DASHBOARD VIDEO SPEED PREDICTION

Jared Ucherek

The University of Texas at Austin

ABSTRACT

In this paper, we cover several approaches for accurate speed prediction from dashboard video. We specifically focus our attention to providing a sound solution to the Comma AI speed challenge [1]. We ultimately show great performance with a simple approach involving sparse optical flow calculations and keypoint detection derived from classical computer vision algorithms. This performance is compared to results obtained by replacin the classical algorithms with a deep learning model. Additionally, we cover our attempts using more sophisticated deep learning approaches that span topics such as interest point detection, visual odometry, and dense optical flow.

Index Terms— Speed Prediciton, Optical Flow, Keypoint Detection, Visual Odometry

1. INTRODUCTION

Speed prediction is an important area of research for autonomous robotics, where flying drones and underwater vehicles have no reliable method of tracking odometry. Numerous publications show effective speed prediction for various forms of data feeds. For vehicle speed estimation, previous work has shown accurate simultaneous object tracking and speed estimation [2, 3].

In this paper, we focus on speed prediction from a moving vehicle, using its dashboard footage as the only available source of data. The intuitive solution involves finding a keypoint that is a predefined distance from the car, suitably located near the center of the image. By determining how far this point moves between consecutive frames, we may use the training data to determine how much to scale each of these distance measurements, which results in predictions for unlabeled frames. Smoothing these predictions will also help reduce the variation between consecutive predictions. This process is described thoroughly in [4], which proposes the use of classical corner detectors and optical flow algorithms to determine how far each keypoint moves.

After replicating the effectiveness of this method on our dataset, we explore alternative approaches that would replace classical algorithms with more modern techniques in computer vision. This involves an architecture capable of detecting and tracking keypoints between frames. We also suggest the use of a segmentation network to remove extraneous features from the input images. Lastly, we briefly cover our expedited attempts at end-to-end approaches, where we feed in images and receive speed predictions. These methods involve architectures that compute optical flow and depth from a monocular input feed.

2. DATASET

The dataset is comprised of 20400 labeled training frames, and 10798 unlabeled testing frames. Test predictions can be emailed to Commaai for scoring, with MSE less than 5 being reported as very good performance. The dataset is provided in video mp4 format at 20 frames per second. We display some examples frames from both the training and testing set in figures 1 and 2.

The entire set is cropped uniformly to remove the dashbaord present throughout the video. We remove the bottom dashboard, top dashboard, and a small portion on the left and right sides of the image, as shown in figure 3. Most methods below operate on the grayscale image while adding Gaussian blur to ensure there is some robustness for variation between frames.

Given the time series nature of the dataset, we choose to reserve the final 30% of the training the training dataset for validation purposes. When training a neural network on the dataset, a portion of this is also split between validation and holdout, where validation MSE is used throughout training to monitor overfitting, and holdout MSE is checked once we have fully trained the network.

3. FEATURE POINT REGRESSION

We begin by describing the classical approach to this problem and best performing method, feature point regression. Given two consecutive frames, we first detect keypoints on the previous frame, and remove any that fall outside of a simple polygon mask. This mask ensures we are utilizing keypoints on the road, and not other objects or moving vehicles. After masking, the points and current frame are used to calculate a sparse Lucas-Kanade optical flow [5]. This calculates where the previous keypoints have moved in the new frame. We display the flow vectors in figure 4, with red circles representing

Fig. 1. Frame 1 of the training dataset. Each frame is originally 480 pixels tall by 640 pixels wide before being cropped to remove the dashboard.

Fig. 2. Frame 1 and 10797 of the testing dataset, respectively. This dataset contains a different environment of driving and will require a robust model for accurate predictions.

their location in the current frame, and green circles representing their location in the previous frame. Each flow vector has a position and orientation given by its keypoint pair. By recentering these flow vectors to the center of the image, and normalizing their length by this recentered location, we may calculate a reasonable statistic for predictions.

Due to the variable number of feature points for any given consecutive pair of frames, we use the median of these values as our singular feature fed into a regression model. More importantly, we do not fit any intercept, leaving us with a single parameter linear regression model that simply scales our median values to an accurate representation of the true speed. The predictions of our single-feature regression model are smoothed using a rolling window mean. These smoothed predictions are displayed in figure 5 and MSE scores are compared with several variations of the next method in table 1.

4. DEEP FEATURE POINT REGRESSION

Our alternative method to feature point regression simply replaces the classical algorithms used for feature point detection and sparse optical flow with a single deep learning architecture. We aim to extract a rich set of feature points tracked across consecutive pairs of frames using Superpoint, a self-supervised interest point detection architecture [6]. We specifically load a Superpoint model pretrained on the Kitti dataset[7].

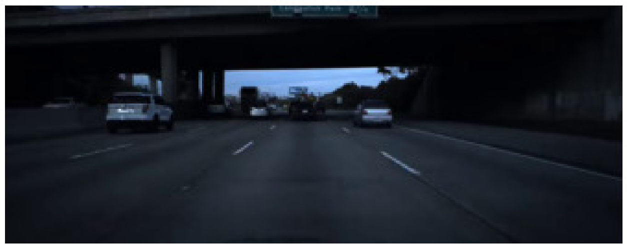

Fig. 3. Frame 1 of the training dataset cropped to remove the dashbaord. Resulting image is 220 pixels tall and 570 pixels wide.

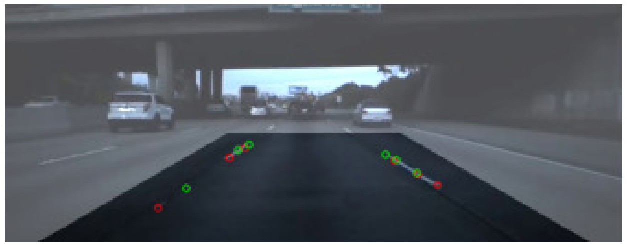

Fig. 4. Visualized feature points detected on the masked image using OpenCV.

For keypoint masking, rather than a handtuned polygon, we use a popular pretrained segmentation network to separate the road from other parts of the image [8]. We also experiment with combining the two masks, which ensures that our keypoints are both on the road and directly in front of the vehicle.

Superpoint can provide either keypoint anchors or a pair of keypoints tracked across consecutive frames. We test Superpoint's tracking accuracy by comparing it to optical flow calculation on its keypoint anchors. Given a set of keypoints in the previous frame, we apply the same method as before,

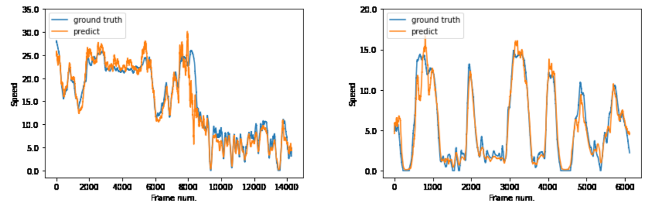

Fig. 5. Training and validation ground truth vs prediction plots for our feature point regression method.

remove points outside of the mask, calculate the flow, calculate the vector norms, calculate the median, and training a single parameter regression model. The process is simplified when using the tracked pair of keypoints. We remove points outside of the mask, calculate the vector norms, calculate the median, and train the regression model.

The results are shown in table 1. While calculating optical flow manually outperforms the keypoint tracking with Superpoint, further tuning of the various parameters in Superpoint could result in better performance. We found that the train MSE is generally lower for the full tracking method, suggesting that we are using too many inaccurate tracks. Parameters that could be explored for tuning include the confidence thresholds for keypoint detection, matching pair confidence threshold, and non-maximal suppression distance.

Method	Train MSE	Val MSE
Feature Point Regression	**5.36**	**1.32**
Superpoint, flow, segmentation	33.76	4.24
Superpoint, flow, original mask	12.11	2.41
Superpoint, flow, combined	12.03	2.70
Superpoint, tracks, segmentation	64.10	3.14
Superpoint, tracks, original mask	9.81	4.62
Superpoint, tracks, combined	9.12	3.80

Table 1. Comparison of the various regression methods tested. Superpoint tracks keypoints across frames, but the comparison to applying flow to the previous keypoints it finds is helpful to determine if its tracking is accurate.

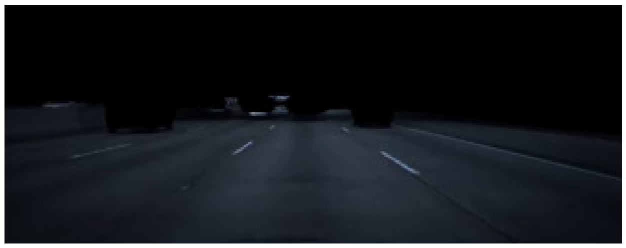

Fig. 6. Road segmentation mask used for filtering keypoints in our deep feature point regression method. This pretrained model had great visual results with no preprocessing of the images in our dataset. Slightly eroding the edges of the mask would also ensure no points are used too close to vehicle silhouettes.

5. IMAGE REGRESSION

While deep learning proves effective at finding features points to be used for our regression problem, there are numerous alternative deep learning approaches. The most straightforward

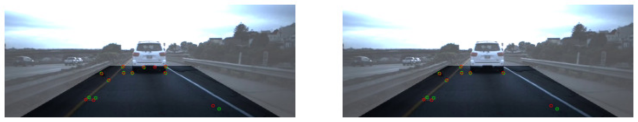

Fig. 7. Frame 1 of the validation dataset. Using the polygon mask results in keypoints placed on a close vehicle. Taking the pixel-wise or of the segmented road and original polygon mask results in more suitable keypoints being selected.

approach would be to feed in raw image data into a regression network. When using a robust architecture that can capture the time series properties of the data, one may expect this method to perform exceptionally well.

We explored the feasibility of feeding in a series of low dimensional vectors for consecutive image frames using a transformer model. This transformer network would operate similarly to the groundbreaking language models for natural language processing [9]. Furthermore, transformer models have recently been applied to vision tasks with great results [10]. However, this is an extremely novel research area to explore. Due to the time constraints and size of the dataset, we ultimately do not pursue this idea in great detail and opt to develop simpler approaches to the problem.

6. OPTICAL FLOW REGRESSION

Several online resources approach this problem by applying deep learning to dense optical flow fields [11] [12]. Dense optical flow does not require keypoints for calculation like sparse optical flow, and offers the potential for a richer feature set for our deep learning model. These dense flow images shown in figure 8 can be computed during preprocessing and saved. During training, we load these flow images in batches to remove the need for repetitive calculation of the flow image.

We explore tuning the various parameters for dense optical flow by visually inspecting the resulting images. Several parameters within the dense optical flow calculation have a major impact on results. Without properly tuning the parameters, the optical flow is too granular, and allows any deep learning model to quickly overfit to the unique set of several thousand images. One potential solution is to mask the flow not contained within the mask we used for our feature point regression problems.

We trained an EfficientNet model from scratch using PyTorch [13] [14]. Without masking the optical flow, this model quickly overfits to the small amount of training data. We briefly experimented with randomized masking and cropping with limited improvement, but were not able to fully test these methods. Overall, this method helped us understand some of the limitations of dense optical flow for deep learning on this dataset, which we consider while exploring the next method.

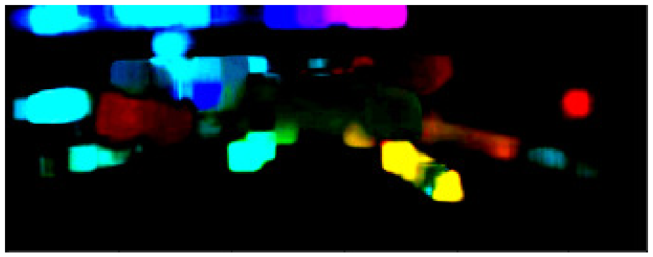

Fig. 8. Dense optical flow example from frame 1 of training data. A large window size of 21 is used to blur the flow, increasing robustness to overfitting.

7. VISUAL ODOMETRY

Visual odometry is a method used in robotics for tracking a robot's position with visual sensors. Recent works have applied deep learning to this problem, with specialized architectures replacing their counterparts found in classical geometric odometry [15, 16].

Most work within this area focuses on the Kitti dataset, a large dataset with 11 sequences of synchronized stereo camera images and lidar data for depth information [7]. For our problem, utilizing these pretrained models for monocular visual odometry could help track a map position, which infers speed and orientation. We attempted to apply pretrained models to our dataset, and found that the depth and flow predictions were too noisy, resulting in poor performance.

Given more time, this tasks would prove more advantageous than vanilla speed prediction. Accurately estimating odometry using monocular video would prove useful in several areas of robotics. More time spent ensuring proper data preprocessing and model loading would help determine how accurate the depth and flow predictions are on this unique dataset.

7.1. CONCLUSION

We detailed two promising approaches for robust speed prediction from dashboard video, showing great performance on the Comma AI challenge dataset. The intuitive process these approaches build on involves detecting keypoint movement through a pair of frames and training regression on this pixel-wise distance traveled. Both the classical algorithms and deep learning frameworks to measure this keypoint movement between frames performed similarly for our dataset, with minimal need for finetuning parameters.

More intensive deep learning approaches proved to be much more challenging to implement than originally expected, but show great promise as research advances. Given the size of the dataset and presence of only a monocular camera, most approaches are quickly deemed unreliable with out-of-the-box models. More analysis and work will need to be done before completely ruling out these methods and architectures for generalized speed prediction and visual odometry tasks. With time, computer vision models will begin to be more robust for time-series data and have the ability to be trained on video with generalized architecture.

Given the current necessity for thorough preprocessing and architecture selection on these models, they have minimal benefits compared to their simple regression based counterparts. Generalization is difficult, and training specialized models on datasets does not imply that the model will work with minimal changes for visually similar data. For speed prediction, simpler may be better, and we should focus our attention to developing proper deep learning models for accurate visual odometry instead. This will continue to be an important topic as deep learning and vision models continue to find harder tasks to solve.

8. REFERENCES

[1] commaai, "speedchallenge," *GitHub repository*, 2018.

[2] Tarun Kumar and Dharmender Singh Kushwaha, "An efficient approach for detection and speed estimation of moving vehicles," *Procedia Computer Science*, vol. 89, pp. 726 – 731, 2016.

[3] Wang Jing-zhong and Xu Xiaoqing, "A real-time detection of vehicle's speed based on vision principle and differential detection," in *2009 IEEE/INFORMS International Conference on Service Operations, Logistics and Informatics*, 2009, pp. 493–496.

[4] X. Qimin, L. Xu, W. Mingming, L. Bin, and S. Xianghui, "A methodology of vehicle speed estimation based on optical flow," in *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics*, 2014, pp. 33–37.

[5] Bruce D. Lucas and Takeo Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, San Francisco, CA, USA, 1981, IJCAI'81, p. 674–679, Morgan Kaufmann Publishers Inc.

[6] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal on Computer Vision*, 2018.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, June 2019, pp. 4171–4186, Association for Computational Linguistics.

[10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, Eds., Cham, 2020, pp. 213–229, Springer International Publishing.

[11] RyanChesler, "comma-speed-challenge," *GitHub repository*, 2019.

[12] Gunnar Farnebäck, "Two-frame motion estimation based on polynomial expansion," 06 2003, vol. 2749, pp. 363–370.

[13] lukemelas, "Efficientnet-pytorch," *GitHub repository*, 2020.

[14] M. Tan and Quoc V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[15] Stefan Milz, Georg Arbeiter, Christian Witt, Bassam Abdallah, and Senthil Yogamani, "Visual slam for automated driving: Exploring the applications of deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[16] H. Zhan, C. S. Weerasekera, J. W. Bian, and I. Reid, "Visual odometry revisited: What should be learnt?," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4203–4210.