Intelligent Intersection: Two-stream Convolutional Networks for Real-time Near-accident Detection in Traffic Video

XIAOHUI HUANG, PAN HE, ANAND RANGARAJAN, and SANJAY RANKA, University of Florida, USA

Camera-based systems are increasingly used for collecting information on intersections and arterials. Unlike loop controllers that can generally be only used for detection and movement of vehicles, cameras can provide rich information about the traffic behavior. Vision-based frameworks for multiple-object detection, object tracking, and near-miss detection have been developed to derive this information. However, much of this work currently addresses processing videos offline. In this article, we propose an integrated two-stream convolutional networks architecture that performs real-time detection, tracking, and near-accident detection of road users in traffic video data. The two-stream model consists of a spatial stream network for object detection and a temporal stream network to leverage motion features for multiple-object tracking. We detect near-accidents by incorporating appearance features and motion features from these two networks. Further, we demonstrate that our approaches can be executed in real-time and at a frame rate that is higher than the video frame rate on a variety of videos collected from fisheye and overhead cameras.

CCS Concepts: • **Computer systems organization → Neural networks**; **Real-time system architecture**;

Additional Key Words and Phrases: Intelligent transportation systems, ITS, near-accident detection, multiple-object tracking, convolutional neural networks

ACM Reference format:
Xiaohui Huang, Pan He, Anand Rangarajan, and Sanjay Ranka. 2020. Intelligent Intersection: Two-stream Convolutional Networks for Real-time Near-accident Detection in Traffic Video. *ACM Trans. Spatial Algorithms Syst.* 6, 2, Article 10 (January 2020), 28 pages.
https://doi.org/10.1145/3373647

1 INTRODUCTION

The rapid changes in growth of exploitable and, in many cases, open data have the potential to mitigate traffic congestion and improve safety. Despite significant advances in vehicle technology, traffic engineering practices, and analytics based on crash data, the number of traffic crashes and fatalities are still too many. Many drivers are frustrated due to long (but potentially preventable)

This research was supported, in part, by the Florida Department of Transportation (FDOT) and NSF CNS 1922782. The opinions, findings, and conclusions expressed in this publication are those of the author(s) and not necessarily those of the Florida Department of Transportation or the U.S. Department of Transportation
Authors' address: X. Huang, P. He, A. Rangarajan, and S. Ranka, University of Florida, 432 Newell Drive, Gainesville, FL, 32611, USA; emails: {xiaohuihuang, pan.he}@ufl.edu, {anand, ranka}@cise.ufl.edu.

2374-0353/2020/01-ART10 $15.00
https://doi.org/10.1145/3373647

delays at intersections. The use of video or light detection and ranging (LIDAR) processing, big data analytics, artificial intelligence, and machine learning can profoundly improve the ability to address these challenges. The collection and exploitation of large data sets is not new to the transportation sector. However, the confluence of ubiquitous digital devices and sensors, significantly lower hardware costs for computing and storage, enhanced sensing and communication technologies, and open-source analytics solutions have enabled novel applications. The latter may involve insights into otherwise unobserved patterns that may positively influence individuals and society.

The technologies of artificial intelligence (AI) and the Internet of Things (IoTs) are ushering in a new promising era of "Smart Cities" where billions of people around the world can improve the quality of their lives in aspects of transportation, security, information, communications, and so on. One example of data-centric AI solutions is computer vision technologies that enable vision-based intelligence for edge devices across multiple architectures. Sensor data from smart devices or video cameras can be analyzed immediately to provide real-time analysis for intelligent transportation systems (ITS). At traffic intersections, there is a greater volume of road users (pedestrians and vehicles), traffic movement, dynamic traffic events, near-accidents, and so on. It is a critically important application to enable global monitoring of traffic flow, local analysis of road users, and automatic near-accident detection.

As a new technology, vision-based intelligence has many applications in traffic surveillance and traffic management [Buch et al. 2011; Coifman et al. 1998; He et al. 2017; Kamijo et al. 2000; Valera and Velastin 2005; Veeraraghavan et al. 2003]. Among them, many research works have focused on traffic data acquisition with aerial videos [Angel et al. 2002; Salvo et al. 2017]; the aerial view provides better perspectives to cover a large area and focus resources for surveillance tasks. Unmanned aerial vehicles (UAVs) and omnidirectional cameras can acquire useful aerial videos for traffic surveillance, especially at intersections, with a broader perspective of the traffic scene and with the advantage of being both mobile and spatio-temporal. A recent trend in vision-based intelligence is to apply computer vision technologies to these acquired intersection aerial videos [Scotti et al. 2005; Wang et al. 2006] and process them at the edge across multiple ITS architectures.

Object detection and multiple-object tracking are widely used applications in transportation, and real-time solutions are significant, especially for the emerging area of big transportation data. A near-miss is an event that has the potential to develop into a collision situation between two vehicles or between a vehicle and a pedestrian or bicyclist. These events are important to monitor and analyze for preventing collisions in the future. They also serve as a proxy for potential timing and design issues at the intersection. Camera monitored intersections produce video data in gigabytes per camera per day. Analyzing thousands of trajectories collected per hour at an intersection from different sources to identify near-miss events, quickly becomes challenging, given the amount of data to be examined and the relatively rare occurrence of such events.

In this article, we investigate the use of traffic video data for near-accident detection. However, to our best knowledge, a unified system that performs real-time detection and tracking of road users and near-accident detection for aerial videos is not available. Therefore, we have collected video datasets and presented a real-time deep-learning-based method to tackle these problems.

Generally, a vision-based surveillance tool for ITS should meet several requirements: (1) segment vehicles from their surroundings (including other road objects and the background) to detect all road objects (still or moving); (2) classify detected vehicles into categories: cars, buses, trucks, motorbikes, and so on; (3) extract spatial and temporal features (motion, velocity, and trajectory) to enable more specific tasks, including vehicle tracking, trajectory analysis, near-accident detection, anomaly detection, and so on; (4) function under various traffic conditions and lighting conditions; and (5) operate in real time.

Over the decades, although an increasing amount of research on vision-based systems for traffic surveillance has been proposed, many of the criteria listed above still cannot be met. Early solutions [Hoose 1992] do not identify individual vehicles as unique targets and progressively track their movements. Methods have been proposed to address individual vehicle detection and vehicle tracking problems [Coifman et al. 1998; Koller et al. 1993; McLauchlan et al. 1997] with tracking strategies and optical flow deployment. Compared to traditional hand-crafted features, deep-learning methods [Girshick et al. 2016; He et al. 2019a, 2019b; Redmon et al. 2016; Ren et al. 2015; Tian et al. 2016] in object detection have illustrated the robustness of specialization of the generic detector to a specific scene. Recently, automatic traffic accident detection has become an important topic. Before detecting accident events [Hommes et al. 2011; Jiang et al. 2007; Jiansheng et al. 2014; Kamijo et al. 2000; Sadeky et al. 2010], one typical approach is to apply object detection or tracking methods, using histogram of flow gradient (HFG), hidden Markov model (HMM), or Gaussian mixture model (GMM). Other approaches [Chen et al. 2010, 2016; Ihaddadene and Djeraba 2008; Karim and Adeli 2002; Liu et al. 2010; Tang and Gao 2005; Wang and Dong 2012; Wang and Miao 2010; Xia et al. 2015] use low-level features (e.g., motion features) to demonstrate better robustness. Neural networks have also been employed for automatic accident detection [Ghosh-Dastidar and Adeli 2003; Ohe et al. 1995; Srinivasan et al. 2004; Yu et al. 2008].

Intersections tend to experience more, and potentially more severe, near-accidents, due to factors such as angles and turning collisions. The first type of intersection video considered here is drone video, which monitors an intersection with a top-down view. The second type is real traffic video acquired by omnidirectional fisheye cameras that monitor small or large intersections. This is widely used in transportation surveillance. These can be directly used as inputs for any vision-based framework.

We propose a unified vision-based framework for near-incident detection that has the following novel ideas:

(1) **A combined spatial and temporal stream convolutional networks architecture that performs object detection, classification, and tracking.** The spatial stream network detects individual vehicles and likely near-accident regions at the single frame level by capturing appearance features with a state-of-the-art object-detection method. The temporal stream network leverages motion features extracted from detected candidates to perform multiple-object tracking and generates corresponding trajectories of each tracked target;
(2) **A frame-based spatial pixel segmentation approach to derive accurate bounding boxes for multiple objects.** This approach is considerably more accurate as compared to rectangular bounding boxes approaches;
(3) **A metric learning approach for improved tracking in the presence of occlusion.** Occlusion can lead to several frames where the object goes undetected;
(4) **A novel distance measure between tracks to detect near-misses between objects.** This measure detects near-accidents by incorporating appearance features and motion features to compute probabilities of near-accident candidate regions.

In addition, we show that this can be executed on a GPU at the rate of 33+ frames per second, which is useful in real-time video applications. We also note that these methods are executed on fisheye lens-based video, which is considerably more difficult than standard rectilinear video frame geometry. Experiments demonstrate the advantage of our framework with an overall competitive performance at high frame rates. The overall pipeline of our method is depicted in Figure 1.

The organization of the paper is as follows. Section 2 describes the background of convolutional neural networks, object detection, and multiple-object-tracking methods. Section 3 describes the

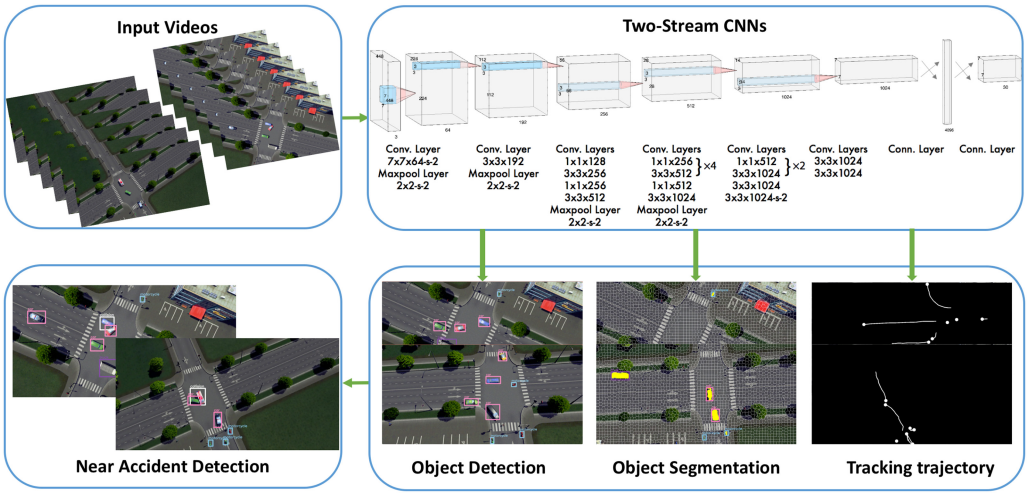

Fig. 1. The overall pipeline of our two-stream convolutional neural networks for near-accident detection.

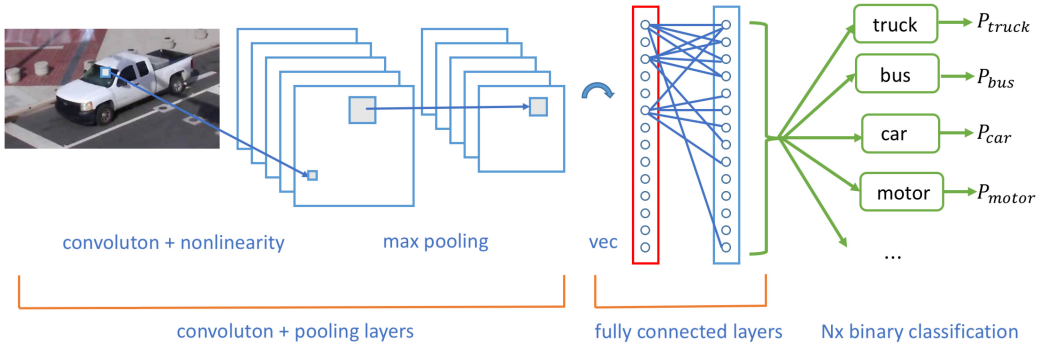

Fig. 2. Architecture of convolutional neural networks for image classification.[1]

overall architecture, methodologies, and implementation of our method. This is followed in Section 4 by an introduction of our traffic near-accident detection dataset (TNAD) and presentation of a comprehensive evaluation of our approach and other state-of-the-art near-accident detection methods, both qualitatively and quantitatively. Section 5 describes a recent literature of object detection, multiple-object tracking and near-accident detection. Section 6 summarizes our contributions and also discusses the scope of future work.

2 A BRIEF DESCRIPTION OF STANDARD COMPUTER VISION APPROACHES

2.1 Convolutional Neural Networks (CNNs)

CNNs have shown strong capabilities in representing objects, thereby boosting the performance of numerous vision tasks, especially when compared to traditional features [Dalal and Triggs 2005]. A CNN is a class of deep neural network that is widely applied in image analysis and computer vision. A standard CNN usually consists of both the input layer and the output layer, as well as multiple hidden layers (e.g., convolutional layers, fully connected layers, pooling layers) as shown in Figure 2. The input to a convolutional layer is the original image $\boldsymbol{X}$. We denote the feature map

[1]Figure based on the blog article published at https://adeshpande3.github.io/.

of the ith convolutional layer as $\boldsymbol{H}_i$ and $\boldsymbol{H}_0 = \boldsymbol{X}$. Then $\boldsymbol{H}_i$ can be described as

$$\boldsymbol{H}_i = f\left(\boldsymbol{H}_{i-1} \otimes \boldsymbol{W}_i + \boldsymbol{b}_i\right), \tag{1}$$

where $\boldsymbol{W}_i$ is the weight for the ith convolutional kernel for the $i-1$th image or feature map and $\otimes$ is the convolution operation. The output of the convolution operation includes a bias $\boldsymbol{b}_i$. Then, the feature map for the ith layer can be computed by applying a standard nonlinear activation function. We briefly describe a 32×32 RGB image with a simple ConvNet for CIFAR-10 image classification [Krizhevsky et al. 2009]:

- Input layer: the original 32×32 image with three color channels (R,G,B).
- Convolutional layer: local operations in the image passed through nonlinear activation functions. If we use 12 filters, then the size of the result is $[32 \times 32 \times 12]$.
- Pooling layer: a downsampling operation, resulting in a volume such as $[16 \times 16 \times 12]$.
- Fully connected layer: output scores for each class, resulting in a volume of size $[1 \times 1 \times 10]$ for 10 classes.

In this way, CNNs transform the original image into multiple high-level feature representations layer by layer, finally obtaining class-specific outputs or scores.

2.2 YOLO Object Detection

The real-time You Only Look Once (YOLO) detector, proposed in Redmon et al. [2016], is an end-to-end state-of-the-art deep-learning approach without using region proposals. The pipeline of YOLO [Redmon et al. 2016] is rather straightforward: Given an input image, YOLO [Redmon et al. 2016] passes it through the neural network only once, as its name implies (You Only Look Once), and outputs the detected bounding boxes and class probabilities in prediction. Figure 4 demonstrates the detection model and system of YOLO [Redmon et al. 2016]. YOLO [Redmon et al. 2016] is orders of magnitude faster (45 frames per second) than other object-detection approaches, which means it can process streaming video in real time. Compared to other systems, it achieves a higher mean average precision as well. In this work, we leverage the extension of YOLO [Redmon et al. 2016], Darknet-19, a classification model used as the basis of YOLOv2 [Redmon and Farhadi 2017]. Darknet-19 [Redmon and Farhadi 2017] consists of 19 convolutional layers and five max-pooling layers, where batch normalization is utilized to stabilize training, speed up convergence, and regularize the model [Ioffe and Szegedy 2015].

2.3 Simple Online Real-time Tracking (SORT)

SORT [Bewley et al. 2016] is a simple, popular, and fast multiple-object-tracking (MOT) algorithm. The core idea is a combination of Kalman filtering [Kalman 1960] and frame-by-frame data association. The data association is implemented with the Hungarian method [Kuhn 1955], by measuring the bounding box overlap. With this rudimentary combination, SORT [Bewley et al. 2016] achieves a state-of-the-art performance compared to other online trackers. Moreover, due to its simplicity, SORT [Bewley et al. 2016] can update at a rate of 260 Hz on a single machine, which is over 20 times faster than other state-of-the-art trackers.

2.4 DeepSORT

DeepSORT [Wojke et al. 2017] is an extension of SORT [Bewley et al. 2016]. DeepSORT integrates appearance information to improve the performance of SORT [Bewley et al. 2016], by adding one pre-trained association metric. DeepSORT [Wojke et al. 2017] helps to solve a large number of identity switching problems in SORT [Bewley et al. 2016], and it can track occluded objects in

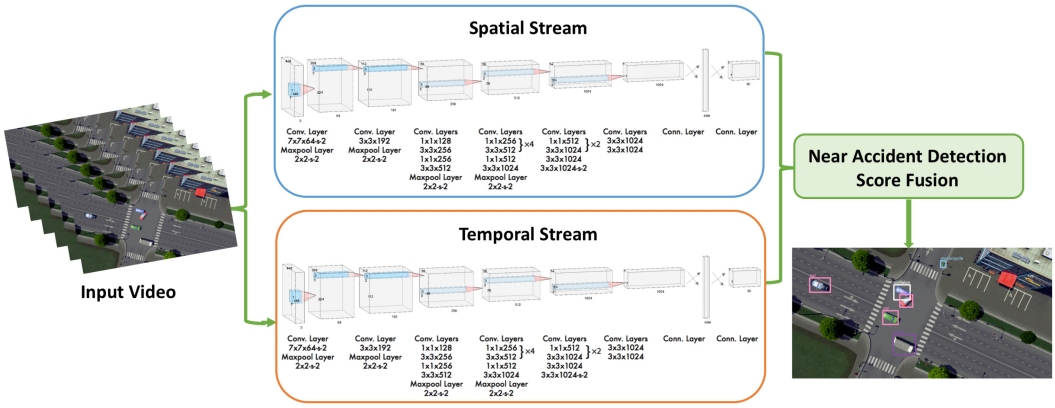

Fig. 3. Two-stream convolutional neural networks architecture for near-accident detection.

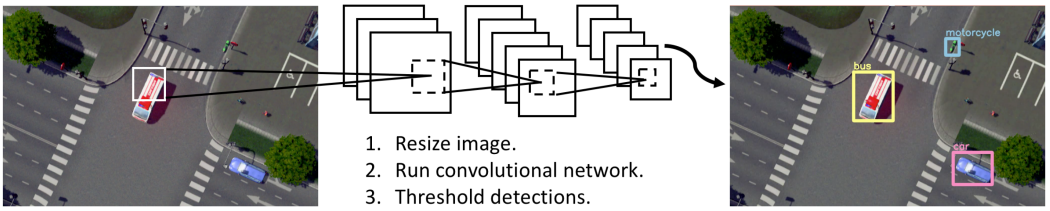

Fig. 4. Object-detection pipeline of our spatial stream [Redmon et al. 2016]: (1) resizes the input video frame, (2) runs convolutional network on the frame, and (3) thresholds the resulting detection by the model's confidence.

a longer term. During online application, the measurement-to-track association is established in visual appearance space, using nearest-neighbor queries.

3 TWO-STREAM ARCHITECTURE FOR NEAR-ACCIDENT DETECTION

In this section, we present our computer vision-based two-stream architecture for real-time near-accident detection. The architecture is primarily driven by real-time object detection and multiple-object tracking (MOT). The goal of near-accident detection is to detect likely collision scenarios across video frames and report these near-accident records. Because videos have both spatial and temporal components, we divide our framework into a two-stream architecture as shown in Figure 3. The spatial aspect comprises individual frame appearance information about scenes and objects. The temporal aspect comprises motion information of objects. For the spatial stream convolutional neural network, we utilize a standard convolutional network designed for state-of-the-art object detection [Redmon et al. 2016] to detect individual vehicles and mark near-accident regions at the single-frame level. The temporal stream network leverages object candidates from object-detection CNNs and integrates their appearance information with a fast MOT method to extract motion features and compute trajectories. When two trajectories of individual objects intersect or come closer than a certain threshold (whose estimation is described below), we label the region covering the two objects as a high probability near-accident region. Finally, we take the average near-accident probability of both the spatial stream network and the temporal stream network and report the near-accident record.

3.1 Object Detection and Classification

In our framework, each stream is implemented using a deep convolutional neural network. Near-accident scores are combined by averaging. Since our spatial stream ConvNet is essentially an object-detection architecture, we base it on recent advances in object detection—essentially the YOLO detector [Redmon et al. 2016]—and pre-train the network from scratch on our dataset containing multi-scale drone, fisheye, and simulation videos. As most of our videos contain traffic scenes with vehicles and traffic movement captured in a top-down view, we specify different vehicle classes such as motorbike, car, bus, and truck as object classes for training the detector. Additionally, near-accidents or collisions can be detected from single still frames even at the beginning of a video or from stopped vehicles associated in an accident after collision. Therefore, we train our detector to localize these likely near-accident scenarios. Since static appearance is a useful cue, the spatial stream network performs object detection by only operating on individual video frames.

The spatial stream network regresses the bounding boxes and predicts the class probabilities associated to these boxes, using a simple end-to-end convolutional network. It first takes the image and splits it into a $S \times S$ grid. For each grid cell,

- it predicts B boundary boxes, and each box has a confidence score
- it detects one object only, regardless of the number of boxes B
- it predicts C conditional class probabilities (one per class for the likelihood of the object class)

For each bounding box, the CNN outputs a class probability and offset values for the bounding box. Then, it selects bounding boxes that have the class probability above a threshold value and uses them to locate the object within the image. In essence, each boundary box contains five elements: (x, y, w, h) and a box confidence. The (x, y) are coordinates that represent the center of the box relative to the bounds of the grid cell. The (w, h) parameters are the width and height of the object. These elements are normalized such that x, y, w and h lie in the interval $[0, 1]$. The intersection over union (IoU) between the predicted bounding box and the ground-truth box is used in confidence prediction, which reflects the likelihood that the box contains an object (objectness) and the accuracy of the boundary box. The mathematical definitions of the scoring and probability terms are:

$$\begin{gathered}\text{box confidence score } P_r(object) \cdot IoU,\\ \text{conditional class probability } P_r(class_i|object),\\ \text{class confidence score } P_r(class_i) \cdot IoU,\\ \text{class confidence score} = \text{box confidence score} \times \text{conditional class probability},\end{gathered}$$

where $P_r(object)$ is the probability that the box contains an object. IoU is the intersection over union (IoU) between the predicted box and the ground-truth box. $P_r(class_i)$ is the probability that the object belongs to $class_i$. $P_r(class_i|object)$ is the probability that the object belongs to $class_i$ given an object is present. The network architecture of the spatial stream simply contains 24 convolutional layers followed by two fully connected layers, reminiscent of AlexNet and even earlier convolutional architectures. The last convolution layer is flattened, which outputs a $(7, 7, 1024)$ tensor. It performs a linear regression using two fully connected layers to make boundary box predictions and to make a final prediction using the threshold of box confidence scores. The final loss adds the localization (the first and the second terms), confidence (the third and the fourth

terms), and classification (the fifth term) losses together. The objective function is

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

where $\mathbb{1}_i^{\text{obj}}$ denotes if the object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

3.2 Multiple-object Tracking

In general, MOT can be regarded as a multi-variable estimation problem [Luo et al. 2014], and the objective of MOT can be modeled by performing MAP (maximum *a posteriori*) estimation to find the *optimal* sequential states of all the objects from the conditional distribution of the sequential states, given all the observations:

$$\widehat{\mathbf{S}}_{1:t} = \arg\max_{\mathbf{S}_{1:t}} P\left(\mathbf{S}_{1:t} | \mathbf{O}_{1:t}\right), \quad (3)$$

where $\mathbf{s}_t^i$ denotes the state of the ith object in the tth frame, $\mathbf{S}_t = (\mathbf{s}_t^1, \mathbf{s}_t^2, \ldots, \mathbf{s}_t^{M_t})$ denotes states of all the M_t objects in the tth frame, and $\mathbf{S}_{1:t} = \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_t\}$ denotes all the sequential states of all the objects from the first frame to the tth frame. In tracking-by-detection, $\mathbf{o}_t^i$ denotes the collected observations for the ith object in the tth frame. $\mathbf{O}_t = (\mathbf{o}_t^1, \mathbf{o}_t^2, \ldots, \mathbf{o}_t^{M_t})$ denotes the collected observations for all the M_t objects in the tth frame. $\mathbf{O}_{1:t} = \{\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_t\}$ denotes all the collected sequential observations of all the objects from the first frame to the tth frame.

The spatial stream network is not able to extract motion features and compute trajectories due to single-frame inputs. To leverage this useful information, we present our temporal stream network, a ConvNet model that implements a tracking-by-detection MOT algorithm [Bewley et al. 2016; Wojke et al. 2017] with a data association metric that combines deep appearance features. The inputs are identical to the spatial stream network using the original video. Detected object candidates (only vehicle classes) are used for tracking, state estimation, and frame-by-frame data association using SORT [Bewley et al. 2016] and DeepSORT [Wojke et al. 2017]—the real-time MOT method. MOT models the state of each object and describes the motion of objects across video frames. The tracking information allows us to stack trajectories of moving objects across several consecutive frames, which are useful cues for near-accident detection.

Estimation Model. For each target, its state is modeled as

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T, \quad (4)$$

where u, v denotes the 2D pixel location of the target's center, s denotes the scale of the target's bounding box, and r is the aspect ratio (usually considered to be constant). The target's state is updated when a new detected bounding box is associated to it, by solving the velocity via a Kalman filter [Kalman 1960]. The target's state is simple predicted (without correction via the Kalman filter) if no bounding box is associated to it.

Data Association. To assign all new detection boxes to existing targets, we predict each target's new location (one predicted box) in the current frame and compare the IoU distance between new detection boxes and all predicted boxes (the detection-to-target overlaps), forming as the assignment cost matrix. The matrix is further used in the Hungarian algorithm [Kuhn 1955] to

solve the assignment problem. One assignment is rejected if the detection-to-target overlap is less than a threshold (the minimum IoU IoU_{min}). The IoU distances of the bounding boxes are utilized to handle occlusion caused by passing targets in short term.

Creation and Deletion of Track Identities. When new objects enter or old objects vanish in video frames, we need to add or remove certain tracks (or objects) accordingly to main unique identities. We treat any detection fails to associate to existing targets as one potential untracked object. This target further undergoes a probationary period to accumulate enough evidence, by associating the target with detection. This strategy can prevent tracking of false positives. The track object is removed from the tracking list if they failed to be detected for T_{Lost} frames, which enables to maintain an relatively unbounded number of trackers and reduces localization errors accumulated over long duration.

Track Handling and State Estimation. This part is mostly identical to SORT [Bewley et al. 2016]. The tracking scenario is represented as an eight-dimensional state space $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$. (u, v) represents the bounding box center location. γ is the aspect ratio and h is the height. The rest are its velocities relative to other bounding boxes. This representation is used to describe the constant velocity motion $(\dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$ and linear observation model (u, v, γ, h) in Kalman filtering [Kalman 1960].

Data Association. To solve the frame-by-frame association problem, SORT uses the Hungarian algorithm [Kuhn 1955] where both motion and appearance information are considered.

To handle motion information, the (squared) Mahalanobis distance is utilized to measure the distance between newly arrived measurements and Kalman states:

$$d^{(1)}(i, j) = (\boldsymbol{d}_j - \boldsymbol{y}_i)^T (S_i)^{-1} (\boldsymbol{d}_j - \boldsymbol{y}_i), \tag{5}$$

where $\boldsymbol{y}_i$ and $\boldsymbol{d}_j$ denote the ith track distribution and the jth bounding box detection, respectively. The smallest cosine distance between the ith track and jth detection is considered to handle appearance information:

$$d^{(2)}(i, j) = \min \left\{ 1 - \boldsymbol{r}_j{}^T \boldsymbol{r}_k^{(i)} \middle| \boldsymbol{r}_k^{(i)} \in \mathcal{R}_i \right\}. \tag{6}$$

Both motion and appearance are combined with a linear combination where the influence of each is controlled by a hyperparameter λ:

$$c_{i,j} = \lambda\, d^{(1)}(i, j) + (1 - \lambda) d^{(2)}(i, j). \tag{7}$$

Matching Cascade. Previous methods try to solve measurement-to-track associations globally, while SORT adopts a matching cascade, introduced in Wojke et al. [2017] to solve a series of subproblems. In some situations, when an object is occluded for a longer period of time, the subsequent Kalman filter [Kalman 1960] predictions would increase the uncertainty associated with the object location. As a consequence, the probability mass will spread out in the state space, and the observation likelihood will be less peaked. The measurement-to-track distance should be increased by considering the spread of probability mass. Therefore, objects seen more frequently are given more priority in the matching cascade strategy to encode the notion of probability spread in the association likelihood.

3.3 Metric Learning for Vehicle Re-identification

Metric learning and various hand-crafted features are widely used in person re-identification problems. We apply a cosine metric learning method to train a neural network for vehicle re-identification and use it to make our temporal stream more robust in terms of tracking performance. Metric learning aims to solve the clustering problem by constructing an embedding in which the metric distance corresponding to the same identity is likely to be closer than features

Fig. 5. Vehicle samples selected from the VeRi dataset [Liu et al. 2016b]. All images are the same size, and we resize to 64 × 128 for training. Left: Diversity of vehicle colors and types; right: variation of the viewpoints, illuminations, resolutions, and occlusions for the vehicles.

from different identities. The cosine metric measures the degree of similarity by calculating the cosine distance between two objects. We observe that in traffic video, especially in crowded traffic scenes or scenes with heavy occlusion, our tracking algorithm produces switched road user identities. To generate accurate and consistent track data, we introduce a deep cosine metric learning method to learn the cosine distance between objects. The cosine distance involves appearance information that provides useful cues for recovering identities in crowded scenes or after long-term occlusion when motion information is less discriminative. Through this deep network, the feature expression vector obtained by any object is placed in the cluster corresponding to the nearest neighbor. We trained the network on a vehicle re-identification dataset (VeRi dataset) [Liu et al. 2016b] (Figure 5) and integrated it as the second metric measure for the assignment problem of our temporal stream.

Our idea is to use a unified end-to-end framework that automatically learns the best metrics. Compared to the standard distance metric (L1, L2), the learned metric can obtain more discriminative features for Re-ID and is more robust to cross-view vehicle images. We adopt the architecture of [Wojke and Bewley 2018] and train a deep network with a cosine softmax classifier, which can generate feature vectors of fixed length (128) for the input images (vehicles). The cosine metric finds the nearest cluster exemplar and matches the vehicle on (sometimes far-flung) different video frames to solve tracking problems in the presence of heavy occlusion.

Given a re-identification dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ of N training images $\boldsymbol{x}_i \in \mathbb{R}^D$ and associated class labels $y_i \in \{1, \ldots, C\}$, deep metric learning is to find a parametrized encoder function. The deep metric neural network is $\boldsymbol{r} = f_\Theta(\boldsymbol{x})$ with parameters Θ, which projects the input images $\boldsymbol{x} \in \mathbb{R}^D$ into a feature representation $\boldsymbol{r} \in \mathbb{R}^d$ that follows a predefined notion of cosine similarity.

The cosine softmax classifier can be adapted from a standard softmax classifier and is expressed as

$$p(y = k|r) = \frac{\exp\left(\kappa \cdot \tilde{\boldsymbol{w}}_k^{\mathrm{T}} \boldsymbol{r}\right)}{\sum_{n=1}^{C} \exp\left(\kappa \cdot \tilde{\boldsymbol{w}}_n^{\mathrm{T}} \boldsymbol{r}\right)}, \tag{8}$$

where κ is a free-scale parameter. To generate compact clusters in the feature representation space, the first modification is to apply the ℓ_2 normalization to the final layer of the encoder network so that the representation has unit length $\|f_\Theta(\boldsymbol{x})\|_2 = 1, \forall \boldsymbol{x} \in \mathbb{R}^D$. The second modification is to normalize the weights to unit length as well, i.e., $\tilde{\boldsymbol{w}}_k = \boldsymbol{w}_k / \|\boldsymbol{w}_k\|_2, \forall k = 1, \ldots, C$. The training of the encoder network can be carried out using the cross-entropy loss, as usual. In particular, the authors in Salimans and Kingma [2016] have proposed a way to accelerate convergence of stochastic gradient descent by decoupling the length of the weight vector κ from its direction.

3.4 Using Image Segmentation to Determine Object Gaps

To present our method in a fully end-to-end fashion, we apply a learning-based dynamic distance threshold rather than a manually defined one to determine near-accident detection. Near-accident is detected if the distance between two objects in image space is below the threshold. We introduce a threshold learning method by estimating the gap between objects using object segmentation. We present a supervised learning method, using gap distance between road users in the temporal stream to determine a near-accident while continuing to update the threshold for more convergence and precision. The straightforward way is to compute the gap using bounding boxes from detections, but the bounding boxes are not accurate in terms of representing boundaries of objects, particularly when rotation is involved. We need a more compact representation and therefore combine detections with superpixel segmentation on the video frame to get an object mask for gap distance estimation.

The main steps of our gap estimation method can be summarized as follows:

(1) Extract object detections (bounding boxes) from the spatial stream network.
(2) Tessellate the video frame into homogeneous superpixels.
(3) Compute the Intersection over Union (IoU) metric of bounding boxes and superpixels at frame level.
(4) Generate object mask per detections (using the IoU metric), and compute the gap between objects using the mask and bounding boxes.
(5) Update the gap threshold while processing the video.

In our gap estimation method, the Intersection over Union (IoU) metric of boxes and superpixels is given by

$$IoU = \frac{area(B_{1:t}) \cap area(S_{1:t})}{area(B_{1:t}) \cup area(S_{1:t})}, \tag{9}$$

where $\mathbf{b}_t^i$ denotes the detection for the ith object in the tth frame, $\mathbf{B}_t = (\mathbf{b}_t^1, \mathbf{b}_t^2, \ldots, \mathbf{b}_t^{M_t})$ denotes the collected detection observations for all the M_t objects in the tth frame, and $\mathbf{B}_{1:t} = \{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_t\}$ denotes all the collected sequential detection observations of all the objects from the first frame to the tth frame. Similarly, $\mathbf{s}_t^i$ denotes the ith superpixel in the tth frame. $\mathbf{S}_t = (\mathbf{s}_t^1, \mathbf{s}_t^2, \ldots, \mathbf{s}_t^{N_t})$ denotes all the N_t superpixels in the tth frame. $\mathbf{S}_{1:t} = \{\mathbf{S}_1, \mathbf{B}_2, \ldots, \mathbf{S}_t\}$ denotes all the collected sequential superpixels from the first frame to the tth frame.

In 2D superpixel segmentation, the popular SLIC (simple linear iterative clustering) [Achanta et al. 2012] and ultrametric contour map (UCM) [Arbelaez et al. 2010] methods have established themselves as the state of the art. Recently, we have seen deep neural networks [Jampani et al. 2018] and generative adversarial networks (GANs) [Yan et al. 2018] integrated with these methods. In our architecture, we adopt the GPU-based SLIC (gSLICr) [Ren et al. 2015] approach to produce the tessellation of the image data to achieve an excellent frame rate (400 fps). SLIC is widely applicable to many computer vision applications, such as segmentation, classification, and object recognition, often achieving state-of-the-art performance. The contours forming the homogeneous superpixels are shown in Figure 6. In Figure 7, we show an illustration for the definition of intersection and union, and present some object masks we generated from detections and superpixels.

3.5 Near-accident Detection

The most typical motion cue is optical flow, which is widely utilized in video processing tasks such as video segmentation [Huang et al. 2018]. A trajectory is defined as a data sequence

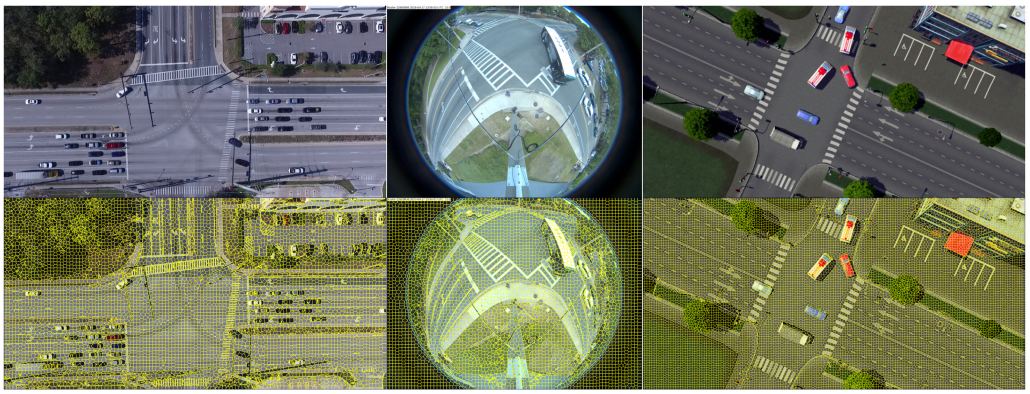

Fig. 6. 2D image superpixel segmentation using SLIC [Achanta et al. 2012]. Top: original image; bottom-left: SLIC segmentation for drone video (1,600 superpixels); bottom-middle: SLIC segmentation for fisheye video (1,600 superpixels); bottom-right: SLIC segmentation for simulation video (1,600 superpixels).

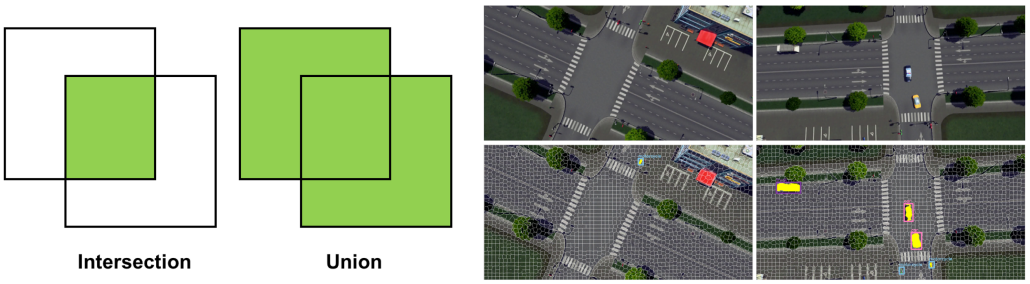

Fig. 7. (a) Illustration depicting the definitions of intersection and union. (b) Top: original video frames; bottom: object detections and object masks produced by our method.

containing several concatenated state vectors from tracking and an indexed sequence of positions and velocities over a given time window.

When utilizing the multiple-object-tracking algorithm, we compute the center of each object in several consecutive frames to form stacking trajectories as our motion representation. These stacking trajectories can provide accumulated information through image frames, including the number of objects, their motion history, and timing of their interactions, such as near-accidents. We stack the trajectories of all objects for L consecutive frames, as illustrated in Figure 8, where $\mathbf{p}_t^i$ denotes the center position of the ith object in the tth frame. $\mathbf{P}_t = (\mathbf{p}_t^1, \mathbf{p}_t^2, \ldots, \mathbf{p}_t^{M_t})$ denotes trajectories of all the M_t objects in the tth frame. $\mathbf{P}_{1:t} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_t\}$ denotes the sequence of trajectories of all the objects from the first frame to the tth frame. For L consecutive frames, the stacking trajectories are defined by the sequence

$$\mathbf{P}_{1:L} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_L\}, \mathbf{P}_{L+1:2L} = \{\mathbf{P}_{L+1}, \mathbf{P}_{L+2}, \ldots, \mathbf{P}_{2L}\}, \ldots \tag{10}$$

where $\mathbf{O}_t = (\mathbf{o}_t^1, \mathbf{o}_t^2, \ldots, \mathbf{o}_t^{M_t})$ denotes the collected observations for all the M_t objects in the tth frame, and $\mathbf{O}_{1:t} = \{\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_t\}$ denotes all the collected sequential observations of all the objects from the first frame to the tth frame. We use a simple detection algorithm that finds collisions between simplified forms of the objects, using the center of bounding boxes.

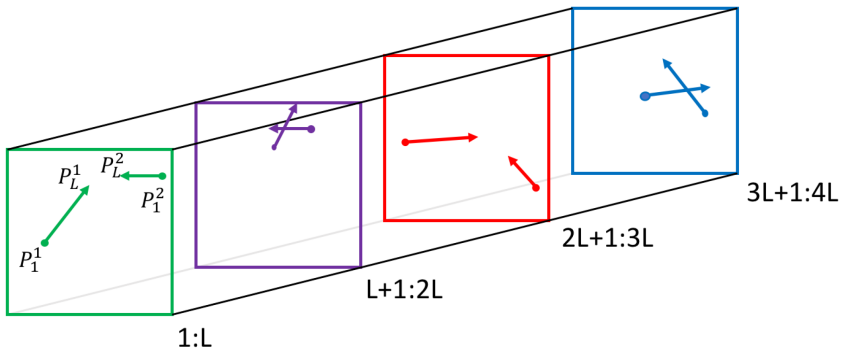

Fig. 8. The stacking trajectories extracted from multiple-object tracking of the temporal stream. Consecutive frames and the corresponding displacement vectors are shown with the same color.

ALGORITHM 1: Collision Detection

Input: current frame $t_{current}$, collision state list *Collision*
Output: collision state list *Collision*
for $t_L \leftarrow t_{previous}$ *to* $t_{current}$ *in steps of L frames* **do**
  for *each pair of object trajectory* $(\mathbf{p}^1_{:t_L}, \mathbf{p}^2_{:t_L})$ **do**
    if $(\mathbf{p}^1_{:t_L}$ *intersects* $\mathbf{p}^2_{:t_L}$ *as of* $t_L)$ **then**
      add $\mathbf{o}_1$, $\mathbf{o}_2$ to *Collision*
    end
  end
  if *(Collisions)* **then**
    $t_{previous} \leftarrow t_L$; return TRUE
  end
end
$t_{previous} \leftarrow t_d$; return FALSE

Our algorithm is depicted in Algorithm 1. Once a collision is detected, we set the region covering collision-associated objects to be a new bounding box with class probability of near-accident of 1. By averaging the near-accident probabilities from the spatial stream network and the temporal stream network, we are able to obtain a final confidence measure of near-accident detection.

4 EXPERIMENTS

Here, we present qualitative and quantitative evaluation in terms of the performance of object detection, MOT, and near-accident detection, and a comparison between other methods and our framework.

4.1 A Traffic Near-accident Dataset (TNAD)

To our best knowledge, we know of no comprehensive traffic near-accident dataset containing top-down-view videos such as drone or UAV videos or omnidirectional camera videos for traffic analysis. Therefore, we have built our own dataset, the traffic near-accident dataset (TNAD), which is depicted in Figure 9. Intersections tend to experience more near-accidents and more potentially severe ones due to factors such as angles and turning collisions. The traffic near-accident dataset contains three types of video data from traffic intersections that could be utilized for not only for near-accident detection but also for other traffic surveillance tasks, including turn movement counting.

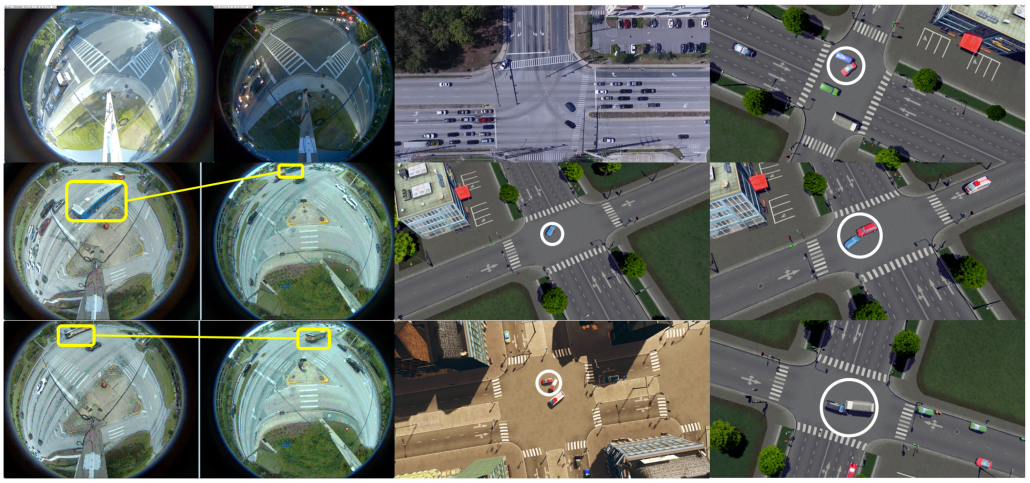

Fig. 9. Samples of traffic near-accident: Our data consists of a large number of diverse intersection surveillance videos and different near-accidents (cars and motorbikes). Yellow rectangles and lines represent the same object in video from multiple cameras. White circles represent the near-accident regions.

The first type is drone video that monitors an intersection with a top-down view. The second type of intersection video is real traffic video acquired by omnidirectional fisheye cameras that monitor small or large intersections. They are widely used in transportation surveillance. These video data can be directly used as input for our vision-based intelligent framework. Furthermore, pre-processing and fisheye correction can be applied for better surveillance performance. The third type of video is video game-engine simulations that train on near-accident samples as they are accumulated. The dataset consists of 106 videos with total duration over 75 minutes, with frame rates between 20 and 50 fps. The drone video and fisheye surveillance videos were recorded in Gainesville, Florida, at several different intersections. Our videos are more challenging than videos in other datasets for the following reasons:

- Diverse intersection scene and camera perspectives: The intersections in drone video, fisheye surveillance video, and simulation video are very diverse. Additionally, the fisheye surveillance video has distortion, and a fusion technique is needed for multi-camera fisheye videos.
- Crowded intersection and small object: The number of moving cars and motorbikes per frame is large, and these objects are relatively smaller than normal traffic video.
- Diverse accidents: Accidents involving cars and motorbikes are all included.
- Diverse lighting conditions: Lighting conditions such as daylight and sunset are included.

We manually annotated the spatial location and temporal location of near-accidents and the still or moving objects and their vehicle class in each video. Thirty-two videos with sparsely sampled frames (only 20% of the frames in these 32 videos are used for supervision) were used, but only for training the object detector. The remaining 74 videos were used for testing.

4.2 Fisheye and Multi-camera Video

We have large amounts of fisheye traffic videos from across the city. The fisheye surveillance videos were recorded from real traffic data in Gainesville. We collected 29 single-camera fisheye surveillance videos and 19 multi-camera fisheye surveillance videos monitoring a large intersection. We

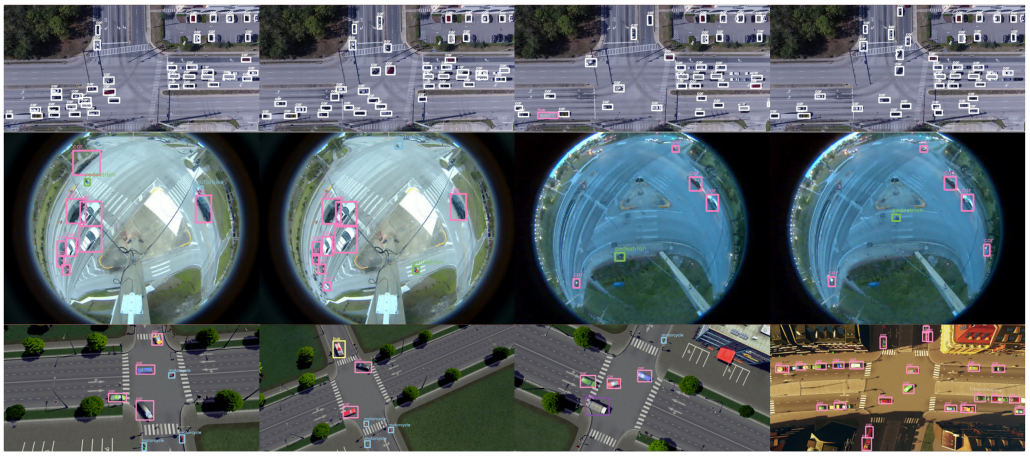

Fig. 10. Object-detection result samples of our spatial network on TNAD dataset. Top: samples for drone video; middle: samples for fisheye video; bottom: samples for simulation video.

conducted two experiments, one directly using these raw videos as input for our system and another in which we first preprocessed the video to correct for fisheye distortion and then fed them into our system. As the original surveillance video has many visual distortions, especially near the circular boundaries of the cameras, our system performed better on these after preprocessing. In this article, we do not discuss issues related to fisheye unwarping, leaving these for future work.

For large intersections, two fisheye cameras placed opposite to each other are used for surveillance, and each of them shows almost half the roads and real traffic. We apply a simple *object-level* stitching method by assigning the object identity for the same objects across the left and right video using similar features and appearing and vanishing positions.

4.3 Model Training

We adopt Darknet-19 [Wojke et al. 2017] for classification and detection with DeepSORT, using a data association metric that combines deep appearance features. We implement our framework on Tensorflow and perform multi-scale training and testing with a single GPU (Nvidia Titan X Pascal). Training a single spatial convolutional network takes one day on our system with one Nvidia Titan X Pascal card. For classification and detection training, we use the same training strategy as YOLO9000 [Wojke et al. 2017]. We train the network on our dataset with four classes of vehicle (motorbike, bus, car, and truck) for 160 epochs, using stochastic gradient descent with a starting learning rate of 0.1 for classification and 10^{-3} for detection (dividing it by 10 at 60 and 90 epochs.), weight decay of 0.0005, and momentum of 0.9 using the Darknet neural network framework [Wojke et al. 2017].

4.4 Qualitative Results

We present some example experimental results of object detection, MOT, and near-accident detection on our traffic near-accident dataset (TNAD) for drone videos, fisheye videos, and simulation videos. For object detection (Figure 10), we present some detection results of our spatial network with multi-scale training based on YOLOv2 [Redmon and Farhadi 2017]. These visual results demonstrate that the spatial stream is able to detect and classify road users with good accuracy even if the original fisheye videos have large distortions and occlusions. Different from drone and simulation video, we have added pedestrian detection on real traffic fisheye video. From

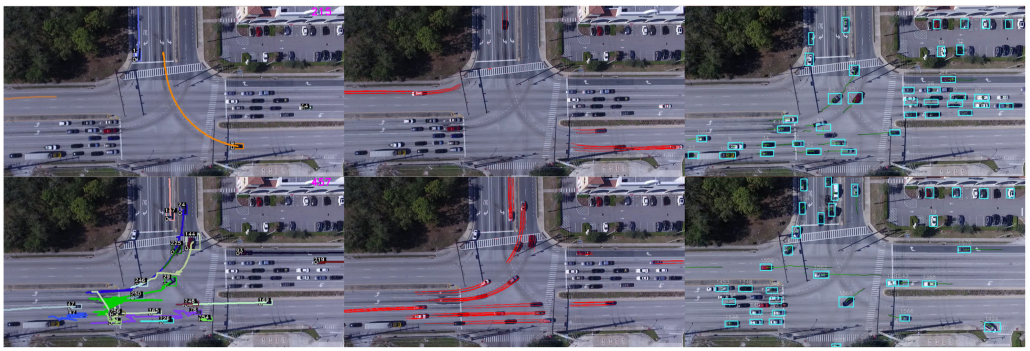

Fig. 11. Tracking and trajectory comparison with Urban Tracker [Jodoin et al. 2014] and TrafficIntelligence [Jackson et al. 2013] on drone videos of TNAD dataset. Left: tracking results of Urban Tracker [Jodoin et al. 2014] (BSG with Multilayer and Lobster Model; middle: tracking results of TrafficIntelligence [Jackson et al. 2013]; right: tracking results of our temporal stream network.

Figure 10, we see that our detector has good performance on pedestrian detection on daylight and dawn fisheye videos. The vehicle detection capabilities are good on top-down view surveillance videos, even for small objects. In addition, we can achieve a fast detection rate at 20 to 30 frames per second. Overall, this demonstrates the effectiveness of our spatial neural network.

For MOT (Figure 11), we present comparison of our temporal network based on DeepSORT [Wojke et al. 2017] with Urban Tracker [Jodoin et al. 2014] and TrafficIntelligence [Jackson et al. 2013]. We also present more visual MOT comparisons of our baseline temporal stream (DeepSort) with cosine metric learning-based temporal stream (Figure 12). For the tracking part, we use a tracking-by-detection paradigm, thus our methods can handle still objects and measure their state. This is especially useful, since Urban Tracker [Jodoin et al. 2014] and TrafficIntelligence [Jackson et al. 2013] can only track moving objects. However, Urban Tracker [Jodoin et al. 2014] and TrafficIntelligence [Jackson et al. 2013] can compute dense trajectories of moving objects with good accuracy, but they have slower tracking speed—around one frame per second. For accident detection, our two-stream convolutional networks are able to do spatial localization and temporal localization for diverse accident regions involving cars and motorbikes. The three subtasks (object detection, MOT, and near-accident detection) can always achieve real-time performance at a high frame rate—40 to 50 frames per second—and this depends on the frame resolution (e.g., 50 fps for 960×480 image frames).

In Figure 12, we demonstrate the effectiveness of applying cosine metric learning with the temporal stream for solving ID switching (and for the case where new object IDs emerge). For example, the first row shows that the ID of the green card (inside the red circle) got changed from 32 to 41. The third row shows that the IDs of the red car and a pedestrian (inside the red circle) were both changed (from 30 to 43 and from 38 to 44, respectively). The second and fourth row demonstrate that with cosine metric learning, the IDs of tracks are kept consistent. In Figure 13, we show an example of more accuracy and compact road user mask generated by superpixel segmentation and detections for the learning-based gap estimation. With a segmentation mask, we can estimate the gap distance between road users in a more accurate way than directly using object detections. Segmentation would be more useful for fisheye videos where the distortion is large and occlusion is heavier. For near-accident detection (Figure 14), we present final near-accident detection results along with tracking and trajectories using our two-stream convolutional networks method. Overall, the qualitative results demonstrate the effectiveness of our spatial and temporal networks.

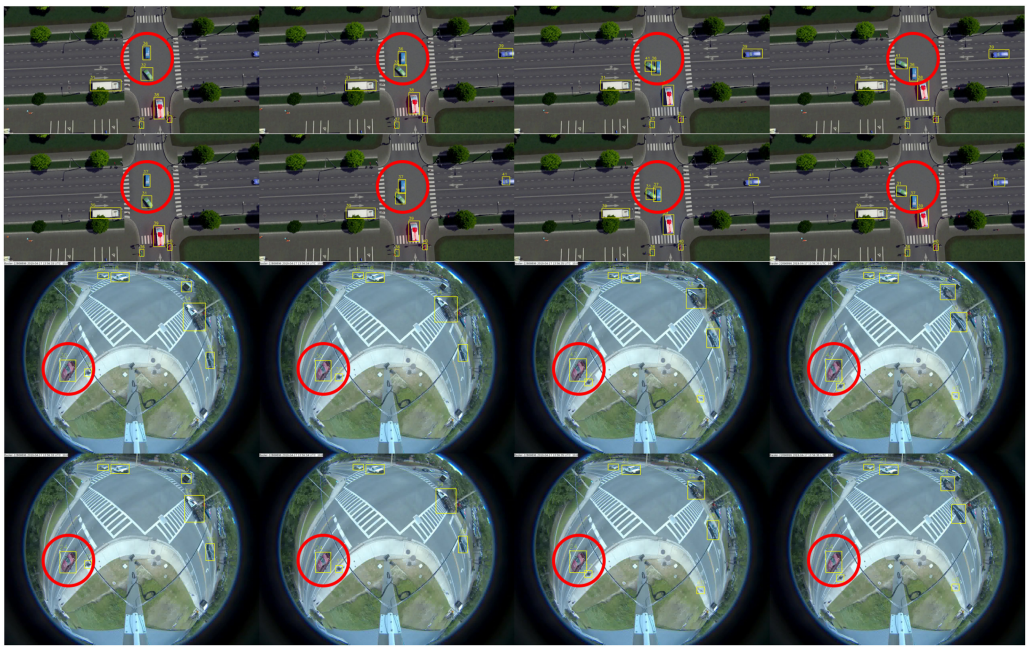

Fig. 12. Multiple-object-tracking comparisons of our baseline temporal stream (SORT) with cosine metric learning-based temporal stream. Row 1: results from SORT (ID switching issue); row 2: results from cosine metric learning + SORT (ID consistency); row 3: results from SORT (ID switching issue); row 4: results from cosine metric learning + SORT (ID consistency).

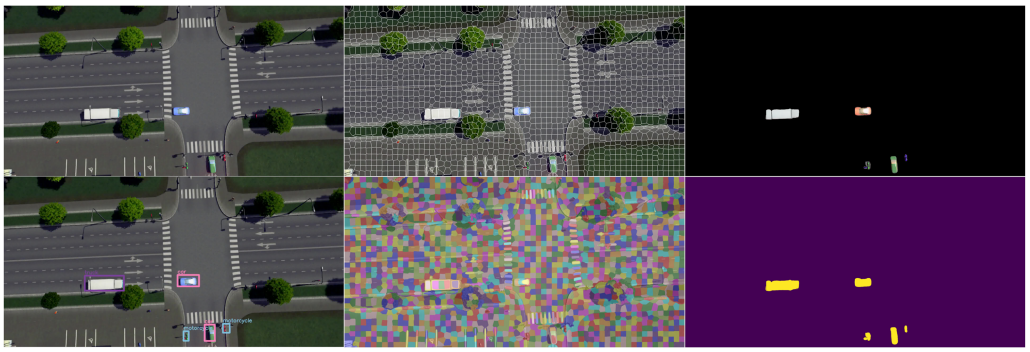

Fig. 13. Object segmentation using detections and superpixels. Top-left: original image; top-middle: superpixels generated by SLIC [Achanta et al. 2012]; top-right: final object mask; bottom-left: object detections; bottom-middle: colored superpixels generated by SLIC [Achanta et al. 2012]; bottom right: final binary object mask.

4.5 Quantitative Results

4.5.1 Speed Performance. We present timing results for the tested methods in Table 1. All experiments have been performed on a single GPU (NVIDIA TITAN V). The GPU-based SLIC segmentation [Achanta et al. 2012] has excellent speed and runs from 110 to 400 fps on videos of different resolutions. The overall two-stream CNNs (object detection, MOT, and near-accident

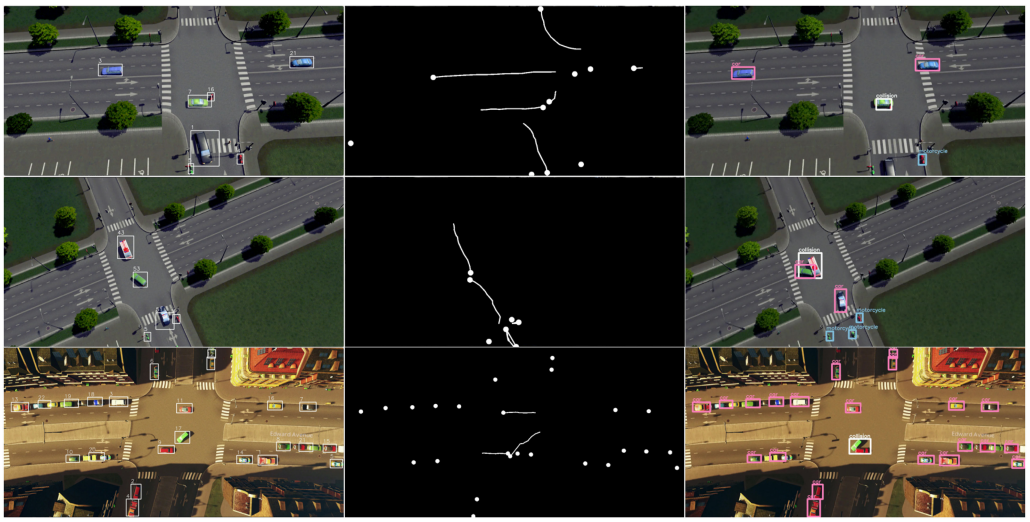

Fig. 14. Sample results of tracking, trajectory, and near-accident detection of our two-stream convolutional networks on simulation videos of TNAD dataset. Left: tracking results from the temporal stream; middle: trajectory results from the temporal stream; right: final near-accident detection results from the two-stream convolutional networks.

Table 1. Quantitative Evaluation of Timing Results for the Tested Methods

Methods	GPU	Drone	Simulation		Fisheye
		1,920 × 960	2,560 × 1,280	3,360 × 1,680	1,280 × 960
SLIC segmentation	NVIDIA TITAN V	300 fps	178 fps	110 fps	400 fps
Two-Stream CNNs	NVIDIA TITAN V	43 fps	38 fps	33 fps	50 fps

detection) can achieve real-time performance at a high frame rate, 33–50 fps, depending on the frame resolution.

4.5.2 Improved Multiple-object Tracking with Cosine Metric Learning. We present some quantitative results for gap estimation and cosine metric learning in Figure 15. The results for cosine metric learning have been established after training the network for a fixed number of steps (100,000 iterations). The batch size was set to 120 images and the learning rate was 0.001. All configurations after training have fully converged as depicted in Figure 15.

Triplet Loss. The triplet loss for cosine metric learning network [Weinberger and Saul 2009] is defined by

$$\mathcal{L}_{\mathrm{t}}(\boldsymbol{r}_a, \boldsymbol{r}_p, \boldsymbol{r}_n) = \left\{ \|\boldsymbol{r}_a - \boldsymbol{r}_n\|_2 - \|\boldsymbol{r}_a - \boldsymbol{r}_p\|_2 + m \right\}_+, \tag{11}$$

where $\boldsymbol{r}_a$, $\boldsymbol{r}_p$, and $\boldsymbol{r}_n$ are three samples in which a positive pair, $y_a = y_p$, and a negative pair, $y_a \neq y_n$ are included. With a pre-defined margin $m \in \mathbb{R}$, the triplet loss demands the distance between the positive and negative is larger than it. In this experiment, we introduce a soft-margin-based triplet loss where we replace the hinge of the original triplet loss [Hermans et al. 2017] by a soft plus function, $\{x + m\}_+ = \log(1 + \exp(x))$, to resolve non-smoothness [Rippel et al. 2015] issues. Further, to avoid potential issues in the sampling strategy, we directly generate the triplets on GPU as proposed by [Hermans et al. 2017].

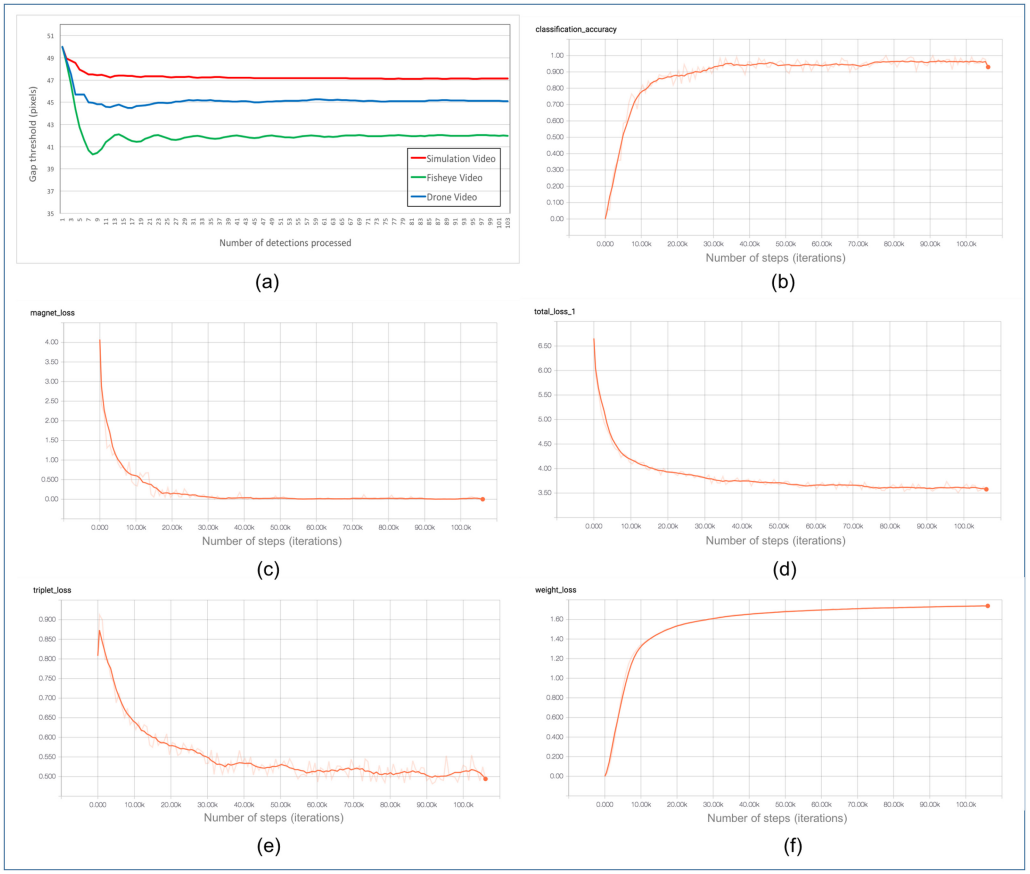

Fig. 15. Gap estimation and cosine metric learning. (a) Evolution of gap threshold for three types of videos; (b) classification accuracy for training VeRi dataset; (c) evolution of total loss for training; (d) evolution of magnet loss for training; (e) evolution of triplet loss for training; (f) evolution of weight loss for training.

Magnet Loss. The magnet loss is defined as a likelihood ratio measure that demands separation of all samples away from the means of other classes. Instead of using a multimodal form in its original proposition [Rippel et al. 2015], we use an unimodal, adapted version of this loss to fit the vehicle re-identification problem:

$$\mathcal{L}_{\mathrm{m}}(y, \boldsymbol{r}) = \left\{ -\log \frac{\mathrm{e}^{-\frac{1}{2\hat{\sigma}^2}\|\boldsymbol{r}-\hat{\boldsymbol{\mu}}_y\|_2^2 - m}}{\sum_{k\in\bar{C}(y)} \mathrm{e}^{-\frac{1}{2\hat{\sigma}^2}\|\boldsymbol{r}-\hat{\boldsymbol{\mu}}_k\|_2^2}} \right\}_+, \tag{12}$$

where $\bar{C}(y) = \{1, \ldots, C\} \setminus \{y\}$, and the m is the pre-defined margin parameter, $\hat{\boldsymbol{\mu}}_y$ is used to represent the sample mean of class y, and $\hat{\sigma}^2$ denotes the variance of each sample separated away from their class mean. They are all established for each batch individually on GPU.

4.5.3 Object Segmentation. For learning the gap threshold with superpixel segmentation and detections, the plot (top left) demonstrates the converging evolution of gap distance for three types of video. The initial gap threshold is set to be 50 pixels, the average gap thresholds from experiments are about 47 pixels, 45 pixels, and 42 pixels for simulation video, fisheye video, and drone video, respectively.

Table 2. Quantitative Evaluation of All Tasks for Our Two-stream Method

Task or Methods	Data	Precision	Recall	F1-score
Object Detection	Simulation	0.92670	0.95255	0.93945
	Fisheye	0.93871	0.87978	0.90829
Multiple-object Tracking	Simulation	0.89788	0.86617	0.88174
	Fisheye	0.91239	0.84900	0.87956
Multiple-object Tracking (cosine metric)	Simulation	0.91913	0.90310	0.91105
	Fisheye	0.92663	0.87725	0.90127
SLIC Segmentation Mask	Simulation	0.93089	0.81321	0.86808
Near-Accident Detection (Spatial Stream)	Simulation	0.90395	0.86022	0.88154
Near-Accident Detection (Temporal Stream)	Simulation	0.83495	0.92473	0.84108
Near-Accident Detection (Two-Stream)	Simulation	0.92105	0.94086	0.93085

The evaluation method we use for instance segmentation is quite similar to that of object detection, with the exception that we now calculate IoU of masks instead of bounding boxes.

Calculating mask precision, recall, and F1 score. To evaluate our collection of predicted masks, we will compare each of our predicted masks with each of the available target masks for a given input.

- A true positive is observed when a prediction-target mask pair has an IoU score that exceeds some predefined threshold (we set it to 0.7).
- A false positive indicates a predicted object mask had no associated ground-truth object mask.
- A false negative indicates a ground-truth object mask had no associated predicted object mask.

4.5.4 Precision, Recall, and F1 Score. Since our framework has three tasks and our dataset is quite different from other object-detection datasets, tracking datasets, and near-accident datasets such as dashcam accident dataset [Chan et al. 2016], it is difficult to compare the individual quantitative performance for all three tasks with other methods. One of our motivations was to propose a vision-based solution for ITS; therefore, we focus more on near-accident detection and present quantitative analysis of our two-stream convolutional networks. In Table 2, we present quantitative evaluations of three sub-tasks: object detection; multiple-object tracking (with and without cosine metric learning); SLIC segmentation (mask vs. bounding box); and near-accident detection (spatial stream only, temporal stream only, and two-stream model). For the fisheye video, we present evaluations regrading object detection and multiple-object tracking. The simulation videos are for the purposes of training and testing with more near-accident samples, and we have 57 simulation videos totaling over 51,123 video frames. We sparsely sample only 1,087 frames from them for training processing. We present the analysis of near-accident detection for 30 testing videos (18 had positive near-accident; 12 had negative near-accident).

Calculating Near-accident Detection Precision, Recall, and F1 Score. To evaluate our prediction of near-accidents, we'll compare each of our predicted detection with each ground truth for a given input.

- A true positive is observed when a prediction-target detection pair has an IoU score that exceeds some predefined threshold (we set it to 0.7).
- A false positive indicates a predicted near-accident had no associated ground-truth near-accident.
- A false negative indicates a ground-truth near-accident had no associated predicted detections.

5 RELATED WORK

5.1 Object Detection

Object detection has achieved striking improvements in recent years, as demonstrated in popular object-detection competitions such as PASCAL VOC detection challenge [Everingham et al. 2015, 2010], ILSVRC large-scale detection challenge [Russakovsky et al. 2015], and MS COCO large-scale detection challenge [Lin et al. 2014]. Object detection aims at outputting instances of semantic objects with a certain class label such as "humans" or "car." It has wide applications in many vision tasks, including pedestrian detection, face detection, facial recognition, video object co-segmentation, image retrieval, object tracking, and video surveillance. Different from image classification, object detection does not have the goal of classifying the whole image. Position and category information of the objects are both needed, which means we have to segment instances of objects from backgrounds and label them with position and class. The inputs are images or video frames while the outputs are lists where each item represents the position and category information of candidate objects. In general, object detection seeks to extract discriminative features to help in distinguishing the classes.

Methods for object detection generally fall into three categories: (1) traditional machine learning-based approaches; (2) region proposal-based deep-learning approaches; and (3) end-to-end deep-learning approaches. For traditional machine learning-based approaches, one of the important steps is to design features. Many methods have been proposed to first design features [Dalal and Triggs 2005; Lowe 1999; Viola and Jones 2001, 2004] and apply techniques such as support vector machine (SVM) [Hearst et al. 1998] to do the classification. The main steps of traditional machine learning-based approaches are:

- Region selection: using sliding windows at different sizes to select candidate regions from whole images or video frames
- Feature extraction: extract visual features from candidate regions using techniques such as Harr feature for face detection, HOG feature for pedestrian detection, or general object detection;
- Classifier: train and test classifier using techniques such as SVM.

Traditional machine learning-based approaches have their limitations. The scheme using sliding windows to select RoIs (Regions of Interest) increases computation time with many window redundancies. However, these hand-crafted features are not robust, due to the diversity of objects, deformation, lighting condition, background, and so on, while the feature selection has a huge effect on classification performance of candidate regions.

Recent advances in deep learning, especially in computer vision, have shown that CNNs have a strong capability of representing objects and help to boost the performance of numerous vision tasks, compared to traditional heuristic features [Dalal and Triggs 2005]. For deep-learning-based approaches, there are CNNs to extract features of region proposals or end-to-end object detection without specifically defining features of a certain class. Well-performing deep-learning-based approaches of object detection include R-CNN [Girshick et al. 2014], Fast R-CNN [Girshick 2015], Faster R-CNN [Ren et al. 2015]), SSD [Liu et al. 2016a], and YOLO [Redmon et al. 2016].

Usually, we adopt region proposal methods (Category 2) to produce multiple-object proposals, and then apply a robust classifier to further refine the generated proposals; this is referred to as a two-stage method. The first example of a region-proposal-based deep-learning approach was R-CNN [Girshick et al. 2014]. The main pipeline of R-CNN [Girshick et al. 2014] is (1) gathering input images, (2) generating a number of region proposals (e.g., 2,000), (3) extracting CNN features, and (4) classifying regions using SVM. It usually adopts selective search (SS), one of the state-of-art

object proposal methods [Uijlings et al. 2013], which is applied in numerous detection tasks on several fascinating systems [Girshick 2015; Girshick et al. 2014; Ren et al. 2015] to extract these regions from an image and designate region proposals. Instead of trying to classify all the possible proposals, R-CNN selects a fixed set of proposals (e.g., 2,000) to work with. The selective search algorithm used to generate these region proposals includes (1) generate initial sub-segmentation, generate many candidate regions, (2) use a greedy algorithm to recursively combine similar regions into larger ones, and (3) use the generated regions to produce the final candidate region proposals.

These candidate region proposals are warped into a square and fed into a CNN, which acts as the feature extractor. The resulting dense layer consists of the extracted features, which are fed into an SVM [Hearst et al. 1998] to classify the presence of the object within that candidate region proposal. The main problem of R-CNN [Girshick et al. 2014] is that it is limited by the inference speed, due to a huge amount of time spent extracting features of each individual region proposal. And it cannot be applied in applications requiring real-time performance (such as online video analysis). Later, Fast R-CNN [Girshick 2015] was proposed to improve the speed by avoiding feeding raw region proposals every time. Instead, the convolution operation is done only once per image, and RoIs over the feature map are generated. Faster-R-CNN [Ren et al. 2015] further exploits the shared convolutional features to extract region proposals used by the detector. Sharing convolutional features leads to substantially faster speed for the object-detection system.

The third type is the end-to-end deep-learning approach, which does not need region proposals (also referred to as one-stage method). The pioneering works are SSD [Liu et al. 2016a] and YOLO [Redmon et al. 2016]. An SSD detector [Liu et al. 2016a] works by adding a sequence of feature maps of progressively decreasing spatial resolution to replace the two stage's second classification stage, allowing a fast computation and multi-scale detection on one single input. YOLO detector is an object-detection algorithm much different from the region-based algorithms. In YOLO [Redmon et al. 2016], object detection is regarded as an end-to-end regression problem and uses a single convolutional network to predict the bounding boxes and the corresponding class probabilities. It first takes the image and splits it into a $S \times S$ grid; within each cell of the grid, we take m bounding boxes. For each bounding box with multiple scales, the convolutional neural network outputs a class probability and offset values for the bounding box. Then, it selects bounding boxes that have the class probability above a threshold value and uses them to locate the object within the image. YOLO [Redmon et al. 2016] is orders of magnitude faster (45 frames per second) than other object-detection approaches, but the limitation is that it struggles with small objects within the image.

5.2 Multiple-object Tracking

Video object tracking attempts to locate objects in successive video frames, and it has various important applications in robotics, video surveillance, and video scene understanding. Based on the number of moving objects that we wish to track, there is the single-object tracking (SOT) problem and the multiple-object-tracking (MOT) problem. In addition to detecting objects in the video frame, the MOT solution requires robust association of multiple detected objects between frames to get consistent tracking, and this data association part remains very challenging. In MOT tasks, for each frame in a video, we aim at localizing and identifying all objects of interest, so that the identities are consistent throughout the video. Typically, the main challenges are speed, data association, appearance change, occlusions, disappear and re-entry of objects, and so on. In practice, it is desirable that the tracking be performed in real time, running as fast as the frame-rate of the video. Also, it is challenging to provide consistent labeling of the detected objects in complex scenarios such as when objects change appearance, disappear, or involve severe occlusions.

MOT approaches can be categorized by their different models. A distinction based on *initialization method* is that of detection-based tracking (DBT) versus detection-free tracking (DFT). In DBT, object detection is performed on video frames before tracking. DBT methods involve two distinct tasks: detection of objects and tracking of objects. In this article, we focus on DBT (also referred to as tracking-by-detection for MOT). The reason is that DBT methods are widely used due to their excellent performance with deep-learning-based object detectors, while DFT methods require manual annotations of the targets, which could yield poor results if a new, previously unseen object appears. Another important distinction based on *processing mode* is that of online versus offline models. An online model receives video input on a frame-by-frame basis and gives output per frame. This means that only information from past frames and the current frame can be used. Offline models have access to the entire video, which means that information from both past and future frames can be used.

Tracking-by-detection methods are usually utilized in online tracking models. A simple and classic pipeline is (1) detect objects of interest, (2) predict new locations of objects from previous frames, and (3) associate objects between frames by similarity of detected and predicted locations. Well-performed CNN architectures, such as Faster R-CNN [Ren et al. 2015], YOLO [Redmon et al. 2016], and SSD [Liu et al. 2016a], can be used for object detection. For prediction of new locations of tracked objects, approaches model the velocity of objects and predict their positions in future frames using optical flow, recurrent neural networks, or Kalman filters. The association task is to determine whether a detection is associated with a previously detected object or a new object.

One popular dataset for MOT is MOTChallenge [Leal-Taixé et al. 2015]. In MOTChallenge [Leal-Taixé et al. 2015], detections for each frame are provided in the dataset, and the tracking capability is measured, as opposed to the detection quality. Video sequences are labeled with bounding boxes for each pedestrian, collected from multiple sources. This motivates the use of the tracking-by-detection paradigm. MDPs [Xiang et al. 2015] is a tracking-by-detection method that achieved state-of-the-art performance on the MOTChallenge [Leal-Taixé et al. 2015] Benchmark when it was proposed. Major contributions can be solving MOT by learning a MDP policy taking advantages of both offline-learning and online-learning for data association. SORT [Bewley et al. 2016] is a simple, real-time MOT method, where it only applies classical tracking methods to achieve state-of-the-art performance. It is the most widely used real-time online MOT method and is very efficient for real-time application in practice. On the MOTChallenge [Leal-Taixé et al. 2015], SORT [Bewley et al. 2016] achieves better performance than MHT [Kim et al. 2015] on standard detections by combining with a well-performed person detector. DeepSort [Wojke et al. 2017] is an extension of SORT [Bewley et al. 2016] that further improves the performance.

5.3 Near-accident Detection

In addition to vehicle detection and vehicle tracking, interaction and behavior analysis of vehicles play an important role in traffic modeling [Hermes et al. 2009; Sivaraman et al. 2011; Wiest et al. 2012]. Near-accident detection is one of the highest levels of semantic interpretation in characterizing the interactions of vehicles on the road. The basic task of near-accident detection is to locate near-accident regions and report them over video frames. To detect near-accidents in traffic scenes, robust vehicle detection and vehicle tracking are prerequisites.

Most of the near-accident detection approaches are based on motion cues and trajectories. The most typical motion cues are optical flow and trajectory. Optical flow is widely utilized in video processing tasks such as video segmentation [Huang et al. 2018]. In recent years, researchers have worked on long-term vehicle motion prediction. Based on tracking algorithms such as Kalman filtering, we can estimate future vehicle motion state. Vehicle motion can be estimated for a longer

duration with trajectory modeling approaches [Hermes et al. 2009; Sivaraman et al. 2011; Wiest et al. 2012]. Typical highway trajectories are modeled using clustering and classified via hidden Markov modeling in Sivaraman et al. [2011]. In Hermes et al. [2009], it uses the rotation-invariant-based longest common subsequence to measure the similarity between trajectories. Wiest et al. [2012] classifies and predicts long-term vehicles trajectories by applying variational Gaussian mixture modeling.

Approaches such as decision trees, Kalman filters, or time-series analysis have been applied in traffic accident or near-accident detection [Bhonsle et al. 2000; Jiansheng et al. 2014; Shuming et al. 2002; Srinivasan et al. 2001, 2003; Xu et al. 1998]. Neural network is utilized in Ohe et al. [1995] to detect traffic incidents. In Ikeda et al. [1999], the authors propose a system to classify different types of incidents for automatic incident detection. Kimachi et al. [1994] investigated the abnormal behavior of vehicles related to accidents, based on the concepts of fuzzy theory, where accident occurrence is related to the behavioral abnormality of multiple continual images. Zeng et al. [2008] integrates multi-class SVMs with D-S evidence theory data fusion. Sadeky et al. [2010] presents a real-time method using trajectory and histogram of flow gradient (HFG). Kamijo et al. [2000] develops an event recognition system based on the hidden Markov model (HMM), which is robust for traffic monitoring and accident detection. Chen et al. [2010] proposed a SVM-based method on traffic flow measurement. A similar approach using BP-ANN for accident detection has been proposed in Ghosh-Dastidar and Adeli [2003] and Srinivasan et al. [2004]. Road-user interaction are analyzed using a refined probabilistic framework, where collision probabilities are estimated by identifying potential collision points. Other methods for traffic accident detection have also been presented using matrix approximation [Xia et al. 2015], optical flow and scale-invariant feature transform (SIFT) [Chen et al. 2016], smoothed particles hydrodynamics (SPH) [Ullah et al. 2015], and adaptive traffic motion flow modeling [Maaloul et al. 2017]. With advances in object detection with deep neural networks, several automatic traffic accident detection methods based on CNNs [Singh and Mohan 2018; Sultani et al. 2018] and traffic accident anticipation methods based on recurrent neural networks (RNNs) [Chan et al. 2016; Suzuki et al. 2018] have been proposed along with traffic accident datasets [Kataoka et al. 2018; Shah et al. 2018; Sultani et al. 2018; Suzuki et al. 2018] of surveillance videos or dashcam videos [Chan et al. 2016]. However, these methods either do not have real-time performances for online accident detection without using future frames or give unsatisfactory results. Besides that, no proposed dataset contains videos with top-down views such as drone or unmanned aerial vehicle (UAVs) videos or omnidirectional camera videos for traffic analysis.

6 CONCLUSION

We have proposed a two-stream convolutional network architecture that performs real-time detection, tracking, and near-accident detection for vehicles in traffic video data. The two-stream convolutional network comprises a spatial stream network and a temporal stream network. The spatial stream network detects individual vehicles and likely near-accident regions at the single-frame level by capturing appearance features with a state-of-the-art object-detection method. The temporal stream network leverages motion features of detected candidates to perform multiple-object tracking and generates individual trajectories of each tracked target. We detect near-accidents by incorporating appearance features and motion features to compute probabilities of near-accident candidate regions. Experiments have demonstrated the advantage of our framework with an overall competitive qualitative and quantitative performance at high frame rates. Future work will include image stitching methods that will be deployed on multi-camera fisheye videos.

ACKNOWLEDGMENT

The authors thank the City of Gainesville for access to the fisheye video data that was used in this article.

REFERENCES

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. 2012. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 11 (2012), 2274–2282.

Alejandro Angel, Mark Hickman, Pitu Mirchandani, and Dmesh Chandnani. 2002. Methods of traffic data collection, using aerial video. In *Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems*. IEEE, 31–36.

Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. 2010. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 5 (2010), 898–916.

Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple online and real-time tracking. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'16)*. IEEE, 3464–3468.

Shailendra Bhonsle, Mohan Trivedi, and Amarnath Gupta. 2000. Database-centered architecture for traffic incident detection, management, and analysis. In *Proceedings of the IEEE Intelligent Transportation Systems (ITSC'00)*. IEEE, 149–154.

Norbert Buch, Sergio A. Velastin, and James Orwell. 2011. A review of computer vision techniques for the analysis of urban traffic. *IEEE Trans. Intell. Transport. Syst.* 12, 3 (2011), 920–939.

Fu-Hsiang Chan, Yu-Ting Chen, Yu Xiang, and Min Sun. 2016. Anticipating accidents in dashcam videos. In *Proceedings of the Asian Conference on Computer Vision*. Springer, 136–153.

Lairong Chen, Yuan Cao, and Ronghua Ji. 2010. Automatic incident detection algorithm based on support vector machine. In *Proceedings of the 6th International Conference on Natural Computation*, Vol. 2. IEEE, 864–866.

Yu Chen, Yuanlong Yu, and Ting Li. 2016. A vision-based traffic accident detection method using extreme learning machine. In *Proceedings of the International Conference on Advanced Robotics and Mechatronics (ICARM'16)*. IEEE, 567–572.

Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. 1998. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transport. Res. Part C: Emerg. Technol.* 6, 4 (1998), 271–288.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1. IEEE, 886–893.

Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. 2015. The PASCAL visual object classes challenge: A retrospective. *Int. J. Comput. Vision* 111, 1 (2015), 98–136.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. 2010. The PASCAL visual object classes (VOC) challenge. *Int. J. Comput. Vision* 88, 2 (2010), 303–338.

Samanwoy Ghosh-Dastidar and Hojjat Adeli. 2003. Wavelet-clustering-neural network model for freeway incident detection. *Comput.-Aided Civil Infrastruct. Eng.* 18, 5 (2003), 325–338.

Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*. 1440–1448.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 580–587.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2016. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 1 (2016), 142–158.

Pan He, Weilin Huang, Tong He, Qile Zhu, Yu Qiao, and Xiaolin Li. 2017. Single shot text detector with regional attention. In *Proceedings of the IEEE International Conference on Computer Vision*. 3047–3055.

Pan He, Aotian Wu, Xiaohui Huang, Jerry Scott, Anand Rangarajan, and Sanjay Ranka. 2019a. Deep-learning-based geometric features for effective truck selection and classification from highway videos. In *Proceedings of the International IEEE Conference on Intelligent Transportation Systems (ITSC'19)*. IEEE, 824–830.

Pan He, Aotian Wu, Anand Rangarajan, and Sanjay Ranka. 2019b. *Truck Taxonomy and Classification Using Video and Weigh-In Motion (WIM) Technology Final Report*. Technical Report. Final Research Report Prepared for: Florida Department of Transportation.

Marti A. Hearst, Susan T. Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intell. Syst. Appl.* 13, 4 (1998), 18–28.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.

Christoph Hermes, Christian Wohler, Konrad Schenk, and Franz Kummert. 2009. Long-term vehicle motion prediction. In *Proceedings of the IEEE Intelligent Vehicles Symposium*. IEEE, 652–657.

Stefan Hommes, Radu State, Andreas Zinnen, and Thomas Engel. 2011. Detection of abnormal behaviour in a surveillance environment using control charts. In *Proceedings of the 8th IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS'11)*. IEEE, 113–118.

Neil Hoose. 1992. IMPACTS: An image analysis tool for motorway surveillance. *Traffic Eng. Control* 33, 3 (1992).

Xiaohui Huang, Chengliang Yang, Sanjay Ranka, and Anand Rangarajan. 2018. Supervoxel-based segmentation of 3D imagery with optical flow integration for spatiotemporal processing. *IPSJ Trans. Comput. Vision Appl.* 10, 1 (2018), 9.

Nacim Ihaddadene and Chabane Djeraba. 2008. Real-time crowd motion analysis. In *Proceedings of the 19th International Conference on Pattern Recognition*. IEEE, 1–4.

Hiromi Ikeda, Yukihiro Kaneko, Toshihiro Matsuo, and Kunihiko Tsuji. 1999. Abnormal incident detection system employing image processing technology. In *Proceedings of the IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*. IEEE, 748–752.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Stewart Jackson, Luis F. Miranda-Moreno, Paul St-Aubin, and Nicolas Saunier. 2013. Flexible, mobile video camera system and open source video analysis software for road safety and behavioral analysis. *Transport. Res. Rec.* 2365, 1 (2013), 90–98.

Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. 2018. Superpixel sampling networks. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 352–368.

Fan Jiang, Ying Wu, and Aggelos K. Katsaggelos. 2007. Abnormal event detection from surveillance video by dynamic hierarchical clustering. In *Proceedings of the IEEE International Conference on Image Processing*, Vol. 5. IEEE, V–145.

Fu Jiansheng et al. 2014. Vision-based real-time traffic accident detection. In *Proceedings of the 11th World Congress on Intelligent Control and Automation*. IEEE, 1035–1038.

Jean-Philippe Jodoin, Guillaume-Alexandre Bilodeau, and Nicolas Saunier. 2014. Urban tracker: Multiple-object tracking in urban mixed traffic. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*. IEEE, 885–892.

Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. *J. Basic Eng.* 82, 1 (1960), 35–45.

Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. 2000. Traffic monitoring and accident detection at intersections. *IEEE Trans. Intell. Transport. Syst.* 1, 2 (2000), 108–118.

Asim Karim and Hojjat Adeli. 2002. Incident detection algorithm using wavelet energy representation of traffic patterns. *J. Transport. Eng.* 128, 3 (2002), 232–242.

Hirokatsu Kataoka, Teppei Suzuki, Shoko Oikawa, Yasuhiro Matsui, and Yutaka Satoh. 2018. Drive video analysis for the detection of traffic near-miss incidents. *arXiv preprint arXiv:1804.02555*.

Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M. Rehg. 2015. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*. 4696–4704.

Masatoshi Kimachi, Kenji Kanayama, and Kenbu Teramoto. 1994. Incident prediction by fuzzy image sequence analysis. In *Proceedings of the Vehicle Navigation and Information Systems Conference (VNIS'94)*. IEEE, 51–56.

Dieter Koller, Kostas Daniilidis, and Hans-Hellmut Nagel. 1993. Model-based object tracking in monocular image sequences of road traffic scenes. *Int. J. Comput.* 10, 3 (1993), 257–281.

Alex Krizhevsky, Geoffrey Hinton et al. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart.* 2, 1-2 (1955), 83–97.

Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. 2015. MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision*. Springer, 740–755.

Chang Liu, Guijin Wang, Wenxin Ning, Xinggang Lin, Liang Li, and Zhou Liu. 2010. Anomaly detection in surveillance video using motion direction statistics. In *Proceedings of the IEEE International Conference on Image Processing*. IEEE, 717–720.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016a. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*. Springer, 21–37.

Xinchen Liu, Wu Liu, Huadong Ma, and Huiyuan Fu. 2016b. Large-scale vehicle re-identification in urban surveillance videos. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'16)*. IEEE, 1–6.

David G. Lowe. 1999. Object recognition from local scale-invariant features. In *The Proceedings of the 7th IEEE International Conference on Computer Vision*, Vol. 2. IEEE, 1150–1157.

Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, Xiaowei Zhao, and Tae-Kyun Kim. 2014. Multiple-object tracking: A literature review. *arXiv preprint arXiv:1409.7618*.

Boutheina Maaloul, Abdelmalik Taleb-Ahmed, Smail Niar, Naim Harb, and Carlos Valderrama. 2017. Adaptive video-based algorithm for accident detection on highways. In *Proceedings of the 12th IEEE International Symposium on Industrial Embedded Systems (SIES'17)*. IEEE, 1–6.

Philip McLauchlan, David Beymer, Benn Coifman, and Jitendra Malik. 1997. A real-time computer vision system for measuring traffic parameters. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 495–501.

Iwao Ohe, Hironao Kawashima, Masahiro Kojima, and Yukihiro Kaneko. 1995. A method for automatic detection of traffic incidents using neural networks. In *Proceedings of the 6th International Pacific Rim TransTech Conference: Vehicle Navigation and Information Systems (VNIS'95)*. IEEE, 231–235.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779–788.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7263–7271.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. MIT Press, 91–99.

Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. 2015. Metric learning with adaptive density discrimination. *arXiv preprint arXiv:1511.05939*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein et al. 2015. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision* 115, 3 (2015), 211–252.

Samy Sadeky, Ayoub Al-Hamadiy, Bernd Michaelisy, and Usama Sayed. 2010. Real-time automatic traffic accident recognition using hfg. In *Proceedings of the 20th International Conference on Pattern Recognition*. IEEE, 3348–3351.

Tim Salimans and Durk P. Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*. MIT Press, 901–909.

Giuseppe Salvo, Luigi Caruso, Alessandro Scordo, Giuseppe Guido, and Alessandro Vitale. 2017. Traffic data acquirement by unmanned aerial vehicle. *Eur. J. Remote Sens.* 50, 1 (2017), 343–351.

G. Scotti, L. Marcenaro, C. Coelho, F. Selvaggi, and C. S. Regazzoni. 2005. Dual camera intelligent sensor for high definition 360 degrees surveillance. *IEE Proc.-Vision Image Signal Process.* 152, 2 (2005), 250–257.

Ankit Shah, Jean Baptiste Lamare, Tuan Nguyen Anh, and Alexander Hauptmann. 2018. Accident forecasting in CCTV traffic camera videos. *arXiv preprint arXiv:1809.05782*.

Tang Shuming, Gong Xiaoyan, and Wang Feiyue. 2002. Traffic incident detection algorithm based on non-parameter regression. In *Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems*. IEEE, 714–719.

Dinesh Singh and Chalavadi Krishna Mohan. 2018. Deep spatio-temporal representation for detection of road accidents using stacked autoencoder. *IEEE Trans. Intell. Transport. Syst.* 20, 3 (2018), 879–887.

Sayanan Sivaraman, Brendan Morris, and Mohan Trivedi. 2011. Learning multi-lane trajectories using vehicle-based vision. In *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV'11)*. IEEE, 2070–2076.

Dipti Srinivasan, Ruey Long Cheu, and Young Peng Poh. 2001. Hybrid fuzzy logic-genetic algorithm technique for automated detection of traffic incidents on freeways. In *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC'01)*. IEEE, 352–357.

Dipti Srinivasan, Xin Jin, and Ruey Long Cheu. 2004. Evaluation of adaptive neural network models for freeway incident detection. *IEEE Trans. Intell. Transport. Syst.* 5, 1 (2004), 1–11.

Dipti Srinivasan, Wee Hoon Loo, and Ruey Long Cheu. 2003. Traffic incident detection using particle swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS'03)*. IEEE, 144–151.

Waqas Sultani, Chen Chen, and Mubarak Shah. 2018. Real-world anomaly detection in surveillance videos. *Center for Research in Computer Vision, University of Central Florida* (2018).

Tomoyuki Suzuki, Hirokatsu Kataoka, Yoshimitsu Aoki, and Yutaka Satoh. 2018. Anticipating traffic accidents with adaptive loss and large-scale incident db. *arXiv preprint arXiv:1804.02675*.

Shuming Tang and Haijun Gao. 2005. Traffic-incident detection-algorithm based on nonparametric regression. *IEEE Trans. Intell. Transport. Syst.* 6, 1 (2005), 38–42.

Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. 2016. Detecting text in natural image with connectionist text proposal network. In *Proceedings of the European Conference on Computer Vision*. Springer, 56–72.

Jasper R. R. Uijlings, Koen E. A. Van De Sande, Theo Gevers, and Arnold W. M. Smeulders. 2013. Selective search for object recognition. *Int. J. Comput. Vision* 104, 2 (2013), 154–171.

Habib Ullah, Mohib Ullah, Hina Afridi, Nicola Conci, and Francesco G. B. De Natale. 2015. Traffic accident detection through a hydrodynamic lens. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, 2470–2474.

Maria Valera and Sergio A. Velastin. 2005. Intelligent distributed surveillance systems: A review. *IEE Proc.-Vision Image Signal Process.* 152, 2 (2005), 192–204.

Harini Veeraraghavan, Osama Masoud, and Nikolaos P. Papanikolopoulos. 2003. Computer vision algorithms for intersection monitoring. *IEEE Trans. Intell. Transport. Syst.* 4, 2 (2003), 78–89.

Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1. IEEE, 3.

Paul Viola and Michael J. Jones. 2004. Robust real-time face detection. *Int. J. Comput. Vision* 57, 2 (2004), 137–154.

Lijun Wang and Ming Dong. 2012. Real-time detection of abnormal crowd behavior using a matrix approximation-based approach. In *Proceedings of the 19th IEEE International Conference on Image Processing*. IEEE, 2701–2704.

Ming-Liang Wang, Chi-Chang Huang, and Huei-Yung Lin. 2006. An intelligent surveillance system based on an omnidirectional vision sensor. In *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*. IEEE, 1–6.

Shu Wang and Zhenjiang Miao. 2010. Anomaly detection in crowd scene. In *Proceedings of the IEEE 10th International Conference on Signal Processing Proceedings*. IEEE, 1220–1223.

Kilian Q. Weinberger and Lawrence K. Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.* 10, Feb (2009), 207–244.

Jürgen Wiest, Matthias Höffken, Ulrich Kreßel, and Klaus Dietmayer. 2012. Probabilistic trajectory prediction with gaussian mixture models. In *Proceedings of the IEEE Intelligent Vehicles Symposium*. IEEE, 141–146.

Nicolai Wojke and Alex Bewley. 2018. Deep cosine metric learning for person re-identification. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV'18)*. IEEE, 748–756.

Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple online and real-time tracking with a deep association metric. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'17)*. IEEE, 3645–3649.

Siyu Xia, Jian Xiong, Ying Liu, and Gang Li. 2015. Vision-based traffic accident detection using matrix approximation. In *Proceedings of the 10th Asian Control Conference (ASCC'15)*. IEEE, 1–5.

Yu Xiang, Alexandre Alahi, and Silvio Savarese. 2015. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE International Conference on Computer Vision*. 4705–4713.

Hongli Xu, CM Kwan, L. Haynes, and J. D. Pryor. 1998. Real-time adaptive on-line traffic incident detection. *Fuzzy Sets and Systems* 93, 2 (1998), 173–183.

Yupeng Yan, Xiaohui Huang, Anand Rangarajan, and Sanjay Ranka. 2018. Densely labeling large-scale satellite images with generative adversarial networks. In *Proceedings of the IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech'18)*. IEEE, 927–934.

Liu Yu, Lei Yu, Jianquan Wang, Yi Qi, and Huimin Wen. 2008. Back-propagation neural network for traffic incident detection based on fusion of loop detector and probe vehicle data. In *Proceedings of the 4th International Conference on Natural Computation*, Vol. 3. IEEE, 116–120.

Dehuai Zeng, Jianmin Xu, and Gang Xu. 2008. Data fusion for traffic incident detection using DS evidence theory with probabilistic SVMs. *J. Comput.* 3, 10 (2008), 36–43.

Received December 2018; revised June 2019; accepted November 2019