Distance-Dependent Velocity Estimation using pre-computed Features

Moritz Kampelmuehler[1*], Michael Mueller[2*] and Christoph Feichtenhofer[1]

Institute of Electrical Measurement and Measurement Signal Processing[1]
Institute for Theoretical Computer Science[2]
Graz University of Technology, Austria
* These authors contributed equally.

Overview

Single-view velocity estimation is a relatively recent task in autonomous driving which has not yet been thoroughly explored. The task is to estimate the velocity of a specific vehicle from a set of monocular camera images to aid autonomous driving algorithms.

Given the camera calibration and ground truth position of the vehicle, this problem could accurately be solved using geometry only. However, in absence of the ground truth data, the representations need to be learned.

It is well known that machine learning algorithms grealty benefit from mapping the input data into a beneficial feature space. Here, we present a two-stage approach to single-view velocity estimation that is based on a) feature extraction using models pre-trained on different datasets, and b) representation learning based thereupon.

Feature Extraction

The feature extraction stage makes use of several state-of-the-art methods in computer vision.

Disparity. To obtain dense disparity maps of the RGB images, we use *monodepth* [4], which is an unsupervised monocular depth estimator. It is trained the on *Cityscapes* [2] and *KITTI* [3] Datasets.

Flow. 2D optical flow data is calculated using FlowNet 2.0 [5] trained on FlyingThings3D [7]. This network allows to calculate dense optical flow between pairs of frames.

Bound Box Tracking. Since the bounding box is only given for the final frame of the input image sequences, we perform tracking (backwards through time) to obtain bounding boxes for all frames. For this task, we use a fusion of the Medianflow [6] and MIL [1] tracking algorithms.

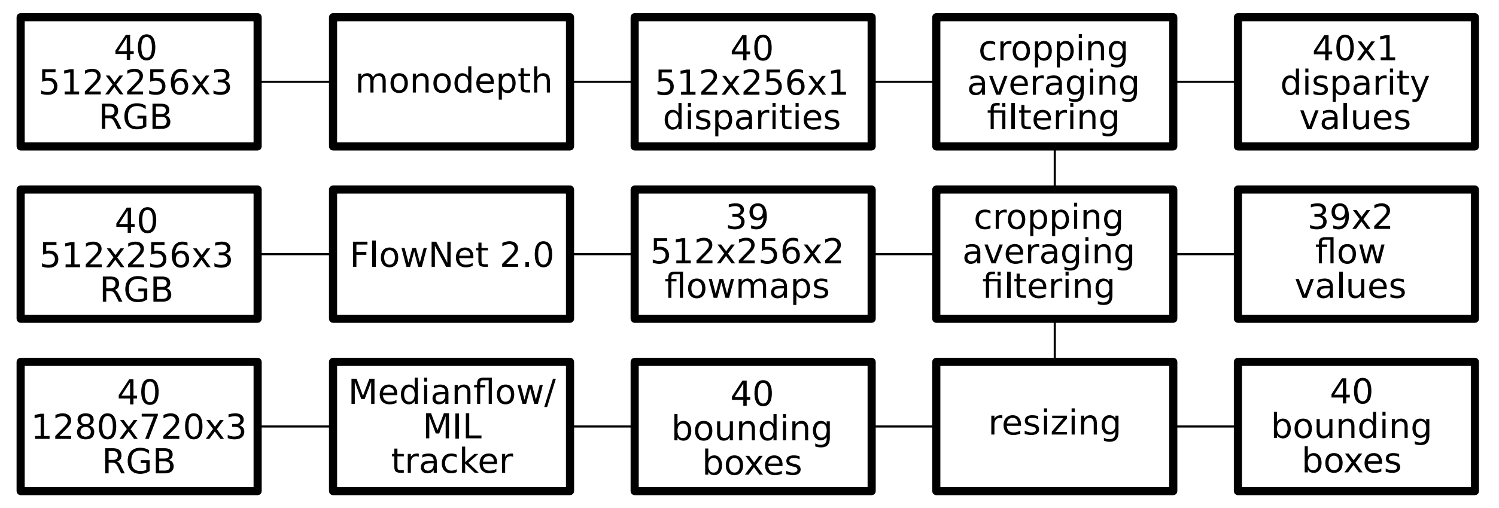

Figure 1: Feature extraction procedure.

Pipeline. The full procedure of the feature extraction stage is depicted in Figure 1. Due to architecture constraints, the HD input RGB images are resized to 512x256. For both flow and disparity the network outputs are cropped and averaged over the respective bounding boxes. To provide smooth estimates over the frames, these averaged values are filtered using a Gaussian kernel. While the tracking is applied on the 1280x720 inputs, the obtained bounding boxes are later resized to match the disparity and flow map size.

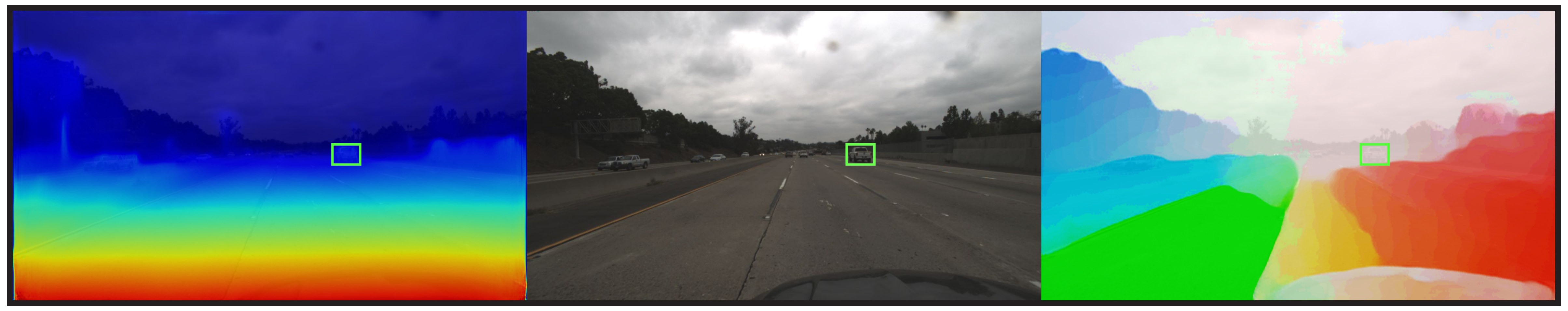

Figure 2: (from left to right) Disparity map, RGB, Flow map.

Figure 2 shows a visualization of the feature outputs. The left image depicts the dense disparity map over the RGB image. The middle image shows the raw mono RGB image. On the right the dense optical flow representation is given. An example bounding box of a vehicle is shown for reference.

Model

These pre-computed features allow us to use rather shallow fully-connected neural networks to achieve good estimation performance.

Data Split. The relationship of the precomputed features to the learning target is highly nonlinear. To simplify learning, the data set is split into three subsets according to the bounding box area of the last frame to obtain disjoint sets for near, medium, and far away vehicles. This classification shows some variance near the borders between different sets, but nevertheless simplifies the learning task.

Architecture. On each of these three sets, cross-validation was performed to obtain a good architecture. The resulting network topologies are (layers x units) 3x40 (near), 4x60 (medium), and 4x70 (far). All hidden units use complementary rectified linear units [8], thus, the layer output sizes are twice the input size.

Training. For all distances, we train for 2000 epochs using minibatches of 50 samples on the MSE between network output and targets. For regularization, weight decay and Dropout were used. The training data for each distance is split up into five partitions. Four of these are used as training set, the fifth is used for validation. After 2000 epochs, the model with the lowest validation error is saved. This results in 3x5 models for the entire dataset. Note that the number of examples per neural network is quite small, so overfitting may occur easily as the validation score is no longer an accurate estimate of the error on the test set.

Model Averaging. When evaluating the test set, we split the data according to the computed bounding box areas using the same procedure as for the training set. Then, the average over all five models for the respective distance is computed.

Discussion

Results. The final results are given in terms of the mean squared velocity error

$$E_{V,C} = \frac{1}{|C|}\sum_{c\in C} ||V_c - \hat{V}_c||^2 \qquad (1)$$

which is computed separately for near, medium, and far vehicles and averaged to get the final score. (The classes are split by position according to the ground truth distance of target car to camera, so this split does not necessarily correspond to our split of models.) The final average score across all three classes was $\sum_C \boldsymbol{E_{V,C}} = \mathbf{1.3021}$.

Discussion. The presented approach builds on pre-computed features and achieves good results (winning score on the TU Simple Velocity Estimation Challenge 2017). Better results could be achieved by merging the pre-trained models for feature extraction with the final layers to allow error backpropagation to the feature extraction layers for fine-tuning in a single pipeline.

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[4] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[5] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[6] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 2756–2759. IEEE, 2010.

[7] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016.

[8] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. *CoRR*, abs/1603.05201, 2016.