Comma AI Challenge: Predicting Car Speed Using Optical Flow with CNNs for Autonomous Driving

Luis Ahumada
George Washington University
lahumada@gwu.edu

Sam Cohen
George Washington University
cohen_samuel@gwu.edu

Rich Gude
George Washington University
richgude@gmail.com

Abstract

Neural networks are being used to help accelerate the development and commercialization of autonomous vehicles. In this paper, we use convolutional neural networks (CNNs) combined with optical flow to predict vehicle speed from a video of that vehicle driving. The videos are taken from the Comma AI speed challenge. Our process aims to improve upon an existing solution to the Comma AI speed challenge published by Millin Gabani. We developed two side-by-side CNN models for predicting speed from the outside quarters of pairs of images. We find that this technique does not achieve the desired performance, yet it is a unique way to tackle the problem of speed prediction and may have usefulness with enhanced experimentation.

I. Introduction

As autonomous driving technology continues to progress, public and private sector institutions are pursuing different technological solutions to produce a safe autonomous vehicle. These technologies need to be able to sense the car's external environment, such as other cars, people around the car, and traffic lights and signs, and regulate a car's internal environment, such as the speed of the car and the angle of the steering wheel, based on these external factors. Current technological solutions in both hardware and software include lidar, advanced cameras, and neural networks. Neural network solutions are promising, given the advances in the field of machine learning in the past few decades. One issue with using neural networks is that they require large quantities of training data to be trained appropriately.

One company creating autonomous vehicle technology is Comma AI[1]. Comma AI produces a open-source driver assistance system that can perform such functions as cruise control, lane centering, collision warning, and driver monitoring for asleep drivers[2]. This software can be connected to certain make and models of mainstream commercial vehicles through hardware produced by the company called *comma two*. Given their open-source nature, Comma AI has a programming challenge that they request applicants to their jobs to complete[3]. This challenge asks participants to predict the speed of a car from a video of that car. The challenge and the team's attempt at that challenge is the subject of the paper.

II. Related Work

We found prior attempts to complete the Comma AI speed challenge and based our work on improving the performance of previous solutions. The solution we based our work on is

[1] https://comma.ai/
[2] https://github.com/commaai/openpilot
[3] https://github.com/commaai/speedchallenge

published by Millin Gabani on his Github[4]. This approach uses analysis of the optical flow of a pair of images and convolutional neural networks (CNNs) to predict the vehicle speed. Optical flow is "the pattern of apparent motion" between two successive images[5]. It assumes that the brightness and intensities of the pixels do not change from one image to another, but the physical location of those pixels changes. As such, optimal flow analysis attempts to find the direction of movement of each pixel or a group of pixels in two images. For optical flow, there are two main methods: the Lucas-Kanade method and the Farneback method. The Lucas-Kanade method is used in sparse optical flow analysis where one is concerned with the movement of a few key points between the images. The Farneback method is used to calculate the movement of all the points between images. Gabani uses the Farneback method in his solution to the Comma AI speed challenge for a more comprehensive analysis, as do we. For the CNN model, Gabani relies on an NVIDIA paper on end to end learning for self-driving cars[6].

Other methods to find the optical flow of a pair of images exists. One method that has gained popularity in the past few years is using a CNN to create the optical flow[7]. This method is implemented by NVIDIA in their CNN called FlowNet 2.0[8].

III. Dataset

The training dataset used for the challenge consists of a 17 minute video of a car driving. There are 20,400 total frames with 20 frames per second of video. The camera used to film the

---

[4] Gabani, Millan. (September 1, 2017). *deeps*. GitHub. https://github.com/millingab/deeps/blob/master/Full%20Article.md.

[5] Mordvintsev & Revision. (2013). *Optical Flow*. OpenCV. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html.

[6] Bojarksi et al. (April 25, 2016). *End to End Learning For Self-Driving Cars.* Arxiv. https://arxiv.org/pdf/1604.07316v1.pdf.

[7] Fischer et al. (May 4, 2015). *FlowNet: Learning Optical Flow with Convolutional Networks.* Arxiv. https://arxiv.org/pdf/1504.06852.pdf.

[8] https://github.com/NVIDIA/flownet2-pytorch

video is located on the rearview mirror of the car. Throughout the video, the car moves through different environments, such as a highway and a more suburban area. The time-of-day of the video is close to dusk, so there are significant shadows throughout. This mp4 file is accompanied with a text file containing a speed of the vehicle at each frame. The speed is rounded to the nearest one-millionth, so it is a very accurate speed measure.

The test dataset consists of a 9 minute video of a car driving. There are 10,798 frames with 20 frames per second of video. There is no accompanying speed values, as our model is intended to predict speed in the test video.

IV. Training

1. Pre-Processing

We implemented a pre-processing pipeline that includes re-sizing, changing the brightness, cropping, and finding the optical flow between a pair of images. Then, this optical flow image gets re-sized once more before entering the model. Appendix A shows the pair of images going through multiple steps in this pipeline, including the original re-sized images, the brightened and cropped images, and the optical flow images according to our implementation in the model. Appendix B shows an original 480X640 image going through the a similar pre-processing pipeline. The reader is better able to make out the contents of each image, so it is recommended to visit Appendix B to get a better understanding of the image transformations.

For our images, after we read them in from the mp4 file, we re-sized the images from 480X640 to 60X320. This was done to save storage space and because height is not as important to the CNN to calculate the movement from one frame to the next. Next, we pair two images together in order to find the change in movement between them. We use a step size of five images between each pair to insure we have enough movement for the model to pick up. Next, we

changed the brightness of each pair of images using a function created by Gabani. We need to change the brightness of the images because optical flow depends on the assumption that the intensity of the pixels remains constant from one image to another and that the only change is the location of the pixels. We next split each image into two images, one of the left quarter of the image and one of the right quarter of the image. Therefore, we take the middle half of the image out. This part of the image often contains other cars and objects that are likely to be moving at other speeds, which may throw off the prediction. Finally, we run the dense optical flow function, also borrowed from Gabani's code, on the left image pairs and the right image pairs. We then re-size the optimal flow images to 60X80X3.

2. Model

For our model architecture, we diverged from Gabani's approach and used two identical CNN models. We put the left optical flow image through one CNN model and the right optical flow image through the other CNN model and took the maximum prediction from each model as our final predicted speed. We are taking the maximum speed because it is likely that other cars are present in one quarter of the image and not the other quarter, depending on where the other side of the road is. We think that other cars in the pairs of images will decrease the predicted speed because these cars are moving as well, leading us to see less of a movement in pixels between images. Therefore, we take the maximum of the two quarters.

The model architecture is show below in Figure 1.

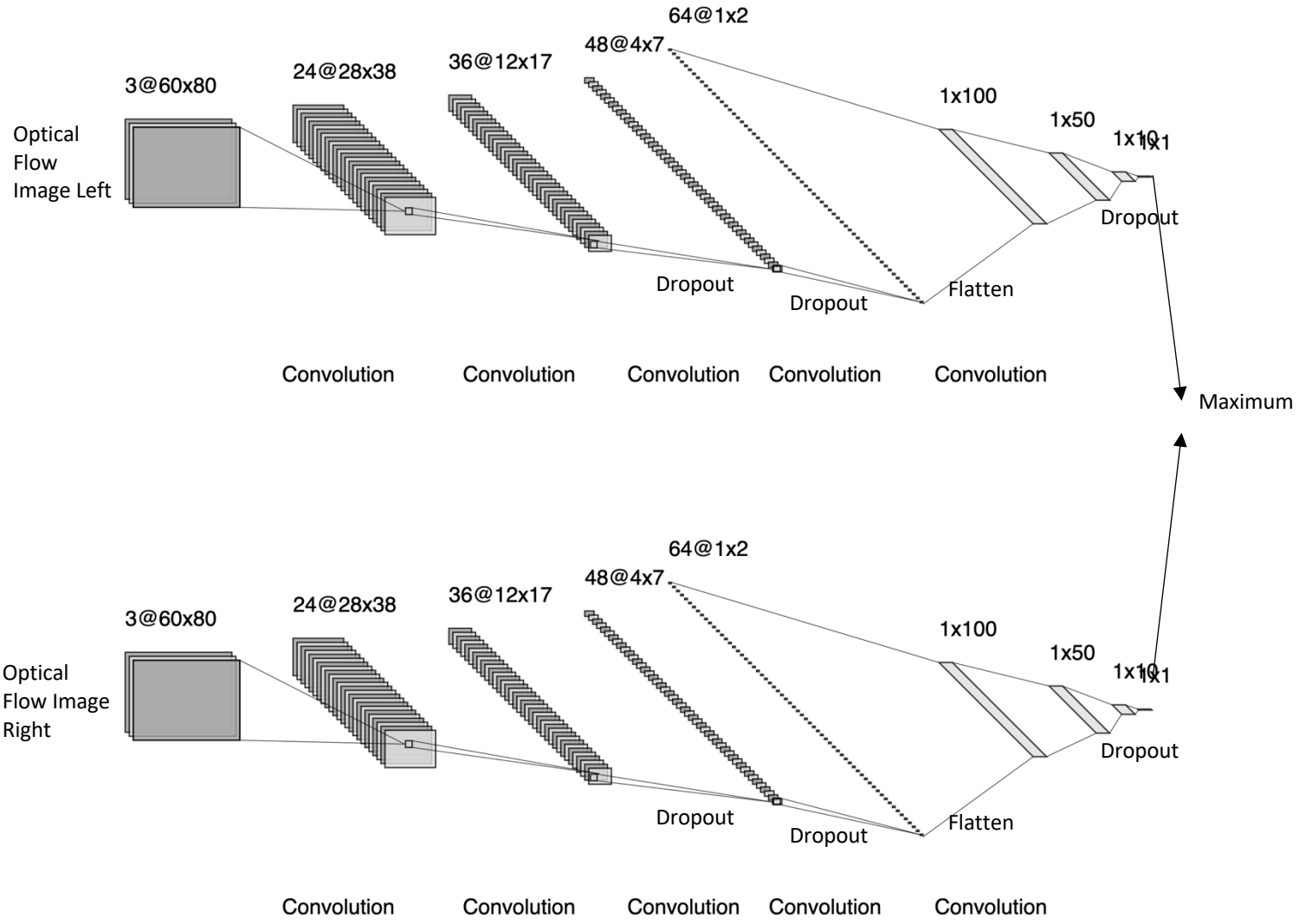


Figure 1. CNN Network Architecture

The activation function we used in each layer was the 'elu' activation function. We used Adam as our optimizer with a learning rate of 10^-4. We used Mean Squared Error as our performance metric. Finally, we used a batch size of 10 for our mini-batches.

V. Results

The Comma AI Speed Challenge calls for participants to submit their predicted speeds on the provided test set to the company, who will then score their results. We did not provide Comma

AI with our predicted speed values and therefore did not get a score. Yet, we did separate our given training data into a train set, a test set, and a validation set and all three of these sets have labels. So we were able to get a Mean Squared Error (MSE) metric from our resulting model, although we have nothing to compare it to. Regardless, our final MSE is 33.2, a higher metric than desired based on the facilitators of the Comma AI Speed Challenge who recommended an MSE of under 10.

You can see based on Figure 2 below that the predictions of speed followed the correct general trend, yet oscillated considerably.

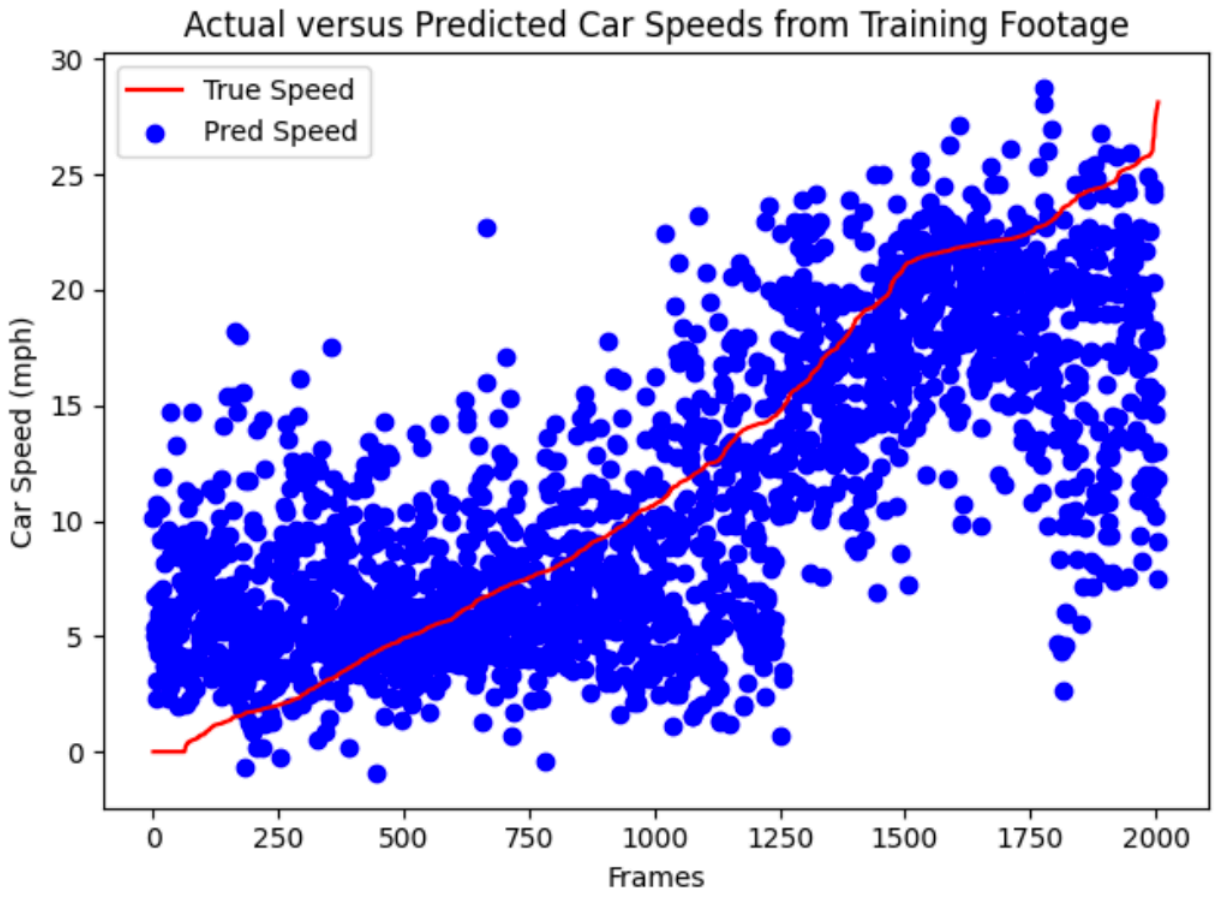


Figure 2. Actual vs. Predicated Car speeds on the Test Dataset

In addition, we recorded our own video while driving to test the accuracy of our model's speed predictions on the speedometer that we recorded in our home video. This test allowed us to see how generalizable our model was. The results of this test are shown in Figure 3 below.

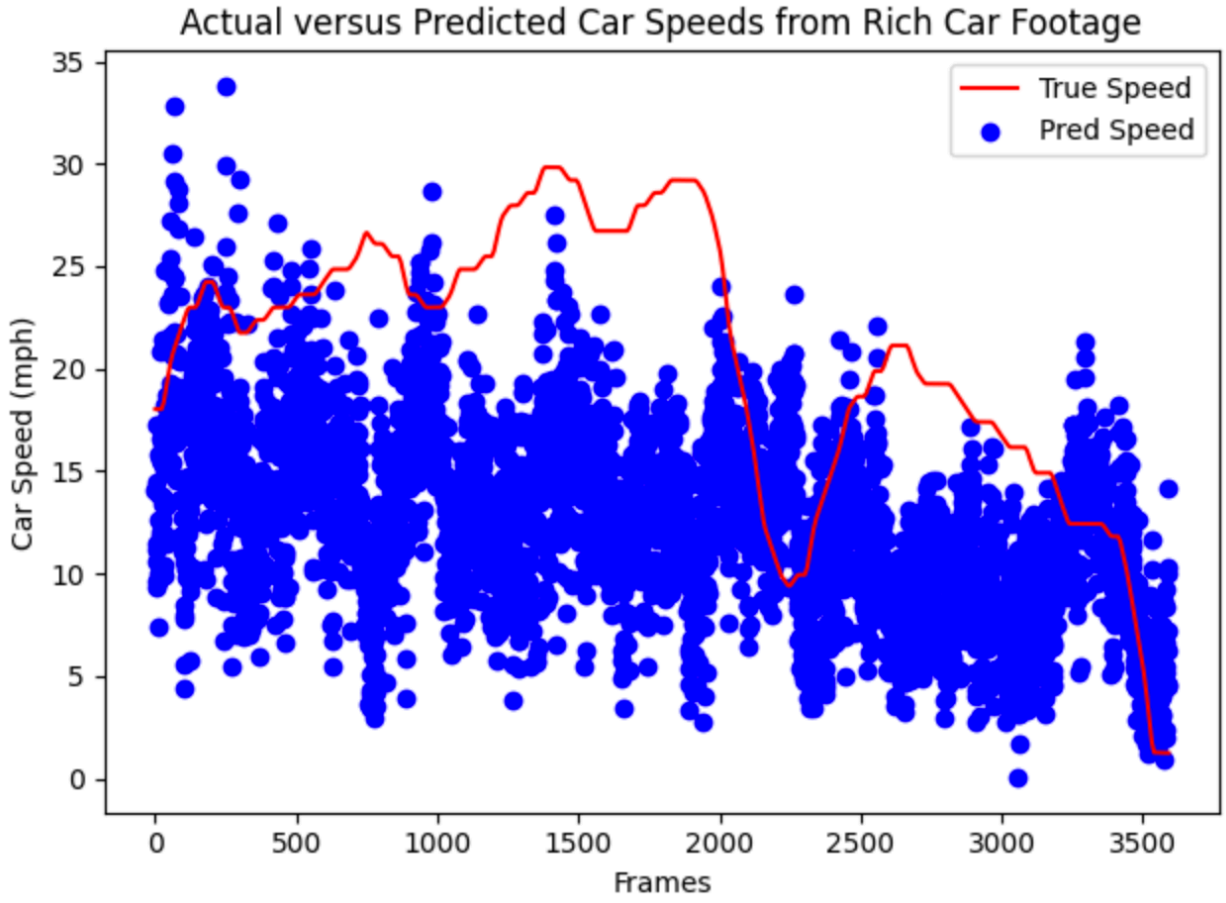


Figure 3: Actual vs. Predicted Speed on Our Test Drive Video

As you can see, the predictions are not as good as our initial mode. The poor results are likely due to the fact that we were holding the phone, instead of having it mounted to the rearview mirror, resulting in unnecessary movement that threw the optical flow generator off.

VI. Conclusion and Future Work

We have shown that it is possible to use a two-tiered CNN approach with the outside quarters of pairs of images of self-driving car footage to predict speed with decent prediction error

metrics. Though this approach did not directly improve performance from prior examples of the Comma AI speed challenge or from the recommended target metrics as set by Comma AI, it is still a valuable approach and can be improved upon with further experimentation. For future improvements, we will look into using CNNs to predict the optical flow image instead of the dense Farneback method used in this paper. In addition, we will continue to play around with the two-tiered ensemble model we created to see if it can be generalizable.

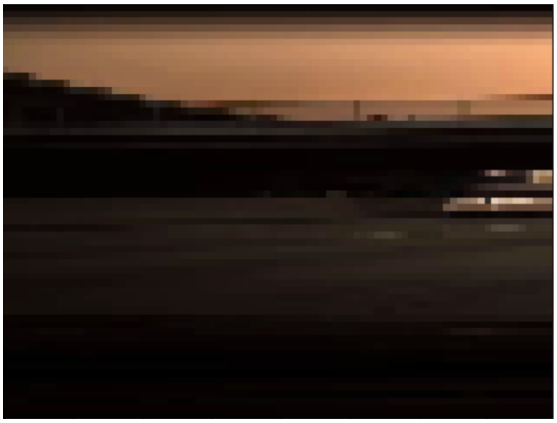
VII. Appendices

Appendix A: Pre-Processing Pipeline

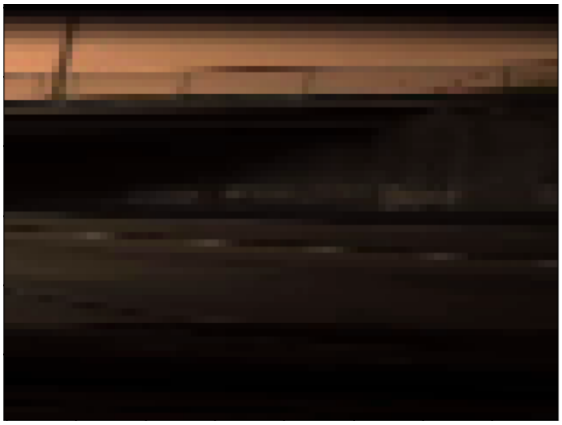
Image 1

Image 2

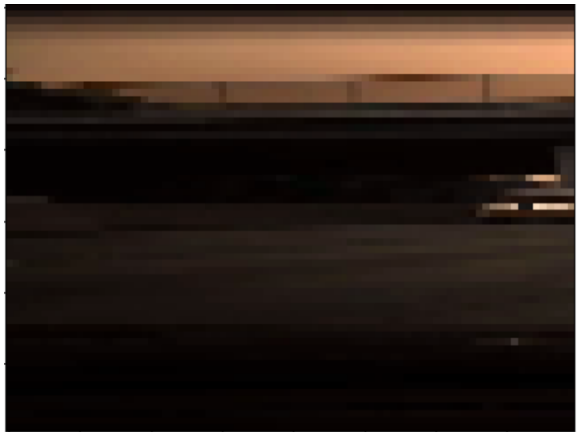
Left-Side Bright/Cropped Image 1



Right-Side Bright/Cropped Image 1



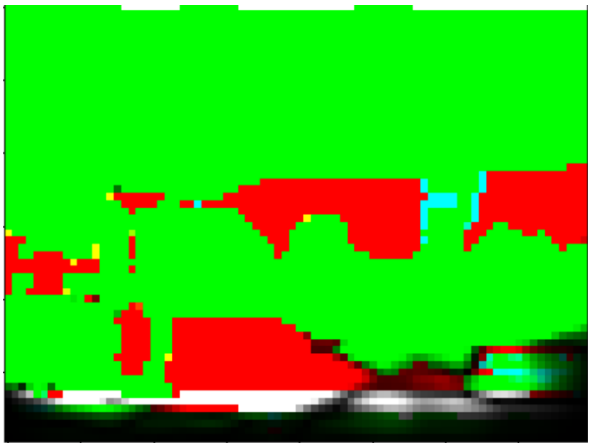
Left-Side Bright/Cropped Image 2



Right-Side Bright/Cropped Image 2

Left-Side Dense Optical Flow

Right-Side Dense Optical Flow

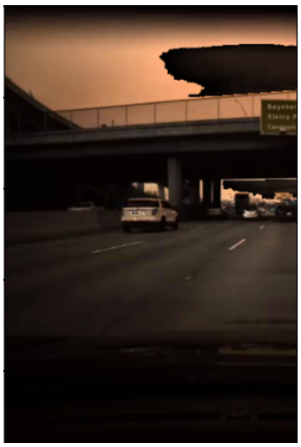


Appendix B: Full-Size Image Pipeline (only for enhanced visualization for the reader)

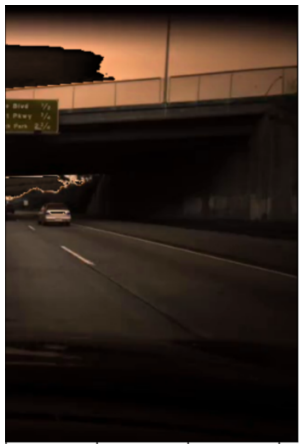
Full-Size Image 1

Full-Size Image 2

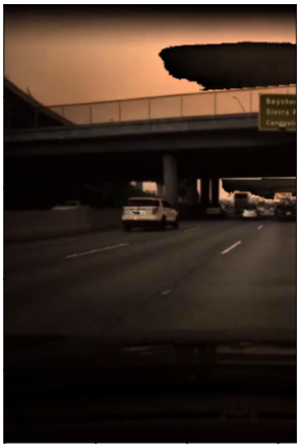
LS Bright Image 1



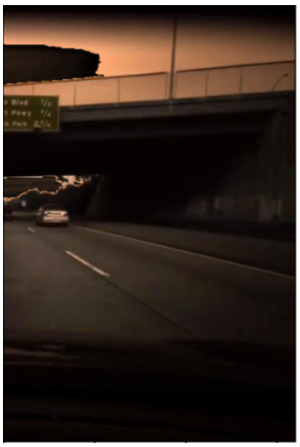
RS Bright Image 1



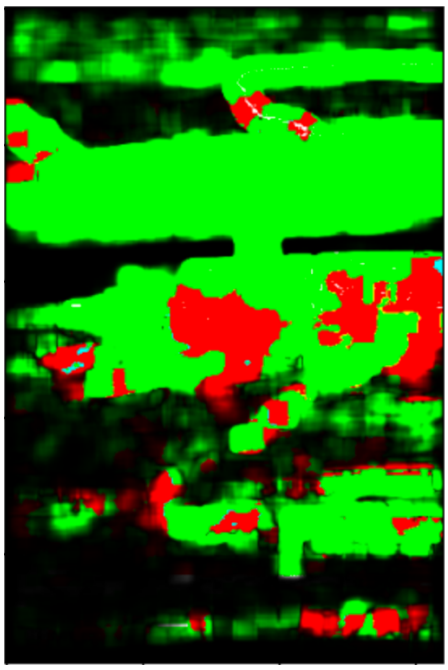
LS Bright Image 2



RS Bright Image 2



Left-Side Dense Optical Flow



Right-Side Dense Optical Flow