Predicting Dash-Camera Misalignment with Recurrent Neural Networks

Matt Bowring and Nick Bridges

The University of New Hampshire

CS 750 Machine Learning (Spring 2021) Final Project Report

Abstract

Predicting the misalignment between the vehicle and camera reference frame is essential for devising a `calibrated` frame of reference needed for training self-driving models. In this work, we develop a pipeline for extracting feature information from driving footage, which is then utilized by a recurrent neural network (RNN) to predict the pitch and yaw misalignment angles. The model is trained and evaluated using labeled open-source driving footage from a mounted dash-camera. We show that our method can reliably predict the correct misalignment on the training set but fails to generalize on an arbitrary testing video.

1. Introduction

It is helpful for self-driving models to train on video from the same reference frame. A dash-camera will often be misaligned from the car's reference frame due to suspension effects and variations between installations. Predicting the pitch and yaw misalignment is essential for transforming the video into a calibrated frame, which is used in the model's learning pipeline. A typical autonomous vehicle sensor suite consists of cameras, LiDARs, radars and other equipment. The goal of calibration is to find transformations between these sensors that map the raw data into a standard frame of reference. It is common for a trained professional to calibrate the sensors on these vehicles. If calibration is not performed correctly, operations like lane detection and collision detection will provide skewed results. Recent work has investigated the effects of camera misalignment on lane keeping assist systems to show that the mean lane position of the vehicle can shift considerably [1]. Previous solutions use a multi-camera apparatus for determining angular misalignment through a sequence of images used in calculating the camera orientation parameters [2]. Other developing technologies use the coarse geometric shapes of vehicles in front of the camera to derive the calibration parameters through forward and lateral vanishing points [3].

2. Methodology

With the assumption that the dash-camera is perfectly fixed to the vehicle, there should be no movement of the dash horizon, i.e., the boundary where the dash meets the road should not move with respect to the camera's reference frame. Of course, the background scenery will change, but the location of this boundary should remain fixed. Therefore, tracking this dash horizon over sequential frames indirectly provides information on how the misalignment changes over time. Figure 1 outlines the pipeline used to predict the pitch and yaw misalignment angles from an input video taken from a dash-camera.

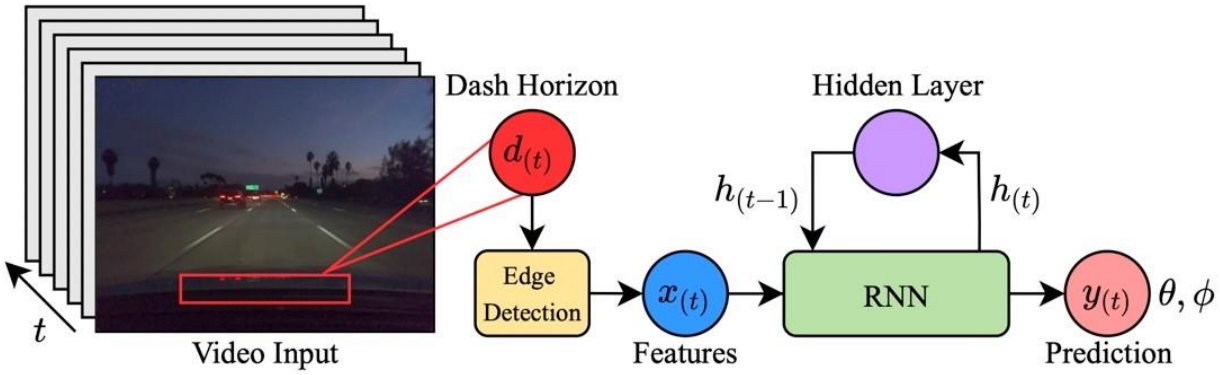

Figure 1: An outline of the proposed method for feature extraction using edge detection, and a recurrent neural network for predictions.

Edge Detection. A cropped input frame $d_{(t)}$ focuses on the region where the dash boundary meets the road. Greyscale and blurring transformations are then applied to reduce image complexity and noise. The well-known Canny Algorithm [4] is applied for edge detection, which relies on image intensity gradients. Otsu's method [5] is used in determining the upper and lower Canny threshold parameters, which effectively separates pixels into two classes, foreground, and background. The output $x_{(t)}$ is a binary sparse matrix describing where the dash horizon is located during frame t.

RNN. This specific neural network architecture was chosen for its ability to efficiently handle sequential information. Schematically, an RNN layer iterates over the timesteps of a sequence while maintaining an internal hidden state h that encodes information about the timesteps previously observed. The hidden state for a frame t is calculated according to

$$h_{(t)} = \tanh\left(W_x x_{(t)} + b_x + W_h h_{(t-1)} + b_h\right) \tag{1}$$

where W_x, b_x and W_h, b_h are the respective weights and biases for the features and hidden state, and $h_{(t-1)}$ is the hidden state from the previous frame. Note that these learnable weights and biases are sequentially updated at each frame and are initialized from $\mathcal{U}\left(-\sqrt{k}, \sqrt{k}\right)$ where $k = \frac{1}{hidden\_size}$. The output is then determined by

$$y_{(t)} = W_y h_{(t)} + b_y \tag{2}$$

where W_y, b_y are the weights and biases for the output and are initialized from $\mathcal{U}\left(-\sqrt{m}, \sqrt{m}\right)$ where $m = \frac{1}{in\_features}$ [6]. The loss metric is evaluated as the mean-squared-error and the Adam optimizer is used for training via stochastic gradient descent. The hyperparameters for the RNN model are provided in the appendix.

Model Evaluation. 5 open-source dash-camera videos from [7] are used to train and test the model; videos 1-4 are used for training, and testing is performed on the final video. Each video is approximately 1 minute long and is shot at 20 fps. Training and testing are both performed sequentially one frame at a time. For frames where the vehicle speed is less than 4 m/s, the label

is NaN, so the previous real angle value is used instead. The source code is written in Python 3.8 and utilizes the PyTorch library[1].

3. Results & Discussion

The RNN model can achieve accurate predictions on the training set but cannot generalize to the test set. Figure 2 depicts the model's performance on a single video in the training set. Here, convergence occurs around the 300th frame where the model can accurately predict both misalignment angles. Across the training videos, the RNN appears to converge better on the pitch (up/down) angle prediction than the yaw (left/right).

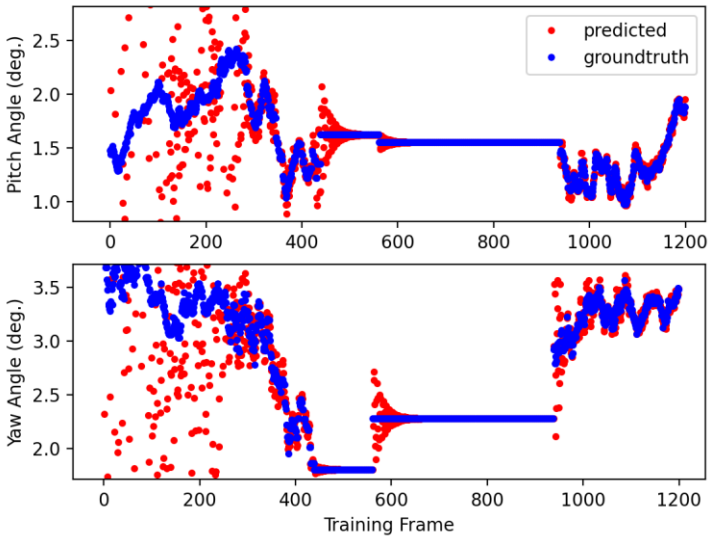

Figure 2: Pitch and yaw predictions for a single training video.

Figure 3 displays the effects of overfitting, where the model is not capable of generalizing to an arbitrary dash-camera video. It should be noted that each of the videos come from a different vehicle, and are highly susceptible to varying lighting conditions, reflections, and other noisy effects. Additionally, the cropped region $d_{(t)}$ is hard-coded and is naively used across all training and testing videos. Ideally, a trained convolutional neural network (CNN) would be used to identify the optimal dash horizon region, however, this is outside the scope of this work. While the RNN computations consider historical information, the weights and biases seen in (1) and (2) were derived from previous independent driving scenes, and so the model has difficulty generalizing to a new scene with a different dash horizon.

[1] The full code repository is available at https://github.com/bowrango/calib_challenge.

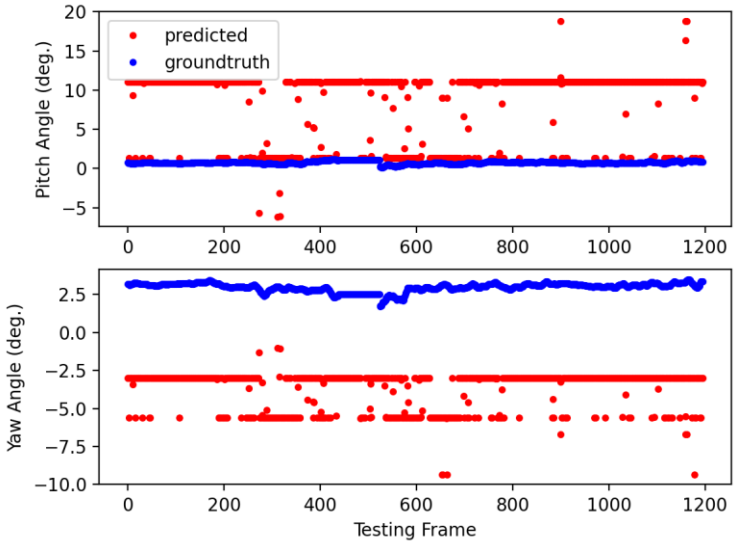

Figure 3: Pitch and yaw predictions for the test video.

Only a simple `vanilla` RNN design was considered here, with its implementation adapted from [8]. It is possible that modifying the architecture could improve performance. While the RNN was able to converge on the training set, its architecture may be ill-suited for handing video footage; processing spatial information has seen much better success with CNNs.

4. Conclusion

The proposed method outlined by Figure 1 overfits the training data and produces invalid results on the testing set. Using a fixed dash horizon boundary across the entire dataset likely made it more difficult for the model to accurately make predictions. A simple network architecture, and highly noisy data also contributed to the poor testing performance of the model. We hope this work provides a foundation for further research involving applications of recurrent neural networks for dash-camera calibration methods.

References

[1] Richard Romano, Davide Maggi , Toshiya Hirose , Zara Broadhead & Oliver Carsten (2021) Impact of lane keeping assist system camera misalignment on driver behavior, Journal of Intelligent Transportation Systems, 25:2, 157-169

[2] Oleg, Konevsky. Calibration Method and Apparatus for Automotive Camera System, and Method and ECU for Determining Angular Misalignments of Automotive Camera System. 7 Oct. 2010. U.S. Patent 0253784.

[3] Gopi Krishna Tummala, Tanmoy Das, Prasun Sinha, and Rajiv Ramnath (2019) SmartDashCam: automatic live calibration for DashCams. In Proceedings of the 18th International Conference on Information Processing in Sensor Networks. Association for Computing Machinery, New York, NY, USA, 157–168.

[4] J. Canny, "A Computational Approach to Edge Detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[5] Otsu, N. “A Threshold Selection Method from Gray-Level Histograms.” *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 9, No. 1, 1979, pp. 62-66.

[6] “RNN.” RNN - PyTorch 1.8.1 Documentation, Torch Contributors, 2019, pytorch.org/docs/stable/generated/torch.nn.RNN.html.

[7] Schafer, Harald. “Commaai/calib_challenge.” GitHub, Commaai, 30 Apr. 2021, github.com/commaai/calib_challenge.

[8] MorvanZhou. “MorvanZhou/PyTorch-Tutorial.” GitHub, 29 Oct. 2020, github.com/MorvanZhou/PyTorch-Tutorial.

Appendix

RNN Hyperparameters

Learning Rate	Time Step	Batch Size	Hidden Layer
0.02	1	1	1 x (32, 2)