Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses –Supplement–

Eric Brachmann and Carsten Rother
Visual Learning Lab
Heidelberg University (HCI/IWR)
http://vislearn.de

In this supplementary material, we provide additional implementation details that did not fit into the main paper but can be helpful when reproducing our results. Note that we will also make our code publicly available[1], including pre-trained models. This document contains:

Essential Matrix Estimation:

- list of scenes used for training and testing
- network architecture
- initialization procedure
- implementation details
- qualitative results
- detailed comparison with USAC
- runtime discussion

Fundamental Matrix Estimation:

- implementation details
- qualitative results

Horizon Lines:

- network architecture
- implementation details
- qualitative results

Camera Re-Localization:

- network architecture
- implementation details
- learned 3D representations

[1] vislearn.de/research/neural-guided-ransac/

1. Essential Matrix Estimation

List of Scenes Used for Training and Testing.

Training:

- Staint Peter's (Outdoor)
- brown_bm_3 - brown_bm_3 (Indoor)

Testing (Outdoor):

- Buckingham
- Notre Dame
- Sacre Coeur
- Reichstag
- Fountain
- HerzJesu

Testing (Indoor):

- brown_cogsci_2 - brown_cogsci_2
- brown_cogsci_6 - brown_cogsci_6
- brown_cogsci_8 - brown_cogsci_8
- brown_cs_3 - brown_cs3
- brown_cs_7 - brown_cs7
- harvard_c4 - hv_c4_1
- harvard_c10 - hv_c10_2
- harvard_corridor_lounge - hv_lounge1_2
- harvard_robotics_lab - hv_s1_2
- hotel_florence_jx - florence_hotel_stair_room_all
- mit_32_g725 - g725_1
- mit_46_6conf - bcs_floor6_conf_1
- mit_46_6lounge - bcs_floor6_long
- mit_w85g - g_0
- mit_w85h - h2_1

Network Architecture. As mentioned in the main paper, we replicated the architecture of Yi *et al*. [19] for our experiments on epipolar geometry (estimating essential and fundamental matrices). For a schematic overview see Fig. 1. The network takes a set of feature correspondences as input, and predicts as output a weight for each correspondence which we use to guide RANSAC hypothesis sam-

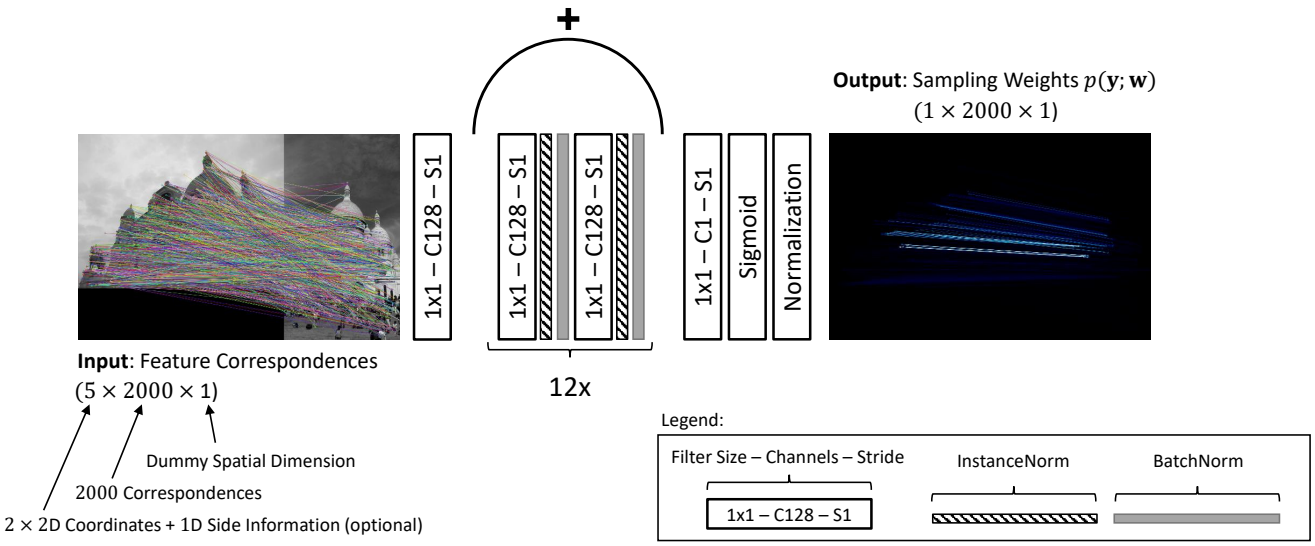

Figure 1. **NG-RANSAC Network Architecture for F/E-matrix Estimation.** The network takes a set of feature correspondences as input and predicts as output a weight for each correspondence. The network consists of linear layers interleaved by instance normalization [17], batch normalization [8] and ReLUs [6]. The arc with a plus marks a skip connection for each of the twelve blocks [7]. This architecture was proposed by Yi *et al*. [19].

pling. The network consists of a series of multilayer perceptrons (MLPs) that process each correspondence independently. We implement the MLPs with 1×1 convolutions. The network infuses global context via instance normalization layers [17], and it accelerate training via batch normalization [8]. The main body of the network is comprised of 12 blocks with skip connections [7]. Each block consists of two linear layers followed by instance normalization, batch normalization and a ReLU activation [6] each. We apply a Sigmoid activation to the last layer, and normalize by dividing by the sum of outputs.[2]

Initialization Procedure. We initialize our network in the following way. We define a target sampling distribution $g(\mathbf{y}; E^*)$ using the ground truth essential matrix E^* given for each training pair. Intuitively, the target distribution should return a high probability when a correspondence $\mathbf{y}$ is aligned with the ground truth essential matrix E^*, and a low probability otherwise. We assume that correspondence $\mathbf{y}$ is a 4D vector containing two 2D image coordinates $\mathbf{x}$ and $\mathbf{x}'$ (3D in homogeneous coordinates). We define the epipolar error of a correspondence w.r.t. essential matrix E:

$$d(\mathbf{y}, E) = \frac{(\mathbf{x}'^{\top} E \mathbf{x})^2}{[E\mathbf{x}]_0^2 + [E\mathbf{x}]_1^2 + [E^{\top}\mathbf{x}']_0^2 + [E^{\top}\mathbf{x}']_1^2}, \quad (1)$$

where $[\cdot]_i$ returns the ith entry of a vector. Using the epipolar error, we define the target sampling distribution:

$$g(\mathbf{y}; E^*) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{d(\mathbf{y}, E^*)}{2\sigma^2}\right). \quad (2)$$

[2]The original architecture of Yi *et al.* [19] uses a slightly different output processing due to using the output as weights for a robust model fit. They use a ReLU activation followed by a tanh activation.

Parameter σ controls the softness of the target distribution, and we use $\sigma = 10^{-3}$ which corresponds to the inlier threshold we use for RANSAC. To initialize our network, we minimize the KL divergence between the network prediction $p(\mathbf{y}; w)$ and the target distribution $g(\mathbf{y}; E^*)$. We initialize for 75k iterations using Adam [10] with a learning rate of 10^{-3} and a batch size of 32.

Implementation Details. For the following components we rely on the implementations provided by OpenCV [2]: the 5-point algorithm [13], epipolar error, SIFT features [12], feature matching, and essential matrix decomposition. We extract 2000 features per input image which yields 2000 correspondences for image pairs after matching. When applying Lowe's ratio criterion [12] for filtering and hence reducing the number of correspondences, we randomly duplicate correspondences to restore the number of 2000. We minimize the expected task loss using Adam [10] with a learning rate of 10^{-5} and a batch size of 32. We choose hyperparameters based on validation error of the *Reichstag* scene. We observe that the magnitude of the validation error corresponds well to the magnitude of the training error, *i.e*. a validation set would not be strictly required.

When calculating the AUC for evaluation, we adhere to the protocol of Yi et al. [19] to ensure comparability. Yi et al. approximate the AUC via the area under the cumulative histogram with a bin width of 5°.

Qualitative Results. We present additional qualitative results for indoor and outdoor scenarios in Fig. 2. We compare results of RANSAC and NG-RANSAC, also visualizing neural guidance. We obtain these results in the high-outlier setup, *i.e*. without using Lowe's ratio criterion and without using side information as additional network input.

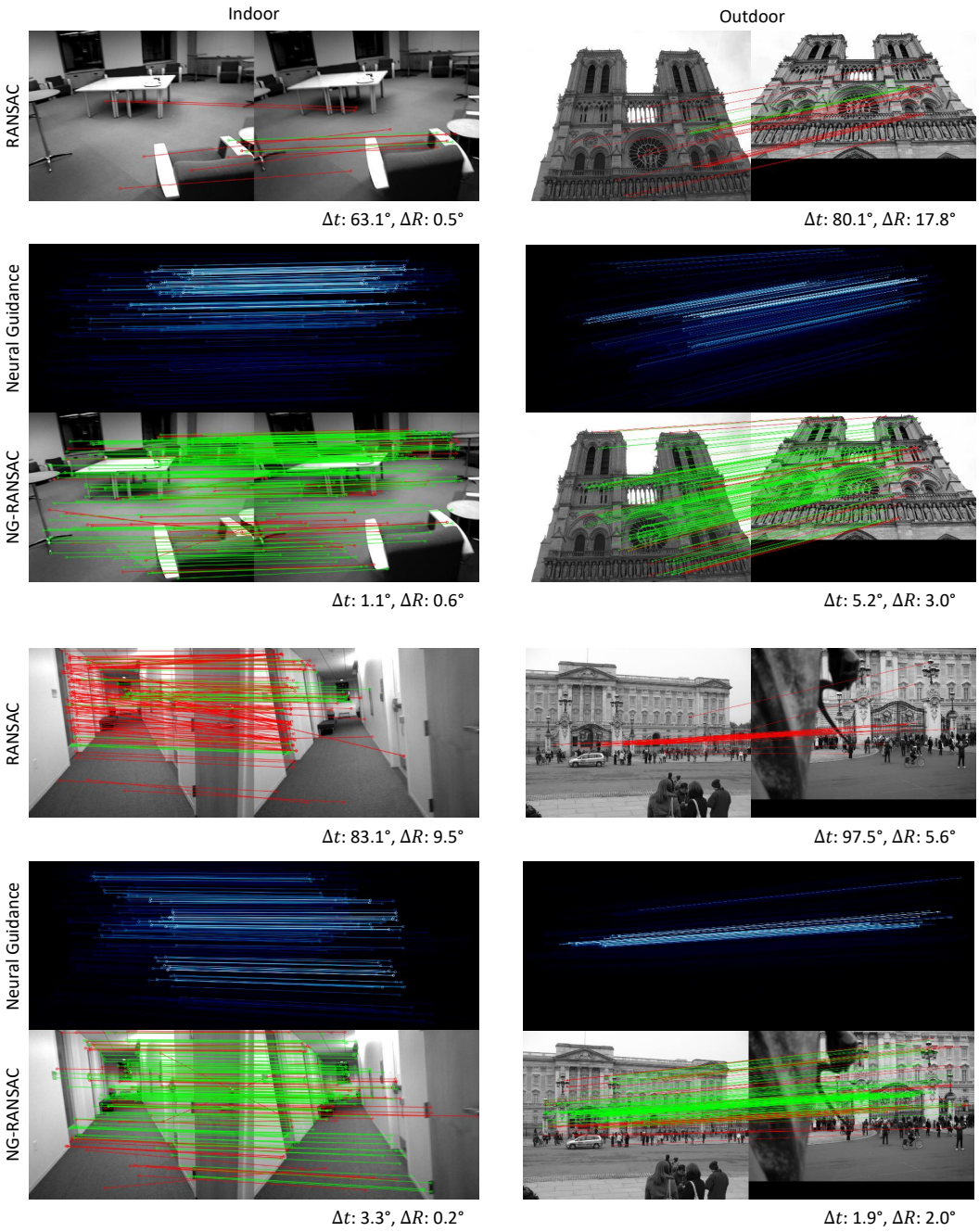

Figure 2. **Qualitative Results for Essential Matrix Estimation.** We compare results of RANSAC and NG-RANSAC. Below each result, we give the angular error between estimated and true translation vectors, and estimated and true rotation matrices. We draw correspondences in green if they adhere to the ground truth essential matrix with an inlier threshold of 10^{-3}, and red otherwise.

Detailed Comparison with USAC. The accuracy of USAC [14] and NG-RANSAC depend on the hypothesis budget M, see Fig. 3. NG-RANSAC finds good hypotheses much earlier than USAC, and achieves a reasonable accuracy by drawing as few as 10 hypotheses. Fig. 4 shows a visualization of progressive hypotheses search. USAC is designed to draw the same hypotheses as RANSAC but in a different order. Therefore, USAC samples degenerate hypotheses (poor accuracy but high inlier count) eventually, even if it gives them a low priority, see Fig. 4 bottom. NG-RANSAC learns to suppress such hypotheses more effectively.

Interestingly, passing our learned weights to USAC achieves significantly lower accuracy than passing matching ratios to USAC. For example, for the outdoor setting, w/o ratio filter and $M = 10^3$, USAC achieves -0.27/-0.24/-0.34 AUC for $5°/10°/20°$ when using our

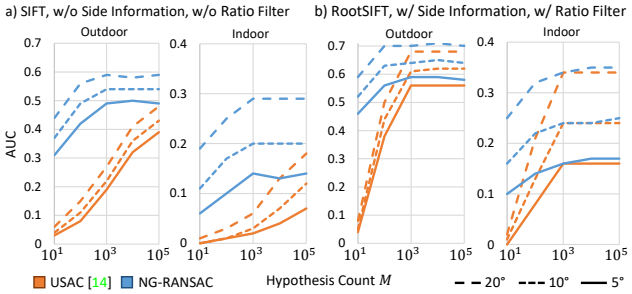

Figure 3. **Accuracy vs. Hypothesis Budget.** We compare the AUC of NG-RANSAC and USAC [14] for increasing number of hypotheses M. **a)** with and **b)** without side information.

weights. The USAC/PROSAC sampling scheme assumes that the probability of correspondences being inliers increases monotonically with the sampling weight [3]. In contrast, our training objective optimizes over entire pools of hypotheses where correspondences are sampled independently. Individual outlier correspondences might be ranked high by the neural network, without affecting accuracy negatively, thus violating the assumption of PROSAC.

Runtime. A forward pass of the network takes 3ms on CPU (similar for GPU). The total runtime (and accuracy) depends on the hypothesis count M. For $M = 10^3$, our implementation of NG-RANSAC takes 90ms per image pair. For $M = 10$, it takes 21ms.

2. Fundamental Matrix Estimation

Implementation Details. We reuse the architecture of Fig. 1. To normalize image coordinates of feature matches, we subtract the mean coordinate and divide by the coordinate standard deviation, where we calculate mean and standard deviation over the training set. Ranftl and Koltun [15] fit the final fundamental matrix to the top 20 weighted correspondences as predicted by their network. Similarly, we re-fit the final fundamental matrix to the largest inlier set found by NG-RANSAC. This refinement step results in a small but noticeable increase in accuracy. For the following components we rely on the implementations provided by OpenCV [2]: the 7-point algorithm, epipolar error, SIFT features [12] and feature matching.

Qualitative Results. We present additional qualitative results for the Kitti dataset [5] in Fig. 5. We compare results of RANSAC and NG-RANSAC, also visualizing neural guidance as predicted by our network.

3. Horizon Lines

Network Architecture. We provide a schematic of our network architecture for horizon line estimation in Fig. 6. The network takes a 256×256px image as input. We re-scale images of arbitrary aspect ratio such that the long side is 256px. We symmetrically zero-pad the short side to 256px.

The network has two output branches. The first branch predicts a set of $8 \times 8 = 64$ 2D points, our observations $\mathbf{y}(\mathbf{w})$, to which we fit the horizon line. We apply a Sigmoid and re-scale output points to [-1.5,1.5] in relative image coordinates to support horizon lines outside the image area. We implement the network in a fully convolutional way [11], *i.e.* each output point is predicted for a patch, or restricted receptive field, of the input image. Therefore, we shift the coordinate of each output point to the center of its associated patch.

The second branch predicts sampling probabilities $p(\mathbf{y}; \mathbf{w})$ for each output point. We apply a Sigmoid to the output of the second branch, and normalize by dividing by the sum of outputs. During training, we block the gradients of the second output branch when back propagating to the base network. The sampling gradients have larger variance and magnitude than the observation gradients of the first branch, especially in the beginning of training with a negative effect on convergence of the network as a whole. Intuitively, we want to give priority to the observation prediction because they determine the accuracy of the final model parameters. The sampling prediction should address deficiencies in the observation predictions without influencing them too much. The gradient blockade ensures these properties.

Implementation Details. We use a differentiable soft inlier count [1] as scoring function, *i.e.*:

$$s(\mathbf{h}, \mathcal{Y}) = \alpha \sum_{\mathbf{y} \in \mathcal{Y}} 1 - \text{sig}[\beta d(\mathbf{y}, \mathbf{h}) - \beta\tau], \tag{3}$$

where $d(\mathbf{y}, \mathbf{h})$ denotes the point-line distance between observation $\mathbf{y}$ and line hypothesis $\mathbf{h}$. Hyperparameter α determines the softness of the scoring distribution in DSAC, β determines the softness of the Sigmoid, and τ is the inlier threshold. We use $\alpha = 0.1$, $\beta = 100$ and $\tau = 0.05$.

We convert input images to grayscale, and apply the following data augmentation strategy during training: We randomly adjust brightness and contrast in the range of $\pm 10\%$. We randomly rotate/scale/shift images (and ground truth horizon lines) in the range of $\pm 5°/20\%/8$px.

As discussed in the main paper, we use the normalized maximum distance between a line hypothesis and the ground truth horizon in the image as task loss ℓ. This can lead to stability issues when we sample line hypotheses with very steep slope. Therefore, we clamp the task loss to a maximum of 1, *i.e.* the normalized image height.

As mentioned before, some images in the HLW dataset [18] have their horizon outside the image. Some of these images contain virtually no visual cue where the horizon exactly lies. Therefore, we find it beneficial to use a robust variant of the task loss ℓ' that limits the influence of such outliers. We use:

$$\ell' = \begin{cases} \ell & \ell < 0.25 \\ 0.25\sqrt{\ell} & \text{otherwise} \end{cases}, \tag{4}$$

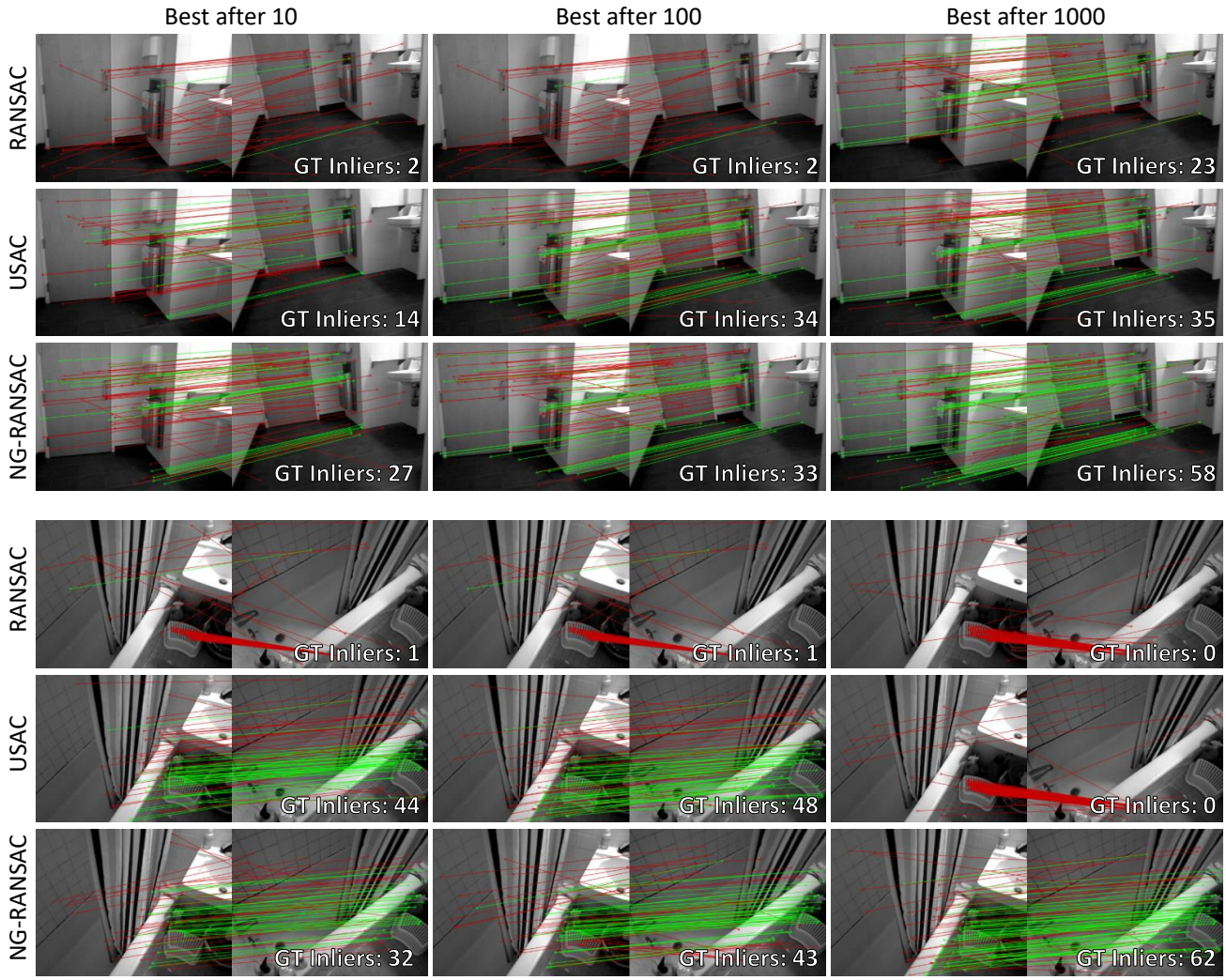

Figure 4. **Hypothesis Search.** We visualize the best hypothesis found after $M \in \{10, 100, 1000\}$ iterations for RANSAC [4], USAC [14] and NG-RANSAC. For each result, we give the number of correspondences which are also inliers for the ground truth model (*GT Inliers*, drawn in green). We perform this experiment in the *Indoor* scenario, using side information and RootSIFT but without Lowe's ratio filter.

i.e. we use the square root of the task loss after a magnitude of 0.25, which is the magnitude up to which the AUC is calculated when evaluating on HLW [18].

Qualitative Results. We present additional qualitative results for the HLW dataset [18] in Fig. 7.

4. Camera Re-Localization

Network Architecture. We provide a schematic of our network architecture for camera re-localization in Fig. 8. The network is a FCN [11] that takes an RGB image as input, and predicts dense outputs, sub-sampled by a factor of 8. The network has two output branches. The first branch predicts 3D scene coordinates [16], our observations $\mathbf{y}(\mathbf{w})$, to which we fit the 6D camera pose. The second output branch predicts sampling probabilities $p(\mathbf{y}; \mathbf{w})$ for the scene coordinates. We apply a Sigmoid to the output of the second branch, and normalize by dividing by the sum of outputs. During training, we block the gradients of the second output branch when back propagating to the base network. The sampling gradients have larger variance and magnitude than the observation gradients of the first branch, especially in the beginning of training. This has a negative effect on convergence of the network as a whole. Intuitively, we want to give priority to the scene coordinate prediction because they determine the accuracy of the pose estimate. The sampling prediction should address deficiencies in the scene coordinate predictions without influencing them too much. The gradient blockade ensures these properties.

Implementation details. We follow the three-stage training procedure proposed by Brachmann and Rother for DSAC++ [1].

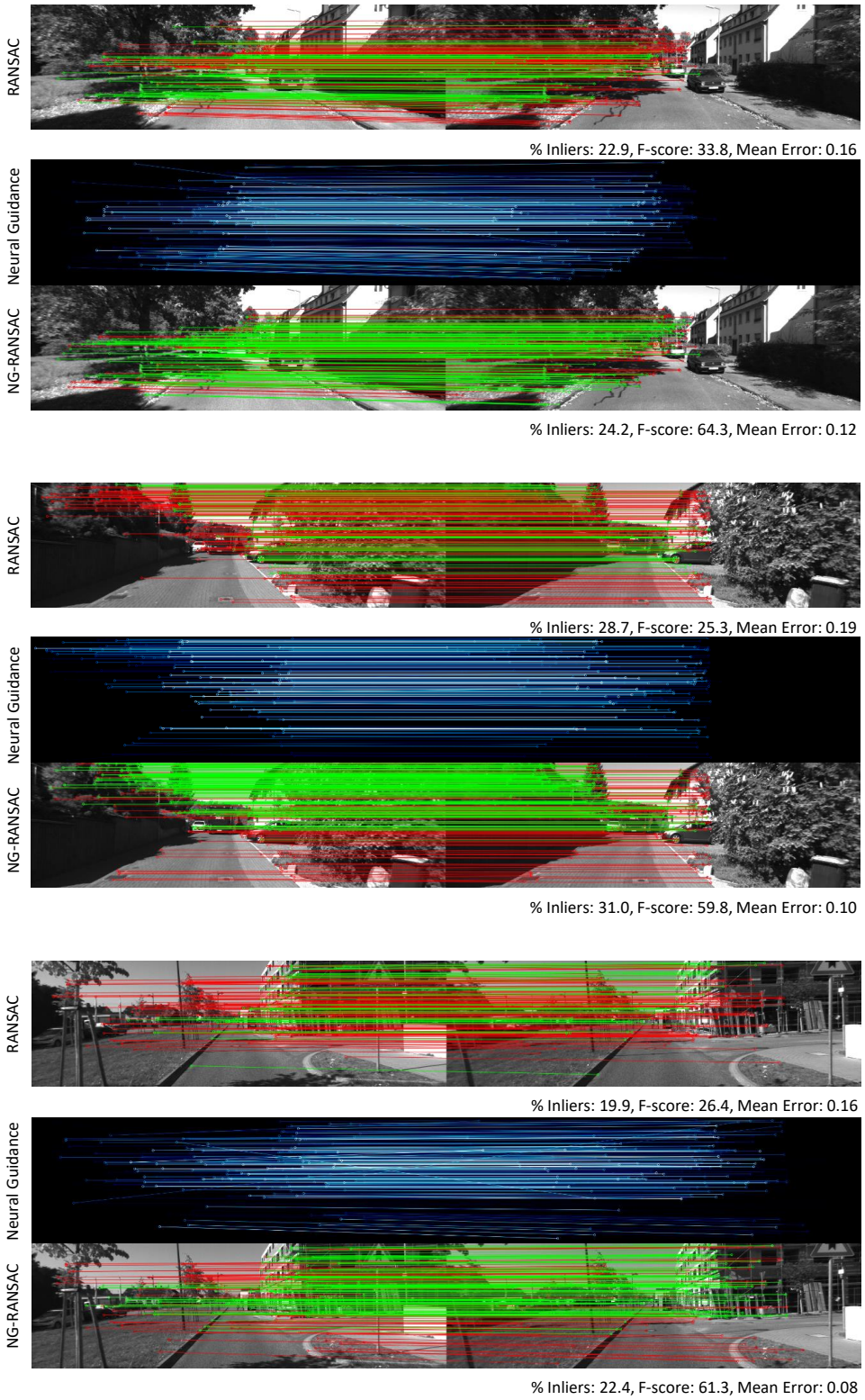

Figure 5. **Qualitative Results for Fundamental Matrix Estimation.** We compare results of RANSAC and NG-RANSAC. Below each result, we give the percentage of inliers of the final model, the F-score which measures the alignment of estimated and true fundamental matrix, and the mean epipolar error of estimated inlier correspondences w.r.t. the ground truth fundamental matrix. We draw correspondences in green if they adhere to the ground truth fundamental matrix with an inlier threshold of 0.1px, and red otherwise.

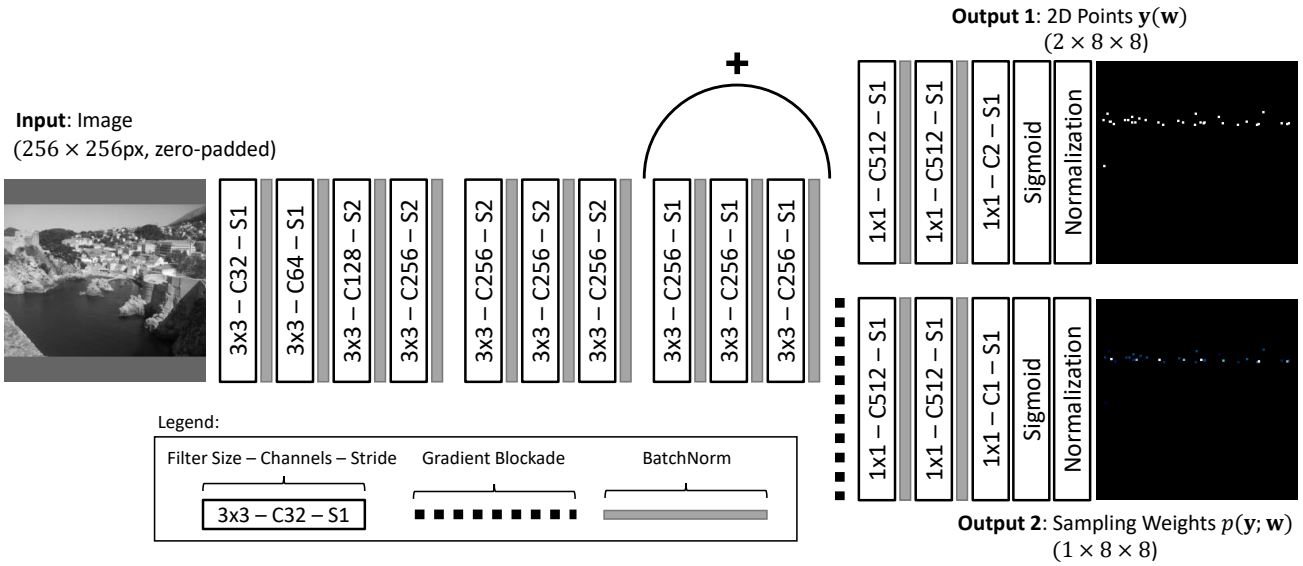

Figure 6. **NG-DSAC Network Architecture for Horizon Line Estimation.** The network takes a grayscale image as input and predicts as output a set of 2D points and corresponding sampling weights. The network consists of convolution layers interleaved by batch normalization [8] and ReLUs [6]. The arc with a plus marks a skip connection [7]. We use the gradient blockage during training to prevent direct influence of the sampling prediction (second branch) to learning the observations (first branch).

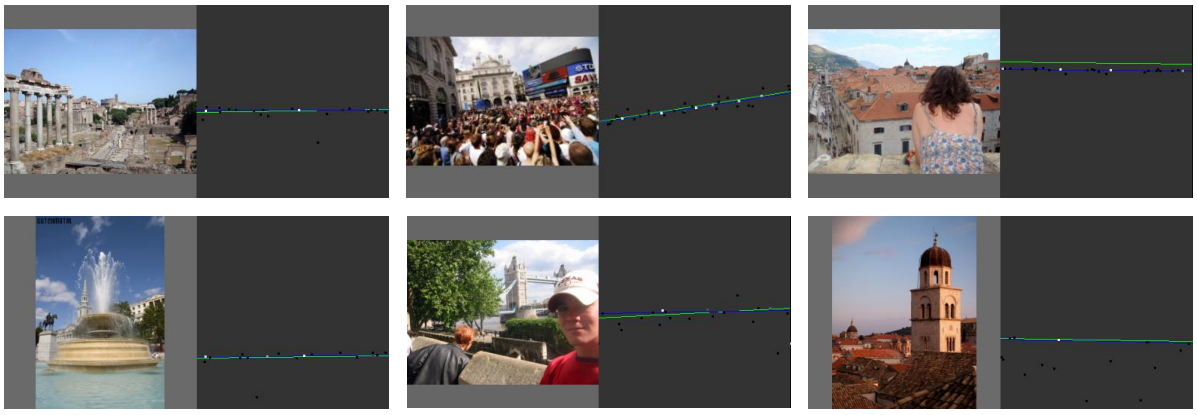

Figure 7. **Qualitative Results for Horizon Line Estimation.** Next to each input image, we show the estimated horizon line in blue and the true horizon line in green. We also show the observation points predicted by our network, colored by their sampling weight (dark = low).

Firstly, we optimize the distance between predicted and ground truth scene coordinates. We obtain ground truth scene coordinates by rendering the sparse reconstructions given in the Cambridge Landmarks dataset [9]. We ignore pixels with no corresponding 3D point in the reconstruction. Since the reconstructions contain outlier 3D points, we use the following robust distance:

$$(\mathbf{y}, \mathbf{y}^*) = \begin{cases} ||\mathbf{y} - \mathbf{y}^*||_2 & ||\mathbf{y} - \mathbf{y}^*||_2 < 10 \\ 10\sqrt{||\mathbf{y} - \mathbf{y}^*||_2} & \text{otherwise} \end{cases}, \quad (5)$$

i.e. we use the Euclidean distance up to a threshold of 10m after which we use the square root of the Euclidean distance. We train the first stage for 500k iterations using Adam [10] with a learning rate of 10^{-4} and a batch size of 1 image.

Secondly, we optimize the reprojection error of the scene coordinate predictions w.r.t. to the ground truth camera pose. Similar to the first stage, we use a robust distance function with a threshold of 10px after which we use the square root of the reprojection error. We train the second stage for 300k iterations using Adam [10] with a learning rate of 10^{-4} and a batch size of 1 image.

Thirdly, we optimize the expected task loss according to the NG-DSAC objective as explained in the main paper. As task loss we use $\ell = \angle(\boldsymbol{\theta}, \boldsymbol{\theta}^*) + ||\mathbf{t} - \mathbf{t}^*||_2$. We measure the angle between estimated camera rotation $\boldsymbol{\theta}$ and ground truth rotation $\boldsymbol{\theta}^*$ in degree. We measure the distance between the

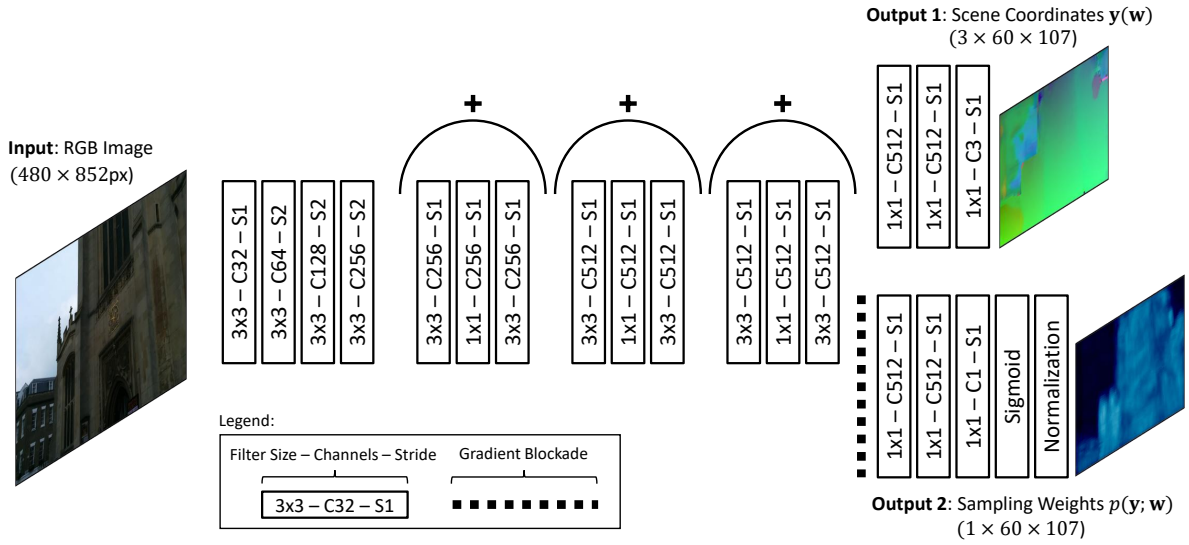

Figure 8. **NG-DSAC++ Network Architecture for Camera Re-Localization.** The network takes an RGB image as input and predicts as output dense scene coordinates and corresponding sampling weights. The network consists of convolution layers followed by ReLUs [6]. Am arc with a plus marks a skip connection [7]. We use the gradient blockage during training to prevent direct influence of the sampling prediction (second branch) to learning the scene coordinates (first branch).

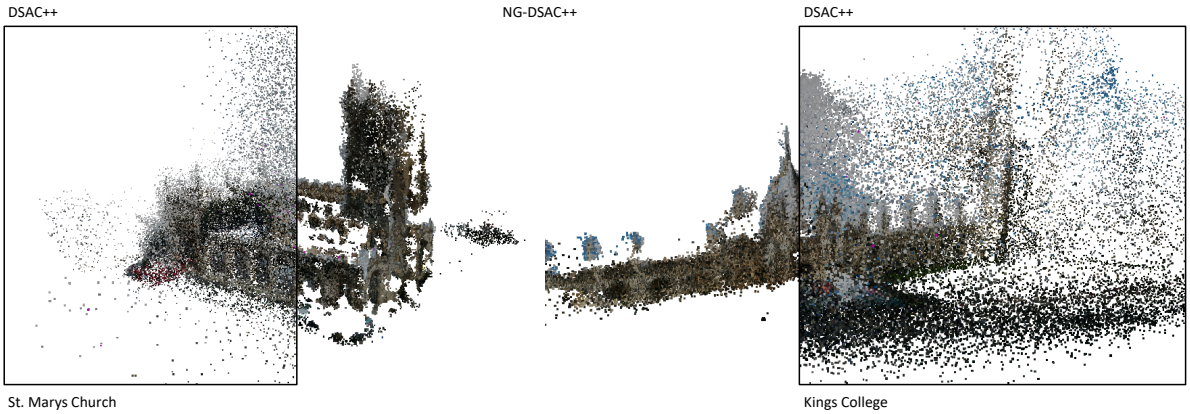

Figure 9. **Learned 3D Representations.** We visualize the internal representation of the neural network. We predict scene coordinates for each training image, plotting them with their RGB color. For DSAC++ we choose training pixels randomly, for NG-DSAC++ we choose randomly among the top 1000 pixels per training image according to the predicted distribution.

estimated camera position $\mathbf{t}$ and ground truth position $\mathbf{t}^*$ in meters. As with horizon line estimation (see previous section), we use a soft inlier count as hypothesis scoring function with hyperparameters $\alpha = 10$, $\beta = 0.5$ and $\tau = 10$. We train the third stage for 200k iterations using Adam [10] with a learning rate of 10^{-6} and a batch size of 1 image.

In the main paper, we report the translational accuracy. The median rotational accuracies are between 0.2° to 0.3° for all scenes, and do hardly vary between methods. In our experiments, we omitted the *Street* scene. Like DSAC++ [1] we failed to achieve sensible results for it. By visual inspection, the corresponding SfM reconstruction seems to be of poor quality, which potentially harms training.

Learned 3D Representations. We visualize the internal 3D scene representations learned by DSAC++ and NG-DSAC++ in Fig. 9 for two more scenes.

References

[1] Eric Brachmann and Carsten Rother. Learning less is more-6D camera localization via 3D surface regression. In *CVPR*, 2018. 4, 5, 8

[2] Gary Bradski. OpenCV. *Dr. Dobb's Journal of Software Tools*, 2000. 2, 4

[3] Ondřej Chum and Jiří Matas. Matching with PROSAC - Progressive sample consensus. In *CVPR*, 2005. 4

[4] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981. 5

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012. 4

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015. 2, 7, 8

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 7, 8

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2, 7

[9] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *ICCV*, 2015. 7

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2, 7, 8

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 4, 5

[12] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 2, 4

[13] David Nistér. An efficient solution to the five-point relative pose problem. *TPAMI*, 2004. 2

[14] Rahul Raguram, Ondřej Chum, Marc Pollefeys, Jiří Matas, and Jan-Michael Frahm. USAC: A universal framework for random sample consensus. *TPAMI*, 2013. 3, 4, 5

[15] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 4

[16] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, 2013. 5

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, 2016. 2

[18] Scott Workman, Menghua Zhai, and Nathan Jacobs. Horizon lines in the wild. In *BMVC*, 2016. 4, 5

[19] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 1, 2