Driving among Flatmobiles: Bird-Eye-View occupancy grids from a monocular camera for holistic trajectory planning

Abdelhak Loukkal[1,3] Yves Grandvalet[1] Tom Drummond[2]
You Li[3]
[1] Université de technologie de Compiègne, CNRS, Heudiasyc UMR 7253, France
[2] Monash University, department of Electrical and Computer Systems Engineering, Australia
[3] Renault S.A.S, Guyancourt, France
{aloukkal,yves.grandvalet}utc.fr tom.drummond@monash.edu you.li@renault.com

1. Data description

1.1. Nuscenes dataset

Real-world experiments are conducted on the nuScenes dataset [1], which records the measurements of a complete suite of sensors: 6 cameras, 32-channels LIDAR, long-range radars. The whole dataset comprises 40 000 annotated frames but for intellectual property reasons we are currently limited to the preview version, with 3 340 annotated frames from five driving sequences in Boston and 81 sequences in Singapore. Each driving sequence lasts around 20s and images are acquired at a framerate of 2Hz. The two most important features of nuScenes for this paper are the availability (i) of 3D bounding boxes, (ii) of layers of the drivable area are also provided as binary semantic masks, where each pixel corresponds to 0.1×0.1 square meters.

1.2. Carla simulator dataset

Simulated-world experiments are run on the CARLA simulator [5] version 0.9.6, as implemented in [2]. We collect 50K frames at 10 fps for training, which amounts to 1.4 hours of driving in Town 1 under 4 training weather conditions with 100 vehicles and 250 pedestrians. Each frame contains a $576 * 240$ rgb camera image, the world position, the 3D bounding boxes coordinates of the vehicles in the scene and a road layout raster image. Town 2 is used for the on-line evaluation of our model.

2. Homography estimation

On nuScenes, we take advantage of the annotated 3D bounding boxes to get corresponding points in the BEV and the camera planes. The pixel positions of the 3D bounding boxes "ground" face corners in the camera image are matched with their 3D position in the BEV plane, and the homography matrix is obtained using the

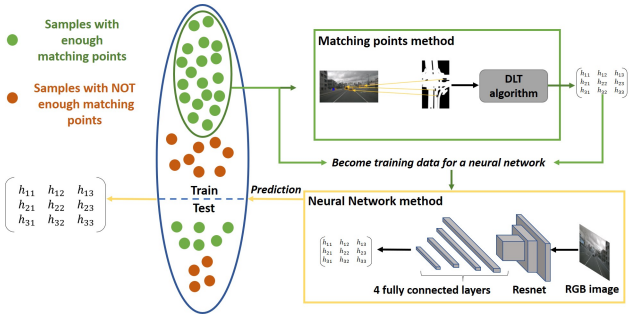

Figure 1: For training images with enough matching points, the homography is computed with the matching method. These samples become training data for a neural network that learns to estimate the homography from RGB input images. This trained network is then used to pre-compute the homographies of all the samples in the dataset. Better viewed in color.

`getPerspectiveTransform()` function of OpenCV that applies the Direct Linear Transformation (DLT) algorithm [6]. Not all training samples contain enough matching points, so only a subset of the original training set is available for fitting a network predicting homographies (see Fig. 1). The homography network is composed of a ResNet-18 encoder and 4 fully connected layers with the fourth layer outputting 9 values corresponding to the elements of the homography matrix. The homography matrix contains 4 rotational terms and 2 translation terms. The difference in magnitude between these terms must be taken into account during training. The authors of [4] circumvent this issue by predicting 4 matching points instead of the homography matrix elements as there is a one-to-one correspondence between the two representations; we simply rescale the rotational and translation terms with a constant factor

such that all the elements of the homography have the same magnitude. The L2 loss function is then used as the training criterion.

For each frame in the dataset, the predicted homography is used to warp the drivable area and vehicle OGMs in the camera plane. These warped masks are used as ground truths in the holistic end-to-end network. Each sample has a sequence of 6 input camera images separated by 0.5s, the camera view semantic masks for every image in the sequence, the homography matrices for every mask in the sequence, the past and future trajectories.

3. NoCrash benchmark

On the Carla simulator, our model is evaluated on the NoCrash benchmark [3]. This benchmark consists of 3 driving scenarios with a varying number of vehicles and pedestrians in the simulated town: empty (no traffic), regular traffic and dense traffic. The vehicle drives 25 predefined routes with different starting points and an episode is counted as successful if the vehicle reaches its goal within a certain time limit without colliding with a static or a dynamic object. The training weathers are "Clear noon", "Clear noon after rain", "Heavy raining noon", and "Clear sunset" and the test-only weathers are "After rain sunset" and "Soft raining sunset".

Our approach is adapted to the task and a one-hot encoded high-level command is provided instead of a goal position. We also modify the trajectory part of the network and adopt the same architecture and loss function as in [2] with a single image being used as input. The set of waypoints is converted to driving commands using a low-level controller as described in [2]. The mid-to-mid baseline input is also changed to a 7 channels grid map similar to [2] containing information about the road layout, the vehicles, the pedestrians and the traffic lights. Pedestrians in Carla simulator have a very erratic behavior as they cross frequently the road. This required access to pedestrian information in the mid-to-mid network. Information about traffic lights is also necessary for navigation, because the model is fed with a high-level command instead of a goal position. Even though the intermediate representation of our network does not inform about the pedestrians, it is interesting to observe that our model successfully stops before the pedestrians and avoids collision. When looking at the intermediate representation, we observe that the pedestrians are detected as vehicles when they are on the drivable area. The vehicles class in the simulator contains cars and trucks but also cyclists which look very similar to pedestrians. This observation highlights the importance of having an intermediate representation to delve the decisions of the neural network.

References

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[2] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning (CoRL)*, 2019.

[3] Felipe Codevilla, Antonio López, V. Koltun, and A. Dosovitskiy. On offline evaluation of vision-based driving models. In *ECCV*, 2018.

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *CoRR*, abs/1606.03798, 2016.

[5] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. CARLA: an open urban driving simulator. In *1st Annual Conference on Robot Learning (CoRL 2017)*, pages 1–16, 2017.

[6] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.