Machine Learning based Video Processing for Real-time Near-Miss Detection

Xiaohui Huang, Tania Banerjee, Ke Chen, Naga Venkata Sai Varanasi, Anand Rangarajan and Sanjay Ranka

Modern Artificial Intelligence and Learning Technologies (MALT) Lab, University of Florida, U.S.A.

Keywords: Near-Miss Detection, Fisheye Camera, Intersection Video, Calibration, Thin-plate Spline, Deep Learning.

Abstract: Video-based sensors are ubiquitous and are therefore indispensable in understanding traffic behavior at intersections. Deriving near-misses from large scale video processing is extremely useful in assessing the level of safety of intersections. In this paper, we develop real-time or near real-time algorithms for detecting near-misses for intersection video collected using fisheye cameras. We propose a novel method consisting of the following steps: 1) extracting objects and multiple object tracking features using convolutional neural networks; 2) densely mapping object coordinates to an overhead map; 3) learning to detect near-misses by new distance measures and temporal motion. The experimental results demonstrate the effectiveness of our approach with real-time performance at 40 fps and high specificity.

1 INTRODUCTION

The advent of nominally priced video-based systems, open source tools for video processing and deep learning, and the availability of low-cost GPU processors have opened the door for their use in real-time transportation decision systems. While video-based systems for intersection traffic measurement can perform multiple object detection and tracking, their use for more complex tasks such as anomaly detection and near-misses is limited. The recent proposed AI city challenge (Tang et al., 2019) also focuses on similar applications. In general, monitoring activities of road users and understanding traffic events have shown to be useful for modeling, analyzing and improving road-based transportation.

In order to derive intersection scenes with wider angles, omnidirectional fisheye cameras are widely installed and used for street video surveillance [also known as closed-circuit television (CCTV)]. It is non-trivial to directly apply learning-based methods to detect near-misses in fisheye videos as they suffer from two types of distortions: fisheye lens distortion and perspective distortion. Due to both these distortions, road users (pedestrians, cars, etc.) can appear to be very close to each other in image space and to the human eye while remaining far apart in the physical world. Figure 1 illustrates a real fisheye traffic scene and one false near-miss case that can easily mislead.

Figure 1: Illustration of near-Miss Detection Problem.

The focus of our work is on building a platform that allows one to collect sufficient samples and visual cues corresponding to near-misses, intending to detect and even anticipate dangerous scenarios in real-time so that appropriate preventive steps can be undertaken. In particular, we focus on near-miss problems from large-scale intersection videos collected from fisheye cameras. The goal is to temporally and

Huang, X., Banerjee, T., Chen, K., Varanasi, N., Rangarajan, A. and Ranka, S.
Machine Learning based Video Processing for Real-time Near-Miss Detection.
DOI: 10.5220/0009345401690179
In *Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2020)*, pages 169-179
ISBN: 978-989-758-419-0

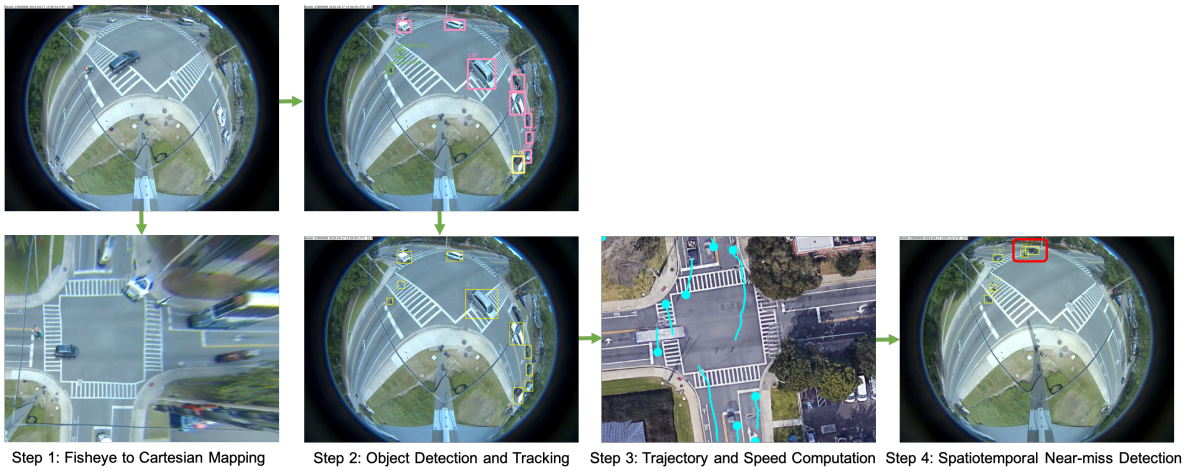

Figure 2: The Pipeline Overview of the Proposed Framework.

spatially localize and recognize near-miss cases from fisheye video. The main motivation for resolving distortion instead of using original fisheye videos are to compute accurate distance among objects as well as their accurate speeds using rectangular coordinates that better represent the real-world. The projections are done on an overhead satellite map of the intersection. We specify five categories of objects of interest: pedestrians, motorbikes, cars, buses and trucks. The overhead satellite maps of intersections are derived from Google Earth®. The main steps of our detection framework (Figure 2) can be summarized as follows:

1. *Fisheye to Cartesian Mapping:* We first apply camera calibration methods on a fisheye background image (with no road objects) to make an initial correction. We take the calibrated image as the target image, an overhead satellite map as the reference image and select corresponding landmark points in both images for mapping. Given these landmark points, we adopt the thin-plate spline (TPS) (Bookstein, 1989; Chui and Rangarajan, 2003) as basis function for coordinate mappings from the reference to the target and store the point-to-point outputs.
2. *Object Detection and Multiple Object Tracking:* We train an object detector using deep learning techniques and design a vehicle re-identification model with deep cosine metric learning to handle occlusion problems. We integrate these two models into our multiple-object tracking pipeline. Given fisheye videos as the input, the framework supports real-time object detection and multiple object tracking.
3. *Trajectory and Speed Computation:* Using the point-to-point TPS mappings, we correct and scale road object trajectories and speed information from the perspective of the overhead satellite map with learned deep features. As the complexity of coordinates transfer is $O(1)$, it allows us to process data both online and offline.
4. *Spatial and Temporal Near-Miss Detection* We define two scenarios for near-misses in videos: 1) spatial scenario: proximity of road objects in image space, 2) temporal scenario: a dramatic speed decrease to avoid near-misses (a sudden break). We use the distance-based and speed-based measures to compute the near-miss probabilities of road objects and aggregate scores via averaging as the final output.

The main contributions of this paper can be summarized as follows:

- We propose a novel method that combines distance measures and temporal motion to detect near-misses in fisheye traffic video.
- We propose a combined calibration and spline-based mapping method that maps fisheye video features to an overhead map to correct fisheye lens distortion and camera perspective distortion.
- We present a unified approach that performs real-time object recognition, multiple object tracking, and near-miss detection in fisheye video.
- We show a promising pipeline to be customized to several fisheye video understanding applications such as accident anticipation, anomaly detection, and trajectory prediction.

We have obtained accurate and real-time object detection and multiple object tracking results for long-duration video. The object detector has good performance to localize and classify road objects even

for tiny pedestrians. With cosine metric learning, the tracker generates more consistent and robust tracks and trajectories. With aid of calibration and TPS mapping, the location and speed information of objects has been corrected and scaled to a large extent. Compared to non-mapping based methods, the experimental results demonstrated that our methods have better performance in filtering out non-near-miss cases.

The overall organization of the paper is as follows: Section 1 introduces the significant challenges in near-miss detection from fisheye video, and summarizes the proposed method. Section 2 discusses related work on video-based near-miss detection. Section 3 presents preliminary and methodology details for the proposed method. Section 4 describes the dataset and demonstrates qualitative and quantitative evaluation. Section 5 gives an overall summary of the work in this paper and discussion of future opportunities for extending this work to other applications.

2 RELATED WORK

We have conducted a literature survey on near-miss or accident detection. However, these methods have limitations to process large-scale video data due to performance issues and difficulties in handling the distortion characteristics of omnidirectional fisheye videos. Therefore, we proposed a method with real-time performance and distortion correction. Future opportunities consist of extending our method to near-miss anticipation and/or applying specific spherical coordinate-based deep learning models.

Near-Miss or Accident Detection. In general, we have two types of methods for near-miss or accident detection: sensor-based and video-based. **Sensor-based** methods typically use data collected from loop detectors or multi-sensors and most of them apply machine learning or signal processing techniques: Kalman filters, time series analysis and decision trees etc. in (Ohe et al., 1995; Srinivasan et al., 2001; Srinivasan et al., 2003; Ghosh-Dastidar and Adeli, 2003; Srinivasan et al., 2004; Zeng et al., 2008; Chen et al., 2010). **Video-based** methods attempt to recognize near-miss events from image and video and this is the focus of the present work. The literature in this area uses a variety of machine learning and computer vision technologies (Jiansheng et al., 2014; Kamijo et al., 2000; Saunier et al., 2010; Chen et al., 2020; Banerjee et al., 2020; He et al., 2020). Specific techniques include histograms of flow gradients (HFG) (Sadeky et al., 2010), smoothed particles hydrodynamics (SPH) (Ullah et al., 2015), matrix approximation (Xia et al., 2015), optical flow and scale-invariant feature transform (SIFT) (Chen et al., 2016), adaptive traffic motion flow modeling (Maaloul et al., 2017) and convolutional neural networks (CNNs) and stacked autoencoders (Singh and Mohan, 2018). However, these methods are restricted to offline processing and not really applicable for the real-time analysis of fisheye video.

Near-Miss or Accident Anticipation. The early work in accident prediction is mainly based on anomaly detection.With advances in deep neural networks and object detection, several automatated traffic accident anticipation methods based on deep learning have been proposed. (Chan et al., 2016) proposed a method for anticipating accidents in dashcam videos using a Dynamic-Spatial-Attention (DSA) recurrent neural network (RNN). Meanwhile, more large-scale annotated video accident datasets have been proposed along with these learning based methods such as surveillance videos (Sultani et al., 2018; Shah et al., 2018) or drive (dashcam) videos (Chan et al., 2016). In our work, we detect near-misses 5 to 20 frames ahead of an actual near-miss or accident.

Fisheye Video Processing. A few deep network models have been proposed to learn and handle spherical representations in fisheye videos for problems such as object detection, tracking and segmentation. (Lee et al., 2019) proposed a method to directly apply CNNs to omnidirectional images. (Li et al., 2019) proposed a method with a CNN (trained on a synthetic distortion dataset) to predict displacement fields between distorted images and corrected images. (Wei et al., 2011) presented an interactive fisheye correction method that integrates natural scene appearance and use energy minimization of time-varying distortion. (Dhane et al., 2012) presented a fisheye correction method using non-linear radial stretching and scaling down of pixels in X and Y directions. (Yin et al., 2018) proposed a multi-context collaborative deep network to rectify distortions from single fisheye images. These approaches process videos offline and are not applicable for real-time traffic applications.

3 METHODOLOGY

Figure 3 demonstrates the pipeline and the overall architecture of the proposed method.

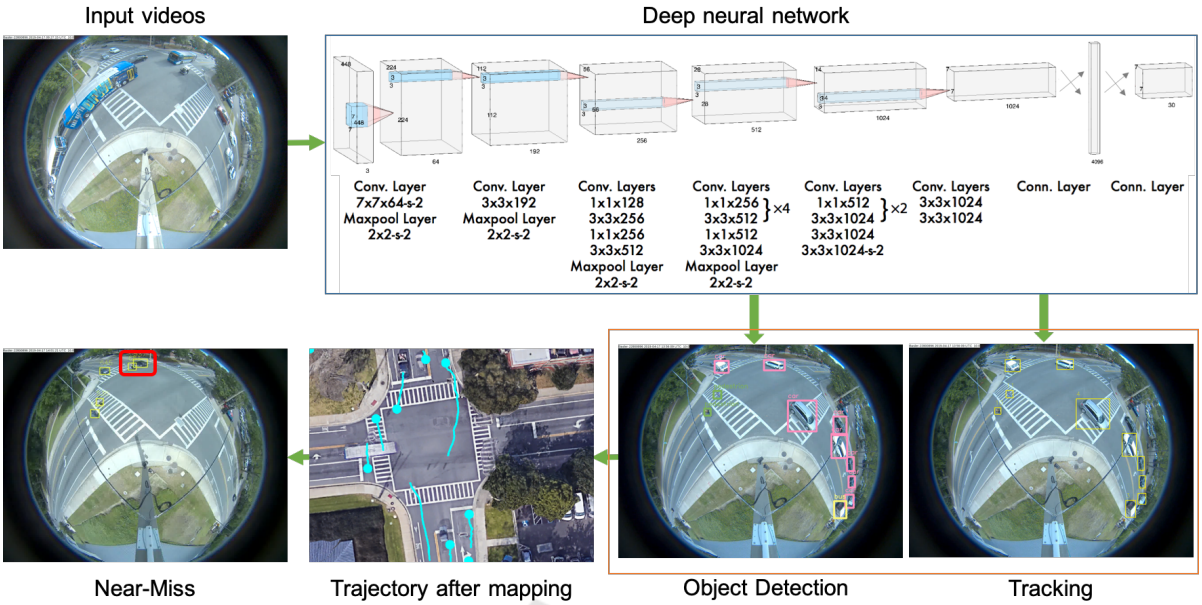

Figure 3: The Pipeline and the Deep Model Architecture of the Proposed Method.

3.1 Fisheye to Cartesian Mapping

3.1.1 Calibration and Perspective Correction

Due to fisheye lens distortion and perspective distortion, we found that directly applying mapping methods between fisheye images and satellite maps does not result in good quality mappings. Therefore, we wish to utilize fisheye camera parameters to make an initial calibration. For our fisheye camera model, points in real 3D world are first transformed to fisheye coordinates via extrinsic parameters (rotation and translation), and these fisheye coordinates are mapped into the 2D image plane via the intrinsic parameters (including the polynomial mapping coefficients of the projection function). For a point P in the 3D world, the transformation from world points to points in the camera reference image is:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} Xc \\ Yc \\ Zc \end{pmatrix} = R \begin{pmatrix} Xw \\ Yw \\ Zw \end{pmatrix} + T \qquad (1)$$

where R is the rotation matrix and T is for translation. The pinhole projection coordinates of P is (a,b) where $a = x/z$, $b = y/z$, $r^2 = a^2 + b^2$, $\theta = atan(r)$. The fisheye distortion is defined as

$$\theta_{distortion} = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \qquad (2)$$

where the vector of distortion coefficients is (k_1, k_2, k_3, k_4) and camera matrix is

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

The distorted point coordinates are $(x' = (\theta_d/r)a, y' = (\theta_d/r)b)$. The final pixel coordinates vector is (u,v) where $u = f_x(x' + \alpha y') + c_x$ and $v = f_y y' + c_y$, where skew coefficient α is set to zero and stay zero.

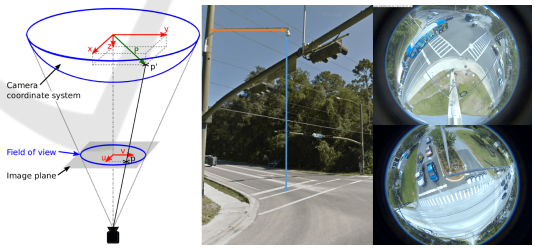

Figure 4: Illustration of Omnidirectional Fisheye Camera Used for Data Collection and Examples of Fisheye Video.

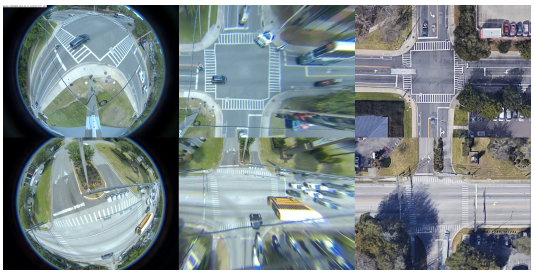

Figure 5: Calibration and TPS Mapping Are Used for Fisheye to Cartesian Mapping. **left:** Original Fisheye Image. **middle:** Mapping Result. **right:** Reference Satellite Map.

The procedure of distortion correction involves three major stages—Calibration correction, Perspective correction, and TPS mapping—in order to transform the object location in fisheye to Cartesian coordinates. The calibration process involves getting parameters using a checkerboard reference. The image obtained after the calibration has a noticeable perspective distortion which is adjusted by selecting four points in the output image of the first stage and then mapping them to a reference satellite image. There are small but noticeable distortions in the image after perspective correction which are caused by the geographic structure of the road like ridges and grooves or may be due to small errors caused in the calibration. To address these distortions, the TPS mapping is used, where multiple points are selected on the image obtained after perspective transformation and mapped to points on the satellite map. It approximates the transformation using a spline-based method. Thus by performing TPS, we get an image whose ground (road) and the map ground almost overlap. As our application goes beyond distortion correction, we can actually track the vehicles and get the exact location in Cartesian coordinates.

3.1.2 Thin-plate Spline Mapping

After calibration and perspective correction steps, we can compute an initial fisheye to cartesian mapping. To refine the mapping between the corrected fisheye image and satellite map, we adopt the thin-plate spline (TPS) as the parameterization of the non-rigid spatial mapping connecting fisheye geometry to a Cartesian grid. The choice of TPS to handle the spatial warping in our problem is driven by the fact that it is a natural non-rigid extension of the affine map. Furthermore, we do not have any information regarding physics-based mappings that can augment fisheye calibration. Therefore, we adopt the TPS to generate mappings. Given the point-sets V and Y in 2D ($D = 2$) consisting of points $v_a, a = 1,2,...,K$ and $y_a, a = 1,2,...,N$ respectively, the TPS fits a mapping function $f(x,y)$ using corresponding landmark sets y_a and v_a by minimizing the following energy function (Chui and Rangarajan, 2003):

$$E_{TPS}(f) = \sum_{a=1}^{K} \|y_a - f(v_a)\|^2 + \lambda \int\int \left[(\frac{\partial^2 f}{\partial x^2})^2 + 2(\frac{\partial^2 f}{\partial x \partial y})^2 + (\frac{\partial^2 f}{\partial y^2})^2 \right] dxdy. \quad (4)$$

Homogeneous coordinates are used for the landmarks with each point y_a represented as a vector $(1, y_{ax}, y_{ay})$.

With a fixed regularization parameter λ, a unique minimizer f can be obtained as follows (Chui and Rangarajan, 2003):

$$f(v_a, d, w) = v_a \cdot d + \phi(v_a) \cdot w \quad (5)$$

where d is a $(D+1) \times (D+1)$ matrix representing the affine transformation and w is a $K \times (D+1)$ warping coefficient matrix representing the nonaffine deformation. The vector $\phi(v_a)$ is a $1 \times K$ vector related to the TPS kernel. When combined with the warping coefficients w, the TPS generates a non-rigid warping.

Figure 4 illustrates the omnidirectional fisheye camera model, camera placement and examples of collected video data. We present mapping results of two intersections in Figure 5.

3.2 Object Detection and Multiple Object Tracking

The pipeline of our framework is to first detect and track road objects using deep learning models and then compute distortion corrected speeds followed by map-based trajectories. The deep object detector—trained on fisheye video samples—is based on the architecture of YOLO (Redmon and Farhadi, 2017). According to the intersection attributes, we specify five object categories: pedestrian, motorbike, car, bus, and truck.

The multiple object tracker is built upon DeepSort (Wojke et al., 2017), which uses a conventional single hypothesis tracking method with recursive Kalman filtering (Kalman, 1960) and frame-by-frame data association. However, there exists an occlusion problem when the intersection becomes crowded or when big buses or trucks appear. Therefore, some road objects can get a new identification after occlusion disappears and this forces us to integrate object signatures or object re-identification features. We introduce a deep cosine metric learning component to learn the cosine distance between road objects and integrated it as the second metric measure for the assignment problem in multiple object tracking. The cosine distance includes appearance information of road objects to provide useful cues for recovering identities when the motion feature is less discriminative. We trained the deep cosine metric learning model on the VeRi dataset (Liu et al., 2016).

Given the dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ of N training samples $\boldsymbol{x}_i \in \mathbb{R}^D$ and associated class labels $y_i \in \{1,\ldots,C\}$, it fits a parametrized deep neural network encoder function $\boldsymbol{r} = f_\Theta(\boldsymbol{x})$ with parameters Θ project 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. We modified a standard softmax classifier

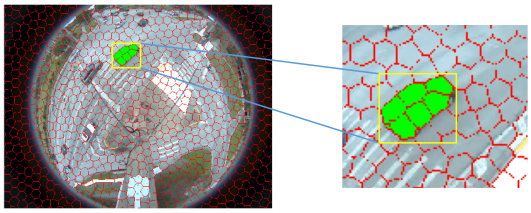

Figure 6: Example of Superpixel Segmentation on Fisheye Video. It Is Used for Extracting More Detailed Object Features (with Object Boundaries and Shapes) than Detection.

into a cosine softmax classifier as (Wojke et al., 2017)

$$p(y=k \mid \boldsymbol{r}) = \frac{\exp\left(\kappa \cdot \tilde{\boldsymbol{w}}_k^{\mathrm{T}} \boldsymbol{r}\right)}{\sum_{n=1}^{C} \exp\left(\kappa \cdot \tilde{\boldsymbol{w}}_n^{\mathrm{T}} \boldsymbol{r}\right)}, \tag{6}$$

where κ is a free scale parameter. The training of the cosine metric encoder network can also be carried out using the cross-entropy loss.

3.3 Trajectory and Speed Computation

We leverage tracking results to generate a trajectory for each object in terms of frame, track id, class, x, y coordinates. We transform the x, y coordinates from fisheye image space to overhead satellite map space using the point-to-point mapping matrix obtained in the mapping pipeline. We estimate the speed of objects using distance after mapping. In order to leverage more accurate and compact object masks than rectilinear bounding boxes, we also investigate the (Huang et al., 2020) use of gSLICr (Ren et al., 2015), a GPU-based implementation of SLIC (Achanta et al., 2012)—a superpixel segmentation method—instead of standard rectangular bounding boxes. Figure 6 demonstrates the use of superpixels for generating object masks. This integration performed in real-time results in better distance measures that can be utilized for detecting near-misses.

3.4 Near-Miss Detection

Our method performs object detection and multiple object tracking in real-time and has the capability to handle large-scale and city-scale intersection video for traffic understanding and analysis. Using our TPS-based non-rigid mapping tool we can do both online and offline coordinates correction to project road object locations to satellite maps and then form refined trajectories for near-miss detection.

Two near-miss scenarios are defined for videos: 1) spatial scenario: road objects collide or are very close in image space, 2) temporal scenario: a dramatic speed decrease to avoid near-misses (a sudden break). We use distance-based measures and speed measures to compute the near-miss probability of road objects with the average of two scores serving as the final output as described below.

Spatial Distance Measure We use tracks data to form trajectories of road objects and compute distances between two road objects using center coordinates of detected bounding boxes in image space at frame level. The probability of a spatial near-miss is computed using the Euclidean distance of road objects with ratio to the size of object according to object class (vehicles size of each class does not varies much) and is computed as follows:

$$P_{spatial}(\mathbf{b}_t^p, \mathbf{b}_t^q) = \frac{1}{2} \cdot \frac{(\mathbf{w}_t^p + \mathbf{w}_t^p + \mathbf{h}_t^q + \mathbf{h}_t^q)}{\sqrt{(\mathbf{x}_t^p - \mathbf{x}_t^q)^2 + (\mathbf{y}_t^p - \mathbf{y}_t^q)^2}} \tag{7}$$

where $\mathbf{b}_t^p$ and $\mathbf{b}_t^q$ denote the detected bounding boxes for the p-th and q-th objects in the t-th frame. $\mathbf{w}_t^p$, $\mathbf{h}_t^q$, $\mathbf{w}_t^p$, $\mathbf{h}_t^q$ denote the object width, object height, x coordinate, and y coordinates for the p-th object in the t-th frame respectively.

Temporal motion measure The speed of the road object is computed by adjacent displacement over multiple time frames. The probability of motion-based near-miss is computed by the fractional decrease in speed and is computed as follows:

$$P_{temporal}(\mathbf{b}_{k:t}^p) = \frac{\max \sum_{i=k}^{t} (\mathbf{s}_{i+1}^p - \mathbf{s}_i^p)}{average(\mathbf{s}_{k:t}^p)} \tag{8}$$

where $\mathbf{b}_{1:t}^p$ denotes the detected bounding boxes for the p-th from its first frame (k-th frame) to its last frame (t-th frame). $\mathbf{s}_i^p$ denotes speed for the p-th object in the i-th frame.

We use a weighted average of the above two probabilities of near-miss to compute the overall score.

4 EXPERIMENTS

We first describe the dataset used for our experimental evaluation. We then present qualitative performance and quantitative evaluation of our methods for object detection, multiple object tracking, superpixel segmentation, thin-plate spline, and near-miss detection. For near-miss detection, we present a performance comparison between non-mapping-based method and our proposed calibration+TPS-based method.

4.1 Fisheye Video Data

We have collected large-scale fisheye traffic video from omnidirectional cameras at several intersections. Figure 7 shows gallery images of the dataset with several collected fisheye video samples at multiple intersections under different lighting conditions. We collected 8 hours of videos on a daily basis for each

Figure 7: Gallery Images of Fisheye Video with a Variety of Different Locations (4 Cameras) and Different Lighting Conditions.

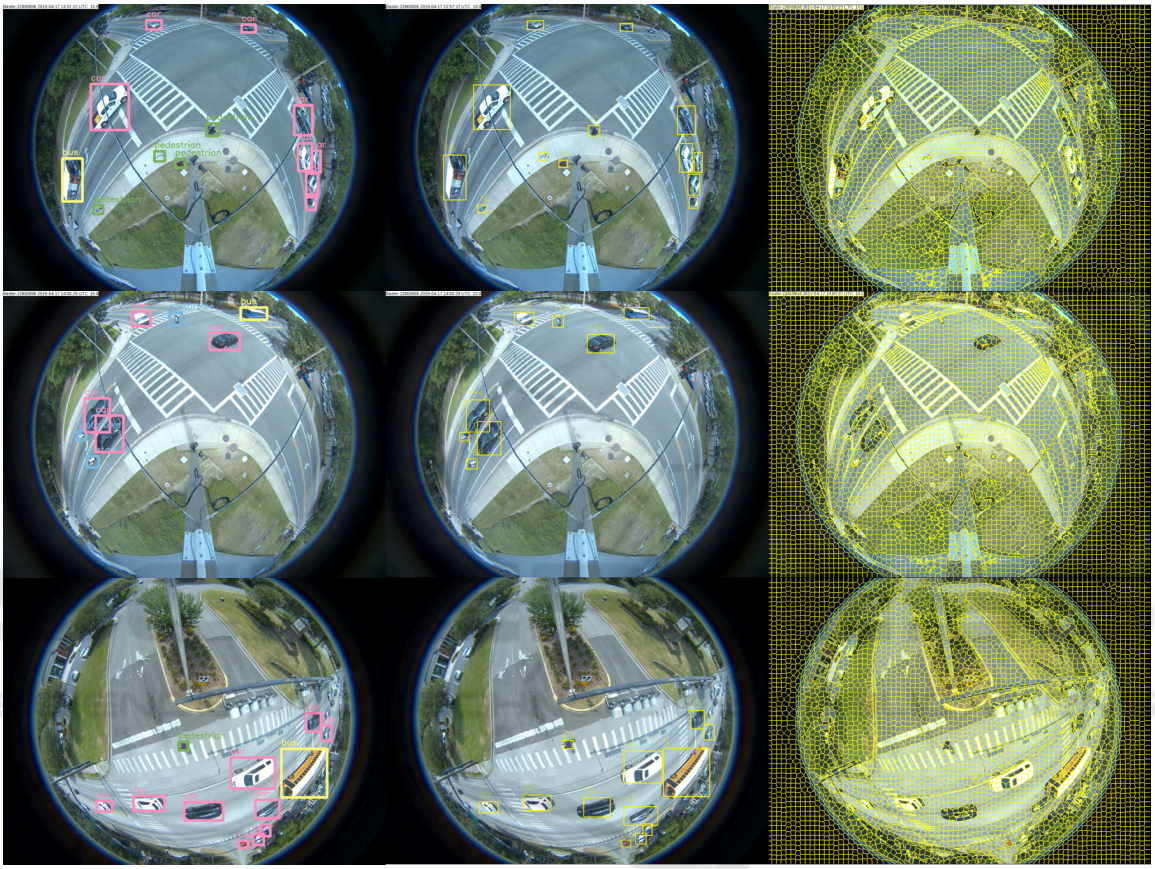

Figure 8: Qualitative Results of Detection, Tracking and Segmentation Tasks. **left:** Object Detection, Outputs Object Class (Car, Pedestrian, Bus, Motorbike, Etc.) and Localization (Bounding Box). **middle:** Multiple Object Tracking, Associates Object in Consecutive Video Frames (Track Id). **right:** Superpixel Segmentation, Aids to Compute Object Boundaries and Shapes.

intersection, 2 hours for the morning, noon, afternoon, evening time respectively. The total video datasets used the experiments has a duration of more than 100 hours. As discussed earlier, fisheye intersection videos are more challenging than videos in other datasets collected by surveillance cameras for reasons including fisheye distortion, multiple object types (pedestrians and vehicles) and diverse lighting conditions. For generating ground truth for object detection, tracking, and near-miss detection, we manually annotated the spatial location (bounding boxes) and temporal location (frames) for each object and near-miss. We also annotated the corresponding vehicle class.

4.2 Qualitative Performance

Fisheye to Cartesian Mapping. The results for calibration+TPS pipeline (Figure 5) shows that fisheye distortion and perspective distortion are effectively addressed by our method. The qualitative results in terms of performance for object detection, multiple object tracking, and superpixel segmentation (Figure 8) show that the deep learning based detector is effective in classifying objects even when the image footprint is small (e.g. pedestrians and motorbikes). The use of deep cosine metric learning allows the tracker to generate more consistent and stable tracks. The superpixel segmentation assists in out-

Table 1: Quantitative Evaluation of Speed Performance.

Methods	CPU/GPU	Speed
TPS mapping	CPU	10 s
SLIC segmentation	NVIDIA TITAN V	400 fps
Overall pipeline	NVIDIA TITAN V	40 fps

Table 2: Quantitative Performance of Object Detection and Multiple Object Tracking.

Methods	TP	FN	FP	Precision	Recall	F1-score
Object Detection	7649	102	82	0.98940	0.98684	0.98812
Multiple Object Tracking	7540	483	314	0.96002	0.93980	0.94980

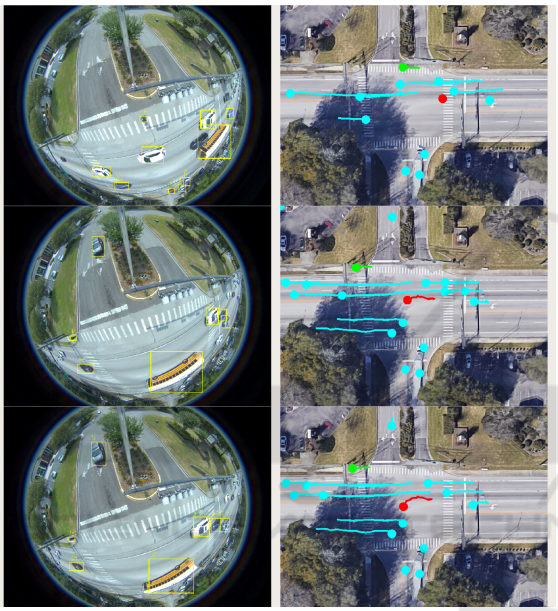

Figure 9: Qualitative Results of Object Trajectories Mapping to Satellite Map. **left:** Tracking in Fisheye Video. **right:** Trajectories after Mapping. Different Color Represents Different Object Class (Red for Bus, Green for Pedestrian, Blue for Car).

putting compact contours of objects. The latter can then be used for an effective signature for tracking.

Trajectory and near-Miss Detection. The trajectories of road objects projected on the satellite map along with referenced tracking frames are shown in Figure 9. These trajectory maps give an easier to understand traffic pattern for the intersection than that from the perspective of the original fisheye camera. Samples of near-miss we detected at different intersections are presented in Figure 10. The first example shows a spatial near-miss between two road objects. The second example shows a temporal near-miss as the front white car suddenly stopped in the middle of the intersection, forcing a sudden break for the car that was following.

4.3 Quantitative Evaluation

We present a quantitative evaluation of the overall performance of our proposed method in terms of speed performance, improvement object speed measures based on mapping, and the precision and recall for each subtask of the pipeline.

Computational Requirements. We present speed performance for the tested methods in Table 1. The fisheye video resolution is 1280×960 and our implementation for thin-plate spline takes 10s for one-to-one corresponding mapping for 1,228,800 points. This is one-time setup cost.

After getting mapping point-sets, all video processing experiments have been performed on a single GPU (NVIDIA TITAN V). The GPU-based SLIC segmentation (Achanta et al., 2012) has excellent speed performance and can process 400 fps on fisheye videos. The overall pipeline of our methods (object detection, multiple object tracking, and near-miss detection) achieve about 40 fps. This rate is sufficient to address a variety of real traffic surveillance and near-miss detection for large-scale daily video data.

Trajectory and near-Miss Detection. A quantitative prediction of object representation and near-miss is achieved by comparing predicted detection with the ground truth at frame level. A true positive corresponds to a high level of overlap between prediction and ground truth detection pair. It is computed using an Intersection over Union (IoU) score. If this overlap exceeds a predefined threshold (e.g 0.7) then the track is correctly associated. A true negative means no prediction and no associated ground truth. A false positive is that a prediction had no associated ground truth. A false negative is that a ground truth had no associated prediction. The true negative rate (TNR) also refers to specificity and false positive rate (FPR) refers to fall-out. The specificity, fall-out, precision,

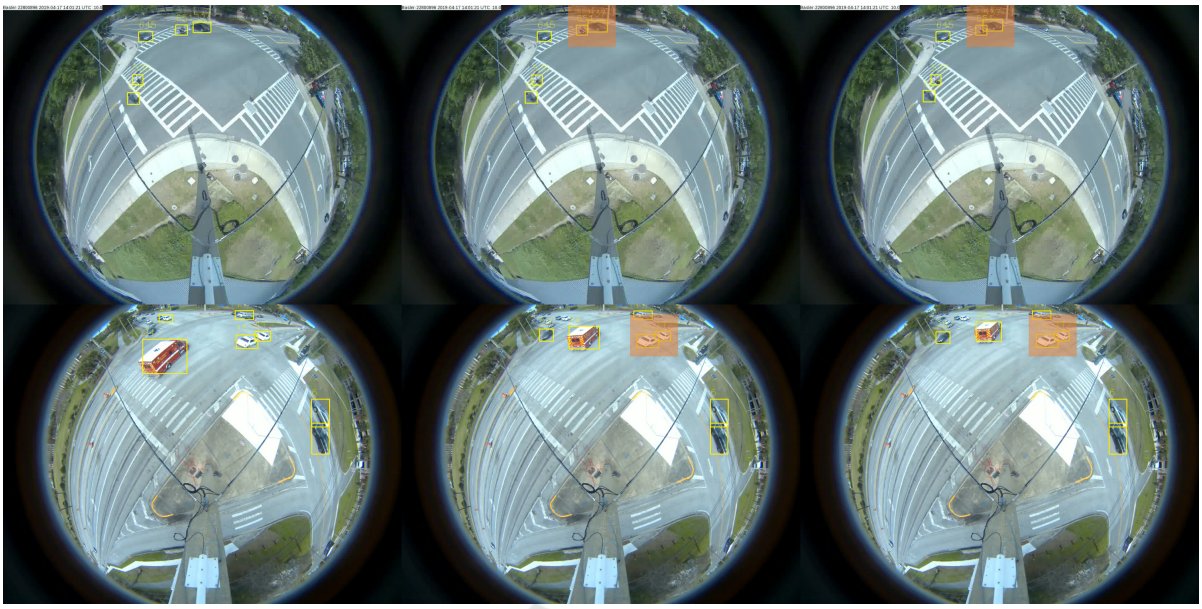

Figure 10: Qualitative Results of Two Types of near-Miss Detected. **top 3 Images:** a Spatial near-Miss Case: A Motorbike and a Car Are Colliding. **bottom 3 Images:** a Temporal near-Miss Case Caused a Sudden Break.

Table 3: Quantitative comparison of near-miss detection between non-mapping and calibration+TPS-based method.

Methods	Intersection	TN	FP	Specificity (TNR)	Fall-out (FPR)
Non-mapping based (Baseline)	intersection 01	2869	32	0.98897	0.01103
	intersection 02	2659	162	0.94257	0.05743
Calibration + TPS mapping	intersection 01	2895	6	0.99793	0.00207
	intersection 02	2818	3	0.99894	0.00106

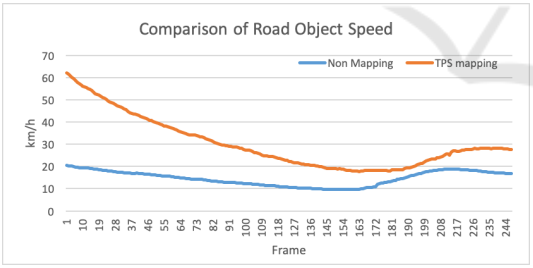

Figure 11: Quantitative Comparison of Computed Object Speed between Non-Mapping and Proposed Methods.

recall, and F1-score are defined as

$$TNR = \frac{TN}{TN+FP} = 1 - FPR \tag{9}$$

$$Precision = \frac{TP}{TP+FP} \quad Recall = \frac{TP}{TP+FN} \tag{10}$$

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall} \tag{11}$$

We compute object speed information based on trajectories by converting pixels to actual meters and frame intervals to seconds. Figure 11 shows an example of the comparison of computed object speed information where a car is approaching the intersection with speed decreasing from 60 km/h to 20 km/h and then back to 30 km/h. With non-mapping methods, object speed computing suffers from fisheye and perspective distortion and yields inaccurate results. We also present accuracy evaluation for object detection and multiple object (cosine metric learning) in Table 2. As real near-miss is rare in terms of two camera video data in a week, it is more reasonable to exam specificity (selectivity or true negative rate) and fall-out (false positive rate) for near-miss detection. In Table 3, we present the comparison of non-mapping based detection and calibration+TPS mapping based detection in terms of true negative rate (TNR) and false positive rate (FPR). The quantitative evaluation demonstrates the overall effectiveness of our proposed method for near-miss detection in large-scale fisheye traffic videos.

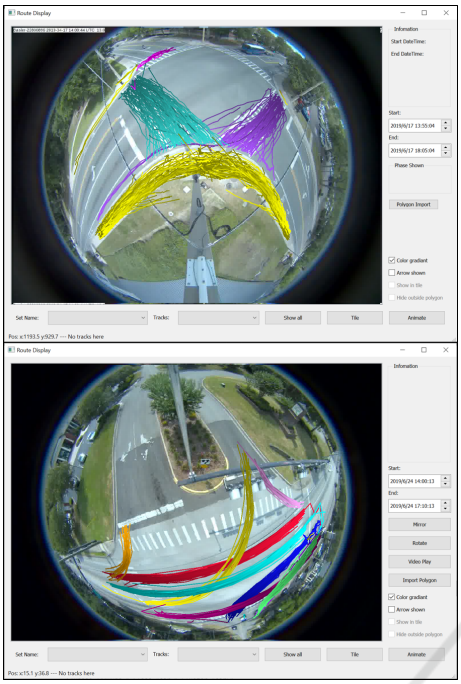

Figure 12: Example of Trajectory Visualization Used in Our Visualization Tool. **top:** Pedestrian Trajectories in Intersection 01. **bottom:** Vehicle Trajectories in Intersection 02. Color Means Different Clusters after Clustering.

5 CONCLUSIONS

We presented a novel unsupervised method to detect near-misses in fisheye intersection video using an end-to-end deep learning model integrated with a combined camera calibration and spline-based mapping method. It maps road objects coordinates in fisheye images to a satellite based overhead map to correct fisheye lens distortion and camera perspective distortion. This allows for computing distance and speed more accurately. This unified approach performs real-time object recognition, multiple object tracking, and near-miss detection in fisheye video. It is efficient and robust to handle geometry and uncertainty on object-level analysis in fisheye video, resulting in more accurate near-miss detection. The experimental results demonstrate the effectiveness of our approach and we show a promising pipeline broadly applicable to fisheye video understanding applications such as accident anticipation, anomaly detection, and trajectory prediction.

Intersection SPAT data can be integrated with video data to develop interesting traffic analyses, e.g. cars crossing the intersection during a red light. The generated tracks can be plotted over extended periods (shown in Figure 12) to visualize macro trends.

ACKNOWLEDGMENTS

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

REFERENCES

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

Banerjee, T., Huang, X., Chen, K., Rangarajan, A., and Ranka, S. (2020). Clustering object trajectories for intersection traffic analysis. In *6th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2020)*.

Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585.

Chan, F.-H., Chen, Y.-T., Xiang, Y., and Sun, M. (2016). Anticipating accidents in dashcam videos. In *Asian Conference on Computer Vision*, pages 136–153. Springer.

Chen, K., Banerjee, T., Huang, X., Rangarajan, A., and Ranka, S. (2020). A Visual Analytics System for Processed Videos from Traffic Intersections. In *6th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2020)*.

Chen, L., Cao, Y., and Ji, R. (2010). Automatic incident detection algorithm based on support vector machine. In *2010 Sixth International Conference on Natural Computation*, volume 2, pages 864–866. IEEE.

Chen, Y., Yu, Y., and Li, T. (2016). A vision based traffic accident detection method using extreme learning machine. In *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 567–572. IEEE.

Chui, H. and Rangarajan, A. (2003). A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141.

Dhane, P., Kutty, K., and Bangadkar, S. (2012). A generic non-linear method for fisheye correction. *International Journal of Computer Applications*, 51(10).

Ghosh-Dastidar, S. and Adeli, H. (2003). Wavelet-clustering-neural network model for freeway incident detection. *Computer-Aided Civil and Infrastructure Engineering*, 18(5):325–338.

He, P., Wu, A., Huang, X., Rangarajan, A., and Ranka, S. (2020). Video-based machine learning system for commodity classification. In *6th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2020)*.

Huang, X., He, P., Rangarajan, A., and Ranka, S. (2020). Intelligent intersection: Two-stream convolutional networks for real-time near-accident detection in traffic video. *ACM Trans. Spatial Algorithms Syst.*, 6(2).

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

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kamijo, S., Matsushita, Y., Ikeuchi, K., and Sakauchi, M. (2000). Traffic monitoring and accident detection at intersections. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):108–118.

Lee, Y., Jeong, J., Yun, J., Cho, W., and Yoon, K.-J. (2019). Spherephd: Applying cnns on a spherical polyhedron representation of 360deg images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9181–9189.

Li, X., Zhang, B., Sander, P. V., and Liao, J. (2019). Blind geometric distortion correction on images through deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4855–4864.

Liu, X., Liu, W., Ma, H., and Fu, H. (2016). Large-scale vehicle re-identification in urban surveillance videos. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE.

Maaloul, B., Taleb-Ahmed, A., Niar, S., Harb, N., and Valderrama, C. (2017). Adaptive video-based algorithm for accident detection on highways. In *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–6. IEEE.

Ohe, I., Kawashima, H., Kojima, M., and Kaneko, Y. (1995). A method for automatic detection of traffic incidents using neural networks. In *Pacific Rim TransTech Conference. 1995 Vehicle Navigation and Information Systems Conference Proceedings. 6th International VNIS. A Ride into the Future*, pages 231–235. IEEE.

Redmon, J. and Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

Sadeky, S., Al-Hamadiy, A., Michaelisy, B., and Sayed, U. (2010). Real-time automatic traffic accident recognition using hfg. In *2010 20th International Conference on Pattern Recognition*, pages 3348–3351. IEEE.

Saunier, N., Sayed, T., and Ismail, K. (2010). Large-scale automated analysis of vehicle interactions and collisions. *Transportation Research Record: Journal of the Transportation Research Board*, (2147):42–50.

Shah, A., Lamare, J. B., Anh, T. N., and Hauptmann, A. (2018). Accident forecasting in cctv traffic camera videos. *arXiv preprint arXiv:1809.05782*.

Singh, D. and Mohan, C. K. (2018). Deep spatio-temporal representation for detection of road accidents using stacked autoencoder. *IEEE Transactions on Intelligent Transportation Systems*.

Srinivasan, D., Cheu, R. L., and Poh, Y. P. (2001). Hybrid fuzzy logic-genetic algorithm technique for automated detection of traffic incidents on freeways. In *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*, pages 352–357. IEEE.

Srinivasan, D., Jin, X., and Cheu, R. L. (2004). Evaluation of adaptive neural network models for freeway incident detection. *IEEE Transactions on Intelligent Transportation Systems*, 5(1):1–11.

Srinivasan, D., Loo, W. H., and Cheu, R. L. (2003). Traffic incident detection using particle swarm optimization. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*, pages 144–151. IEEE.

Sultani, W., Chen, C., and Shah, M. (2018). Real-world anomaly detection in surveillance videos. *Center for Research in Computer Vision (CRCV), University of Central Florida (UCF)*.

Tang, Z., Naphade, M., Liu, M.-Y., Yang, X., Birchfield, S., Wang, S., Kumar, R., Anastasiu, D., and Hwang, J.-N. (2019). Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ullah, H., Ullah, M., Afridi, H., Conci, N., and De Natale, F. G. (2015). Traffic accident detection through a hydrodynamic lens. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 2470–2474. IEEE.

Wei, J., Li, C.-F., Hu, S.-M., Martin, R. R., and Tai, C.-L. (2011). Fisheye video correction. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1771–1783.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE.

Xia, S., Xiong, J., Liu, Y., and Li, G. (2015). Vision-based traffic accident detection using matrix approximation. In *2015 10th Asian Control Conference (ASCC)*, pages 1–5. IEEE.

Yin, X., Wang, X., Yu, J., Zhang, M., Fua, P., and Tao, D. (2018). Fisheyerecnet: A multi-context collaborative deep network for fisheye image rectification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 469–484.

Zeng, D., Xu, J., and Xu, G. (2008). Data fusion for traffic incident detection using ds evidence theory with probabilistic svms. *Journal of computers*, 3(10):36–43.