GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization – Supplementary Material –

Lukas von Stumberg[1,2*] Patrick Wenzel[1,2*] Qadeer Khan[1,2] Daniel Cremers[1,2]
[1]Technical University of Munich [2]Artisense

Abstract

In this document, additional information complementing the original paper is given. Details of our relocalization tracking *benchmark are also described. Furthermore, we provide results of some additional experiments evaluating the performance of our method to indoor and homogeneous scenes. Lastly, additional details on the network architecture used for this work is also provided. The video and the benchmark dataset can be found at* `https://vision.in.tum.de/gn-net`.

A. Details on the Evaluation Benchmark

This section describes the data included in our benchmark which is used for *relocalization tracking*. It contains simulated data created with the CARLA [2] simulator. It also provides Lidar aligned real-world sequences from the Oxford RobotCar dataset [4]. The CARLA benchmark is constructed with the stable version 0.8.2 of the simulator. We add ground-truth poses and camera intrinsics for all images collected under different lighting and weather conditions. The dataset presents the challenge of relocalization against different weather conditions and poses. We have collected data from 6 different cameras. The positions and orientations of the cameras are given in Table 1. A subset of the different positions and orientations of 3 of the 6 cameras are shown in Figure 1 for a certain time step. The cameras are mounted relative to the ego-vehicle. For each camera, 500 images are collected. For training, validation, and testing sets, 9 different sequences for each are recorded under three different weather conditions. The conditions and the overall statistics of the training, validation, and testing datasets are given in Table 2. Along with the camera images, we also provide access to their corresponding dense depth maps and semantic labels. The data is generated by driving around *Town1* of the CARLA simulator.

Relocalization tracking: For each one weather sequence, we have created a *relocalization.txt* and for each all weathers sequence a *relocalization_other_weathers.txt* file. The difference between the two relocalization files is that the latter one contains cameras to be relocalized against different sequences and hence different weather conditions. Each row of the relocalization file is arranged as follows: Current camera index, camera to be localized against, and relative pose between these cameras. The relative pose between the two cameras is in the coordinate system of the left camera. The *stereocalibration.txt* provides the relative pose information between Cam0 and Cam1. Furthermore, *transforms.json* contains the extrinsic parameters of all cameras along with *camera_intrinsics.json*. We have withheld the ground truth data for the testing sets which can be evaluated by submitting the relocalized camera poses to our servers which shall be established upon acceptance.

Camera Id	X	Y	Z	Roll	Pitch	Yaw
Cam0	2.2	0.0	1.3	8	0	0
Cam1	2.2	0.5	1.3	8	0	0
Cam2	2.2	-0.5	1.3	8	0	0
Cam3	2.5	0.5	2.3	4	-20	0
Cam4	2.6	-0.7	2.3	-10	27	0
Cam5	3.0	-1.2	2.1	20	14	0

Table 1. This table describes the camera positions and orientations mounted relative to the ego-vehicle.

condition	#cameras	#images individual	#images sequences	#images total
WetNoon	6	500	3	9,000
SoftRainNoon	6	500	3	9,000
WetCloudySunset	6	500	3	9,000
total	-	1,500	9	27,000

Table 2. This table provides a summarized information about the proposed CARLA benchmark. It describes the weather scenarios under which data was collected, the number of cameras, sequences and the images for each.

Note that only for the training and validation sets, we additionally provide dense depth maps and the semantic segmentations for all the images. In order to understand the general structure of the benchmark, the folder tree is given

*These authors contributed equally.

below.

```
benchmark_sample
  └── episode_000
        ├── relocalization.txt
        ├── relocalization_other_weathers.txt
        ├── transforms.json
        ├── calibs
        │     ├── camera.txt
        │     └── stereocalibration.txt
        ├── CameraDepth0
        │     ├── image_00000.png
        │     ├── image_00001.png
        │     └── ...
        ├── .....
        │     └── ....
        ├── CameraDepth5
        │     ├── image_00000.png
        │     ├── image_00001.png
        │     └── .....
        ├── CameraRGB0
        │     ├── image_00000.png
        │     ├── image_00001.png
        │     └── ...
        ├── .....
        │     └── ....
        ├── CameraRGB5
        │     ├── image_00000.png
        │     ├── image_00001.png
        │     └── .....
        ├── CameraSemSeg0
        │     ├── image_00000.png
        │     ├── image_00001.png
        │     └── ...
        ├── .....
        │     └── ....
        └── CameraSemSeg5
              ├── image_00000.png
              ├── image_00001.png
              └── .....
  └── episode_001
        ├── relocalization.txt
        ├── ......
        │     └── .......
        └── .......
```

Our benchmark for the Oxford RobotCar [4] sequences follows the same structure as that for CARLA described above, but only utilizes images recorded from a stereo camera setup rather than having 6 different cameras. The resulting point clouds from oxford sequences are aligned with the global registration followed by ICP alignment using the implementation of Open3D [7]. This alignment was performed for the following sequences: *2014-12-02-15-30-08 (overcast)* and *2015-03-24-13-47-33 (sunny)* for training. For testing, we use the reference sequence *2015-02-24-12-32-19 (sunny)* and align it with the sequences *2015-03-17-11-08-44 (overcast)*, *2014-12-05-11-09-10 (rainy)*, and *2015-02-03-08-45-10 (snow)*.

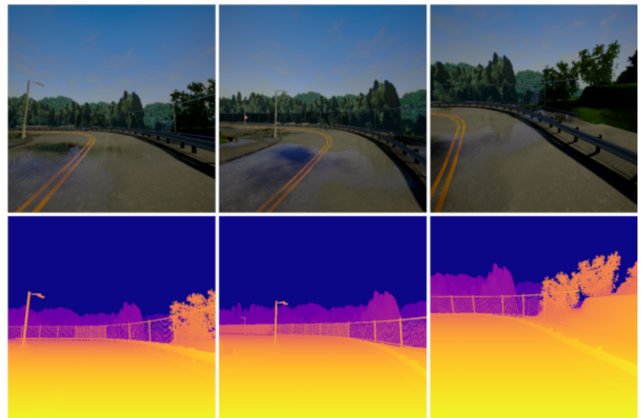

Figure 1. This figure shows a subset of 3 of the 6 camera images (top row) from the same time step along with the corresponding dense depth map (bottom) rendered from the simulation engine. The cameras are oriented at different positions and orientations with respect to each other. This provides 6 DOF. Multiple cameras are used to enhance the variety in the dataset for training a robust deep feature prediction network. However, it is important to mention here that the correspondences used to train our models were determined from the point clouds using DSO [6].

B. Additional Experiments

In these supplementary experiments, we show that our method significantly improves the robustness for large-baseline tracking even when there are no weather/lighting changes involved. This is done by evaluating on our CARLA benchmark with only one weather, as well as on the indoor EuRoC dataset [1].

B.1. CARLA results for different weathers

As mentioned in the main paper we show results on our CARLA benchmark for relocalization between different weather conditions in Figure 2. Figure 3 shows feature matrices produced by our model, showing similar feature maps even for images with differing lighting/weather conditions.

B.2. Evaluation of robustness to large baselines/low frame rates

We demonstrated in the main paper that our method greatly improves robustness and accuracy for tracking across different weathers.

However, even when tracking images with similar lighting conditions our deep features greatly improve tracking

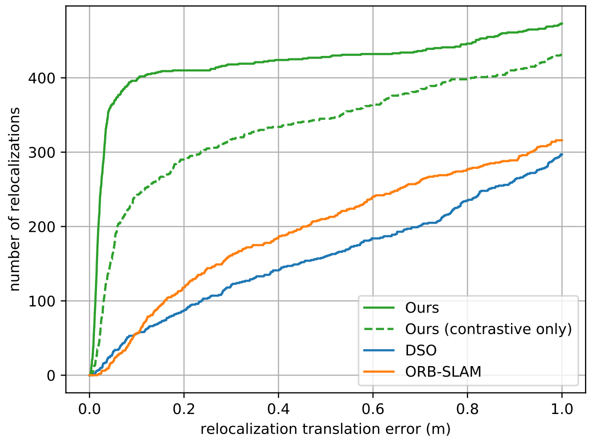

Figure 2. This figure shows the cumulative relocalization accuracy for tracking against different weathers on the CARLA benchmark. ORB-SLAM is more robust to changes in lighting, and weather, whereas DSO shows the worst performance. By utilizing our trained deep descriptors, we are able to outperform both methods by a large margin. Notice that our novel Gauss-Newton loss has a large impact as the model trained only with the contrastive loss performs significantly worse.

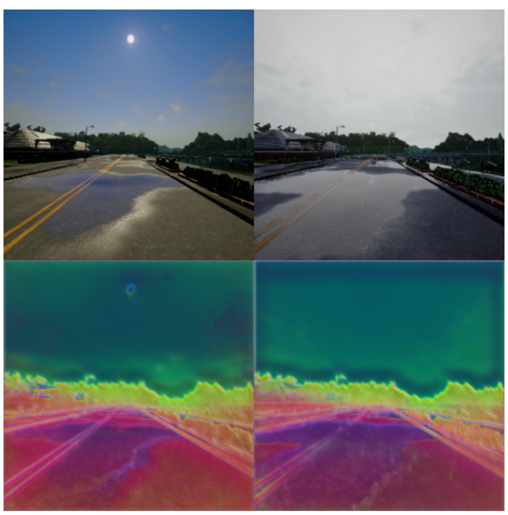
Figure 3. This figure shows images and their corresponding feature maps predicted by our GN-Net for two different weather conditions included in our CARLA benchmark. Despite shadows, raindrops, and water puddles the feature maps are very much similar. Note that the feature maps are displayed via a lower-dimensional representation using PCA.

performance for large baselines, which we will show in this section on the CARLA and the EuRoC datasets. For all experiments in this section we have changed the following two hyperparameters: the vicinity for the GN-Loss is changed from 1 to 3 pixels and the second term of the Gauss-Newton loss is weighted by $2/7$.

CARLA: We use the first three sequences of the training, validation, and test set provided by our benchmark, which all capture the same weather condition. As these sequences do not contain substantial illumination changes any differences in the performance will mainly show the general accuracy and robustness of the methods. Again, all hyperparameter tuning is only performed on the training and validation set. The cumulative relocalization accuracy for tracking against the same weather for the 3 evaluated methods is shown in Figure 4. While DSO is more accurate, the indirect ORB-SLAM system has higher robustness. In contrast, our GN-Net is not only as accurate as DSO, but also more robust than ORB-SLAM. This is because of the larger convergence basin of the features maps created by our network.

EuRoC dataset with low framerates: For this experiment we use a more traditional metric and evaluate on the challenging EuRoC MAV dataset [1]. We run each method on the 11 sequences of the dataset and evaluate the absolute trajectory error of the estimated poses against the groundtruth. Note that in this experiment no relocalization tracking is involved. For ORB-SLAM we have disabled loop-closures and relocalization to enable a fair comparison with the other pure odometry methods. Stereo DSO is used without modification.

For our method we have modified the normal frame-to-frame tracking performed by the *coarse tracker* of Stereo DSO to use deep features instead. As our features have a larger convergence basin than normal images, we expect this to improve the robustness of the tracking against large baselines.

In order to evaluate the performance of our method on all the 11 sequences, we split the sequence into 2 sets, while training 2 different models. Set A contains the first 6 of the 11 sequences, while set B comprises of the remaining 5. The first model is trained with set A and evaluated on set B. The second model is trained with set B and evaluated on set A. The final evaluation reported is the combination of the evaluation results from these 2 models. This way we are able to cover all the 11 sequences in our evaluation.

In order to evaluate the performance of the methods with low-framerate cameras (thus including larger-baselines) we subsample the frames included by skipping n frames for each frame that is used. Each method is run 3 times for all 11 sequences, for each $n \in [0, 7]$. For $n = 7$, e.g. this means only every 8th image is used, simulating a frame-rate of 2.5 Hz. The results are shown in Figure 5, again demonstrating that our features significantly improve the robustness of direct SLAM methods.

C. Network Architecture Details

We adopt a similar network architecture as the U-Net model [5] as seen in Figure 6. What is different is that we change the decoder part such that our feature maps can leverage a multiscale hierarchical feature pyramid which allows propagating information from coarser to finer levels of the pyramid. For the encoder part, we followed the conven-

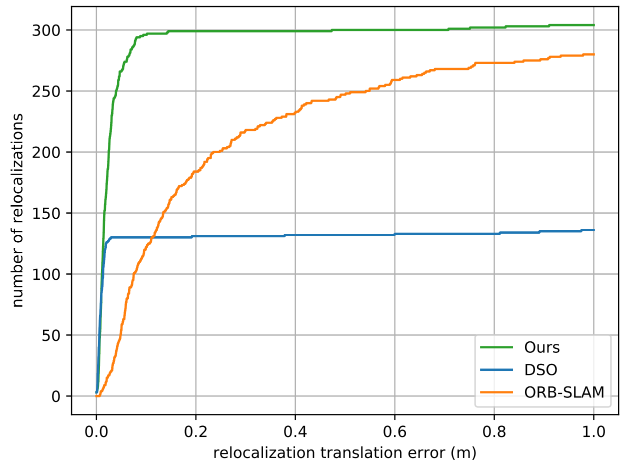

Figure 4. This figure shows the cumulative relocalization accuracy for tracking against the *same weather*. ORB-SLAM tracking is less accurate but more robust than normal direct image alignment. Our approach outperforms both of them. This shows that our approach improves not only robustness to challenging lighting situations, but also to large-baseline tracking.

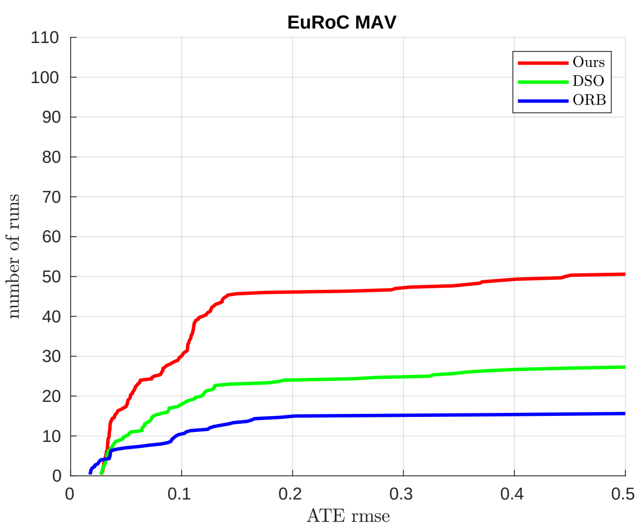

Figure 5. A cumulative plot of the absolute trajectory error (RMSE) on the EuCoC dataset for different numbers of skipped frames. In this experiment, we have replaced the frame-to-frame tracking used in Stereo DSO with our deep feature matrices. We then evaluate the accuracy of the estimated trajectory when running all sequences of the dataset with different frame-rates ranging between 20Hz and 2.5 Hz We compare to normal Stereo DSO and ORB-SLAM without loop-closures. Note that our models for this experiment are trained entirely self-supervised, yet improving DSO robustness almost by a factor of two.

tion of [5]. The encoder part consists of four downsampling blocks, for which each of them uses a 2×2 max pooling operation with stride 2, followed by a convolutional block which is executed two times. The convolutional block applies a 3×3 convolution with padding of 1, followed by batch normalization, and an exponential linear unit (ELU). At each downsampling step, the number of channels is doubled. The number of feature channels is set to 64.

Decoder modification: We change the decoder part of the architecture from the default U-Net architecture in the following way. Beginning from the coarsest level, we upsample (with bilinear interpolation) the feature maps by 2 and concatenate those feature maps with the feature map of the higher level. After this, we apply D number of 1×1 convolution kernels to make the filter maps of the same channel size. This is done in an iterative fashion until the finest level. This results in the final feature pyramid map representation which we use for deep direct SLAM. The feature map sizes are described in Table 3.

Level	S
3 (coarsest)	$D \times H/8 \times W/8$
2	$D \times H/4 \times W/4$
1	$D \times H/2 \times W/2$
0 (finest)	$D \times H \times W$

Table 3. The hyperparameter setting for our network architecture. Level: level of the network. S : size of the feature map. H, W : height and width of the image, respectively.

For all experiments, we use $D = 16$ channels as a feature output vector size. For all trainings, we use the Adam optimizer [3] for optimization with a learning rate of 10^{-6} and a weight decay of 10^{-3}. For the correspondence contrastive loss term, we set the margin $M = 1$. For the Gauss-Newton loss, we set the maximum distance of the start point to the correct point to 1 pixel for all experiments in the paper and to 3 pixels for the experiments in this supplementary material. All steps of the optimizer use a single image pair as input to the network. Each pair of images fed to the Siamese network architecture has a number of positive correspondences and for each of them, a negative correspondence is randomly sampled. For CARLA the input image size is $W = 512$, $H = 512$ and for Oxford RobotCar it is $W = 640$, $H = 480$.

D. Implementation Details

Coupling of GN-Net with DSO: DSO contains two components where images are used for pose estimation. In the Bundle Adjustment (BA) the pose of 8 keyframes together with the inverse depth of all active points is optimized. For this optimization, a very good initialization is assumed. The *coarse tracking* is performed for every image and optimizes the 6DOF pose between the latest keyframe and the current frame together with two affine brightness parameters a and b which represent a brightness transformation between the two images. This is done using normal direct image alignment in a pyramid scheme.

We have adopted the coarse tracker to be able to use our

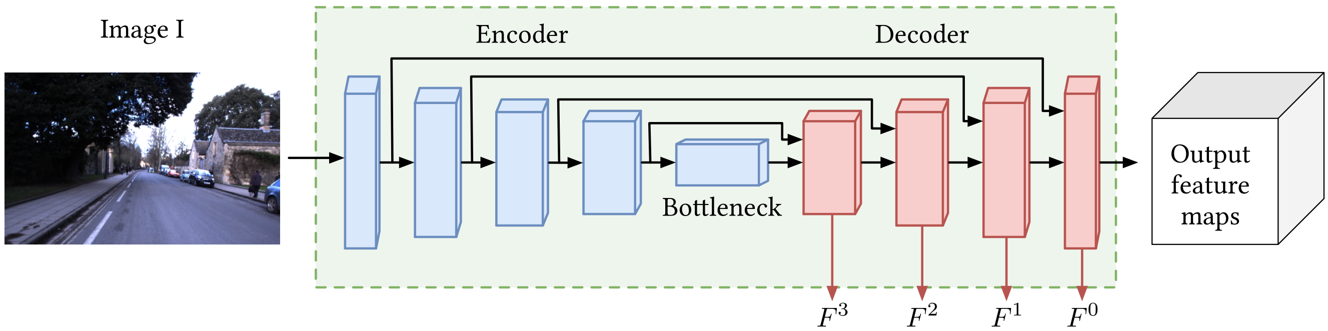

Figure 6. Overview of one branch of the Siamese network architecture. Each branch is a modified U-Net [5] architecture which receives as an input an image and predicts multi-scale feature maps $[F^0, F^1, F^2, F^3]$. The multi-scale feature maps from the decoder network of both branches are then passed and used by DSO. Note that the weights between the two branches are shared.

multi-channel feature maps as an input instead of images. Notice that we directly input *all* the pyramid levels created by our network instead of downscaling the image as it is done in DSO. This modified coarse tracker is then used for relocalization tracking.

Notably, the network only takes a single image as an input to create the feature maps. This means that the runtime of the inference scales linearly with the number of images involved. Therefore it would be possible to also use the features for the BA, although this has not been done in this specific work. Our network extracts features at over 8Hz. In principle this would be enough for real-time operation as we perform relocalization only on keyframes which arrive at less than 5Hz usually.

Details of the relocalization demo: Our method works by performing relocalization tracking as in the previous experiment, which yields a solution for the transformation between the current world and the map $T_{\text{map}}^{\text{world}}$. The results for this transform are optimized in a factor graph with a fixed random walk between successive solutions. As again we do not consider finding candidate images, we supply the first corresponding image in the map to the methods. From there on no supervision signal is given. After bootstrapped with the first image our method finds the next relocalization candidates by determining the keyframe in the map with the minimum distance to the current image, which can be computed using the current solution for the transform $T_{\text{map}}^{\text{world}}$.

E. Additional proofs

Here, we show why the linear system used in the Gauss-Newton algorithm (Equation (6) of the main paper) also defines a Gaussian probability distribution.

The Gauss-Newton algorithm assumes a Taylor expansion of the energy:

$$E(\mathbf{x}') \approx E(\mathbf{x}_0) + \mathbf{b}^T(\mathbf{x}' - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x}' - \mathbf{x}_0)^T\mathbf{H}(\mathbf{x}' - \mathbf{x}_0) \tag{1}$$

The optimal solution according to this approximation can be obtained by finding the point where the derivative of the energy is 0

$$0 \stackrel{!}{=} \frac{dE}{d\mathbf{x}} = \mathbf{b} + \mathbf{H}(\mathbf{x}' - \mathbf{x}_0) \tag{2}$$

$$\mathbf{x}' = \mathbf{x}_0 - \mathbf{H}^{-1} \cdot \mathbf{b} \tag{3}$$

As the taylor expansion is performed around $\mathbf{x_0}$ we from now on set $\mathbf{x} = \mathbf{x}' - \mathbf{x_0}$.

On the other hand, we can try to find the the maximum point of a Gaussian distribution, by minimizing the negative log-likelihood (Equation (7-8) in the main paper):

$$E(\mathbf{x}) = -\log f_X(\mathbf{x}) = \tag{4}$$

$$\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) + \log\left(2\pi\sqrt{|\boldsymbol{\Sigma}|}\right) = \tag{5}$$

$$\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}^T - \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \tag{6}$$

$$\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \log\left(2\pi\sqrt{|\boldsymbol{\Sigma}|}\right)\boldsymbol{\mu} \tag{7}$$

Now we set $\mathbf{H} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\mu} = -\mathbf{H}^{-1}\mathbf{b}$. Then

$$(6) = \frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}^T - \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \text{const} = \tag{8}$$

$$\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{b}^T\mathbf{x} = (1) \tag{9}$$

This equality shows that our the linear system in the Gauss-Newton step also represents a Gaussian probability distribution with covariance $\mathbf{H}^{-1}$ and a mean at the solution of the Gauss-Newton step.

References

[1] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC Micro Aerial Vehicle Datasets. *IJRR*, 35(10), 2016.

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *CoRL*, 2017.

[3] D. P. Kingma and J. L. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[4] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 36(1), 2017.

[5] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015.

[6] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *ICCV*, 2017.

[7] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv 2018*, 2018.