1 Theoretical background

For the task at hand I have implemented the method based on Neural Radiance Fields that was introduced in [1]. This algorithm represents using a fully-connected neural network. The inputs are the three-dimensional coordinates of the spatial location (x, y, z) and the viewing direction vector (d_x, d_y, d_z). The outputs are the density and the view-dependent RGB color information at the given spatial location.

For training, each image is used along with its extrinsic matrix and camera parameters to produce ray origin and direction vectors. For a pixel with coordinates (u, v) on the image plane the ray direction vector is obtained as

$$d_x = (u - c_x)/f_x, \quad dy = (v - c_y)/f_y, \quad d_z = 1, \tag{1}$$

where f_x and f_y are the focal lengths, c_x and c_y are the offsets along each axis. After computing Eq. (1), the vector (d_x, d_y, d_z) is normalized to 1.

For the ray direction vector first we must compute the full extrinsic matrix for the frame. Given the camera-specific `RT_body_from_sensor` matrix found in the calibration data file, which weill be denoted as E_C, and the frame-specific `ECEF_RT_body` matrix found in the egomotion data file, which will be denoted as $E_{P,n}$ with n being the frame label, we can compute the full extrinsic matrix for the nth frame as

$$E_{full,n} = E_{P,0}^{-1} \times E_{P,n} \times E_C. \tag{2}$$

Having obtained the coordinates in the camera's frame of reference, E_C transforms them to the vehicle's frame of reference. Then, $E_{P,n}$ transforms these coordinates to the Earth-centered Earth-fixed coordinate system. Due to having large numbers present in $E_{P,n}$, there's another transformation applied to the coordinates, namely, one that brings us back to the coordinate system that is adjusted to the initial position of the camera. Then, the coordinates of the ray origin vector for the given frame are the first three elements in the last column of $E_{full,n}$.

The obtained origin and direction vectors $\boldsymbol{o}$ and $\boldsymbol{d}$ define a ray for each pixel in the given image. Parametrizing the rays as $\boldsymbol{x} = \boldsymbol{o} + t\boldsymbol{d}$, the rays are sampled between $s > 1$ (near clipping plane) and $s < 10$ (far clipping plane). The coordinates of the location vector $\boldsymbol{x}$ and the viewing direction $\boldsymbol{d}$ are then handled with positional encoding as described in subsection 5.1 of [1] and given as inputs to the neural network.

The output will be the RGB color information and the density. These are used to compute the RGB color values for the pixel with image coordinates (u, v) as described in subsection 5.2 of [1].

2 Implementation of the neural network

The neural network was implemented in pytorch following the setup described in [1]. The structure of the neural network is detailed on figure 7 of the cited paper. The size of the hidden layers, which will be referred to as idden layer dimension, was left as a modifiable parameter in this setup.

For the training, the Adam optimizer was used with an initial learning rate of $5\,10^{-4}$ and the `MultiStepLR` scheduler of pytorch with the `gamma` parameter set to 0.5.

3 Training

For training I used two setups. Due to the short duration of time and available resources, only a subset of camera recordings were considered. The records of the fisheye cameras were discarded due to requiring a separate treatment. In both setups a batch size of 1024 was used.

In one setup, only images from the camera `F_MIDLONGRANGECAM_CL` were used to produce an initial proof of concept for the viability of the method. Camera images were undistorted using the calibration parameters avialable in the `calibration.json` file and were given to the neural network for training for 4 epochs. The hidden dimension parameter was set to 256 and the there were 96 random sampling points on each ray, generated with uniform distribution between the near and far clipping planes. The loss had to be monitored in the beginning of the first training epoch to ensure convergence. The full history of the training can be seen in figure 1.

After training, one of the images in the training sample were randomly selected and was reproduced using the trained network. Then, the extrinsic matrix of the image was modified by applying a $0.5m$ translation along the x axis. The original, reproduced and perturbed images can be seen in figure 2. As it can be seen, the method is able to reproduce the image with low fidelity. A larger sample size would be required that should include images of different cameras recording from different angles.

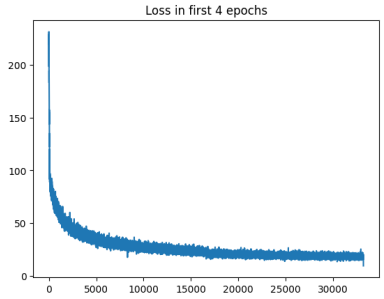

Figure 1: Training loss for the first setup.

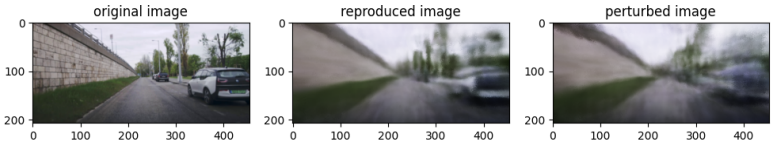

Figure 2: Original, reproduced and perturbed images

For the second setup, images from two cameras were used. Once again, camera images were undistorted using the calibration parameters avialable in the `calibration.json` file and were given to the neural network for training for 2 epochs due to the increased requirement of time and computational resources. The hidden dimension parameter was set to 128 and there were 48 random sampling points on each ray. The full history of the training can be seen in figure 3.

In figures 4 and 5 we show a randomly selected image from the two different camera recordings, similarly as for the previous setup, along with their

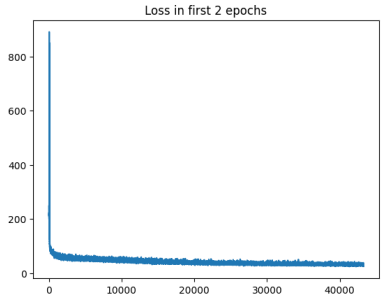

Figure 3: Training loss for the first setup.

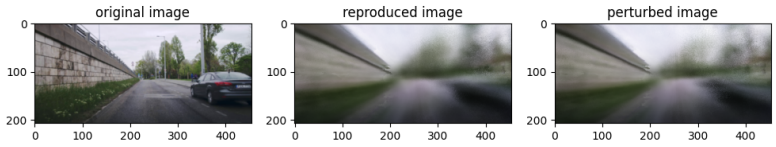

Figure 4: Original, reproduced and perturbed images in the second setup for the camera `F_MIDLONGRANGECAM_CL`

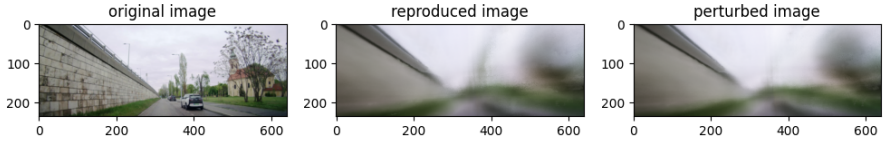

Figure 5: Original, reproduced and perturbed images in the second setup for the camera `F_MIDRANGECAM_C`

reproduced and perturbed versions. As visible, the neural network reproduces the images with low fidelity. Larger sample size and neural network size would be required as well as more training epochs and a finer binning of the rays. Furthermore, it is possible that the egomotion data is not precise enough for using multiple camera images. This is a documented problem of the NeRF method and is usually treated using structure from motion methods o improve egomotion information. However, the method shows promising results.

References

B. Mildenhall, P.P. Srinivasan, M. Tanick, J.T. Barron, R. Ramamoorthi, R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," arXiv [cs] 2003.08934, 2020