Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses

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

Abstract

We present Neural-Guided RANSAC (NG-RANSAC), an extension to the classic RANSAC algorithm from robust optimization. NG-RANSAC uses prior information to improve model hypothesis search, increasing the chance of finding outlier-free minimal sets. Previous works use heuristic side information like hand-crafted descriptor distance to guide hypothesis search. In contrast, we learn hypothesis search in a principled fashion that lets us optimize an arbitrary task loss during training, leading to large improvements on classic computer vision tasks. We present two further extensions to NG-RANSAC. Firstly, using the inlier count itself as training signal allows us to train neural guidance in a self-supervised fashion. Secondly, we combine neural guidance with differentiable RANSAC to build neural networks which focus on certain parts of the input data and make the output predictions as good as possible. We evaluate NG-RANSAC on a wide array of computer vision tasks, namely estimation of epipolar geometry, horizon line estimation and camera re-localization. We achieve superior or competitive results compared to state-of-the-art robust estimators, including very recent, learned ones.

1. Introduction

Despite its simplicity and time of invention, Random Sample Consensus (RANSAC) [12] remains an important method for robust optimization, and is a vital component of many state-of-the-art vision pipelines [39, 40, 29, 6]. RANSAC allows accurate estimation of model parameters from a set of observations of which some are outliers. To this end, RANSAC iteratively chooses random sub-sets of observations, so called minimal sets, to create model hypotheses. Hypotheses are ranked according to their consensus with all observations, and the top-ranked hypothesis is returned as the final estimate.

The main limitation of RANSAC is its poor performance in domains with many outliers. As the ratio of outliers increases, RANSAC requires exponentially many iterations to find an outlier-free minimal set. Implementations of RANSAC therefore often restrict the maximum number of iterations, and return the best model found so far [7].

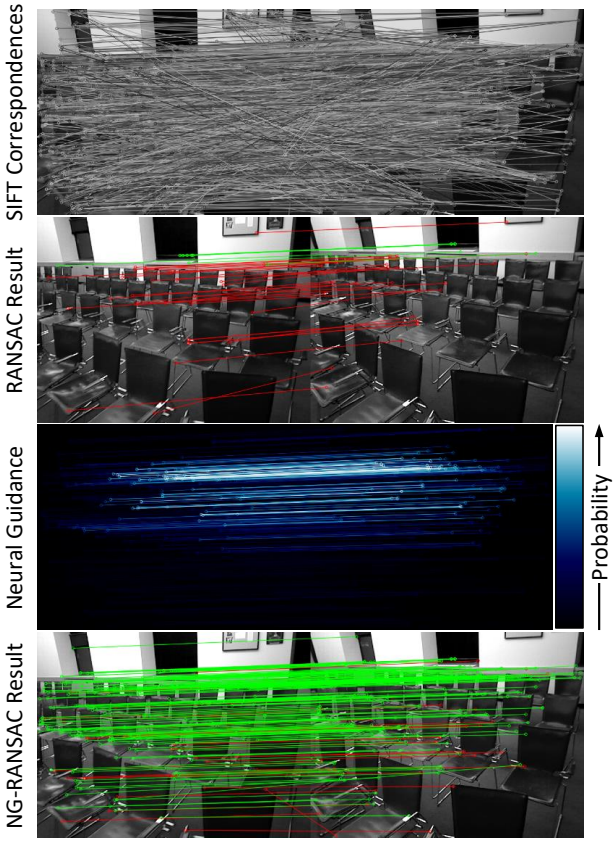

Figure 1. **RANSAC vs. NG-RANSAC.** We extract 2000 SIFT correspondences between two images. With an outlier rate of 88%, RANSAC fails to find the correct relative transformation (green correct and red wrong matches). We use a neural network to predict a probability distribution over correspondences. Over 90% of the probability mass falls onto 239 correspondences with an outlier rate of 33%. NG-RANSAC samples minimal sets according to this distribution, and finds the correct transformation up to an angular error of less than 1°.

In this work, we combine RANSAC with a neural network that predicts a weight for each observation. The weights ultimately guide the sampling of minimal sets. We call the resulting algorithm Neural-Guided RANSAC (NG-RANSAC). A comparison of our method with vanilla RANSAC can be seen in Fig. 1.

When developing NG-RANSAC, we took inspiration from recent work on learned robust estimators [56, 36]. In particular, Yi *et al*. [56] train a neural network to classify observations as outliers or inliers, fitting final model parameters only to the latter. Although designed to replace RANSAC, their method achieves best results when combined with RANSAC during test time, where it would remove any outliers that the neural network might have missed. This motivates us to train the neural network in conjunction with RANSAC in a principled fashion, rather than imposing it afterwards.

Instead of interpreting the neural network output as soft inlier labels for a robust model fit, we let the output weights guide RANSAC hypothesis sampling. Intuitively, the neural network should learn to decrease weights for outliers, and increase them for inliers. This paradigm yields substantial flexibility for the neural network in allowing a certain misclassification rate without negative effects on the final fitting accuracy due to the robustness of RANSAC. The distinction between inliers and outliers, as well as which misclassifications are tolerable, is solely guided by the minimization of the task loss function during training. Furthermore, our formulation of NG-RANSAC facilitates training with any (non-differentiable) task loss function, and any (non-differentiable) model parameter solver, making it broadly applicable. For example, when fitting essential matrices, we may use the 5-point algorithm rather than the (differentiable) 8-point algorithm which other learned robust estimators rely on [56, 36]. The flexibility in choosing the task loss also allows us to train NG-RANSAC self-supervised by using maximization of the inlier count as training objective.

The idea of using guided sampling in RANSAC is not new. Tordoff and Murray first proposed to guide the hypothesis search of MLESAC [48], using side information [47]. They formulated a prior probability of sparse feature matches being valid based on matching scores. While this has a positive affect on RANSAC performance in some applications, feature matching scores, or other hand-crafted heuristics, were clearly *not designed* to guide hypothesis search. In particular, calibration of such ad-hoc measures can be difficult as the reliance on over-confident but wrong prior probabilities can yield situations where the same few observations are sampled repeatedly. This fact was recognized by Chum and Matas who proposed PROSAC [9], a variant of RANSAC that uses side information only to change *the order* in which RANSAC draws minimal sets. In the worst case, if the side information was not useful at all, their method would degenerate to vanilla RANSAC. NG-RANSAC takes a different approach in (i) learning the weights to guide hypothesis search rather than using hand-crafted heuristics, and (ii) integrating RANSAC itself in the training process which leads to self-calibration of the predicted weights.

Recently, Brachmann *et al*. proposed differentiable RANSAC (DSAC) to learn a camera re-localization pipeline [4]. Unfortunately, we can not directly use DSAC to learn hypothesis sampling since DSAC is only differentiable w.r.t. to observations, not sampling weights. However, NG-RANSAC applies a similar trick also used to make DSAC differentiable, namely the optimization of the expected task loss during training. While we do not rely on DSAC, neural guidance can be used in conjunction with DSAC (NG-DSAC) to train neural networks that predict observations and observation confidences at the same time.

We summarize our main contributions:

- We present NG-RANSAC, a formulation of RANSAC with learned guidance of hypothesis sampling. We can use any (non-differentiable) task loss, and any (non-differentiable) minimal solver for training.
- Choosing the inlier count itself as training objective facilitates self-supervised learning of NG-RANSAC.
- We use NG-RANSAC to estimate epipolar geometry of image pairs from sparse correspondences, where it surpasses competing robust estimators.
- We combine neural guidance with differentiable RANSAC (NG-DSAC) to train neural networks that make accurate predictions for parts of the input, while neglecting other parts. These models achieve competitive results for horizontal line estimation, and state-for-the-art for camera re-localization.

2. Related Work

RANSAC was introduced in 1981 by Fischler and Bolles [12]. Since then it was extended in various ways, see *e.g*. the survey by Raguram *et al*. [35]. Combining some of the most promising improvements, Raguram *et al*. created the Universal RANSAC (USAC) framework [34] which represents the state-of-the-art of classic RANSAC variants. USAC includes guided hypothesis sampling according to PROSAC [9], more accurate model fitting according to Locally Optimized RANSAC [11], and more efficient hypothesis verification according to Optimal Randomized RANSAC [10]. Many of the improvements proposed for RANSAC could also be applied to NG-RANSAC since we do not require any differentiability of such add-ons. We only impose restrictions on how to generate hypotheses, namely according to a learned probability distribution.

RANSAC is not often used in recent machine learning-heavy vision pipelines. Notable exceptions include geometric problems like object instance pose estimation [3, 5, 21], and camera re-localization [41, 51, 28, 8, 46] where RANSAC is coupled with decision forests or neural networks that predict image-to-object correspondences. However, in most of these works, RANSAC is not part of the training process because of its non-differentiability. DSAC [4, 6] overcomes this limitation by making the hypothesis

selection a probabilistic action which facilitates optimization of the expected task loss during training. However, DSAC is limited in *which* derivatives can be calculated. DSAC allows differentiation w.r.t. to observations. For example, we can use it to calculate the gradient of image coordinates for a sparse correspondence. However, DSAC does not model observation selection, and hence we cannot use it to optimize a matching probability. By showing how to learn neural guidance, we close this gap. The combination with DSAC enables the full flexibility of learning both, observations and their selection probability.

Besides DSAC, a *differentiable* robust estimator, there has recently been some work on *learning* robust estimators. We discussed the work of Yi *et al*. [56] in the introduction. Ranftl and Koltun [36] take a similar but iterative approach reminiscent of Iteratively Reweighted Least Squares (IRLS) for fundamental matrix estimation. In each iteration, a neural network predicts observation weights for a weighted model fit, taking into account the residuals of the last iteration. Both, [56] and [36], have shown considerable improvements w.r.t. to vanilla RANSAC but require differentiable minimal solvers, and task loss functions. NG-RANSAC outperforms both approaches, and is more flexible when it comes to defining the training objective. This flexibility also enables us to train NG-RANSAC in a self-supervised fashion, possible with neither [56] nor [36].

3. Method

Preliminaries. We address the problem of fitting model parameters $\mathbf{h}$ to a set of observations $\mathbf{y} \in \mathcal{Y}$ that are contaminated by noise and outliers. For example, $\mathbf{h}$ could be a fundamental matrix that describes the epipolar geometry of an image pair [16], and $\mathcal{Y}$ could be the set of SIFT correspondences [27] we extract for the image pair. To calculate model parameters from the observations, we utilize a solver f, for example the 8-point algorithm [15]. However, calculating $\mathbf{h}$ from all observations will result in a poor estimate due to outliers. Instead, we can calculate $\mathbf{h}$ from a small subset (minimal set) of observations with cardinality N: $\mathbf{h} = f(\mathbf{y}_1, \dots, \mathbf{y}_N)$. For example, for a fundamental matrix $N = 8$ when using the 8-point algorithm. RANSAC [12] is an algorithm to chose an outlier-free minimal set from $\mathcal{Y}$ such that the resulting estimate $\mathbf{h}$ is accurate. To this end, RANSAC randomly chooses M minimal sets to create a pool of model hypotheses $\mathcal{H} = (\mathbf{h}_1, \dots, \mathbf{h}_M)$.

RANSAC includes a strategy to adaptively choose M, based on an online estimate of the outlier ratio [12]. The strategy guarantees that an outlier-free set will be sampled with a user-defined probability. For tasks with large outlier ratios, M calculated like this can be exponentially large, and is usually clamped to a maximum value [7]. For notational simplicity, we take the perspective of a fixed M but do not restrict the use of an early-stopping strategy in practice.

RANSAC chooses a model hypothesis as the final estimate $\hat{\mathbf{h}}$ according to a scoring function s:

$$\hat{\mathbf{h}} = \underset{\mathbf{h} \in \mathcal{H}}{\operatorname{argmax}}\, s(\mathbf{h}, \mathcal{Y}). \tag{1}$$

The scoring function measures the consensus of an hypothesis w.r.t. all observations, and is traditionally implemented as inlier counting [12].

Neural Guidance. RANSAC chooses observations uniformly random to create the hypothesis pool $\mathcal{H}$. We aim at sampling observations according to a learned distribution instead that is parametrized by a neural network with parameters $\mathbf{w}$. That is, we select observations according to $\mathbf{y} \sim p(\mathbf{y}; \mathbf{w})$. Note that $p(\mathbf{y}; \mathbf{w})$ is a categorical distribution over the discrete set of observations $\mathcal{Y}$, *not* a continuous distribution in observation space. We wish to learn parameters $\mathbf{w}$ in a way that increases the chance of selecting outlier-free minimal sets, which will result in accurate estimates $\hat{\mathbf{h}}$. We sample a hypothesis pool $\mathcal{H}$ according to $p(\mathcal{H}; \mathbf{w})$ by sampling observations and minimal sets independently, *i.e.*

$$p(\mathcal{H}; \mathbf{w}) = \prod_{j=1}^{M} p(\mathbf{h}_j; \mathbf{w}), \text{ with } p(\mathbf{h}; \mathbf{w}) = \prod_{i=1}^{N} p(\mathbf{y}_i; \mathbf{w}). \tag{2}$$

From a pool $\mathcal{H}$, we estimate model parameters $\hat{\mathbf{h}}$ with RANSAC according to Eq. 1. For training, we assume that we can measure the quality of the estimate with a task loss function $\ell(\hat{\mathbf{h}})$. The task loss can be calculated w.r.t. a ground truth model $\mathbf{h}^*$, or self-supervised, *e.g.* by using the inlier count of the final estimate: $\ell(\hat{\mathbf{h}}) = -s(\hat{\mathbf{h}}, \mathcal{Y})$. We wish to learn the distribution $p(\mathcal{H}; \mathbf{w})$ in a way that we receive a small task loss with high probability. Inspired by DSAC [4], we define our training objective as the minimization of the expected task loss:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{H} \sim p(\mathcal{H}; \mathbf{w})} \left[\ell(\hat{\mathbf{h}}) \right]. \tag{3}$$

We compute the gradients of the expected task loss w.r.t. the network parameters as

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{H}} \left[\ell(\hat{\mathbf{h}}) \, \frac{\partial}{\partial \mathbf{w}} \log p(\mathcal{H}; \mathbf{w}) \right]. \tag{4}$$

Integrating over all possible hypothesis pools to calculate the expectation is infeasible. Therefore, we approximate the gradients by drawing K samples $\mathcal{H}_k \sim p(\mathcal{H}; \mathbf{w})$:

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) \approx \frac{1}{K} \sum_{k=1}^{K} \left[\ell(\hat{\mathbf{h}}) \frac{\partial}{\partial \mathbf{w}} \log p(\mathcal{H}_k; \mathbf{w}) \right]. \tag{5}$$

Note that gradients of the task loss function ℓ do *not* appear in the expression above. Therefore, differentiability of the

task loss ℓ, the robust solver $\hat{\mathbf{h}}$ (*i.e.* RANSAC) or the minimal solver f is not required. These components merely generate a training signal for steering the sampling probability $p(\mathcal{H};\mathbf{w})$ in a good direction. Due to the approximation by sampling, the gradient variance of Eq. 5 can be high. We apply a standard variance reduction technique from reinforcement learning by subtracting a baseline b [45]:

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) \approx \frac{1}{K} \sum_{k=1}^{K} \left[\left[\ell(\hat{\mathbf{h}}) - b \right] \frac{\partial}{\partial \mathbf{w}} \log p(\mathcal{H}_k;\mathbf{w}) \right]. \quad (6)$$

We found a simple baseline in the form of the average loss per image sufficient, *i.e.* $b = \bar{\ell}$. Subtracting the baseline will move the probability distribution towards hypothesis pools with lower-than-average loss for each training example.

Combination with DSAC. Brachmann *et al.* [4] proposed a RANSAC-based pipeline where a neural network with parameters $\mathbf{w}$ predicts observations $\mathbf{y}(\mathbf{w}) \in \mathcal{Y}(\mathbf{w})$. End-to-end training of the pipeline, and therefore learning the observations $\mathbf{y}(\mathbf{w})$, is possible by turning the argmax hypothesis selection of RANSAC (cf. Eq. 1) into a probabilistic action:

$$\hat{\mathbf{h}}_{\text{DSAC}} = \mathbf{h}_j \sim p(j|\mathcal{H}) = \frac{\exp s(\mathbf{h}_j, \mathcal{Y}(\mathbf{w}))}{\sum_{k=1}^{M} \exp s(\mathbf{h}_k, \mathcal{Y}(\mathbf{w}))}. \quad (7)$$

This differentiable variant of RANSAC (DSAC) chooses a hypothesis randomly according to a distribution calculated from hypothesis scores. The training objective aims at learning network parameters such that hypotheses with low task loss are chosen with high probability:

$$\mathcal{L}_{\text{DSAC}}(\mathbf{w}) = \mathbb{E}_{j \sim p(j)} \left[\ell(\mathbf{h}_j) \right]. \quad (8)$$

In the following, we extend the formulation of DSAC with neural guidance (NG-DSAC). We let the neural network predict observations $\mathbf{y}(\mathbf{w})$ and, additionally, a probability associated with each observation $p(\mathbf{y};\mathbf{w})$. Intuitively, the neural network can express a confidence in its own predictions through this probability. This can be useful if a certain input for the neural network contains no information about the desired model $\mathbf{h}$. In this case, the observation prediction $\mathbf{y}(\mathbf{w})$ is necessarily an outlier, and the best the neural network can do is to label it as such by assigning a low probability. We combine the training objectives of NG-RANSAC (Eq. 3) and DSAC (Eq. 8) which yields:

$$\mathcal{L}_{\text{NG-DSAC}}(\mathbf{w}) = \mathbb{E}_{\mathcal{H} \sim p(\mathcal{H};\mathbf{w})} \mathbb{E}_{j \sim p(j|\mathcal{H})} \left[\ell(\mathbf{h}_j) \right], \quad (9)$$

where we again construct $p(\mathcal{H};\mathbf{w})$ from individual $p(\mathbf{y};\mathbf{w})$'s according to Eq. 2. The training objective of NG-DSAC consists of two expectations. Firstly, the expectation w.r.t. sampling a hypothesis pool according to the probabilities predicted by the neural network. Secondly, the expectation w.r.t. sampling a final estimate from the pool according to the scoring function. As in NG-RANSAC, we approximate the first expectation via sampling, as integrating over all possible hypothesis pools is infeasible. For the second expectation, we can calculate it analytically, as in DSAC, since it integrates over the discrete set of hypotheses $\mathbf{h}_j$ in a given pool $\mathcal{H}$. Similar to Eq. 6, we give the approximate gradients $\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w})$ of NG-DSAC as:

$$\frac{1}{K} \sum_{k=1}^{K} \left[\left[\mathbb{E}_j \left[\ell \right] - b \right] \frac{\partial}{\partial \mathbf{w}} \log p(\mathcal{H}_k;\mathbf{w}) + \frac{\partial}{\partial \mathbf{w}} \mathbb{E}_j \left[\ell \right] \right], \quad (10)$$

where we use $\mathbb{E}_j \left[\ell \right]$ as a stand-in for $\mathbb{E}_{j \sim p(j|\mathcal{H}_k)} \left[\ell(\mathbf{h}_j) \right]$. The calculation of gradients for NG-DSAC requires the derivative of the task loss (note the last part of Eq. 10) because $\mathbb{E}_j \left[\ell \right]$ depends on parameters $\mathbf{w}$ via observations $\mathbf{y}(\mathbf{w})$. Therefore, training NG-DSAC requires a differentiable task loss function ℓ, a differentiable scoring function s, and a differentiable minimal solver f. Note that we inherit these restrictions from DSAC. In return, NG-DSAC allows for learning observations and observation confidences, at the same time.

4. Experiments

We evaluate neural guidance on multiple, classic computer vision tasks. Firstly, we apply NG-RANSAC to estimating epipolar geometry of image pairs in the form of essential matrices and fundamental matrices. Secondly, we apply NG-DSAC to horizon line estimation and camera relocalization. We present the main experimental results here, and refer to the supplement for details about network architectures, hyper-parameters and further experimental analysis. Our implementation is based on PyTorch [32], and we will make the code publicly available[1].

4.1. Essential Matrix Estimation

Epipolar geometry describes the geometry of two images that observe the same scene [16]. In particular, two image points $\mathbf{x}$ and $\mathbf{x}'$ in the left and right image corresponding to the same 3D point satisfy $\mathbf{x}'^{\top} F \mathbf{x} = 0$, where the 3×3 matrix F denotes the fundamental matrix. We can estimate F uniquely (but only up to scale) from 8 correspondences, or from 7 correspondences with multiple solutions [16]. The essential matrix E is a special case of the fundamental matrix when the calibration parameters K and K' of both cameras are known: $E = K'^{\top} F K$. The essential matrix can be estimated from 5 correspondences [31]. Decomposing the essential matrix allows to recover the relative pose between the observing cameras, and is a central step in image-based 3D reconstruction [40]. As such, estimating the fundamental or essential matrices of image pairs is a classic and well-researched problem in computer vision.

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

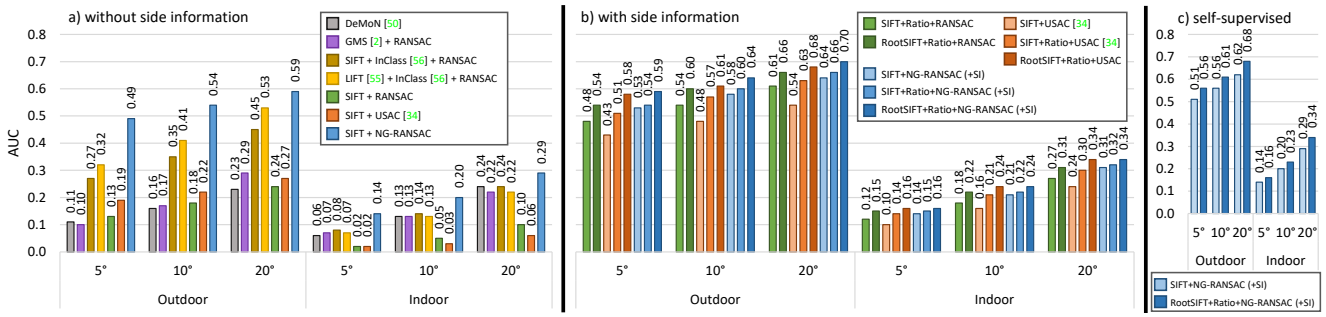

Figure 2. **Essential Matrix Estimation.** We calculate the relative pose between outdoor and indoor image pairs via the essential matrix. We measure the AUC of the cumulative angular error up to a threshold of 5°, 10° or 20°. **a)** We use no side information about the sparse correspondences. **b)** We use side information in the form of descriptor distance ratios between the best and second best match. We use it to filter correspondences with a threshold of 0.8 (*+Ratio*), as an additional input for our network (*+SI*), and as additional input for USAC [34]. **c)** We train NG-RANSAC in a self-supervised fashion by using the inlier count as training objective.

In the following, we firstly evaluate NG-RANSAC for the calibrated case and estimate essential matrices from SIFT correspondences [27]. For the sake of comparability with the recent, learned robust estimator of Yi *et al*. [56] we adhere closely to their evaluation setup, and compare to their results.

Datasets. Yi *et al*. [56] evaluate their approach in outdoor as well as indoor settings. For the outdoor datasets, they select five scenes from the structure-from-motion (SfM) dataset of [19]: *Buckingham*, *Notredame*, *Sacre Coeur*, *St. Peter's* and *Reichstag*. They pick two additional scenes from [44]: *Fountain* and *Herzjesu*. They reconstruct each scene using a SfM tool [53] to obtain 'ground truth' camera poses, and co-visibility constraints for selecting image pairs. For indoor scenes Yi *et al*. choose 16 sequences from the SUN3D dataset [54] which readily comes with ground truth poses captured by KinectFusion [30]. See the supplement for a listing of all scenes. Indoor scenarios are typically very challenging for sparse feature-based approaches because of texture-less surfaces and repetitive elements (see Fig. 1 for an example). Yi *et al*. train their best model using one outdoor scene (*St. Peter's*) and one indoor scene (*Brown 1*), and test on all remaining sequences (6 outdoor, 15 indoor). Yi *et al*. kindly provided us with their exact data splits, and we will use their setup. Note that training and test is performed on completely separate scenes, *i.e*. the neural network has to generalize to unknown environments.

Evaluation Metric. Via the essential matrix, we recover the relative camera pose up to scale, and compare to the ground truth pose as follows. We measure the angular error between the pose rotations, as well as the angular error between the pose translation vectors in degrees. We take the maximum of the two values as the final angular error. We calculate the cumulative error curve for each test sequence, and compute the area under the curve (AUC) up to a threshold of 5°, 10° or 20°. Finally, we report the average AUC over all test sequences (but separately for the indoor and outdoor setting).

Implementation. Yi *et al*. train a neural network to classify a set of sparse correspondences in inliers and outliers. They represent each correspondence as a 4D vector combining the 2D coordinate in the left and right image. Their network is inspired by PointNet [33], and processes each correspondence independently by a series of multilayer perceptrons (MLPs). Global context is infused by using instance and batch normalization [49, 20] in-between layers. We re-build this architecture in PyTorch, and train it according to NG-RANSAC (Eq. 3). That is, the network predicts weights to guide RANSAC sampling instead of inlier class labels. We use the angular error between the estimated relative pose, and the ground truth pose as task loss ℓ. As minimal solver f, we use the 5-point algorithm [31]. To speed up training, we initialize the network by learning to predict the distance of each correspondence to the ground truth epipolar line, see the supplement for details. We initialize for 75k iterations, and train according to Eq. 3 for 25k iterations. We optimize using Adam [23] with a learning rate of 10^{-5}. For each training image, we extract 2000 SIFT correspondences, and sample $K = 4$ hypothesis pools with $M = 16$ hypotheses. We use a low number of hypotheses during training to obtain variation when sampling pools. For testing, we increase the number of hypotheses to $M = 10^3$. We use an inlier threshold of 10^{-3} assuming normalized image coordinates using camera calibration parameters.

Results. We compare NG-RANSAC to the inlier classification (*InClass*) of Yi *et al*. [56]. They use their approach with SIFT as well as LIFT [55] features. We include results for DeMoN [50], a learned SfM pipeline, and GMS [2], a semi-dense approach using ORB features [38]. As classical baselines, we compare to vanilla RANSAC [12] and USAC [34]. See Fig. 2 a) for results. RANSAC achieves poor results across all thresholds, scoring as the weakest method. In this experiment, we assume no side information is available about the quality of correspondences. Therefore, USAC performs similar to RANSAC, since it cannot use guided sampling. Coupling RANSAC with neu-

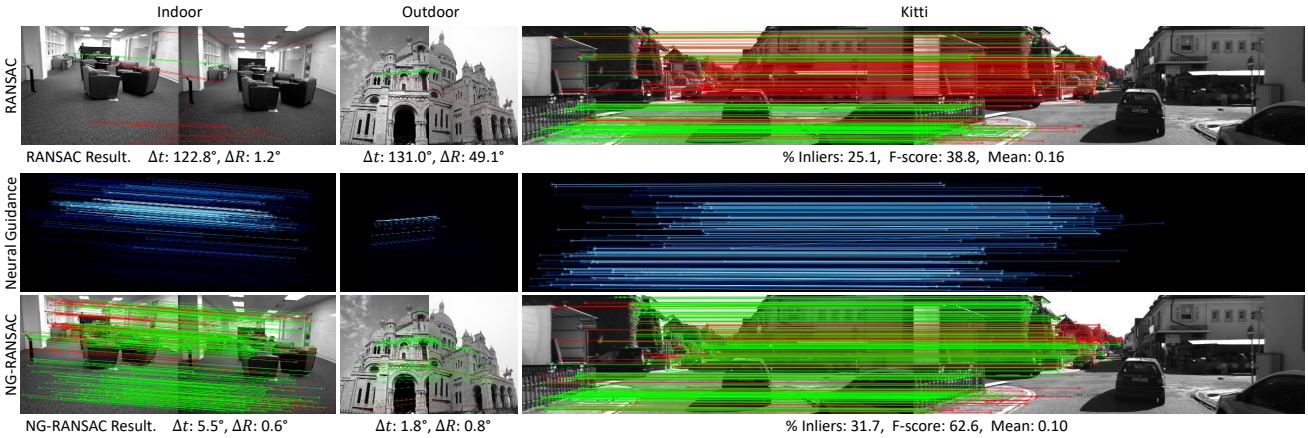

Figure 3. **Qualitative Results.** We compare fitted models for RANSAC and NG-RANSAC. For the indoor and outdoor image pairs, we fit essential matrices, and for the Kitti image pair we fit the fundamental matrix. We draw final model inliers in green if they adhere to the ground truth model, and red otherwise. We also measure the quality of each estimate, see the main text for details on the metrics.

ral guidance (NG-RANSAC) elevates it to the leading position. Different from USAC, NG-RANSAC deduces useful guiding weights solely from the spatial distribution of correspondences. See also Fig. 3 for qualitative results.

NG-RANSAC outperforms *InClass* of Yi *et al*. [56] despite some similarities. Both use the same network architecture, are based on SIFT correspondences, and both use RANSAC at test time. Yi *et al*. [56] train using a hybrid classification-regression loss based on the 8-point algorithm, and ultimately compare essential matrices using squared error. Therefore, their training objective is very different from the evaluation procedure. During evaluation, they use RANSAC with the 5-point algorithm on top of their inlier predictions, and measure the angular error. NG-RANSAC incorporates all these components in its training procedure, and therefore optimizes the correct objective.

Using Side Information. The evaluation procedure of Yi *et al*. [56] is designed to test a robust estimator in high-outlier domains. However, it underestimates what classical approaches can achieve on these datasets. The distance ratio of the best and second-best SIFT match is often an indicator of correspondence quality. This side information can be used by USAC [34] to guide hypothesis sampling according to the PROSAC strategy [9]. Furthermore, Lowe's ratio criterion [27] removes ambiguous matches with a distance ratio above a threshold (we use 0.8) before running RANSAC. We denote the ratio filter as *+Ratio* in Fig. 2 b), and observe a drastic improvement for all methods. Both classic approaches, RANSAC and USAC, outperform all learned methods of Fig. 2 a). RootSIFT normalization of SIFT descriptors [1] improves accuracy further. NG-RANSAC easily incorporates side information. For best accuracy, we train it on ratio-filtered RootSIFT correspondences, using distance ratios as additional network input (denoted as *+SI*). See the supplement for a detailed comparison of NG-RANSAC and USAC with varying hypothesis count M.

Self-supervised Learning. We train NG-RANSAC self-supervised by defining a task loss ℓ to assess the quality of an estimate independent of a ground truth model $\mathbf{h}^*$. A natural choice is the inlier count of the final estimate. We found the inlier count to be a very stable training signal, even in the beginning of training such that we require no special initialization of the network. We report results of self-supervised NG-RANSAC in Fig. 2 c). It outperforms all competitors except USAC [34] which it matches in accuracy. Unsupervised NG-RANSAC achieves slightly worse accuracy than supervised NG-RANSAC. A supervised task loss allows NG-RANSAC to adapt more precisely to the evaluation measure used at test time. For the datasets used so far, the process of image pairing uses co-visibility information, and therefore a form of supervision. In the next section, we learn NG-RANSAC fully self-supervised by using the ordering of sequential data to assemble image pairs.

4.2. Fundamental Matrix Estimation

We apply NG-RANSAC to fundamental matrix estimation, comparing it to the learned estimator of Ranftl and Koltun [36], denoted *Deep F-Mat*. They propose an iterative procedure where a neural network estimates observation weights for a robust model fit. Residuals of the last iteration are an additional input to the network in the next iteration. The network architecture is similar to the one of [56]. Correspondences are represented as 4D vectors, and they use distance ratios as additional inputs. A series of MLPs processes each correspondence with instance normalization interleaved. *Deep F-Mat* was published very recently, and the code is not yet available. We follow the evaluation procedure described in [36] and compare to their results.

Datasets. Ranftl and Koltun [36] evaluate their method on various datasets that involve custom reconstructions not publicly available. Therefore, we compare to their method on the Kitti dataset [14], which is online. Ranftl and Koltun [36] train their method on sequences 00-05 of the Kitti

	Training Objective	% Inliers	F-score	Mean	Median
RANSAC	-	21.85	13.84	0.35	0.32
USAC [34]	-	21.43	13.90	0.35	0.32
Deep F-Mat [36]	Mean	24.61	14.65	**0.32**	**0.29**
NG-RANSAC	Mean	25.05	**14.76**	**0.32**	**0.29**
NG-RANSAC	F-score	24.13	14.72	0.33	0.31
NG-RANSAC	%Inliers	**25.12**	14.74	**0.32**	**0.29**

Figure 4. **Fundamental Matrix Estimation.** We measure the average percentage of inliers of the estimated model, the alignment of estimated and ground truth inliers (*F-score*), and the mean and median distance of estimated inliers to ground truth epilines. For NG-RANSAC, we compare the performance after training with different objectives. *%Inliers* is a self-supervised objective.

odometry benchmark, and test on sequences 06-10. They form image pairs by taking subsequent images within a sequence. For each pair, they extract SIFT correspondences and apply Lowe's ratio filter [27] with a threshold of 0.8.

Evaluation Metric. Ranftl and Koltun [36] evaluate using multiple metrics. They measure the percentage of inlier correspondences of the final model. They calculate the F-score over correspondences where true positives are inliers of both the ground truth model and the estimated model. The F-score measures the alignment of estimated and true fundamental matrix in image space. Both metrics use an inlier threshold of 0.1px. Finally, they calculate the mean and median epipolar error of inlier correspondences w.r.t. the ground truth model, using an inlier threshold of 1px.

Implementation. We cannot use the architecture of *Deep F-Mat* which is designed for iterative application. Therefore, we re-use the architecture of Yi *et al*. [56] from the previous section for NG-RANSAC (also see the supplement for details). We adhere to the training setup described in Sec. 4.1 with the following changes. We observed faster training convergence on Kitti, so we omit the initialization stage, and directly optimize the expected task loss (Eq. 3) for 300k iterations. Since Ranftl and Koltun [36] evaluate using multiple metrics, the choice of the task loss function ℓ is not clear. Hence, we train multiple variants with different objectives (*%Inliers*, *F-score* and *Mean* error) and report the corresponding results. As minimal solver f, we use the 7-point algorithm, a RANSAC threshold of 0.1px, and we draw $K = 8$ hypothesis pools per training image with $M = 16$ hypotheses each.

Results. We report results in Fig. 4. NG-RANSAC outperforms the classical approaches RANSAC and USAC. NG-RANSAC also performs slightly superior to *Deep F-Mat*. We observe that the choice of the training objective has small but significant influence on the evaluation. All metrics are highly correlated, and optimizing a metric in training generally also achieves good (but not necessarily best) accuracy using this metric at test time. Interestingly, optimizing the inlier count during training performs competitively, although being a self-supervised objective. Fig. 3 shows a qualitative result on Kitti.

	AUC (%)
Simon et al. [42]	54.4
Kluger et al. [24]	57.3
Zhai et al. [57]	58.2
Workman et al. [52]	71.2
DSAC	74.1
NG-DSAC	75.2
SLNet [25]	82.3

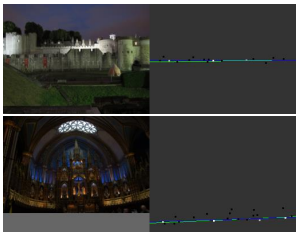

Figure 5. **Horizon Line Estimation. Left:** AUC on the HLW dataset. **Right:** Qualitative results. We draw the ground truth horizon in green and the estimate in blue. Dots mark the observations predicted by NG-DSAC, and the dot colors mark their confidence (dark = low). Note that the horizon can be outside the image.

4.3. Horizon Lines

We fit a parametric model, the horizon line, to a single image. The horizon can serve as a cue in image understanding [52] or for image editing [25]. Traditionally, this task is solved via vanishing point detection and geometric reasoning [37, 24, 57, 42], often assuming a Manhattan or Atlanta world. We take a simpler approach and use a general purpose CNN that predicts a set of 64 2D points based on the image to which we fit a line with RANSAC, see Fig. 5. The network has two output branches predicting (i) the 2D points $\mathbf{y}(\mathbf{w}) \in \mathcal{Y}(\mathbf{w})$, and (ii) probabilities $p(\mathbf{y}; \mathbf{w})$ for guided sampling (see the supplement for details).

Dataset. We evaluate on the HLW dataset [52] which is a collection of SfM datasets with annotated horizon line.

Evaluation Metric. As is common practice on HLW, we measure the maximum distance between the estimated horizon and ground truth within the image, normalized by image height. We calculate the AUC of the cumulative error curve up to a threshold of 0.25.

Implementation. We train using the NG-DSAC objective (Eq. 9) from scratch for 250k iterations. As task loss ℓ, we use the normalized maximum distance between estimated and true horizon. For hypothesis scoring s, we use a soft inlier count [6]. We train using Adam [23] with a learning rate of 10^{-4}. For each training image, we draw $K = 2$ hypothesis pools with $M = 16$ hypotheses. We also draw 16 hypotheses at test time. We compare to DSAC which we train similarly but disable the probability branch.

Results. We report results in Fig. 5. DSAC and NG-DSAC achieve competitive accuracy on this dataset, ranking among the top methods. NG-DSAC has a small but significant advantage over DSAC alone. Our method is only surpassed by SLNet [25], an architecture designed to find semantic lines in images. SLNet generates a large number of random candidate lines, selects a candidate via classification, and refines it with a predicted offset. We could couple SLNet with neural guidance for informed candidate sampling. Unfortunately, the code of SLNet is not online and the authors did not respond to inquiries.

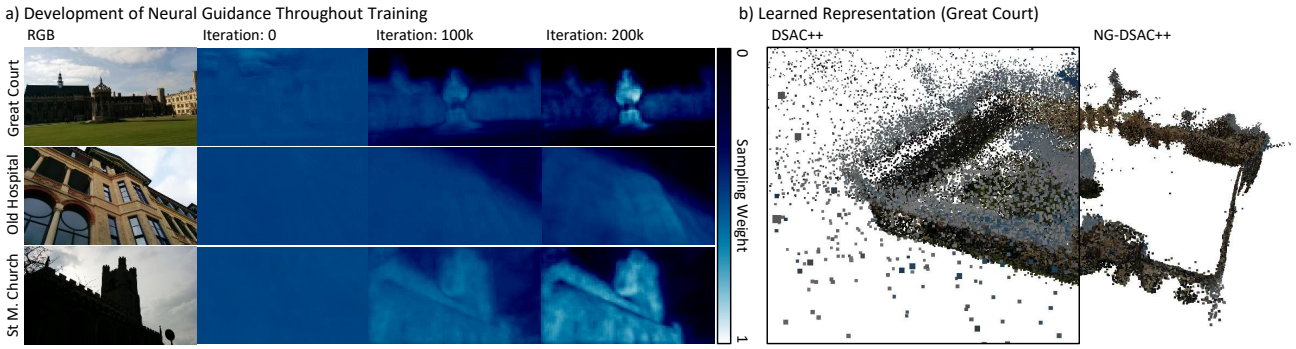

Figure 6. **Neural Guidance for Camera Re-localization. a)** Predicted sampling probabilities of NG-DSAC++ throughout training. **b)** Internal representation of the neural network. We predict scene coordinates for each training image, plotting them with their RGB color. For DSAC++ we choose training pixels randomly, for NG-DSAC++ we choose randomly according to the predicted distribution.

	DSAC++ [6] (VGGNet)	**DSAC++** (ResNet)	**NG-DSAC++** (ResNet)
Great Court	40.3cm	40.3cm	**35.0cm**
Kings College	17.7cm	13.0cm	**12.6cm**
Old Hospital	**19.6cm**	22.4cm	21.9cm
Shop Facade	5.7cm	5.7cm	**5.6cm**
St M. Church	12.5cm	9.9cm	**9.8cm**

Figure 7. **Camera Re-Localization.** We report median position error for Cambridge Landmarks [22]. *DSAC++ (ResNet)* is our re-implementation of [6] with an improved network architecture.

4.4. Camera Re-Localization

We estimate the absolute 6D camera pose (position and orientation) w.r.t. a known scene from a single RGB image.

Dataset. We evaluate on the Cambridge Landmarks [22] dataset. It is comprised of RGB images depicting five landmark buildings in Cambridge, UK.Ground truth poses were generated by running a SfM pipeline.

Evaluation Metric. We measure the median translational error of estimated poses for each scene.

Implementation. We build on the publicly available DSAC++ pipeline [6] which is a scene coordinate regression method [41]. A neural network predicts for each image pixel a 3D coordinate in scene space. We recover the pose from the 2D-3D correspondences using a perspective-n-point solver [13] within a RANSAC loop. The DSAC++ pipeline implements geometric pose optimization in a fully differentiable way which facilitates end-to-end training. We re-implement the neural network integration of DSAC++ with PyTorch (the original uses LUA/Torch). We also update the network architecture of DSAC++ by using a ResNet [18] instead of a VGGNet [43]. As with horizon line estimation, we add a second output branch to the network for estimating a probability distribution over scene coordinate predictions for guided RANSAC sampling. We denote this extended architecture *NG-DSAC++*. We adhere to the training procedure and hyperparamters of DSAC++ (see the supplement) but optimize the NG-DSAC objective (Eq. 9) during end-to-end training. As task loss ℓ, we use the average of the rotational and translational error w.r.t. the ground truth pose. We sample $K = 2$ hypothesis pools with $M = 16$ hypotheses per training image, and increase the number of hypotheses to $M = 256$ for testing.

Results. We report our quantitative results in Fig. 7. Firstly, we observe a significant improvement for most scenes when using DSAC++ with a ResNet architecture. Secondly, comparing DSAC++ with NG-DSAC++, we notice a small to moderate, but consistent, improvement in accuracy. The advantage of using neural guidance is largest for the *Great Court* scene, which features large ambiguous grass areas, and large areas of sky visible in many images. NG-DSAC++ learns to ignore such areas, see the visualization in Fig. 6 a). The network learns to mask these areas solely guided by the task loss during training, as the network fails to predict accurate scene coordinates for them. In Fig. 6 b), we visualize the internal representation learned by DSAC++ and NG-DSAC++ for one scene. The representation of DSAC++ is very noisy, as it tries to optimize geometric constrains for sky and grass pixels. NG-DSAC++ learns a cleaner representation by focusing entirely on buildings.

5. Conclusion

We have presented NG-RANSAC, a robust estimator using guided hypothesis sampling according to learned probabilities. For training we can incorporate non-differentiable task loss functions and non-differentiable minimal solvers. Using the inlier count as training objective allows us to also train NG-RANSAC self-supervised. We applied NG-RANSAC to multiple classic computer vision tasks and observe a consistent improvement w.r.t. RANSAC alone.

Acknowledgements: This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 647769). The computations were performed on an HPC Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

References

[1] Relja Arandjelovic. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. 6

[2] JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan Dat Nguyen, and Ming-Ming Cheng. GMS: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *CVPR*, 2017. 5

[3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *ECCV*, 2014. 2

[4] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-Differentiable RANSAC for camera localization. In *CVPR*, 2017. 2, 3, 4

[5] Eric Brachmann, Frank Michel, Alexander Krull, Michael Y. Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *CVPR*, 2016. 2

[6] Eric Brachmann and Carsten Rother. Learning less is more-6D camera localization via 3D surface regression. In *CVPR*, 2018. 1, 2, 7, 8

[7] Gary Bradski. OpenCV. *Dr. Dobb's Journal of Software Tools*, 2000. 1, 3

[8] Tommaso Cavallari, Stuart Golodetz, Nicholas A. Lord, Julien Valentin, Luigi Di Stefano, and Philip H. S. Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *CVPR*, 2017. 2

[9] Ondřej Chum and Jiří Matas. Matching with PROSAC - Progressive sample consensus. In *CVPR*, 2005. 2, 6

[10] Ondřej Chum and Jiří Matas. Optimal randomized RANSAC. *TPAMI*, 2008. 2

[11] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized RANSAC. In *DAGM*, 2003. 2

[12] 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. 1, 2, 3, 5

[13] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *TPAMI*, 2003. 8

[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012. 6

[15] Richard I. Hartley. In defense of the eight-point algorithm. *TPAMI*, 1997. 3

[16] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. 3, 4

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

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

[19] Jared Heinly, Johannes Lutz Schönberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In *CVPR*, 2015. 5

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[21] Omid H. Jafari, Siva K. Mustikovela, Karl Pertsch, Eric Brachmann, and Carsten Rother. iPose: Instance-aware 6D pose estimation of partly occluded objects. 2018. 2

[22] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *ICCV*, 2015. 8

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5, 7

[24] Florian Kluger, Hanno Ackermann, Michael Y. Yang, and Bodo Rosenhahn. Deep learning for vanishing point detection using an inverse gnomonic projection. In *GCPR*, 2017. 7

[25] Jun-Tae Lee, Han-Ul Kim, Chul Lee, and Chang-Su Kim. Semantic line detection and its applications. In *ICCV*, 2017. 7

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

[27] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 3, 5, 6, 7

[28] Daniela Massiceti, Alexander Krull, Eric Brachmann, Carsten Rother, and Philip H. S. Torr. Random forests versus neural networks - what's best for camera localization? In *ICRA*, 2017. 2

[29] Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *T-RO*, 2017. 1

[30] Richard Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proc. ISMAR*, 2011. 5

[31] David Nistér. An efficient solution to the five-point relative pose problem. *TPAMI*, 2004. 4, 5

[32] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017. 4

[33] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 5

[34] Rahul Raguram, Ondřej Chum, Marc Pollefeys, Jiří Matas, and Jan-Michael Frahm. USAC: A universal framework for random sample consensus. *TPAMI*, 2013. 2, 5, 6

[35] Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In *ECCV*, 2008. 2

[36] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 2, 3, 6, 7

[37] Carsten Rother. A new approach for vanishing point detection in architectural environments. In *BMVC*, 2002. 7

[38] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011. 5

[39] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *TPAMI*, 2016. 1

[40] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016. 1, 4

[41] 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. 2, 8

[42] Gilles Simon, Antoine Fond, and Marie-Odile Berger. A-contrario horizon-first vanishing point detection using second-order grouping laws. In *ECCV*, 2018. 7

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014. 8

[44] Christoph Strecha, Wolfgang von Hansen, Luc J. Van Gool, Pascal Fua, and Ulrich Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, 2008. 5

[45] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998. 4

[46] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 2

[47] Ben Tordoff and David W. Murray. Guided sampling and consensus for motion estimation. In *ECCV*, 2002. 2

[48] Philip. H. S. Torr and Andrew Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *CVIU*, 2000. 2

[49] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, 2016. 5

[50] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and motion network for learning monocular stereo. In *CVPR*, 2017. 5

[51] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip H. S. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015. 2

[52] Scott Workman, Menghua Zhai, and Nathan Jacobs. Horizon lines in the wild. In *BMVC*, 2016. 7

[53] Changchang Wu. Towards linear-time incremental structure from motion. In *3DV*, 2013. 5

[54] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*, 2013. 5

[55] Kwang M. Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned invariant feature transform. In *ECCV*, 2016. 5

[56] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 2, 3, 5, 6, 7

[57] Menghua Zhai, Scott Workman, and Nathan Jacobs. Detecting vanishing points using global image context in a non-manhattan world. In *CVPR*, 2016. 7