Week4: 3D Motion Estimation

November 28, 2021

1 Introduction

In task 4, we are supposed to compute the estimate the 3d motions with CARLA simulator. The ego-vehicle is either stationary in the first exercise or moving in the second exercise. The dynamic objects include other vehicles and pedestrians. There are 2 RGB cameras, 2 depth cameras and 2 semantic segmentation cameras on our vehicle. The ground truth 3d motions can be computed with the help of 3d point cloud and dense optical flow.

The outcomes are shown with the videos, which are stored in:
"https://syncandshare.lrz.de/getlink/fiAgBH17NaoyRV1LUB6QBCRf/"

2 Ex4.1 - Ego-Vehicle stationary

2.1 Compute the ground truth 3d motion

In the exercise, the ego-vehicle is stationary, which means the only the dynamic objects, such as pedestrians and other vehicles are moving in the scene.

To compute the 3d motions of dynamic objects, first the 3d point cloud need to be reconstructed. For each 2d frame, 3d point position for each 2d point can be computed with the depth information from the depth map.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Here the Ego-vehicle is stationary, so the camera coordinate can be set to be equal to the world coordinate, which means the rotation and translation are both 0.

With the 3d point cloud, the 3d motion can be computed by computing the distance between two 3d points which are corresponding to the same point in the scene. To find the matching points in 2 frames for each pixel, the optical flow is used. As shown in the Figure 1, only dynamic objects have relatively large flows, since all the backgrounds are stationary in the scene. The finally computed 3d motion is also shown in the Figure 1, with the first channel (red) showing the x-motion, the second channel (green) the y-motion and the last channel the z-motion. In this figure, all 3d motions tend to be blue since there are almost none x-/y-motion for the moving vehicles and pedestrian.

2.2 Training with CNN

To train a network, first the training dataset need to be prepared. There are 10 episodes, and each with 2 cameras, I randomly choose 2000 frames from them as the first frame, and then randomly choose the 2/3/4 frames after as the second frames. The RGB frames are stored in the dataset as inputs of the NN and the 3d motion are computed as ground truth and stored in the dataset for computing the loss. The final dataset has a shape of (2000,3,3,128,128). The first 3 represents the 2 images: 2 RGB images and the 3d motion. The second 3 means that every image has 3 channels. And I resize the images to (128,128) to train the NN faster.

First I tried to use an classic Autoencoder, which is composed with 2 parts: the encoder and decoder. As 2000 samples are not that enough for training a new encoder, I use the pretrained

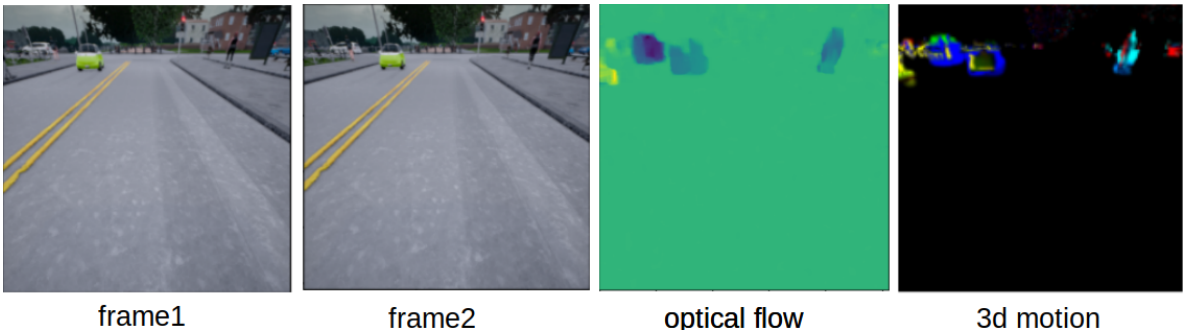

Figure 1: The 3d motion with stationary ego-vehicle.

Alexnet. The 2 RGB images are first put into the encoder separately and then combined together as the input of the decoder. The encoder is supposed to catch the information from the images and put them into a small matrix, decoder analysis the information from the 2 matrices, find the difference from it and reconstruct the motion. The loss function is the MSE Loss.

Before training with the whole dataset, I first try to train the model to overfit with only one sample. The learning rate is set to be 5e-4. After training 300 epochs, the model "memorize" the 3d motion correctly as shown in Figure 2.

However, after training with the whole dataset with 100 epochs, the loss keeps getting down, but the predicted motion always tend to be all-zero especially when the ground truth motion is small. For the samples that have larger ground truth motion, the model can roughly predict the motion but the distance between the ground truth and the output is fairly large. The results are shown in Figure 2.

The reason might be that too much information are lost in the bottleneck of the Autoencoder. The size of the input images are (128, 128), and the size in the bottleneck is only (3, 3). In the samples with small ground truth motion, the dynamic objects may not be caught and stored in the bottleneck, so the predicted motion becomes all-zeros.

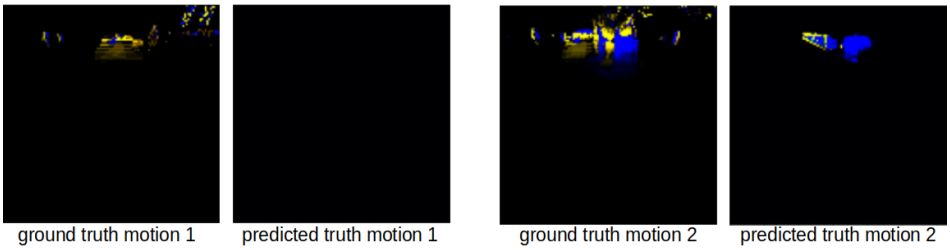

Figure 2: First try with classic Autoencoder.

To improve the model, I change its structure to a UNET-like model. The special point of UNET is that the layers in encoders will be used in decoder, so less information are lost during the down-sampling and up-sampling procedure.

I try to overfit the model with one sample again with the new UNET-like model, and it turns out the model can fit it perfectly, the only problem is that the training seems to be slower since the structure of the new model is more complicated.

2.3 Output Analysis

After training for 300 epochs with learning rate = 2e-4, the loss is 1/100 in comparison to the initialization.

With the UNET-like model, the predicted motion are no longer all-zeros though the ground truth motion is small. The problem is that too much information are kept, so even the static objects also have some "motion", so the predicted motion seems to be more noisy as the ground truth. But most motion of the dynamic objects are captured correctly. The result is shown in Figure 3.

Figure 3: Outcome with UNET-like model for Ex4.a.

3 Ex4.2 - Ego-Vehicle moving

3.1 Compute the ground truth 3d motion

The procedure to compute the 3d motion is very similar to the first exercise. The only difference is that the extrinsic matrices of the 2 frames are different since the ego-vehicle is moving. Due to the same reason, the moving objects between 2 frames are not only other vehicles or pedestrians, but also stationary objects such as trees or buildings. To delete this kind of motion, first the 2 3d point clouds from the 2 frames are needed to be aligned, which means the motion caused by the motion of the ego-vehicle should be eliminated. To align 2 point clouds, the relative pose is computed with the 2 extrinsic matrices and multiply the relative pose with the 3d point positions of the first frame. The aligned 3d point clouds are shown in Figure 4. As can be seen, the most motions of the stationary objects are deleted and the only moving objects are other vehicles and pedestrians.

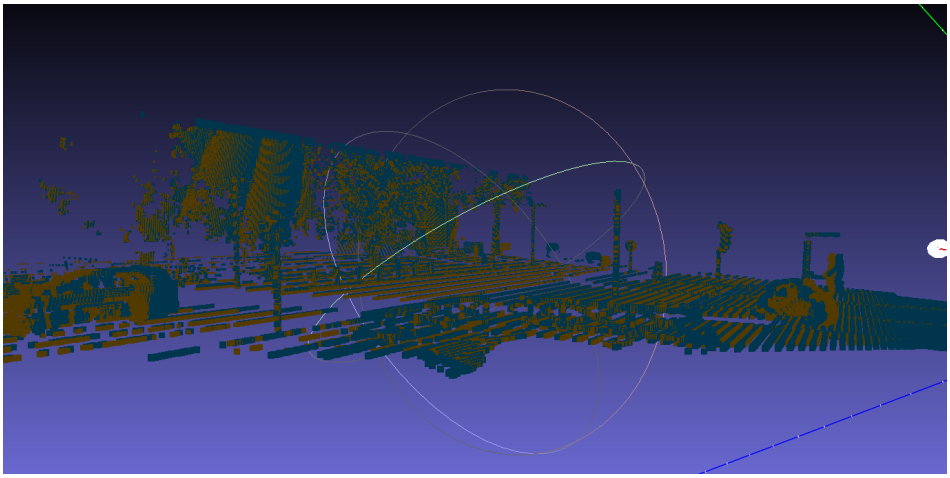

Figure 4: The aligned 3d point clouds.

To compute the optical flow, the 2 frames should be aligned as well. This can be realized by projecting the transformed 3d points of frame 1 to frame 2. After the rendering, the converted frame 1 and frame 2 are the same as in Ex4.1 with the ego-vehicle stationary and the dense optical flow can then be applied on them. The problem is that there are some small errors in this whole process, which leads to the errors in the optical flow.

As can be seen in Figure 5, not only the flows of dynamic objects are detected, but also the static objects like the road and the trees. With the errors in optical flow, there are also errors in the computed 3d motions. To solve this problem, I put a segmentation mask on the computed 3d motion. Only the motions corresponding to other vehicles and pedestrians are kept, all others are deleted. The 3d motion after applying the segmentation mask is shown in Figure 6.

3.2 Training with CNN

The structure of the Network is very similar to the structure in the Ex4.1. The only difference is that the relative translation vector between the two frames are also stored in the dataset as the extra

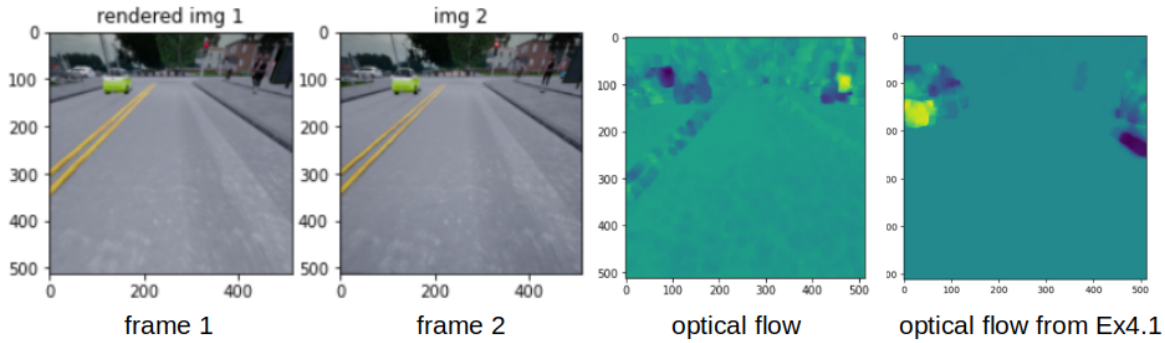

Figure 5: The optical flow in Ex4.2.

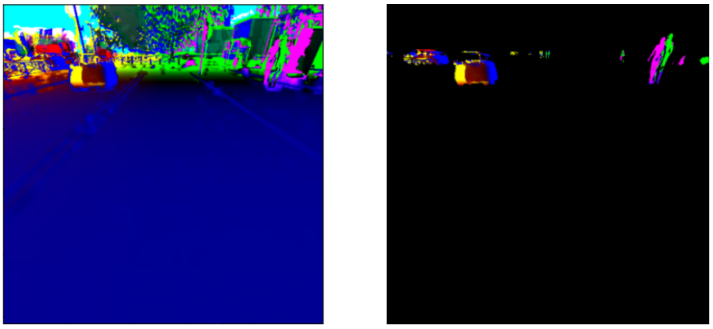

Figure 6: 3d motions with segmentation mask.

information. Since the input images are not aligned, the Network need to now how the ego-vehicle has moved to learn to align the two images. Here I put the whole translation vector rather than its norm since the Network can get more detail information and may learn to the alignment more easily. This translation vector is not put into the encoder, but directly into the decoder together with the combined output of the 2 encoders.

3.3 Output Analysis

It is kind of surprising that the model seems to learn faster and better with the moving ego-vehicle than with the stationary ego-vehicle. After training for 300 epochs with learning rate = 2e-4, the loss is about 1/150 in comparison to the initialization. In comparison to the first exercise, the output motion is less noisy, which means roughly only the motions of dynamic objects are kept.

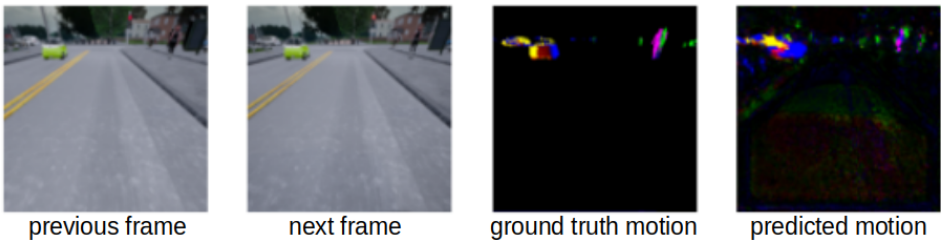

Figure 7: Outcome with UNET-like model for Ex4.b.

4 Conclusion and future work

In this exercise, I learned to compute the 3d motion between 2 frames with the optical flow and try to predict it with a CNN model.

I first tried to use a classic Autoencoder with pretrained Alexnet since I thought the samples maybe not enough to train the model. And it turns out that too much information are lost in the bottleneck. So I change the model to a UNET-like model, and it works much better.