To Turn or Not To Turn, SafeCross is the Answer

Baofu Wu
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
baofu.wu@hdu.edu.cn

Yuankai He
Department of Computer Science
Wayne State University
Detroit, USA
william.he@wayne.edu

Zheng Dong
Department of Computer Science
Wayne State University
Detroit, USA
dong@wayne.edu

Jian Wan
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
wanjian@hdu.edu.cn

Jilin Zhang
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
jilin.zhang@hdu.edu.cn

Weisong Shi
Department of Computer Science
Wayne State University
Detroit, USA
weisong@wayne.edu

***Abstract*—Blind area has plagued drivers' safety ever since the dawn of automobiles. Thanks to the fast-growing vision-based perception technologies, autonomous driving systems can monitor the driving circumstance through a 360-degree view, and hence most blind areas can be avoided. However, in the left turn scenario at an intersection, the opposite road may be blocked by another vehicle parking at the same intersection (see Fig. 1), and in this case, the blind area cannot be observed by the onboard perception module of the autonomous vehicle. A potential fatal collision may occur if the autonomous vehicle turns left while a vehicle is running through the blind area. In this paper, we propose Safecross, a framework that oversees an intersection and delivers blind area warnings to the left-turn vehicles at the intersection if running vehicles are detected in the blind area. In order to provide accurate and reliable real-time warnings in all possible weather conditions, the architecture of Safecross has four major components: video pre-processing (VP) module, video classification (VC) module, few-shot learning (FL) module, and model switching (MS) module. Especially, the VP and VC modules will train a basic model to identify the blind area when a blocking vehicle appears at the intersection. Since the range of the blind area varies in different weather conditions, the FL and MS modules can adapt the basic model to the new condition in real-time to make the blind area identification more accurate. Intuitively, if the blind area is identified timely and accurately, the left-turn throughput of the intersection can be maximized. We have conducted extensive experiments to evaluate our proposed framework. The experiments are performed on a total of 2855 video segments with a time span of 180 days, including sunny, rainy, and snowy weather conditions. Experimental results show how Safecross can guarantee the vehicle's safety while increasing the left-turn traffic throughput by 50%.**

***Index Terms*—SafeCross, Blind zone, Collision avoidance, Real-time, Scenes adaption**

I. Introduction

BLIND zone is one of the major obstacles endangering vehicle safety. According to National Highway Safety Administration statistics (NHTSA), there are more than 800,000 traffic accidents caused by blind areas every year [1]. Autonomous vehicles are equipped with cameras, LiDAR sensors, Radar sensors, and other sensors. Most of the blind areas around the vehicle are monitored by these sensors; however, there are blind areas that are caused by other vehicles, such as at an intersection, is not resolved.

The most typical example is the blind areas in the left turn scenario at a road intersection. According to NHTSA's report in 2010, more than 480,000 crashes involve drivers turning left [2]. The most common reasons cited in the study for left-turn crashes include obstructed view while turning, inadequate surveillance (driving into blind zones), misjudgment of the other driver's speed and miscalculating the distance or "gap" across the intersection.

In the United States and other right-side driving countries, a specific blind area problem plagues the safety of drivers and traffic throughput. An example of such scenes is shown in Fig. 1. In scenes similar to this, the light is green for both the vehicles going straight and turning left. To simplify, we will assume ideal driving conditions. Here we see the vehicle circled in green is looking to make a left turn; however, his view of the opposite road is blocked by the van. Without knowing if there is any on-coming traffic, the vehicle circled in green cannot accurately and confidently make his left turn. In this case, the vehicle circled in red is in the blind area behind the van. If the vehicle circled in green had chosen to make the turn, it would result in a tragic accident. However, if there weren't any cars next to or behind the van, the driver would be wasting his and others' time by waiting for the non-existent vehicle to pass. The problem the vehicle circled in green and many others have stems from incomplete information. To address this challenge, we need a system that can analyze the traffic from a "global" point of view. The system needs to ensure the safety of the driver and to increase traffic throughput. Therefore, the system needs to be able to accurately identify whether or not it is safe for the vehicle circled in green to make the left turn and to output the results in real-time.

Since autonomous vehicle's sensors cannot obtain global information to effectively analyze the risk of blind zones, existing solutions mainly focus on the analysis of the surveillance video of traffic intersections. On the edge server, researchers

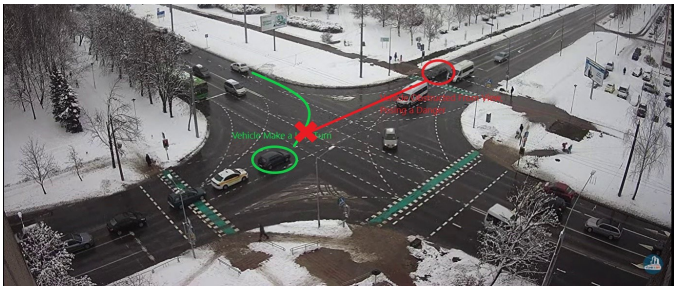


Fig. 1. An overview of left-turn alert system. When the opposite truck obstructs the sight of the vehicle attempting a left turn, the straight-going vehicle moving quickly in the blind area behind the truck is dangerous. The roadside unit equipped with a surveillance camera has a global perspective without blind zones, predicts the risk of collision, and sends out early warning signals.

[3]–[8] use recognition algorithms to locate vehicles to realize the collision warning analysis. However, existing solutions only focus on the risk prediction of a single scenario, and do not consider the impact of weather conditions on road conditions and safe distance. Different weather conditions, such as rain and snow, will affect the friction coefficient of the road, the emergency braking distance, and the deceleration distance of the vehicle. The scope of the safety warning should be adapted to different weather scenarios, which must be considered when the project is implemented.

To address the issues mentioned above, in this paper, we introduce the SafeCross framework, a framework that is added to the existing infrastructure. It oversees the entire junction, analyses the traffic, and makes an informed decision for the driver.

To solve the problem of the limited vision of autonomous vehicles, we adopt a vehicle-road coordination solution to obtain global road information through global surveillance cameras deployed on the roadside. To achieve real-time data processing as much as possible, we will use background subtraction operation to simplify the original RGB data into black and white data; at the same time, we use the video classification algorithm to learn the habits of humans in turning left, and train to get a blind-zone warning model which can judge whether a safety reminder is needed. In the face of problems in different weather scenarios, especially for the scenarios with few samples, we use few-shot learning methods to train a model. Finally, to ensure real-time adaptation of models in different weather scenarios and to ensure uninterrupted business, we choose Pipeswitch to implement real-time switching of multiple models.

Although we studied left-turn blind zones warning issues in the right-driving countries, the SafeCross framework is also suitable for dealing with right-turn blind zones warning in left-driving nations. The difference is just the training data. The major contributions of the SafeCross framework are listed below, of which we will show the results in the experiment and discussion sections.

- We propose the SafeCross framework, which is the first work to identify hidden dangers in blind zones, make early warning judgments in real-time, and adapt real-time models to different weather scenarios.
- We propose a basic model of whether there is danger in the blind zone which is trained based on video preprocessing (VP) and video classification (VC); in addition, we make the multiple scenes real-time adaptation into a true with few-shot learning (FL) and model switching (MS) to ensure uninterrupted recognition.
- We carry out several experiments to verify the performance of SafeCross. We created a dataset from an online live surveillance video. The classification accuracy of Daytime data has reached 98.98% in judging whether there is danger in the blind zone of the left-turn lane. It also achieves a good classification effect in a few-shot scene. It can realize real-time switching of models at a speed of less than 10ms when the scene changes. Through classification and statistics, SafeCross can effectively increase the traffic throughput by 50% in the presence of blind zones.

The other content of the paper is organized as follows. We present the related work in Sec. II, In Sec. III, we present the overall design of SafeCross framework. We will present the system implementation in Sec. IV. In Sec. V, the performance evaluation will be presented. We provide discussion and future work in Sec. VI. Finally, we conclude in Sec. VII.

II. Related Work

A. Detection

Among the potential collision problems caused by blind zones, it is essential to detect moving objects. As shown in Figure 1, the collision risk of a left-turning vehicle comes from the high-speed oncoming vehicle in the opposite blind zone. There are several popular approaches to detection: optical flow, frame difference, background subtraction, and deep learning methods.

Optical flow [9]–[11] focuses on the relationship of pixel characteristics in a sequence of image frames. By reading and comparing the pixel intensities in consecutive frames, the optical flow algorithm determines the apparent motion of objects in the scene. It is widely used to identify the speed and the direction of moving objects, but it has a very high computational complexity.

The frame difference method [12] is based on two or three consecutive frames to detect movement, which has the advantage of fast processing times. Still, it is challenging to separate overlapping targets and may cause false detection.

The background subtraction method [13]–[16] obtains the background image first and then uses semantic thresholds to process the difference image. It calculates the pixel intensity differences between the learned background and the test frame. The processing is fast, but it cannot detect non-moving objects.

The deep learning methods [17], [18] combine the deep learning algorithms with the above three methods to obtain higher accuracy but requires more pre-processing, learning, and memory.

B. Collision Avoidance

State-of-the-art Collision avoidance at traffic intersections mainly has two major approaches, trajectory prediction collision avoidance based on simulation and conflict detection based on deep learning.

Trajectory prediction collision avoidance based on simulation calculates the risk of collision through modeling and simulation. Kim et al. [6] predict the probabilistic motion of surrounding vehicles to analyze the collision risk of surrounding vehicles quantitatively. Park et al. [5] modeled the collision uncertainty through Gaussian distribution and proposed a probabilistic collision detection method between a highly free robot and an imperfect obstacle. The methods these two papers proposed solve the problem of the blind areas around the vehicle but cannot solve the blind areas created at an intersection.

The second category, based on deep learning, can solve several problems. Based on object detection results, researchers propose different object tracking algorithms to predict the probability of collision. For example, Tan et al. [7] use Yolo to implement the function of recognition and combine it with LSTM to achieve object tracking. Fu et al. [8] propose an effective algorithm with five steps for an infrastructure-cooperative intersection accident pre-warning system, which contains defining variables, reasoning the vehicle's evolution state, verifying safe driving behavior, assessing risk, and making decisions. However, it is just in the simulation stage.

C. Scene adaptation

The safety distances of vehicles in different weather scenarios are very different. The vehicles ' safety distance should be longer for the wet and slippery road in rainy and snowy weather. How to adapt to multiple scenes for the monitoring system is a brand new problem. In the existing research, there are two ideas. One is to build a supermodel with different weather scene data as input. The other is to build respective models for different scenarios and use model switching to achieve scene adaptation.

Building a supermodel is one of the solutions to the multi-scenario problem. It takes the data of different scenes as input and trains the model through a large amount of training data to adapt to the recognition task of different scenes. The super program has an essential manifestation in language. For example, the Generative Pre-trained Transformer(GPT-3) [19] model proposed by OpenAI, which has an astonishing 175 billion parameters, can realize functions such as translation, answering, translating, and writing articles. The articles it writes almost reach real people's level. To solve adaptive issues in multiple scenarios, expert mixing is another method. This method sets up models specializing in handling different tasks, namely experts, and has a "gated network" to choose which experts to consult for any given data. Based on this paradigm, Google proposed Switch Transformer-a "sparse activation" technique trained to obtain a model (Switch-C) containing 1.6 trillion parameters. It has 2048 experts. In the Sanford Question Answering Dataset (SQuAD) [20] benchmark test, Switch-C got 87.7. However, the supermodel requires extensive and expensive training data and labeling data. It is difficult to collect enough data at traffic intersections with a small amount of data, extraordinary weather data, for model training.

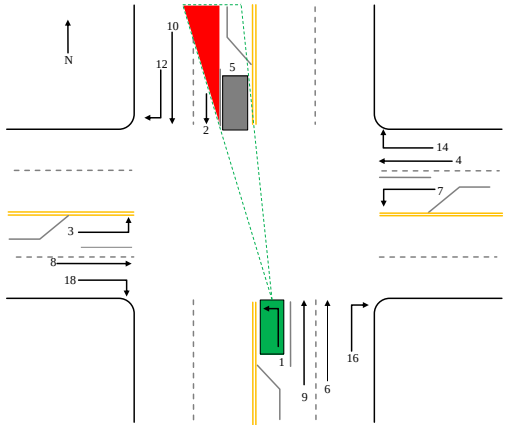


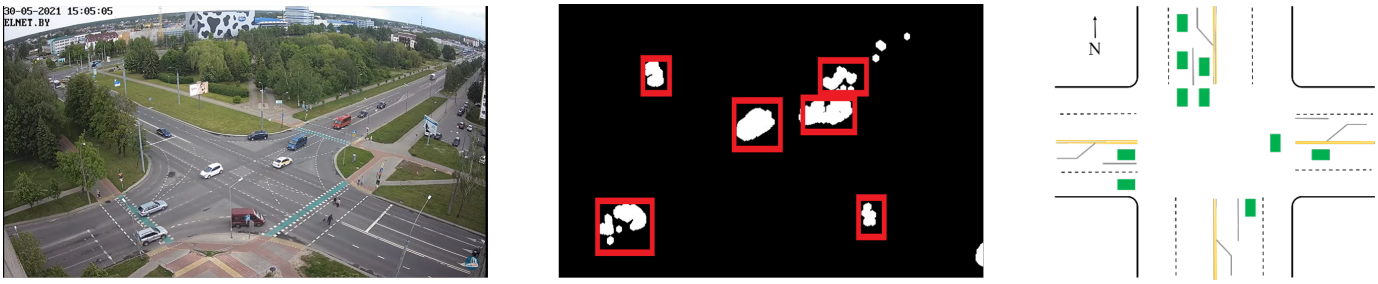
Fig. 2. The danger zone (in red) for a left-turn vehicle at an intersection.

III. SafeCross Framework

Problem Statement: As shown in Fig. 2, we consider a busy road intersection with a green vehicle, which attempts to turn left; however, a grey vehicle is waiting to turn left on the opposite side of the road, and it is blocking the view of the green vehicle. The blind area behind the grey vehicle is called the *danger zone* for the green vehicle, the *danger zone* is highlighted in red in Fig. 2. For the green vehicle to make a safe left turn, whether or not there are vehicles in the *danger zone* is very important. It is also very time costly and dramatically impacts the traffic flow if there aren't any vehicles in the *danger zone*, but the green vehicle does not make the turn because it does not have that information. Thus, we propose the SafeCross framework. It can deliver real-time alerts to left-turning vehicles if vehicles are detected in the *danger zone*. The SafeCross framework learns the size and location of the *danger zone* from drivers' behaviors. The *danger zone* is carefully learned and constrained for several reasons. If we arbitrarily define a very large danger zone, then we would not be helping traffic throughput; on the other hand, if we arbitrarily define a very small zone, then it does not ensure the safety of the left-turning vehicles. Different weather conditions are another factor when SafeCross determines the size and location of the *danger zone*, because the stopping distance and visibility range are different in different weather conditions.

A. System overview

SafeCross is a real-time system deployed in existing infrastructure, designed to ensure the safety of left-turning vehicles and increase left-turning traffic throughput in different scenes. The whole system architecture is shown in Figure 4, which includes four main components:



(a) The original scene. (b) Background subtraction. (c) Mapping.

Fig. 3. From the original scene to our DNN input.

- **Video Pre-processing (VP):** We need to identify all of the moving vehicles on the scene, and we do not need information from vehicles that are not moving. We utilize background subtraction to obtain this information.
- **Video Classification (VC):** After obtaining information about moving vehicles, we train a basic model based on drivers' behaviors when making a left turn under typical weather. This basic model is further divided into four categories: left turn with blind area, left turn without blind area, no left turn with blind area, no left turn without blind area. To fully extract the vehicle's temporal and spatial information, we choose the Slowfast video classification model as the training model.
- **Few shot Learning (FL):** Because it is difficult to find enough data for rainy, snowy, or other weather conditions, we use Few-shot learning to learn new models based on their nearest neighbor in the trained basic models.
- **Model Switching (MS):** When all models are trained, we use Pipe-switch to achieve real-time switching of models so that SafeCross can make the correct prediction for the size and location of the *danger zone*.

B. Video Pre-processing

1) Challenges: Due to the infrastructure available, we can only acquire camera footage from surveillance cameras, sometimes decades old. The quality of the video, the clarity of the objects in the video, and the placement of the cameras all proved to be difficult problems. First, because the quality of the video is low, the objects in each frame cannot be clearly identified with existing object recognition models. We will discuss an example in the experiments sections. Second, if we were to retrain an object detection model, because the cameras are placed at different locations at each intersection, SafeCross would have zero scalability because we would have to train a different object detection model for every intersection. The reasons mentioned above persuaded us not to use an object detection model to recognize vehicles in the scene.

2) Our Approach: The above problems ultimately led us to use background subtraction to track moving objects in each frame. Because we only want to identify the moving objects in the scene, background subtraction is best both for detecting moving objects and scalability. To further improve the accuracy of the method, we use a dynamic background. The background subtracted from each frame is constantly updated.

In Figure 3(a), an unprocessed frame is shown. Because we only want to identify the moving vehicles in each frame, an unprocessed frame contains too much noise and useless information, such as the background, the sky, stopped vehicles, and pedestrians.

In Figure 3(b), we give an example of a snapshot of the intersection that has been processed by the Background Subtraction method. The noise from the low-quality cameras is reduced dramatically by performing opening morphology, erosion then dilation, on the entire scene. We cannot feed this image to the training model yet, because although we have identified the moving vehicles, we still need to map them into a 2-D representation of the intersection.

In Figure 3(c), we map the vehicle location information from b to a 2-D representation of the intersection. This step is crucial for the model to learn the spatial information of the moving vehicles. This is the last step in the pre-processing stage. Once we obtain a 2-D representation of the intersection, we can finally feed it to the training model.

To summarize, we reduced the amount of information contained in a raw frame to retain only relevant information. We further reduce the number of parameters the training model needs to learn by remapping from a 3-D space into a 2-D plane.

By using erosion on the processed scene, we can eliminate most of the noise from the camera because they are small and do not have any structure; however, since erosion is applied to the entire scene, the structures of the detected vehicles are also weakened. Therefore, we need to apply dilation to re-strengthen the structures. Dilation strengthens the structure of all objects on the scene. Since we used erosion to eliminate most of the noise, dilation will not strengthen the noise that has already been eliminated; thus, only the objects we want to detect are strengthened.

The ultimate goal of SafeCross is to create a model that would learn when drivers should make a left turn at an intersection with reduced visibility. So it would be easier if there were less irrelevant information for the model to learn. To achieve this, we can reduce the number of pixels in the processed image while still maintaining the objects' structure.

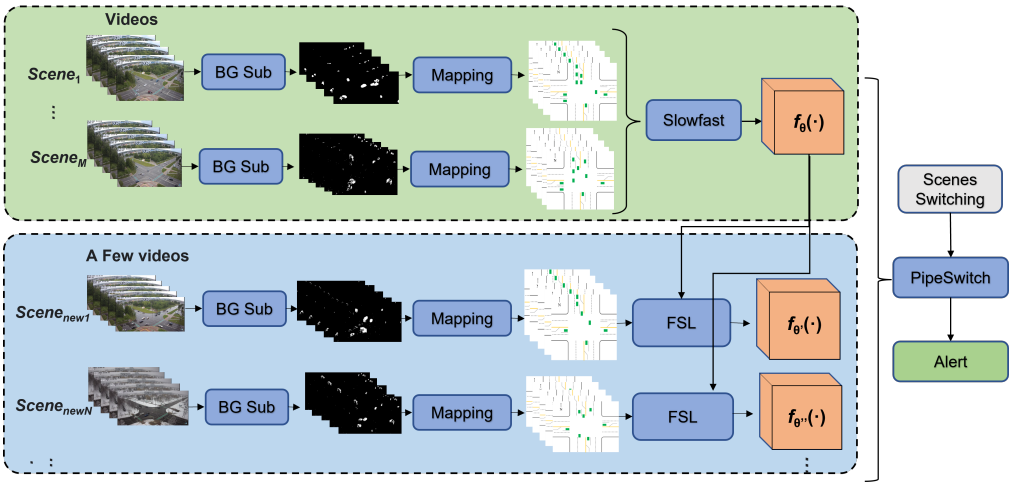


Fig. 4. An overview of SafeCross framework. For all model training, the background subtraction (BG Sub) processing of the video is performed to get the training data. The training of the basic model uses daytime data. And the training of the few-shot data model is obtained through few-shot learning (FSL) training based on the daytime model parameters. For switching between multiple models, real-time Pipeswitch is triggered when the scene changes to realize real-time adaptive safety detection.

C. Video Classification

Model training takes the 2-D representation images inputs and outputs whether or not the vehicle should make the turn.

1) Challenges: There are two challenges with model training. First, while each 2-D representation of the intersection provides spatial information, it does not provide any temporal information across consecutive frames. Without sufficient temporal information, it is difficult for the model to make an accurate prediction. Second, while we do have enough training data to train a good model for normal weather conditions, we do not have enough data to train special weather conditions from scratch. For the second challenge, we will devote the following subsection to discussing our approach.

2) Our Approach: First Challenge: To obtain temporal information across consecutive frames, we apply the SlowFast network to train the basic model. Slowfast [21] network is a single stream architecture that operates at two different frame rates [21]. Figure 5 shows that the SlowFast network. This network takes a series of consecutive frames, interprets the spatial and temporal characteristics of the frames, and learns a model.

We construct the tasks for meta-training as follows: during the meta-training, we have a total of M scene sets that are denoted as $\{S_1, S_2, \ldots, S_M\}$, for every $S_i, i \in \{1, \ldots, M\}$, we construct the corresponding task $\mathcal{T}_i = (\mathcal{D}_i^{train}, \mathcal{D}_i^{test})$ ($\mathcal{D}_i^{train}$ and $\mathcal{D}_i^{val}$ represent the training and test sets in the task $\mathcal{T}_i$). We first split the videos from S_i into many consecutive segments with length $t+1$, so that a segment video set $(V_1, V_2, \ldots, V_{t+1})$ would be generated, and the first t frames would be regarded as the input x, the last frame as the output y ($x = (V_1, V_2, \ldots, V_t)$, $y = V_{t+1}$). So that K input/output pairs (x, y) would be collected for meta-training. In $\mathcal{D}_i^{train}$, every time we randomly sample K (input,output) pairs from the tasks list $\mathcal{T}_i$ as $\mathcal{D}^{train} = \{(x_1, y_1), \ldots, (x_K, y_K)\}$. The result is a prediction model $y = f(x; \theta)$. In the test and validation set, we randomly sampling K input/output pairs to form the corresponding test data $\mathcal{D}_i^{test}$.

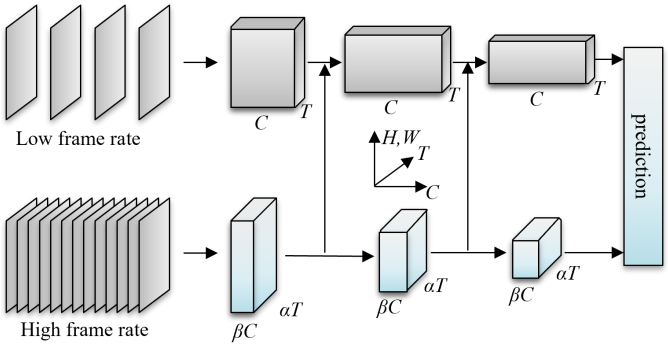


Fig. 5. An overview of Slowfast network. It has a low frame rate, low temporal resolution Slow pathway and a high frame rate, $\alpha \times$ higher temporal resolution Fast pathway. The Fast pathway is lightweight by using a fraction (β, e.g., 1/8) of channels. Lateral connections fuse them.

D. Few shot Learning

1) Challenges: As mentioned in the previous sub-section, we do not have sufficient training data to train special case models from scratch. This is due to the scarceness of the data, difficulties in collecting data, and difficulties in labeling the data.

2) Our approach: To train models for special cases, we apply a few-shot learning technique. On a high level, we take a training case for the special case, find the model with the most similar set of parameters and results, and use transfer learning to train the model for the special case. Specifically, in a N-way, K-shot learning problem, each episode of training cases contains a support set and a query set. The support set consists of K samples each from N unseen categories (K is typically a small integer $<= 10$), the algorithm then has to

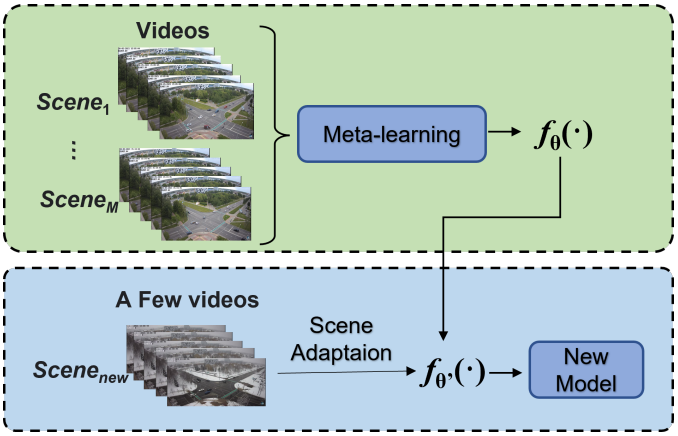


Fig. 6. An overview of our proposed problem setting. During training step (in the up side figure), we have various videos frame collected from M different scenes. We construct meta-training data sets $\mathcal{D}^{train}$ and apply the MAML method to obtain a model $f_\theta(\cdot)$ with parameter θ. When the target scenes (down-side figure), we obtain a small number of video frames from this target scene, and produce a new model $f_{\theta'}(\cdot)$ where the model's parameters θ' are adapted to this rare scene.

determine which of the support set classes each query video belongs to. Here the episodes are randomly sampled from a larger collection of data, we call it the meta sets. The few-shot learning problem here could be solved through learning a distance function $\phi(f_\theta(x_1), f_\theta(x_2))$, here x_1 and x_2 are two video instances that drawn from C_{train}.

Model-Agnostic Meta-Learning (MAML) learns an initialization for the parameters θ of a model f_θ, for a new task, a good model for that task can be learned with only a small number of gradient steps and samples via two optimization loops:

- Outer Loop: Updates the meta initialization of the model parameters, which is used to make the model enable to fast adapt to new tasks.
- Inner Loop: Task adaptation based on the meta-initialization, it performs a few gradient updates over the k labeled examples (the support set) provided for adaptation.

Here, for the task $\mathcal{T}_i$ in support set, let θ_i^k represents the parameter θ after k gradient updates, let $\theta_i^0 = \theta$. In the inner loop, during each update, we compute:

$$\theta_i^k = \theta_i^{k-1} - \alpha \nabla_{\theta_i^{k-1}} \mathcal{L}(f(\mathcal{T}_i; \theta_i^{k-1})) \tag{1}$$

where α is the inner loop learning rate on weights θ, and $\mathcal{L}$ is the loss on the (support set) of $\mathcal{T}_i$ after k-1 inner loop updates. After totally k inner loop updates, the meta initialization update in the outer loop is computed on $\mathcal{T}_i$ in query set, the model parameters θ are then updated, which can be formulated as:

$$\theta = \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \in p_{train}} \mathcal{L}(f(\mathcal{T}_i; \theta_i^k)) \tag{2}$$

where β is the outer learning rate of meta-initialization weights θ.

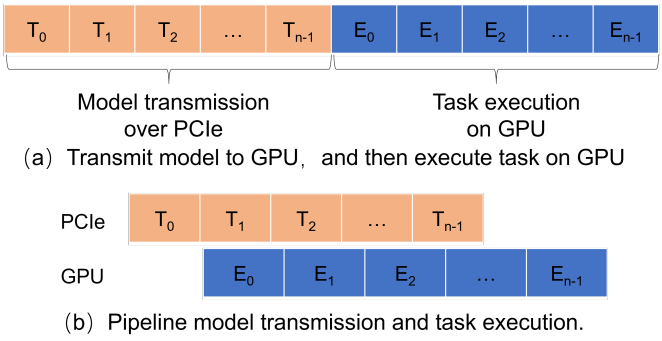


Fig. 7. An overview of PipeSwitch. PipeSwitch pipelines model transmission and task execution. The example shows an inference task that only has a forward pass in task execution.

E. Model Switching

Now that we have trained multiple models for different weather conditions, we need an algorithm to switch between models efficiently.

1) Challenges: The size and location of the *danger zone* vary for different models. Therefore, we need a model-switching algorithm to switch models based on the detected scene; however, non-optimized model switching is slow and tedious for GPUs. Research have found that methods based on task scheduling and context switching have a huge overhead when applied to GPUs. This is because Service Level Objectives (SLO) must be met during the Deep Learning (DL) inference process, which will take tens to hundreds of milliseconds. Taking ResNet as an example, the DNN model in the GPU is switched and preloaded until the first inference request is completed. The average model switching time for ResNet is several seconds [22].

2) Our Approach: We implement the Pipeswitch [22] method to manage model switching. Pipeswitch is a pipelined task switching method that multiplexes GPU resources. It is observed that the DL model has a hierarchical structure, and its reasoning task is calculated layer by layer from the front to the back. Therefore, there is no need for the GPU to load all of the data before starting layer-by-layer inference. Instead, the model can infer while loading hierarchical data, just as Figure 7 shows. with Pipeswitch, the overhead of model switching on the GPU can be dropped down to the millisecond level.

3) Optimal model-aware grouping: Specifically, the algorithm uploads data at the granularity of each layer and blocks one layer of calculation before uploading. Such an operation is bound to bring two aspects of costs. The first is that the amount of data varies between layers, but the cost of the layer with a small amount of data is similar to that of the larger layer, which is obviously not reasonable enough. The second is the synchronization cost of calculation and transmission, which is when the calculation knows to be calculated The data is already in place and it takes time. Based on these two considerations, the algorithm performs layered grouping processing. Use pruning method to achieve. This is also

TABLE I
OVERVIEW OF DATASET.

Scenarios	Daytime	Rain	Snow
Time	6h	1h	3h
Segments	1966	34	855
Segments length	32 frames		
Frame rate	30Hz		
Frame resolution	1376x776 pixels		
Classes	turn_left & no_turn_left		

the basis for synchronizing the flow ways and fast ways of Slowfast.

IV. SYSTEM IMPLEMENTATION

In this section, we will describe the SafeCross framework implementation in detail.

A. Experimental Setup

Hardware configuration: We used an NVIDIA GPU Workstation as a roadside unit to deploy the SafeCross framework. The device deploys Intel Xeon E5-2690 v4 and four GeForce RTX 2080 Ti graphics cards, and the frequency is 2.6GHz. It is equipped with 14 cores, 64GB of memory, and the operating system is 18.04 LTS. It should be noted that, because Pipeswitch in the model switching part cannot support multiple GPUs processing, we only used one GPU in the experiment.

Software configuration: We implement SafeCross in Python 3.7, and use pytorch 1.3.0 as a deep learning library to support Pipeswitch and Slowfast. We use Cuda 10.1 parallel computing architecture to accelerate model training and inference, and we use cudnn 7.6.4 for GPU acceleration of deep neural networks. Another software environment includes torchvision-0.4.1, scipy-1.3.2.

B. Dataset

The existing data set cannot meet the requirements of our experiment. Because of the lack of a suitable data set, we found a live surveillance video of a traffic intersection from Belarus on the Internet. By collecting markers, we sorted out the data needed for training. Below we will introduce the production details of the dataset.

Data overview: We obtained aerial traffic intersection data from a live monitoring of a traffic intersection in Belarus. The live video can be accessed through the Youtube link: https://www.youtube.com/watch?v=RIBTbuT0WkQ. As Figure 3(a), at the intersection, vehicles are allowed to go straight and turn left at the same time. The data set we collected contains three different weather patterns: sunny, rain, and snow. We selected a total of 2855 segments of data with a time span of 180 days. Table I shows the specific details of the data.

Data processing: First, we manually divided and labeled the videos into four different categories, based on driver behaviors, for each weather conditions:

- Left Turn without blind area.
- No Left Turn without blind area.
- Left Turn with blind area.
- No Left Turn with blind area.

For the turn_left category, we select the vehicle that is turning left as the object of interest for segment division, and select the end frame of the segment as the keyframe. In the keyframe, the vehicle's left front wheel is exactly on the lane line. For the no_left_turn category, we choose vehicle waiting to turn left as the object for segment division. Each segment has 32 consecutive frames. If there is a big car on the opposite side in a segment, we consider this segment to be a segment with a blind area, otherwise it is a segment without a blind area.

C. Implementation

Software implementation and specific settings are as follows:

Background Subtraction: The crossroad video is processed frame by frame into 2D data with the background removed, and each frame records the position information of the moving object. Among them, we use mathematical morphology algorithms to reduce noise interference. To further reduce the interference of irrelevant data, we segmented the middle to the upper right corner of the video frame.

Slowfast: We use 32 consecutive 2D frames for each training case. Among them, in Slowfast's Flow section, one frame of data is selected for every eight frames, so a total of 4 frames are obtained. Every frame is included in Slowfast's Fast section, resulting in a total of 32 frames. Then, the four frames of Slow are combined with the 32 frames of Fast to form a new 36 frames of input data for model training. The backbone of Slowfast_r50_4x16x1_256_SafeCross, we used, is ResNet50, and the model trained from scratch using daytime data. We use 80% data as the training data, 10% as test data, and 10% as validation data.

Few-shot learning: We used the daytime model as the pretrained model and used the Few-shot method to train the rain model and snow model.

Pipeswitch: We only use one GPU to carry out the model switching experiment.

V. PERFORMANCE EVALUATION

In this section, we compare and contrast the different methods that worked and did not work.

A. Detection Methods Comparison

In order to determine the most effective detection method, we experimented with a few that has had success on multiple other projects. We will compare and contrast the different techniques we used, including YOLOv3(A Machine Learning model), tracking optical flow, and background subtraction. The data-set we are using is a custom data-set that only includes videos of an intersection. To compare the results of different methods, we chose a specific scenario where there is a vehicle attempting to make a left turn, and the driver has partially obstructed view. In this specific scenario, there is an incoming vehicle on the other side of the traffic in the danger(obstructed

view) zone. We will evaluate the different methods based on if it can identify the vehicle in the danger zone or not and its execution time.

It is important to note that all of the results are based on the same frame.We will show a summary at the end of the experimentation section.

We will start by re-iterating the scenario as shown in 8(a). In this scenario, the vehicle circled in green is attempting to make a left turn; however, the driver cannot determine if it is safe to make that turn. This is because the driver's view of the traffic is partially blocked. In this scenario, there is a vehicle in the danger zone as defined previously; therefore, the driver should not make the left turn.

Next, we will show the results of Sparse Optical Flow, Dense Optical Flow, YOLOv3, and Background Subtraction. We will show whether of not each method can identify the vehicle in the danger zone, circled in red, and each method's execution time.

As shown by the results of Sparse Optical Flow in Figure 8(b), this technique is not ideal. Sparse Optical Flow detects too many edges from the environment and misses too much of the traffic. This is partially due to noise introduced by the low quality camera. While Sparse Optical Flow has an average execution time of only 6.4 milliseconds per frame, it cannot correctly identify the vehicle in the danger zone. If we were to apply Sparse Optical Flow in this scenario, it would cause a crash.

Next, we will demonstrate the detection results from Dense Optical Flow, as shown in Figure 8(c). Dense Optical Flow did successfully identify the vehicle in the danger zone; however,the average execution time process a frame is approximately 224.20 milliseconds, an almost 40 times increase comparing to Sparse Optical Flow.

Moving on to more modern techniques, we implemented YOLOv3 to recognize and extract objects from the scene. We picked YOLOv3 from the categories of object recognition because it is one of the easiest to implement and one of the fastest algorithms available.

As Figure 8(d) shows, YOLOv3 cannot identify the vehicle in the danger zone. This is partially due to the distance, angle, and low-quality of the camera; however, even after we re-trained the weights, we cannot get better results. Adding on to that, YOLOv3 an average execution time of 256.40 milliseconds per frame. Therefore, we did not choose YOLOv3 as our detection method.

Eventually, we dawned on a solution. For our purpose to guide the left-turning driver, with the camera placed at such a faraway distance, we do not necessarily need to recognize the objects. Instead, anything that moves, or introduces pixel changes, should be recorded. It is slightly different from optical flow because they track edges in objects rather than pixel changes. To see pixel changes, background subtraction is the simplest and fastest method. With Background subtraction, we can easily eliminate noise caused by low-quality camera; we can easily track a change in the background; we can easily differ between a moving vehicle and a stationary one.

TABLE II
EXECUTION TIME OF VARIOUS DETECTION METHODS.

Methods	BGS	Sparse OF	Dense OF	YOLOv3
Time	0.74ms	6.43ms	224.20ms	256.40ms
Detected	Yes	No	Yes	No

TABLE III
ACCURACY OF DIFFERENT SCENES VIDEO CLASSIFICATION.

Types	**Top1_acc**	**Mean_class_acc**
Daytime	0.9630	0.9667
Snow	0.9416	0.9510
Rain	0.8518	0.8636

The resulting image, Figure 8(e), has an average execution time of 0.74 milliseconds per frame. This is much faster than Dense Optical Flow, and it produces a slightly better result. Therefore, Background subtraction is our more effective detection method.

To conclude the results, we will demonstrate a Table II of the execution time of each of the detection methods.

B. Accuracy of classification models

We train the video classification models on daytime, snow, and rain, respectively. Among them, we use the daytime data with a large amount of data to train the Slowfast basic model, while the snow and rain models with a small amount of data are trained through few-shot learning. Training videos are classified into two categories, class 0 is marked as danger to turn left, and class 1 is marked as safe to turn left. And both classes include blind and no-blind videos. We randomly divide the data into three parts train:val:test=8:1:1.

The results of Top_1 classification accuracy and Mean_class_acc of the three models are gathered in the table III. From the table, the Top1_accuracy of the daytime scene of the basic model reached 0.9630, and the Mean_class_acc reached 0.9667. The main reason why the classification accuracy is so high is that the number of classification categories is small (two classes). We also found that the accuracy of snow and rain is lower than that of daytime, and the rain dataset achieves the lowest accuracy, but is still get a high result, 0.8636. The above findings indicate that SafeCross can achieve higher accuracy classification in daytime scenes. Rain and snow scenes can also achieve high classification accuracy. However, to achieve more ideal results, more training data is needed.

To verify the effectiveness of SlowFast on SafeCross dataset, we compare it with other two famous video classification models, Convolutional 3D (C3D) [23] and temporal segment network (TSN) [24]. C3D is one of the most famous video classification methods, which uses SVM to classify video. And TSN uses ResNet as the backbone, and combines a sparse temporal sampling strategy and video-level supervision to enable efficient and effective learning. We compare slowfast_r50_4x16x1_256e, c3d_sports1m_16x1x1_45e and tsn_r50_1x1x3_75e on our daytime dataset and the result are as the Table IV shows. From it, we can find that C3D

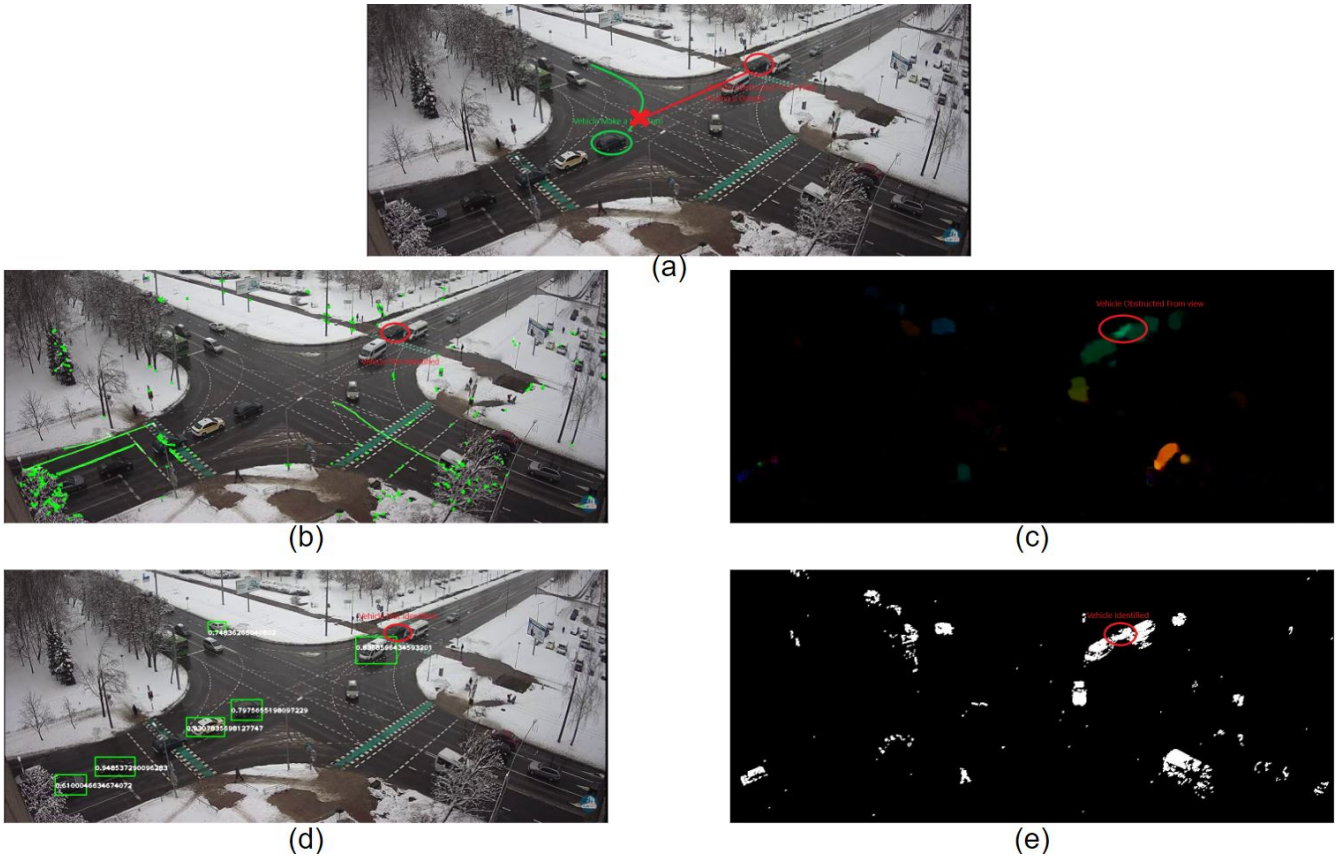


Fig. 8. (a) Original Image (b) Failure of Sparse Optical Flow; (c) Success of Dense Optical Flow; (d) Failure of YOLOv3; (e) Success of Background Subtraction

TABLE IV
ACCURACY OF DIFFERENT CLASSIFICATION METHODS ON DAYTIME DATASET.

Models	Top1_acc	Mean_class_acc
slowfast_r50_4x16x1_256e	0.9630	0.9667
c3d_sports1m_16x1x1_45e	0.9644	0.9340
tsn_r50_1x1x3_75e	0.8855	0.7538

TABLE V
ACCURACY OF FEW SHOT LEARNING.

Experiments	Top1_acc	Mean_class_acc
snow_with_few_shot_learning	0.9416	0.9510
snow_without_few_shot_learning	0.8889	0.8648
rain_with_few_shot_learning	0.8518	0.8636
rain_without_few_shot_learning	0.5455	0.5833

model achieves the highest top_1 accuracy, but the slowfast got the highest mean class accuracy, which is the most important indicate what we want.

To verify the effectiveness of few shot learning. We carry out the ablation study. In our experiments, we design the model training of few shot datasets, snow and rain dataset, with or without with few-shot learning. For these with few-shot learning, we use the model trained by daytime dataset as pretrain model. And for these without few shot learning, we train them directly. Finallywe compare the result of Top1_acc and Mean_class_acc of four experiments, which are shown in the Table V. From it, we can find that both the snow and rain datasets achieve higher Top1_acc and Mean_class_acc with few shot learning method than that of without few shot learning method, which verified the effectiveness of few shot learning.

C. Model Switching

In order to compare the effectiveness and real-time of Pipeswitch in model switching, we carried out an experiment comparing the run-time with and without Pipswitch on one GeForce RTX 2080 Ti graphics. In detail, without Pipeswitch means it stops one model's task in the GPU and then starts a new model's task, which is also called Stop-and-start.

In the experiment, we compared three different types of models: SafeCross, ResNet152 [25], and Inception_v3 [26], and recorded the time when the client sent a task switching request to the GPU server and the time when the GPU switched back to feedback and calculated the task completion time. Among them, SafeCross chooses to switch between day and snow models. The experimental results are shown in the Table VI.

Experiments prove that Pipeswitch allows for much faster

TABLE VI
COMPARISON BETWEEN OF DIFFERENT MODELS SWITCHING.

	Slowfast 4x16,R50	ResNet152	Inception_v3
End-start	5614.75ms	4081.15ms	3612.25ms
Pipeswitch	6.06ms	5.30ms	4.32ms

model switching, and the model switching delay was reduced to the millisecond level (less than 10ms), which is sufficient for real-time model switching tasks. The overhead without Pipeswitch mainly comes from the context initialization of CUDA and the first-time library load in PyTorch [27].

D. Throughput Comparison

In order to verify the improvement of the throughput by SafeCross in the scenes with blind zones, we counted the video segments with blind spots from the 10 hours video data in the daytime, rain, and snow scenes and made the statistics and made a test set. It includes two classes. Class 0 has 32 video segments, meaning there is a car in the blind zone and should not turn left; class 1 has 31 video segments meaning there is no car in the blind area and can turn left.

We use SafeCross to classify the test set data. The result shows that the classification accuracy is 1. This means that 32 scenes with no car in blind zones are judged safe to turn left without waiting; 31 scenes with a car are considered not to turn left and need to wait. In the 63 scenes with blind spots obtained by statistics, SafeCross judged 32 of them should turn, so the traffic throughput of the left turn is significantly increased by 32/63.

VI. DISCUSSION AND FUTURE WORK

A. Discussion

In this part we will discuss the deployment of SafeCross and some observations:

Observation 1: For the detection of moving objects, especially the recognition of moving objects with non-traditional viewing angles and low pixel values, methods based on YOLOv3 and Sparse Optical Flow are not always a good choice and mobile detection is more competent. While Dense Optical Flow shows better results, its high execution time is not always ideal for a fast changing scenario.

In Section 5.1, we evaluated the recognition effect of the vehicle (Figure 8), YOLOv3 recognition has low recognition accuracy or even omission because the camera is placed far away from the objects and placed at a skewed angle. For Optical Flows, due to the low quality of the camera with too much noise, we could not generate an accurate and usable output. Background subtraction can monitor all the moving actions of vehicles and has the lowest execution time.

Observation 2: Although the overall effect of video classification in multi-classification is not good; when there are few categories, such as two-class classification, higher classification accuracy can be obtained.

In Section 5.2, We used a large amount of data to train the two-classification model of Slowfast video classification. The Top_1 accuracy of daytime reached 0.9630 and the Mean_class_acc reached 0.9667. This is significantly higher than the best classification accuracy of Kinetics-400 [28]. However, the accuracy of the binary classification model trained on the video data of snow and rain scenes with a small amount of data decreases significantly. These comprehensive descriptions show that video classification is a good choice when you have a large number of two-category data sets.

B. Future work

SafeCross has realized the real-time adaption danger warning, but there are still many points that can be optimized. The areas to be improved in the future include:

- Change offline danger warning to online danger warning. The current SafeCross is the processing and mining of historical data. For the time being, it cannot realize online early warning of video data.
- Expand the research scope of vehicles coming at intersections. Now our data is based on the danger warning of vehicles turning left in one direction. How to achieve simultaneous warning in four directions requires further research.
- Expand the research scope of blind spot. SafeCross is an architecture to solve the blind spot problem, and the left turn is just a specific scenario of it. Is SafeCross suitable for blind spot pedestrian warning? Is it suitable for highway? It is worth doing further research.
- Increase the number of extreme scenes. The data we are using now contains three scenarios: daytime, rain, and snow. Considering the diverse scenes of traffic intersections, model training can be carried out based on the SafeCross architecture for each scene, but it should be noted that the more training data, the better.
- Based on the SafeCross architecture, build prototypes in real life, and further explore the pain points of the SafeCross architecture based on actual applications, and make targeted optimizations.

VII. CONCLUSION

In this paper, we introduced the SafeCross framework by leveraging Slowfast video classification and Few-shot learning to effectively train models to ensure the safety of drivers. Experiments show that SafeCross is an effective implementation to ensure the safety of drivers and to increase traffic throughput. Its design allows it to be easily implemented in existing infrastructure and be easily scalable.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers for their helpful comments. This work is supported by the National Natural Science Foundation of China under Grant No.62072146, National Key RD Program of China under Grant No.2019YFB2102101, The Key Research and Development Program of Zhejiang Province under Grant No. 2021C03187.

REFERENCES

[1] N. L. Review, "What if my car accident was caused by a blind spot?" [EB/OL], https://www.natlawreview.com/article/what-if-my-car-accident-was-caused-blind-spot Accessed January 31, 2022.

[2] NHTSA, "Crash factors in intersection-related crashes: An on-scene perspective," [EB/OL], 2010, https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811366 Accessed January 31, 2022.

[3] N. Saunier, T. Sayed, and C. Lim, "Probabilistic collision prediction for vision-based automated road safety analysis," in *2007 IEEE Intelligent Transportation Systems Conference*. IEEE, 2007, pp. 872–878.

[4] L. Meng, W. X. Han, and S. Ke, "Traffic conflict identification technology of vehicle intersection based on vehicle video trajectory extraction," *Procedia Computer Science*, vol. 109, pp. 963–968, 2017.

[5] C. Park, J. S. Park, and D. Manocha, "Fast and bounded probabilistic collision detection for high-dof trajectory planning in dynamic environments," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 980–991, 2018.

[6] J. Kim and D. Kum, "Collision risk assessment algorithm via lane-based probabilistic motion prediction of surrounding vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2965–2976, 2017.

[7] L. Tan, X. Dong, Y. Ma, and C. Yu, "A multiple object tracking algorithm based on yolo detection," in *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. Beijing, China: IEEE, 2018, pp. 1–5.

[8] Y. Fu, C. Li, T. H. Luan, Y. Zhang, and G. Mao, "Infrastructure-cooperative algorithm for effective intersection collision avoidance," *Transportation research part C: emerging technologies*, vol. 89, pp. 188–204, 2018.

[9] R. Ke, Z. Li, J. Tang, Z. Pan, and Y. Wang, "Real-time traffic flow parameter estimation from uav video based on ensemble classifier and optical flow," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 54–64, 2018.

[10] Z. Wang, X. Sun, W. Diao, Y. Zhang, M. Yan, and L. Lan, "Ground moving target indication based on optical flow in single-channel sar," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 7, pp. 1051–1055, 2019.

[11] Y. Xin, J. Hou, L. Dong, and L. Ding, "A self-adaptive optical flow method for the moving object detection in the video sequences," *Optik*, vol. 125, no. 19, pp. 5690–5694, 2014.

[12] G. Shi, J. Suo, C. Liu, K. Wan, and X. Lv, "Moving target detection algorithm in image sequences based on edge detection and frame difference," in *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2017, pp. 740–744.

[13] H. Sajid and S.-C. S. Cheung, "Universal multimode background subtraction," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3249–3260, 2017.

[14] S. Jiang and X. Lu, "Wesambe: A weight-sample-based method for background subtraction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 9, pp. 2105–2115, 2017.

[15] B. Garcia-Garcia, T. Bouwmans, and A. J. R. Silva, "Background subtraction in real applications: Challenges, current models and future directions," *Computer Science Review*, vol. 35, p. 100204, 2020.

[16] L. Li, Z. Wang, Q. Hu, and Y. Dong, "Adaptive nonconvex sparsity based background subtraction for intelligent video surveillance," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4168–4178, 2020.

[17] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, "Segflow: Joint learning for video object segmentation and optical flow," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 686–695.

[18] P. W. Patil and S. Murala, "Msfgnet: A novel compact end-to-end deep network for moving object detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 11, pp. 4066–4077, 2018.

[19] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, no. 4, pp. 681–694, 2020.

[20] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[21] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6202–6211.

[22] Z. Bai, Z. Zhang, Y. Zhu, and X. Jin, "Pipeswitch: Fast pipelined context switching for deep learning applications," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. Canada: USENIX Association, Nov. 2020, pp. 499–514.

[23] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[24] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*. Springer, 2016, pp. 20–36.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[28] Open-mmlab, "slowfast-model-zoo," [EB/OL], 2022, https://github.com/open-mmlab/mmaction2/blob/master/configs/recognition/slowfast/README.md Accessed January 31, 2022.