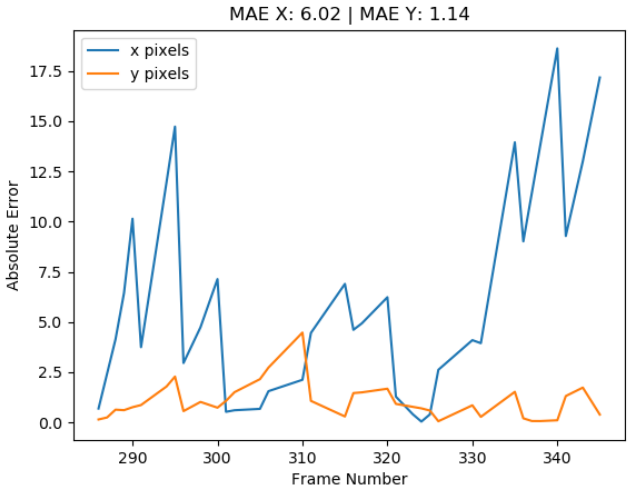

Fig. 3. Absolute Error and Mean Absolute Error (MAE) for Car 3 (ID 15). Using Motion Model 1

the vehicles are following a relatively straight path. The first motion model performs the second best, since it takes consecutive frames to calculate the velocity, which is a reliable since these vehicles were not changing sporadically. Lastly, the second motion model performed the poorest as this filter is too complicated for this scenario. Since the updated velocity is scaled depending on the change in area between consecutive bounding box frames, it would be a better fit for vehicles travelling towards and away from the camera.

TABLE I
MEAN ABSOLUTE ERROR (MAE) USING SECOND MOTION MODEL

	Car 1	**Car 2**	**Car 3**
MAE X	4.70	5.81	6.54
MAE Y	1.57	1.23	0.81

Fig. 4. Test Number One with Car 1 (ID 9) and Car 2 (ID 14).

Fig. 5. Test Number One with Car 3 (ID 15).

TABLE II
MEAN ABSOLUTE ERROR (MAE) USING THIRD MOTION MODEL

	Car 1	Car 2	Car 3
MAE X	3.69	3.83	6.02
MAE Y	0.95	0.81	1.24

B. Runtime and Error With Varied Observation Intervals

The second test focused on the effects of varying the observation intervals. More specifically, we were interested in seeing the runtime and overall error would change when we increased the observation intervals. When the observation intervals are increased, this essentially mimics an occlusion for the same duration of frames. In this test, we varied the observation interval from 2-35 frames/observation. This dataset was also obtained from from the Rawashdeh Research Group [21], but this test used a different video segment. In this video segment, there is a single car which drives quickly across the frame in the horizontal direction.

Figures 6 and 7 show the output of the MAE and runtime with respect to the observation intervals, respectively. The MAE tends to remain low for observation rates below 20 frames/observation. However, after 20, the MAE exponentially grows. This is because the Kalman filter is not receiving any

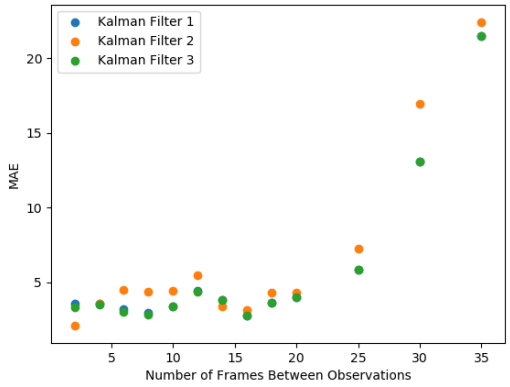

Fig. 6. Mean Squared Error vs. Observation Intervals

observations so the model/tracker is accumulating a lot of uncertainty, which eventually leads to faulty tracking. It can be seen in Figure 6 that the first and third motion model (Kalman Filter in Figure 6 legend) overlap almost perfectly, and the second motion model performs worse.

Next, the runtime follows an exponential decay as the observation interval increases, as seen in Figure 7. This is reasonable because the longest part of execution is the inference of the YOLOv3 detection model. Therefore, as the observation interval increases, the YOLOv3 model is run fewer times. It should be noted, as well, a CNN of this size would perform substantially quicker with access to GPUs for the parallel processing. Since we would like to achieve as close to real-time object tracking, we would have to select an observation interval that is greater than 10 frames/observations. However, we cannot choose the quickest execution time as it will increase the MAE. Therefore, a middle point should be used where the runtime and MAE are both low.

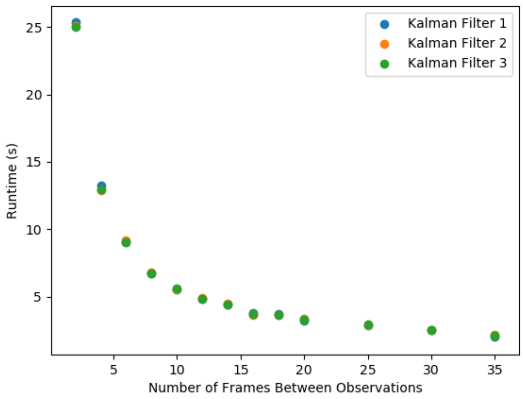

Fig. 7. Runtime vs. Observation Intervals

C. Long Occlusions

Long duration occlusions was another area that was tested. When a vehicle is occluded for several consecutive frames, it puts the motion model to the test to see how accurate it is with

respect to the true vehicle motion. For the test, we collected traffic camera data of a freeway in Florida which had a sign blocking that locked a section of the freeway. As a result, vehicles were occluded for an average of 60 frames, which was about 2 seconds in real-time. Table III shows the performance of the three motion models in two different videos. In this table it can be seen that the third motion model detected 50% of the vehicles after an occlusion of around 60 frames. The results of the third motion model can be seen in Figure 8 and 9, which has two subsequent successful tracks after being occluded. This demonstrates that we have created a model that can track multiple vehicles with substantial occlusions.

TABLE III
NUMBER AND PERCENTAGE OF CARS TRACKED AFTER OCCLUSION

	Video 1		Video 2		
Motion Model	Total Tracked	Tracked (%)	Total Tracked	Tracked (%)	Average (%)
1	2	20	0	0	10
2	6	60	0	0	30
3	4	40	3	60	50

Fig. 8. Properly Tracked Purple Vehicle Through Occlusion.

V. CONCLUSION AND FUTURE WORK

In this paper we extend off of the SORT algorithm by using a YOLO detection algorithm, different motion models, and increase computation speeds by reducing the number of observations seen while tracking. The different motion models (specifically the velocity) were presented, namely, an immediate difference between consecutive frames, an average of the previous n frames (where n is a constant), and an average of the first n frames to obtain a heading vector. The third motion model was found to perform well when there were consecutive frames that occluded the vehicle. Additionally, we showed comparisons between the different motion models and saw that their mean average error remains low as long as the observation interval does not increase too high. As well,

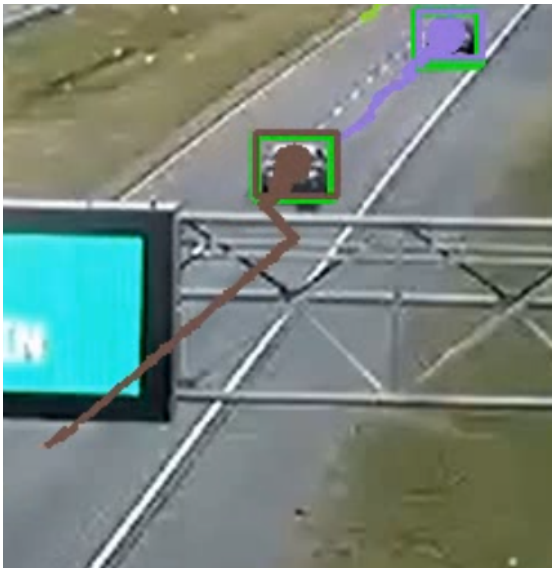

Fig. 9. Subsequent Successful Tracking of Brown Vehicle.

the reason to increase the observation interval is to increase computation speeds. In future work, more complex motion models could be studied. For example, a motion model which learns from its previous trajectory.

REFERENCES

[1] N. Algiriyage, R. Prasanna, E. Doyle, *et al.*, "Towards real-time traffic flow estimation using yolo and sort from surveillance video footage," in *Proceedings of the 18th International Conference on Information Systems for Crisis Response and Management, Blacksburg, VA, USA*, 2021, pp. 23–26.

[2] S. Lee, G. Tewolde, and J. Kwon, "Design and implementation of vehicle tracking system using gps/gsm/gprs technology and smartphone application," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 2014, pp. 353–358. DOI: 10.1109/WF-IoT.2014.6803187.

[3] K. Maurya, M. Singh, and N. Jain, "Real time vehicle tracking system using gsm and gps technology-an anti-theft tracking system," *International Journal of Electronics and Computer Science Engineering*, vol. 1, no. 3, pp. 1103–1107, 2012.

[4] S. S. Teoh and T. Bräunl, "A reliability point and kalman filter-based vehicle tracking technique," in *International Conference on Intelligent Systems*, 2012, pp. 134–138.

[5] S. Denman, V. Chandran, and S. Sridharan, "An adaptive optical flow technique for person tracking systems," *Pattern recognition letters*, vol. 28, no. 10, pp. 1232–1239, 2007.

[6] 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)*, IEEE, Sep. 2016. DOI: 10.1109/icip.2016.7533003.

[Online]. Available: http://dx.doi.org/10.1109/ICIP.2016.7533003.

[7] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[8] J. Jiao and H. Wang, "Traffic behavior recognition from traffic videos under occlusion condition: A kalman filter approach," *Transportation research record*, vol. 2676, no. 7, pp. 55–65, 2022.

[9] 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)*, Ieee, vol. 1, 2005, pp. 886–893.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[11] P. Piccinini, A. Prati, and R. Cucchiara, "Real-time object detection and localization with sift-based clustering," *Image and Vision Computing*, vol. 30, no. 8, pp. 573–587, 2012.

[12] L. Tan, T. Huangfu, L. Wu, and W. Chen, *Comparison of yolo v3, faster r-cnn, and ssd for real-time pill identification*, Jun. 2021. DOI: 10.21203/rs.3.rs-668895/v1.

[13] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[14] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: 1804.02767 [cs.CV].

[15] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.

[16] R. Zhang, P. Ge, X. Zhou, T. Jiang, and R. Wang, "An method for vehicle-flow detection and tracking in real-time based on gaussian mixture distribution," *Advances in Mechanical Engineering*, vol. 5, p. 861 321, 2013.

[17] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *International symposium on visual computing*, Springer, 2016, pp. 234–244.

[18] F. van Beers, A. Lindström, E. Okafor, and M. Wiering, "Deep neural networks with intersection over union loss for binary image segmentation," in *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, SciTePress, 2019, pp. 438–445.

[19] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955. DOI: 10.1002/nav.3800020109.

[20] D. F. Crouse, "On implementing 2d rectangular assignment algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1679–1696, 2016. DOI: 10.1109/TAES.2016.140952.

[21] Z. El Shair and S. A. Rawashdeh, "High-temporal-resolution object detection and tracking using images and events," *Journal of Imaging*, vol. 8, no. 8, p. 210, 2022, ISSN: 2313-433X. DOI: 10.3390/jimaging8080210. [Online]. Available: https://www.mdpi.com/2313-433X/8/8/210.