COSMOS Smart Intersection: Edge Compute and Communications for Bird's Eye Object Tracking

Shiyun Yang†, Emily Bailey‡, Zhengye Yang†, Jonatan Ostrometzky†, Gil Zussman†, Ivan Seskar⋆, Zoran Kostic†
† Dept. of Electrical Engineering, Columbia University, New York City
‡ Dept. of Computer Science, Columbia University, New York City
⋆ Winlab, Rutgers University, New Jersey

***Abstract*—Smart-city intersections will play a crucial role in automated traffic management and improvement in pedestrian safety in cities of the future. They will (i) aggregate data from in-vehicle and infrastructure sensors; (ii) process the data by taking advantage of low-latency high-bandwidth communications, edge-cloud computing, and AI-based detection and tracking of objects; and (iii) provide intelligent feedback and input to control systems. The Cloud Enhanced Open Software Defined Mobile Wireless Testbed for City-Scale Deployment (COSMOS) enables research on technologies supporting smart cities. In this paper, we provide results of experiments using bird's eye cameras to detect and track vehicles and pedestrians from the COSMOS pilot site. We assess the capabilities for real-time computation, and detection and tracking accuracy - by evaluating and customizing a selection of video pre-processing and deep-learning algorithms. Distinct issues that are associated with the difference in scale for bird's eye view of pedestrians vs. cars are explored and addressed: the best multiple-object tracking accuracies (MOTA) for cars are around 73.2, and around 2.8 for pedestrians. The real-time goal of 30 frames-per-second - i.e., a total of 33.3 ms of latency for object detection for vehicles will be reachable once the processing time is improved roughly by a factor of three.**

***Index Terms*—smart-city, smart-intersection, deep learning, detection, tracking, testbed, wireless.**

I. Introduction

Smart-city intersections are the key infrastructural nodes of the emerging technologically-enabled smart cities. A collection of intersections, connecting roads and corresponding vehicle and pedestrian traffic fully describe the real-time dynamics of a smart-city [1]. A traffic intersection is an ideal geographical location for positioning the communications equipment and edge computing nodes used for collection and processing of data, and for interaction with traffic participants. Emerging applications such as autonomous and cloud-connected vehicles, V2I, V2X [2] and multi-sensor/multi-media assistance to handicapped pedestrians, will require significant compute power including machine learning/AI capabilities, very high communication bandwidths, and ultra-low latencies.

Dense urban environments, such as New York City (NYC), present unique challenges to the deployment of the advanced services such as cloud-connected vehicles [3]. This is due to the large number of vehicles moving at various speeds in many directions, obstructions which are opaque to in-vehicle sensors, and erratic behavior of pedestrians. Reliable real-world automation of monitoring, measurement, learning, feedback, and participation in control will require significant further advancements in a number of technologies.

This paper presents the results of experiments with a subset of technologies which will constitute a smart intersection: bird's eye videos, edge computing, and contemporary deep-learning-based detection and tracking of objects in an intersection. The experiments have been performed on the pilot site of the COSMOS testbed [4]–[6], deployed in New York City.

II. Vision of a Smart City Intersection

Smart city intersection will be at the core of an AI-powered traffic management system for future metropolises. The key goals of a smart-intersection are pedestrian protection and real-time collaborative control of cloud-connected vehicles. Real-world experiments are essential to making progress on the concept of smart city intersections, and this paper considers one smart intersection in NYC. Our studies are based on the COSMOS testbed [4] pilot site at Columbia University (Amsterdam Ave. and 120th Street). The paper shows the results of the experiments with videos obtained by bird's-eye cameras located high above the intersection, and deep-learning-based detection and tracking of vehicles and pedestrians. The experiments are performed to asses and provide direction on how to improve the state of art with an eye of achieving two performance targets. One target is to achieve latency around 33 ms for a loop which consists of: (i) sensor data acquisition; (ii) communications between end-users, peripheral, edge cloud and cloud equipment; (iii) real-time AI/ML inference computation; and (iv) feedback towards the end-users. The second target is to achieve accuracies of high 90% for AI-based detection and tracking algorithms.

III. COSMOS testbed as A Platform For Smart Intersection Experimentation

COSMOS testbed [4]–[6] provides technological components for research on smart intersections. It will support the use case of cloud-connected vehicles and handicapped pedestrian assistance. Vehicles will wirelessly share in-vehicle sensor data with other vehicles and with the edge cloud servers. COSMOS will additionally deploy a variety of infrastructure sensors, including street-level and

bird's eye cameras [7], whose data will be aggregated by the edge-cloud servers [8]. The servers will run real-time algorithms to monitor and manage traffic after detecting and tracking the objects in the intersection, and share the processed data in real-time with participants in traffic and with traffic management and control. COSMOS's pilot location is equipped with four cameras (two are bird's eye cameras), a GPU edge computing node, and can support low latency wireless broadcast (see Fig. 1).

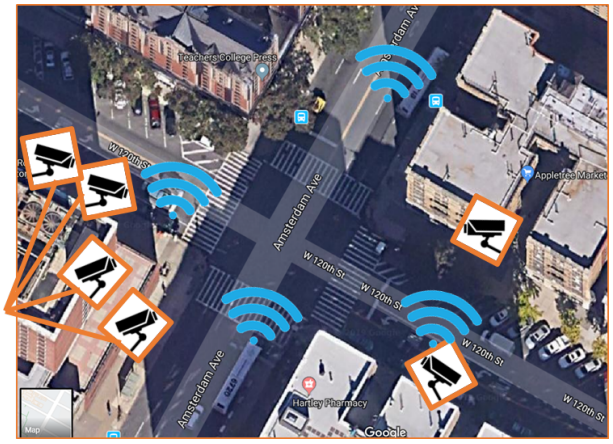

Fig. 1. COSMOS pilot site with cameras and edge-cloud node

COSMOS testbed [6] contains an optical x-haul transport system connecting powerful edge computing clusters for baseband processing with massively scalable CPU/GPU computing resources and FPGA assist, and software defined radios. It provides four technology layers for experimentation: user device layer, radio hardware and front-haul network resources, radio cloud, and general purpose cloud.

IV. Target Use Case: Seeing Around the Corner - Bird's Eye Detection and Tracking of Objects

One of the goals of the smart-intersection research is the ability to support cloud-connected vehicles. That implies multi-modal sensor integration combining data from in-vehicle and infrastructure sensors. On the way to that goal, we have defined an intermediate target which uses infrastructure sensors only, such as cameras, to process videos, detect and track objects, and to construct and broadcast a "radar screen" to all participants in the intersection, similar to the concept described in [9]. The "radar screen" is a real-time-evolving "movie" of positions and velocity vectors of all objects in the intersection. An example snapshot constructed using the first implementation of deep-learning-based tracking is illustrated in Fig. 2, where a car is marked by blue, a pedestrian in red, and a bicycle in teal color. The radar screen is constructed in the edge cloud server using learning algorithms that will ultimately be dynamically distributed across various computing resources based on the application latency requirements and available bandwidth. The radar screen will be wirelessly broadcast to participants in the intersection within the prescribed latency target.

Bird's eye infrastructural cameras are some of the key sensors in dense urban environments such as New York City, because they provide contextual awareness of the whole intersection, and are less problematic from the perspective of protecting privacy/anonymity: Pedestrian profiles are exceptionally small and vehicle license plates are not visible.

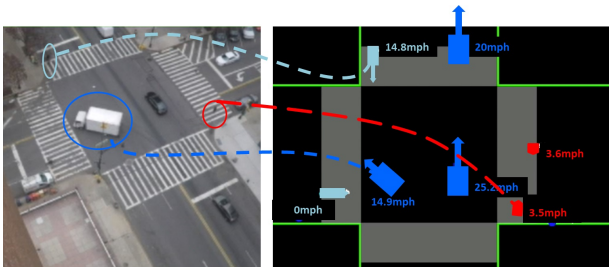

Fig. 2. The "radar screen": one frame of a video containing locations and speed of objects within an intersection.

A. Data Acquisition

Video is the most frequently used data source for detection and tracking of moving objects [7]. Because our application is aimed at object tracking in crowded city intersections and streets from videos taken with bird's eye view cameras, a unique goal when compared to existing literature, there is no applicable public dataset available for training and testing our systems. We therefore created our own dataset. For results presented in this paper, we use videos acquired by COSMOS cameras Hikvision (DS-2CD5585G0-IZHS), and some videos acquired by GOPRO (Hero-6). The cameras were located on the 12th floor of Seeley W. Mudd Building at Columbia University in New York, overlooking the intersection of W 120th St. and Amsterdam Ave. All videos are recorded at 30 frames per second with a resolution of 1920×1080 pixels. Videos were taken across many days in the morning, noon, dusk, and at night.

B. Data Processing

Because different cameras, or cameras installed at different locations, are likely to have different zooms, and objects from different views may vary in scale, it is hard to have comparable results for the detection and tracking algorithms. To remedy this problem, we apply video calibration to the raw videos, which warp the traffic intersection into a predefined rectangle with a uniform scale. We also black out the regions which are related to the buildings and sidewalks to remove irrelevance; we leave the four corners of the intersection where the entrances for pedestrians to enter the crosswalks are located.

We inspected more than ten hours of recorded videos to find video clips that contain busy traffic scenes and different types of vehicles and pedestrians. We then divided the data into one testing dataset and multiple training datasets. The testing dataset contains 10 video clips of 90 seconds each, and is used for benchmarking the model performances. We chose 90-second clip length because this encapsulates one full

Fig. 3. Example of an annotated frame with calibration and shading

cycle of the traffic light, allowing our data to cover all possible standard trajectories in all directions.

To obtain the correct benchmarking results, the testing datasets have been isolated from the training datasets. The training datasets contain individual frames or images cropped from frames, and are used for training of the object detection models. We manually annotated all the calibrated datasets which contain more than $10,000$ frames in total with bounding boxes and corresponding ids. In Fig. 3, we present an example of annotation with location and identity information of objects. In Table I we present an overview of this new COSMOS dataset and compare it to widely-used MOT benchmarking datasets. This test dataset is larger than all but two others, in terms of frames, tracks, and boxes. It is diverse in classes represented and in lighting/weather conditions.

TABLE I
COMPARISON: COSMOS DATASET VS WIDELY-USED MOT BENCHMARKS

	Testing			Properties	
Dataset	**Frames**	**Tracks**	**Boxes**	**Classes**	**Year**
COSMOS	9K	0.8K	243.6K	Ped. & Veh.	2020
MOT17	17.8K	2.4K	564.2K	Ped. & Veh.	2017
MOT16	5.9K	0.8K	182.3K	Ped. & Veh.	2016
MOT15	5.8K	0.7K	61K	Ped.	2015
UA-DETRAC	56K	2.3K	632K	Veh.	2015
KITTI-T*	8K	-	-	Veh.	2014
UR*	3.6K	-	-	Ped. & Veh.	2014
PETS2009	1.5K	0.1K	18.5K	Ped.	2009
TUD*	0.5K	0K	2.6K	Ped.	2008

** Indicates dataset with litle lighting and weather conditions.*

C. AI for Detection, Tracking and Feedback

1) Object Detection: Object detection precedes object tracking, and it requires models to automatically generate the location and class information of objects inside the scenes. Our detection models also create a confidence score for each predicted object and class. We consider three state-of-the-art detection models.

The first object detection algorithm that we consider is Mask R-CNN [10] - an object detection and instance segmentation framework built on top of the Faster R-CNN model within the *Region-based Convolutional Neural Network* (R-CNN) family. Faster R-CNN [11] is a two-stage object detector where, in the first stage, object bounding boxes or Region-of-Interests (RoIs) are proposed for the entire image. In the second stage, a small feature map is extracted from each RoI and is used to perform classification and bounding-box regression. *Mask R-CNN* adds an additional "mask branch" to the second stage that predicts pixel-wise instance segmentation masks.

Because two-stage models such as R-CNNs separate the detection task into branches, they sacrifice speed. Much faster single-stage detectors have been proposed [12], [13]. Current state-of-the-art single-stage detectors include *Single Shot Detector* (SSD) [14] and *You Only Look Once v3* (YOLOv3) [15]. These models follow a similar strategy: they extract features from input images and use several feature layers to represent features with different scales, then predict the offset of predefined anchor boxes based on the information from feature maps to output the bounding boxes with non-max suppression. The predefined anchor box sizes are the clusters of all bounding boxes in the training dataset. These models are designed with the goal to achieve real-time object detection while maintaining competitive accuracy.

2) Object Tracking: In our application, not only do we need to identify the presence and know the location of the objects, but we also want to measure the movement and speed of the objects at each time instance. To obtain this information, a *Multiple Object Tracking* (MOT) [16] system is needed so that we could trace the objects' location in time and predict the direction and speed of the movement. We next discuss two off-the-shelf MOT algorithms, and we also develop an optical-flow based tracking algorithm.

DeepSORT [17] is an extension to the *Simple Online and Realtime Tracking* (SORT) algorithm [18], which is based on the combination of two techniques - the Kalman Filter and the Hungarian algorithm. On top of that, to better solve the object association problem, DeepSORT introduced the deep appearance descriptor, which is a *convolutional neural network* (CNN) model to extract features inside of bounding boxes, then uses cosine distance to represent matching score to associate objects between two frames.

Deep Affinity Network (DAN) [19] learns compact, comprehensive features of pre-detected objects at several abstraction levels, and performs exhaustive pairing permutations of features in any 2 frames to infer object affinities. DAN accounts for multiple objects appearing and disappearing between frames. It exploits the resulting efficient affinity computations to associate objects in the current frame deep into previous frames for reliable on-line tracking.

We developed a tracking algorithm called Multicut, which works with optical flow [20]. Optical flow shows the speed of consistently bright movements, which can be used to describe motion. We use optical flow as traces to help to track objects. We adopted a state-of-the-art optical flow estimation model called PWC-Net [21], which uses pyramidal processing, warping, and cost volume to achieve high estimation accuracy without sacrificing inference speed. With the help of optical flow and object detection results, we can predict objects' movements in the next frame. Then we apply the greedy approach to match the same objects across multiple frames. To remedy the impact of missed detections, we can simply follow

the flow vector of missed objects, with the goal to mitigate the identity switching problem. Furthermore, we keep the partial momentum of optical flow to minimize the impact of object occlusions.

V. Experiments

To evaluate and compare how object detection and tracking models perform in our application, we assess three object detection models and three tracking models. The detection models are trained on our dataset. We measure two sets of performance metrics: (i) object detection performance metrics are measured for each detector, and (ii) MOT performance metrics are measured for each detector-tracker pair.

A. Experimental Setup

Since running neural network models requires considerable computing power, we use computing configurations consisting of clusters of 16 vCPUs, 104GB of memory, and one Nvidia P100 GPU, which can be run on Virtual Machines (VMs) on Google Cloud Platform (GCP) for both training and testing the models.The operating system is Ubuntu 16.04.

B. Training

Training on our own dataset is necessary for the object detection models because most state-of-the-art detectors are pre-trained on popular public benchmark datasets such as the COCO dataset [22], which contains close-up pictures of many different types of objects. In our bird's eye view videos, objects appear much smaller, thus the scale is very different; and the object types are limited to two classes, vehicle and pedestrian.

On the other hand, we find that training the tracking models on our own dataset is not necessary because (i) the tracking models are trained to recognize object similarities between a series of frames, which do not depend on the objects themselves, therefore the pre-trained models work fine; and (ii) training the tracking models requires a training dataset which contains a large number of frames (preferably from different cameras) and where each object is labelled with a consistent ID across all frames. Preparing such a dataset requires significant amount of time. An example tracking dataset is the Vehicle Re-Identification (VeRi) dataset which contains over 50,000 images of 776 vehicles captured by 20 cameras.

The training of the detection models is done on the web platform called Supervisely [23]. The training datasets are the cropped images from our bird's eye video recordings. We ensure that the training images do not overlap with the testing videos. The images are uploaded and hand-labeled using Supervisely annotation tools. After connecting a GCP VM to the platform, we can spawn a Supervisely docker container that trains the model on the labelled datasets.

Since Mask-RCNN is an instance segmentation model, and it therefore needs pixel-wise masks for the training. Annotating pixel-wise masks is time-consuming - to remedy this problem, we applied data augmentation to our training dataset including random crops, horizontal flips, and random rotations all the while keeping the objects' original scale.

The other detectors only need bounding boxes to learn the localization of objects. It is always good to train neural networks with more data, and labeling the datasets is expensive. Inspired by [24], and given Mask-RCNN's outstanding performance, we use Mask-RCNN which is trained on our dataset to predict the calibrated dataset without manual annotation. We then utilize the outputs from Mask-RCNN as the pseudo ground truth to train YOLOV3 and SSD.

C. Inference

After the training is completed, we select the highest accuracy model for each detector to use in the inference phase. To measure the object detection performances, we run each detection model on the entire test dataset, and then use the outputs and the ground truth annotations to compute the detection performance metrics. To measure the MOT performances, we use the output of each detector as an input to run each tracker, then use the tracks' outputs to compute the MOT performance metrics. When running each detector and each tracker, we also record the runtime of the models. To ensure a fair comparison, all models are ran on VMs with the exact same specifications.

VI. Results

In choosing the metrics, we paid particular attention to those used by several widely-known challenges: for detection, the Pascal Visual Object Classes (VOC) Challenge [25] and the Common Objects in Context (COCO) Challenge [22]; for tracking, the MOT Challenge [26].

We utilize the metric called Intersection Over Union (IOU), defined as the area of intersection of the predicted and ground truth bounding boxes, divided by the union of the predicted and ground truth bounding boxes,

$$IOU = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A|+|B|-|A \cap B|}.$$

IOU sets the basis of how we evaluate the detection, and thus the tracking models which depend on detection results. We consider an object to be detected if the IOU between a prediction and a ground truth object is greater than a specified threshold value.

A. Detection Accuracies

We also leverage IOU directly as a metric for detection models: mIOU denotes the mean IOU for all detected objects. The second key metric for measuring detection models is Average Precision (AP), which denotes the mean ratio of matches to detections across a set of confidence intervals. We build ten confidence intervals using the confidence values output by the detectors. Both metrics are calculated using the IOU thresholds of 50% and 75%. Note that in the results below we use the abbreviation Veh. for a vehicle, and Ped. for a pedestrian; we use the abbreviation MRCNN for the MaskRCNN model.

The first set of results, in Table II, shows each detector's performance for each of our two classes. We clearly see that all models perform far better at detecting vehicles than

pedestrians, by an average of 34.8 points of AP_{50}. We also see that MaskRCNN handily outperforms both SSD and YOLOv3 for both classes, by an average of 7.6 points for SSD and 8.2 points for YOLOv3.

TABLE II
DETECTION RESULTS FOR THE TEST DATASET

Class	Detector	AP_{50}	$mIOU_{50}$	AP_{75}	$mIOU_{75}$
Veh.	MRCNN	90.7	77.3	73.3	84.4
	SSD	89.7	71.3	58.5	81
	YOLOv3	86.2	69.3	47.7	80.1
Ped.	MRCNN	62.7	65.6	37.3	80.3
	SSD	48.6	59.1	7.8	72
	YOLOv3	50.8	58.2	12.9	71.4

We further analyze the performance of MaskRCNN on our dataset by breaking down the AP_{50} and AP_{75} results for each class by the time of day. In Table III we see that lighting conditions make a far greater impact on pedestrian detection than on vehicle detection, with pedestrian AP_{50} nearly 8.4 points higher in the day when compared to night (versus just a 3.4 point difference for vehicle detection). We hypothesize this difference exists for two key reasons: first, cars at nighttime shine headlights which are very visible; second, not only are pedestrians unlit, but at dusk we see pedestrians casting very long shadows, which closely resemble pedestrians themselves, leading to large numbers of false positives.

TABLE III
DETECTION RESULTS FOR MASKRCNN BROKEN OUT ACROSS LIGHTING CONDITIONS IN THE TEST DATASET

Class	Daytime		Nighttime	
	AP_{50}	AP_{75}	AP_{50}	AP_{75}
Veh.	92	76	88.6	69.4
Ped.	66.1	44.2	57.7	26.9

To put these results in context, we look to MaskRCNN bounding box performance on the COCO dataset. AP_{50} is reported as 62.3 [10], which is lower than our MaskRCNN performance on both vehicle and pedestrian detection. This can be explained by recalling that the COCO dataset covers more than 80 classes, whereas our dataset focuses in on just 2. In addition, images in the COCO dataset come predominantly from ground-level cameras. Because our data is captured using bird's eye cameras, which are notably further from the subjects, features are less sharp and thus differences between objects of the same class are not as likely to be problematic.

B. Tracking Accuracies

MOT results are measured primarily using MOT Precision (MOTP), defined as the total error in position for valid matched object-hypothesis pairs across all frames, and MOT Accuracy (MOTA), defined as the total number of errors (false positives, false negatives, and mismatches) relative to the total number of ground truth object appearances [27]. Note that while the range of possible MOTP values is $[0, 100]$, MOTA has a range of $[-100, 100]$. We also look closely at the percentages of Mostly Tracked (MT) and Mostly Lost (ML) objects, those ground truths which are tracked $\geq 80\%$ and $< 20\%$ of the time, respectively.

In Table IV we can see that once again our models all perform far better across all metrics on the vehicle class than on the pedestrian class. Note that here we use the abbreviations DSORT for DeepSORT and MCUT for Multicut. Another trend that repeats is that any tracker leveraging MaskRCNN outperforms the same tracker leveraging either of the other detection models, as is expected given the results presented in Table II. Specific to the trackers themselves, we can see that DeepSORT and Multicut both outperform DAN on MOTA; however, we see that MOTP, MT, and ML across the three models are all fairly comparable.

TABLE IV
MULTIPLE OBJECT TRACKING RESULTS FOR ALL TRACKER-DETECTOR PAIRS FROM EXPERIMENTS ON THE TEST DATASET

Class	Tracker	Detector	MOTA	MOTP	MT	ML
Veh.	DAN	MRCNN	55.9	76.4	86.7	4.6
	DAN	SSD	31.7	70.4	46.3	20.4
	DAN	YOLOv3	30.3	68.8	53.5	18.4
	DSORT	MRCNN	73.2	76.6	85.1	4.5
	DSORT	SSD	42	71.8	50.1	21.6
	DSORT	YOLOv3	42.6	69.6	51.2	18.9
	MCUT	MRCNN	72.7	77.5	87.9	4.4
	MCUT	SSD	41.8	71.5	54.1	20.2
	MCUT	YOLOv3	40.8	69.5	59.4	17.5
Ped.	DAN	MRCNN	-21.5	64.4	13.4	28
	DAN	SSD	-50.2	59.8	0	81.5
	DAN	YOLOv3	-66	58.8	0	89
	DSORT	MRCNN	2.8	66.4	19	24.5
	DSORT	SSD	-30.4	60.1	0	79.7
	DSORT	YOLOv3	-47.4	58.8	0	88.3
	MCUT	MRCNN	-4.3	66	16	24.2
	MCUT	SSD	-45.6	60	0	78.8
	MCUT	YOLOv3	-62.1	58.8	0	87.1

In the most recent MOT Challenge [28], we see MOTA values averaged across all classes ranging from -7.3 to 67.2. This range is similarly broad to what we see in our own results, but as they do not report class-level metrics, it is difficult to compare to our work. Also, while the datasets used for this challenge are more similar to ours than that of the COCO challenge, being that they are made up of streetviews containing vehicles and pedestrians, the cameras are still much closer to the objects than our bird's eye cameras.

C. Timing Profiles

Because our ultimate goal is to run real-time tracking and detection, we consider the speed of algorithmic execution using Frames per Second (FPS). We define real-time as a minimum of 30 FPS. This number is convenient because it is equal to FPS of many video recording systems. In the context of the smart-city intersection, this number maps into 1 sec / 30 = 33.3 ms, and the corresponding movement of 10 cm for a vehicle traveling at a velocity of 10 km/h. If a vehicle can be provided feedback with "distance latency" of 10 cm, one can argue that a useful safety action can be actuated by it.

Performance numbers for speed are reported averaged over all classes, as the models do not train separately. The tracker metrics are also reported averaged across the used detectors.

The findings are summarized in Table V. The key learning is that none of our models can yet achieve the targeted real-time speed. A second key takeaway is that MaskRCNN is roughly ten times slower than either of the other two detectors. Of the three tracking models, DeepSORT is the fastest, followed by DAN (37% slower), with Multicut lagging far behind. The reason is that estimating optical flow requires significant computational power, primarily driven by the high resolution of our datasets - this takes up more than 99% of total inference time. While DAN and DeepSORT also process 1920×1080 resolution frames to generate the tracking results.

TABLE V
RUNTIME FOR DETECTORS AND TRACKERS FOR THE TEST DATASET

Detector	FPS	Tracker	FPS
MRCNN	1	DAN	2
SSD	9.2	DSORT	3.2
YOLOv3	11.8	MCUT	0.47

It is valuable to put these results in context. Recent research on real-time detection sees speeds ranging from 2.2 to 145.1 FPS [29]. This shows that state-of-the-art models trained specifically for real-time detection may still lag the threshold considerably; however, it also shows that real-time is achievable. We have not encountered any papers that achieve real-time MOT speed as we define it, though several claim real-time capability by relying on lower thresholds [30], [31].

D. Visualization of the Results

An example of the segmentation-based detection using Mask-RCNN is shown in Fig. 4. Red contours represent pedestrians while green represent vehicles. This figure not only shows the accuracy in detecting objects but also illustrates the difference in scale between vehicles and pedestrians.

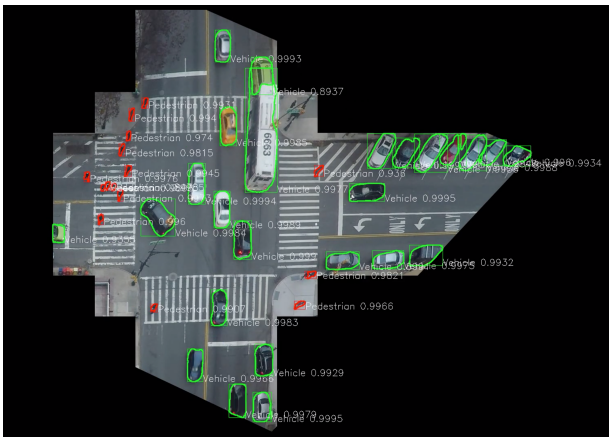

Fig. 4. Example of detection output: MaskRCNN

An example of the output for tracking using MaskRCNN with DeepSort is shown in Fig. 5. Every object in the figure is marked by a colored bounding box which contains the identifier assigned to the object. Many objects of both classes in frame 825 (left) maintain the same color and ID number in frame 800 (right), indicating successful tracking.

Fig. 5. Example of tracking output: Deepsort + MaskRCNN on frame 800 (right) and 825 (left)

VII. CONCLUSION

This paper considers the problem of real-time detection and tracking of objects in traffic intersections using bird's eye cameras, for purposes of supporting smart city use cases such as cloud-connected vehicles and assistance to handicapped pedestrians. Bird's eye infrastructural cameras are some of the key sensors in dense urban environments such as New York City, because they provide contextual awareness of the whole intersection, and are less problematic from the privacy protection perspective: pedestrian profiles are exceptionally small and vehicle license plates are not visible.

The studies presented in this paper are based on data collected on the COSMOS testbed pilot site, and have used its communications and computing equipment. Detection and tracking are facilitated by very-low latency high-bandwidth communications links, and edge-cloud computing resources with deep learning algorithms. The set of bird's eye videos or an urban intersection, acquired from the 12th floor, is a unique data set, which was previously not available for significant studies of detection and tracking. The dataset covers both daytime and nighttime conditions.

The small profile (small number of pixels) representing a pedestrian captured by a bird's eye camera, is in contrast to the significant number of pixels representing a vehicle. This results in significantly different accuracies in detection and tracking of vehicles vs. pedestrians, where pedestrians are much harder to reliably detect. We have presented an overview of the performance of three state-of-the-art detection algorithms (Yolo, MaskRCNN and Single-Shot Detection) and three state-of-the-art multiple object tracking algorithms (Deep-Sort, Multicut and Deep Affinity Networks), observing that the performance is widely spread both in accuracies and in timing. Representative multiple-object tracking accuracies

(MOTA) are around 73.2 for cars, and around 2.8 for pedestrians.

We have defined a latency target of 33.3 ms for detection/tracking of vehicles contending that such a latency would provide useful sampling of vehicle locations, corresponding to 0.1 meter distances for vehicles moving at 10 km/h. This is arguably a sufficiently small distance for purposes of assisting the cloud-connected vehicles. We further observe that pedestrians in intersections may move 3-10 times slower than vehicles, therefore latencies for pedestrian detection may be allowed to be 3-10 times larger that latencies for vehicles. The evaluation of the timing of contemporary deep-learning based methods indicates that detection of vehicles needs to be sped up at least three times to meet the execution time/latency of 33.3 ms.

ACKNOWLEDGMENTS

The authors thank the collaborators in Prof. Kostic's lab Tianyao Hua, Tingyu Mao, Yong Yang, Yanjun Lin, Deepak Ravishankar, Pablo Vicente Juan, Vibhuti Mahajan, Kunjian Liao, and WINLAB collaborator Jakub Kolodziejski. The team is grateful for the support of Columbia School of Engineering, Columbia Data Sciences Institute, and Rutgers University Winlab laboratory. This work was supported in part by NSF grant CNS-1827923, NSF-BSF grant CNS-1910757, and by an AT&T VURI award.

REFERENCES

[1] US DOT - Intelligent Transportation Systems Joint Programs Office, "Connected vehicle pilot deployment program." https://www.its.dot.gov/pilots/. Accessed: 2019-12-13.

[2] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Communications Surveys & Tutorials*, vol. 13, pp. 584–616, Apr. 2011.

[3] NYC DOT, "New york city connected vehicle project for safer transportation." https://cvp.nyc. Accessed: 2019-12-13.

[4] COSMOS team, "Cloud enhanced open software defined mobile wireless testbed for city-scale deployment." https://cosmos-lab.org/. Accessed: 2019-12-13.

[5] D. Raychaudhuri, I. Seskar, G. Zussman, T. Korakis, D. Kilper, T. Chen, J. Kolodziejski, M. Sherman, Z. Kostic, X. Gu, H. Krishnaswamy, S. Maheshwari, P. Skrimponis, and C. Gutterman, "Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless," *In Proc. ACM MOBICOM'20 (to appear)*, 2020.

[6] J. Yu, T. Chen, C. Gutterman, S. Zhu, G. Zussman, I. Seskar, and D. Kilper, "COSMOS: Optical architecture and prototyping," in *Proc. OSA/IEEE Optical Fiber Communications Conference (OFC)*, Mar. 2019.

[7] G. Ananthanarayanan, V. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. R. Sivalingam, and S. Sinha, "Real-time video analytics – the killer app for edge computing," *Computer*, vol. 50/10, pp. 58–67, Oct. 2017.

[8] C.-C. Hung, G. Ananthanarayanan, P. Bodík, L. Golubchik, M. Yu, V. Bahl, and M. Philipose, "Videoedge: Processing camera streams using hierarchical clusters," in *Proc. of ACM/IEEE (SEC)*, Oct. 2018.

[9] MIT, "Helping autonomous vehicles see around corners." http://news.mit.edu/2019/helping-autonomous-vehicles-see-around-corners-1028. Accessed: 2019-12-13.

[10] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[11] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv:1506.02640*, 2015.

[13] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *arXiv:1612.08242*, 2016.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

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

[16] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, "Multiple object tracking: A literature review," *arXiv:1409.7618*, 2014.

[17] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *CoRR*, vol. abs/1703.07402, 2017.

[18] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *CoRR*, vol. abs/1602.00763, 2016.

[19] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, "Deep affinity network for multiple object tracking," *CoRR*, vol. abs/1810.11780, 2018.

[20] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185 – 203, 1981.

[21] D. Sun, X. Yang, M. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," *CoRR*, vol. abs/1709.02371, 2017.

[22] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[23] Maxim Kolomeychenko, "Supervisely platform." https://supervise.ly. Accessed: 2019-12-13.

[24] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, "Learning features by watching objects move," *arXiv:1612.06370*, 2016.

[25] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *Int. J. Computer Vision*, vol. 88, pp. 303–338, June 2010.

[26] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *CoRR*, vol. abs/1603.00831, 2016.

[27] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear MOT metrics," *EURASIP J. on Image and Video Processing*, vol. 2008, pp. 1–10, May 2008.

[28] "MOT17 results." https://motchallenge.net/results/MOT17. Accessed: 2019-12-13.

[29] S. Chinchali, "Distributed perception and learning between robots and the cloud." http://web.stanford.edu/~csandeep/final_slides/csandeep_research_overview.pdf, 2019.

[30] Z. Wang, L. Zheng, Y. Liu, and S. Wang, "Towards real-time multi-object tracking," *arXiv:1909.12605*, 2019.

[31] A. Agarwal and S. Suryavanshi, "Real-time multiple object tracking (MOT) for autonomous navigation." http://cs231n.stanford.edu/reports/2017/pdfs/630pdf, 2017.