Generalized Differentiable RANSAC

Tong Wei[1], Yash Patel[1], Alexander Shekhovtsov[1], Jiří Matas[1], and Daniel Barath[2]
[1] Visual Recognition Group, FEE, Czech Technical University in Prague
[2] Computer Vision and Geometry Group, ETH Zurich
{weitong, patelyas, shekhole, matas}@fel.cvut.cz, danielbela.barath@inf.ethz.ch

Abstract

We propose ∇-RANSAC, a generalized differentiable RANSAC that allows learning the entire randomized robust estimation pipeline. The proposed approach enables the use of relaxation techniques for estimating the gradients in the sampling distribution, which are then propagated through a differentiable solver. The trainable quality function marginalizes over the scores from all the models estimated within ∇-RANSAC to guide the network learning accurate and useful inlier probabilities or to train feature detection and matching networks. Our method directly maximizes the probability of drawing a good hypothesis, allowing us to learn better sampling distributions. We test ∇-RANSAC on various real-world scenarios on fundamental and essential matrix estimation, and 3D point cloud registration, outdoors and indoors, with handcrafted and learning-based features. It is superior to the state-of-the-art in terms of accuracy while running at a similar speed to its less accurate alternatives. The code and trained models are available at https://github.com/weitong8591/differentiable_ransac.

1. Introduction

Robust estimation is a fundamental component in vision pipelines, including relative pose estimation [18], wide baseline matching [54, 47, 48], multi-model fitting [34, 53], image-based localization [9], motion segmentation [70], and pose graph initialization of Structure-from-Motion (SfM) algorithms [61, 63]. While several robust estimators have been proposed throughout the years [30, 32, 41, 75], randomized hypothesize-and-verify approaches, like RANSAC [24] and its recent variants [4, 6, 5, 35], have become the most widely used methods due to their robustness, simplicity, and efficiency. RANSAC repeatedly selects random minimal subsets of the input data sufficient to fit a model hypothesis, *e.g.*, a 3D plane to three points or a fundamental matrix to seven point correspondences. The model score is then computed as the cardinality of the inlier set, formed by the points consistent with the model hypothesis, *i.e.*, having residuals smaller than a threshold. The so-far-the-best model is updated if a new model is found with higher quality. RANSAC terminates when the probability of finding a better hypothesis falls below a threshold. Finally, the model with the highest quality, polished, *e.g.*, by least-squares fitting on all inliers, is returned.

Numerous improvements have been made to the original RANSAC algorithm, including refinement of hypotheses through local optimization [17, 4], better scoring [71, 6, 5], detection of degenerate cases [19], and speed-ups through techniques such as weighted random sampling of hypotheses [15] or preemptive hypothesis verification strategies [14, 16]. See [72] for a recent survey and benchmark. To [72], the most accurate method for relative pose estimation is MAGSAC++ [5] with PROSAC sampling [15] and DEGENSAC-based degeneracy [19] testing.

In recent years, neural networks (NNs) have been employed to estimate tentative matches, including their coordinates and confidences [76, 67, 78, 79, 21, 66]. These predicted confidences could be used for pre-filtering matches or weighted random sampling in RANSAC. However, learning these NNs endowed with RANSAC, particularly for optimizing the desired evaluation metric, such as pose error, remains a challenging problem. One of the main challenges is that minimal solvers are often complex and not readily differentiable. Additionally, learning the sampling distribution for optimal RANSAC performance is challenging, both in terms of formalizing the problem and estimating gradients for the sampling probabilities. Prior work in this direction [9, 10] will be discussed in detail (Section 2).

In this paper, we make contributions to the learnable robust estimation family and propose a new differentiable RANSAC, ∇-RANSAC, with all the components differentiable. The main contributions are as follows:

- We investigate all the components of the RANSAC pipeline and propose a new differentiable alternative that allows learning inlier probabilities while directly optimizing test-time evaluation metrics, *e.g.*, pose error, as a *new* learning objective.

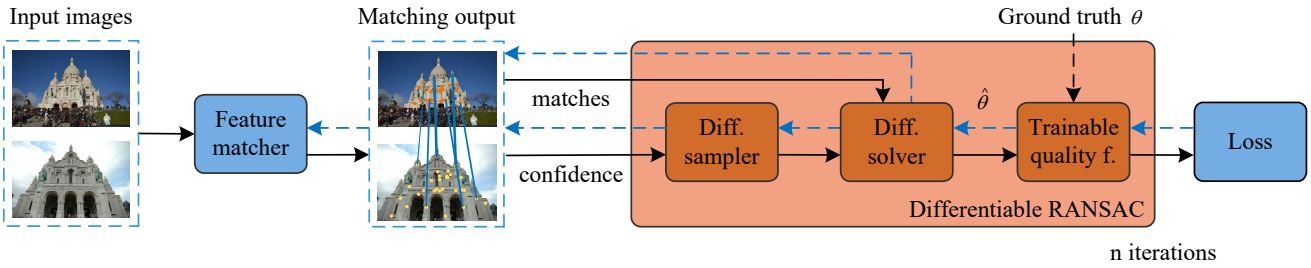

Figure 1. **Pipeline of training ∇-RANSAC (forward and backward).** Given an image pair, we can either use a hand-crafted feature matcher and feed the tentative correspondences into the consensus learning network from [79], or use learning-based matching method that outputs matches and their confidence. The predictions input to the ∇-RANSAC module for robust estimation. In each iteration, the differentiable and randomized Gumbel sampler (Section 3.1) selects a minimal sample of m correspondences. Model $\hat{\theta}$ is estimated by differentiable solvers (Section 3.2) and its loss is calculated based on trainable quality functions (Section 3.3) with the ground truth.

- We propose a new random sampling approach based on a re-parametrization strategy, *i.e*. Gumbel Softmax sampler, that allows gradient propagation through the entire randomized procedure.
- To demonstrate its potential to unlock the end-to-end training of geometric pipelines, ∇-RANSAC is incorporated into an end-to-end feature matcher, LoFTR [66], to improve the predicted matches and confidence.
- Technically, we implement and include a differentiable version of the widely used minimal solvers, *e.g*., five and seven-point algorithms [50], and standard Kabsch algorithm [38] for rigid transformation during training.

∇-RANSAC has significant implications for learning-based vision systems, enabling training such pipelines that were previously difficult or impossible to train.

2. Related Work

Robust Estimation with NNs. Context normalization networks (PointCN) [76] is one of the first papers on the topic, using a PointNet-based [55] structure with batch normalization [33] as a context mechanism to predict inlier probabilities. Attentive context normalization networks [67] improve upon [76] by using a special architectural block. Deep Fundamental matrix estimation [57] iteratively estimates the model using weighted least squares (LS) with weights suppressing the effect of outliers, re-estimated in each iteration. [78, 21] employ NNs to filter outliers, while CLNet [79] estimates the weights to be used in progressive filtering and weighted LS. Although many of these methods use weighted LS for model estimation due to its easy gradient propagation, it has been shown that applying RANSAC on correspondences filtered by CLNet or similar techniques improves results [3]. In other words, RANSAC is still necessary to robustly verify ambiguous hypotheses.

Sampling Distribution. In the worst case, the probability of drawing a good hypothesis at random in RANSAC decreases exponentially with the minimal sample size. Consequently, RANSAC might take excessive time or return an inaccurate solution if stopped early. It was observed that using a weighted random sampling [15, 10, 5], which is more likely to draw inlier points, often significantly improves the performance. This sampling guidance can either come from the feature matching procedure, *e.g*., as the SNN ratio [44] or can be learned from ground truth inliers [10]. PROSAC [15] exploits estimated inlier probabilities to sample the most promising hypotheses first. P-NAPSAC [5] progressively increases the radius of the selection hypersphere according to every unsuccessful iteration, blending into global sampling. Such samplers rely on the given sampling distribution, which might be improved using NNs.

Differentiable RANSAC. Currently, only one method learns the sampling distribution specifically for RANSAC, the Neural-Guided RANSAC [10]. NG-RANSAC maximizes the expected quality of the model found by RANSAC end-to-end by applying REINFORCE [74] gradient estimator. This unbiased estimator requires neither the solver nor the loss function to be differentiable. However, to improve the geometric features of individual tentative correspondences, backpropagation through the solver becomes necessary. It can be achieved by numerically differentiating the solver, using the finite difference method proposed in DSAC [9] for the 4-point P*n*P solver. NG-DSAC [10] combines these two techniques to jointly learn the sampling distribution and refine the coordinates.

Our work differs in several ways. First, we implement differentiable solvers for common minimal problems, enabling us to learn geometric features. In addition, this enables the use of relaxation techniques such as Gumbel-Softmax (GS) [36] to estimate the gradient in the sampling distribution instead of using REINFORCE [74]. Towards this end, we propose a simple GS-like relaxation for drawing a minimal sample of k points without replacement, which performs well in practice in our experiments. However, we remark that other (more complex) estimators, such as NeuralSort [27], can also be applied, enabled by

the differentiable solvers. Another key difference is a new objective function that maximizes the quality of an average sampled model instead of using the best one in the pool (NG-RANSAC). Our objective more directly maximizes the probability of drawing a good hypothesis, allowing ∇-RANSAC to better learn the sampling distribution. Finally, unlike previous work [10], we apply our method to jointly learn the sampling distribution and feature matches for epipolar geometry estimation.

3. Generalized Differentiable RANSAC

In this section, we discuss the algorithmic components of the proposed framework, visualized in Fig. 1. The key component is the differentiable ∇-RANSAC block.

We assume that the input to ∇-RANSAC is a set of tentative correspondences, possibly equipped with extra information from the detector and matcher, *e.g.*, feature orientation, scale, affine shape, jointly referred to as *geometric features* $\Phi \in \mathbb{R}^{N \times D}$ and their confidences represented by scores $s \in \mathbb{R}^N$, where N is the number of matches and D is the number of geometric features per correspondence. From the input image pairs, we adopt leaning-based feature matching methods, such as LoFTR, consensus learning CLNet architecture [79] to generate tentative point correspondences and confidence scores s. These scores are originally used for iterative pruning of matches [79], and we repurpose them for weighted sampling in RANSAC.

At test time, ∇-RANSAC draws minimal samples of k correspondences using weighted random sampling with probabilities $p = \mathrm{softmax}(s)$. The correspondences are drawn from the categorical distribution $\mathrm{Cat}(p)$ one-by-one without replacement until a minimal sample $(i_1, \dots i_k)$ of k correspondences is formed $h = (\Phi_{i_1}, \dots, \Phi_{i_k})$. Namely, the probability to draw a minimal sample $(i_1, \dots i_k)$ follows the Plackett-Luce (PL) model [45]:

$$p(i_1)\frac{p(i_2)}{1-p(i_1)} \cdots \frac{p(i_k)}{1-\sum_{l<k} p(i_l)}. \quad (1)$$

Then a *solver* returns solutions for geometric model $\hat{\theta}$, fitting precisely the minimal sample. The best model is selected, *e.g.*, based on MAGSAC score [5], and its loss, *e.g.*, the relative pose error, is evaluated w.r.t. the ground truth.

At training time, we are interested in learning the NN that produces geometric features and importance scores. We argue that learning importance scores for the best model in the entire RANSAC is impractical. If RANSAC runs long enough, it likely finds a good model independently of the confidence scores. The gradient in the scores is vanishing in the expectation and has high variance. Note that Brachmann *et al.* [10], while considering training the complete RANSAC algorithm, actually limit the number of iterations of RANSAC at training time (the pool of hypothesis) to just 20, which is much less than what is used at test time, creating a discrepancy between theory and practice. We propose instead that the importance scores should be learned to minimize the expected loss of a randomly drawn sample:

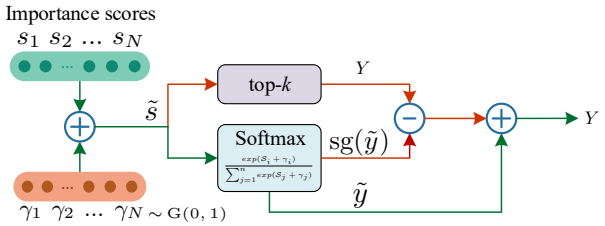

Figure 2. Overview of **Gumbel Softmax Sampler** used in ∇-RANSAC. The input to the sampler is the importance scores $s_1, s_2, ..., s_N$. The process starts by i.i.d random sampling $\gamma_1, \gamma_2, ..., \gamma_N$ from the standard $\mathrm{Gumbel}(0, 1)$ distribution. Then a minimal sample is drawn by selecting indices of top k noisy scores. Since top-k is non-differentiable, the straight through trick is used and the Softmax output is added and subtracted with stop-gradient (sg) to allow backward flow of the gradients. The green arrows show the connections through which gradient flow is possible. Red arrows show connections without any gradients.

$$\mathbb{E}_{\mathrm{data}}\Big[\mathbb{E}_h\left[\mathrm{loss}(\mathrm{solver}(h))\right]\Big], \quad (2)$$

where the first expectation is over the training examples from the dataset and the second one is over a randomly chosen hypothesis. This loss is better aligned with the goal of RANSAC, maximizing the probability of drawing good hypotheses and, thus, triggering the termination criterion in test time earlier. Additionally, this helps achieve a stable learning signal. In the remainder of this section, we will focus on estimating the gradient of (2) in the scores s.

3.1. Gumbel Softmax Sampler

The inner expectation in (2) can be exactly computed in $\mathcal{O}(\binom{N}{k})$ time which is prohibitive in practice. Thus, at training time, we sample a mini-batch of hypotheses per image pair and perform one SGD step for this min-batch. However, by taking a discrete sample, the dependence on the parameters of the sampling distribution is lost. The expectation over hypotheses requires more careful consideration.

Provided that the solver and loss are differentiable, we approximate the derivative of the expectation in scores s in a fashion similar to Gumbel-Softmax relaxation for categorical variables [36, 46]. Sampling from the PL distribution can be equivalently achieved by sorting noisy scores as:

$$(i_1 \dots i_k) = \text{top-}k(s + \gamma), \quad \gamma_i \sim \mathrm{Gumbel}(0, 1), \quad (3)$$

where top-k returns the indices of the largest k elements and $\mathrm{Gumbel}(0, 1)$ is the standard Gumbel distribution. We follow the straight-through GS strategy [36] to draw discrete samples on the forward pass but to propagate back

using the Jacobian of the softmax operator. Let y_j be a one-hot encoded index i_j, the index of the jth element in the sorting order. We define its $N \times N$ Jacobian in scores s as

$$\frac{dy_j}{ds} := \frac{d\,\mathrm{softmax}(\tilde{s}/\tau)}{ds}, \tag{4}$$

where τ is the "temperature of the relaxation" hyper-parameter. Using the vector of one-hot top-k indices $Y = (y_1, \dots y_k)$, we can easily select respective geometric features by matrix-matrix product $h = Y\Phi$. Assuming both the solver and the loss function to be differentiable, (4) is sufficient to complete the chain rule. Note that backpropagation using (4) requires only $\mathcal{O}(N)$ time and not $\mathcal{O}(N^2)$. A convenient way to implement our GS sampler is as:

$$\tilde{s} = s + \gamma, \quad \gamma_i \sim \mathrm{Gumbel}(0, 1), \tag{5a}$$
$$Y = \mathrm{one\_hot}(\text{top-}k(\tilde{s})), \tag{5b}$$
$$\tilde{y} = \mathrm{softmax}(\tilde{s}/\tau), \tag{5c}$$
$$Y = Y + \tilde{y} - \mathrm{sg}(\tilde{y}), \tag{5d}$$
$$h = Y\Phi, \tag{5e}$$

where sg does not propagate gradient (*i.e.*, detach in PyTorch). Step (5e) is a common trick: on the forward pass, the value equals precisely to Y, the one-hot indices of a correct sample. On the backward pass, the gradient flows through $\tilde{y}$ only. This sampler is visualized in Fig. 2.

Let us remark that other differentiable samplers for PL distribution can also be applied, particularly NeuralSort [27]. The proposed sampler is faster and performs well in our experiments even with the default $\tau = 1$. Furthermore, other methods can also be applied, such as unbiased REINFORCE [74] with a relaxation-based baseline [25]. We leave such refinements to future work. Note that all these options require the solver to be differentiable.

3.2. Differentiable Solver

Geometric solvers are a fundamental part of RANSAC-like approaches. Most solvers commonly used in computer vision can be made differentiable by implementing them in an automatic differentiation framework such as Pytorch and carefully considering each algorithmic component.

Learning-based pruning [79] propagates gradients through the well-known normalized eight-point (8PC) algorithm [29] for estimating essential (**E**) or fundamental (**F**) matrices. There are two practical issues with the 8PC algorithm. First, and most importantly, the 8PC and 7PC solvers have a degeneracy when the points stem from a close-to-planar 3D structure. In this case, a degenerate model is estimated that, while often having a large number of inliers [19], is incorrect w.r.t. the scene geometry. This misguides the learning in scenes dominated by planar structures and deteriorates the performance. Second, using eight correspondences instead of the minimal (5 for **E**, and 7 for **F** matrix) in RANSAC substantially decreases the chance to draw an all-inlier minimal sample and thus leads to a larger expected time to find a good solution or worse average quality of a solution found in a fixed budget.

For **E** matrix estimation, numerous 5PC solvers have been developed [50, 42, 8, 65, 11]. However, practical applications (*e.g.*, SfM [62, 68]) use either the method of Stewenius *et al.* [65] due to its stability, or that of Nister [50] due to its effectiveness. We use the method of Nister since it leads to more stable gradients in our experiments. Its steps are as follows: it creates the coefficient matrix from the input correspondences, decomposes it by SVD, solves a linear system of equations, solves a set of polynomial equations, and finally, basic arithmetic operations. We implemented a differentiable polynomial solver based on Sturm sequences [26] and also one using companion matrices. While both algorithms work well, the companion matrix-based solution we applied is the fastest.

Fundamental matrix estimation is an easier problem, where the 7PC [28] solver, besides an SVD decomposition on the coefficient matrix, only solves a cubic polynomial. As we are given a closed-form solution for the cubic problem, we can straightforwardly make the entire algorithm differentiable. Note that, in our experiments, we have not observed improvements by replacing 8PC with 7PC. This might be due to their shared degeneracy of planar scenes.

Minimal solvers often produce multiple solutions that all explain the data. While they are consistent with the constraints, most are inconsistent with the scene geometry. At inference, the best one is selected based on its score. To mimic the test-time evaluation in training, the best algebraic solution is selected for each sample based on the model score and used for the loss computation. Respectively, the gradient propagates back through the best solution.

Note that we are not aware of public implementations of such solvers, neither in open-source libraries [73, 58] nor in standalone public repositories. Therefore, we consider this a technical contribution to the community.

3.3. Trainable Quality Function

In RANSAC, the quality of an estimated model is calculated as the cardinality of its support, *i.e.*, the inlier number. Since RANSAC, a number of algorithms [71, 6, 5, 2] have improved the performance by better modeling the noise both in the inliers and outliers. However, all such methods perform a best model selection step based on the maximal quality, which renders the procedure non-differentiable. Some works [9] tackle this problem by employing soft probabilistic hypothesis selection. Other methods [76, 79] combine classification loss with regression and geometry-induced losses [10] to reason about the quality of a model.

Instead of the above solutions, we exploit *all* models $\{\hat{\theta}_i\}_{i=1}^{t}$ estimated during the fixed $t \in \mathbb{N}^{>0}$ iterations.

In each iteration, the estimated model is compared to the ground truth, and its implied loss is calculated, *e.g.*, as the relative pose error in the case of $\mathbf{E}$ matrix estimation. Specifically, we consider the following loss measures.

In the case of relative pose estimation, the *pose error loss* is defined as follows. The solution $\hat{\theta}$ is decomposed into a rotation and translation $(\hat{R}, \hat{t})$ using SVD [28]. Then

$$L_{\text{pose}} = \frac{1}{2}(\epsilon_{\mathbf{R}}(\hat{\mathbf{R}}, \mathbf{R}) + \epsilon_{\mathbf{t}}(\hat{\mathbf{t}}, \mathbf{t})), \tag{6}$$

where $(\mathbf{R}, \mathbf{t})$ is the ground truth rotation and translation, and functions $\epsilon_{\mathbf{R}}$ and $\epsilon_{\mathbf{t}}$ compute the rotation and translation errors, respectively: $\epsilon_{\mathbf{R}}(\hat{\mathbf{R}}, \mathbf{R}) = \cos^{-1}((\text{tr}(\hat{\mathbf{R}}\mathbf{R}^{\mathsf{T}})-1)/2)$, $\epsilon_{\mathbf{t}}(\hat{\mathbf{t}}, \mathbf{t}) = \cos^{-1}(\hat{\mathbf{t}}^{\mathsf{T}}\mathbf{t}/\|\hat{\mathbf{t}}\|\|\mathbf{t}\|)$. The *average symmetric epipolar error* is defined as follows:

$$L_{\text{epi}} = \frac{1}{|\mathcal{I}|}\sum_{i \in \mathcal{I}} \epsilon_{\text{epi}}(\theta, \Phi_i), \tag{7}$$

where $\mathcal{I}$ is the inlier set selected by the ground truth model θ, Φ_i is the geometric features of a match i and ϵ_{epi} is the respective residual error. The overall loss minimized during training is the linear combination of the above ones as:

$$L = w_\alpha L_{\text{pose}} + w_\beta L_{\text{epi}}, \tag{8}$$

where w_α, w_β are weighting parameters. The above-mentioned metrics are popular for evaluation and are differentiable. Since ∇-RANSAC is differentiable, it enables a direct optimization on such evaluation metrics to learn the sampling probabilities and geometric features.

3.4. Training and Testing Details

The input of ∇-RANSAC is a set of tentative correspondences obtained by any feature detector and matcher. The number of matches is fixed to 2K. We choose the best 2K ones based on the matching score if we are given more. In case of having fewer correspondences, we fill up the missing values with zeros. We extract local and global features from the correspondences by a consensus learning block as backbone [79]. We integrate over the extracted geometric information, *e.g.*, scale, and orientation of the local descriptors, to help learn the qualities of correspondences.

Initialization. We apply a 1K epoch-long weight initialization procedure as in [10]. For this initialization, the KL divergence between the predicted importance $\text{Cat}(p)$ and the target categorical distributions is minimized. The target distribution is chosen proportional to softmax of residuals of all points from the GT model [10]. This initialization scheme does not require sampling the model hypothesis.

Training. Along with the initialized weights, the gradient clipping [13] technique is used to avoid exploding gradients, make the training stable and accelerate the convergence. In the $\mathbf{F}$ matrix case, we normalize the points for consensus learning and use the original unnormalized ones in minimal solvers. We train the pipeline using the 8PC and 7PC algorithms for $\mathbf{F}$ estimation with fixed 1K iterations. For $\mathbf{E}$ case, we use the 5PC algorithm and 100 iterations.

Testing. At test time, we equip state-of-the-art RANSAC components. The model trained end-to-end provides importance scores to the weighted random sampler. Drawing a sample from PL distribution with scores s is simplified as:

$$(i_1 \dots i_k) = \text{top-}k(u_i^{1/s_i}), \quad u_i \sim \text{Uniform}(0, 1). \tag{9}$$

We use the MAGSAC++ [5] model quality function to select the best model, marginalizing over a range of noise scales. We also apply an inner RANSAC-based local optimization [40] to improve the accuracy further. Also, we perform the Levenberg-Marquardt [49] numerical optimization minimizing the pose error on all inliers as a final step. The test code runs in C++ to be fast.

4. Experimental Results

Our main experiments for epipolar geometry estimation were conducted on 13 scenes from the training set of the CVPR IMW 2020 PhotoTourism benchmark [37] that provides images, intrinsic and extrinsic camera parameters from a reference COLMAP reconstruction, and pre-detected RootSIFT features [1]. We train and validate the method on scene St. Peter's Square consisting of 4950 image pairs, split 3 to 1 into training and validation sets. The other 12 scenes are used for testing. We compare ∇-RANSAC on fundamental ($\mathbf{F}$) and essential ($\mathbf{E}$) matrix estimation to classical robust estimators, *i.e.*, RANSAC [24], LMEDS [59], and their recent alternatives, such as GC-RANSAC [4], MAGSAC [6], MAGSAC++ [5] and EAS [23]. Also, we test the provided models by the state-of-the-art learning-based methods, OANet [78], CLNet [79] (both followed by RANSAC), NeFSAC [12], and NG-RANSAC [10]. To make fair comparison, we re-trained NG-RANSAC [10], CLNet [79], and OANet [78] on the same data as what we train ∇-RANSAC on. Moreover, we train and evaluate on ScanNet [20] following the widely used feature matcher SuperGlue [60]. In addition, we apply the proposed method in point cloud registration by training and testing GeoTransformer [56] features of 3DMatch [77] and 3DLoMatch [31], shown in Sec. 4.3.

Technical details. There are two setups of training with ∇-RANSAC. *First*, we train the consensus learning module [79] with off-the-shelf features: **Outdoors** is trained with the SNN ratio [44] coming from RootSIFT descriptors, and the underlying feature scales and orientations as learnable side information besides coordinates, where prefiltering (threshold=0.8) and initialization are needed; **Indoors** is trained with the coordinates and confidence output from the

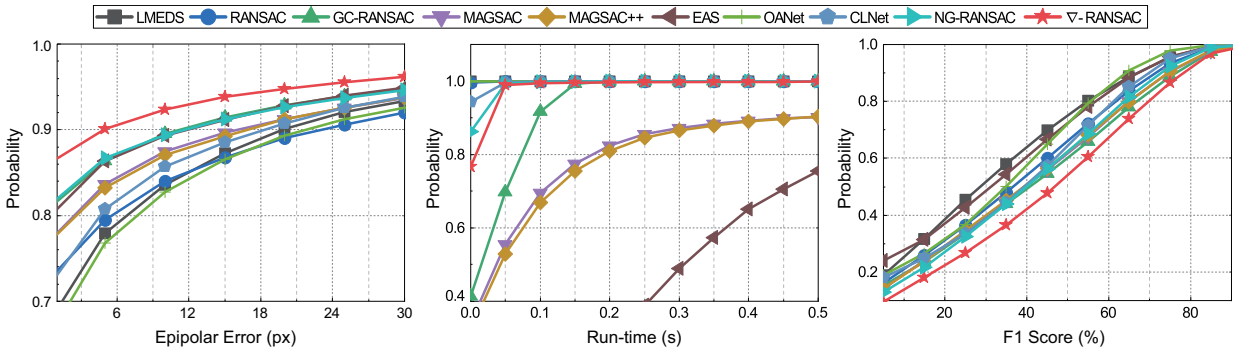

Figure 3. ∇-RANSAC performance on 12 scenes of PhotoTourism measured by the cumulative distribution functions (CDF) of the epipolar errors (*left; in pixels*), run-times (*middle; in seconds*), and F1 score (*right; in percentages*) for **F** matrix estimation. We use the thresholds as in [7] for the traditional algorithms. Besides, OANet, CLNet, and NG-RANSAC were retrained on the same datasets as ∇-RANSAC. In the left two plots, being close to the top-left corner indicates accurate results. The bottom-right corner is preferable in the last plot.

Dataset / Method	LMEDS [59]	RSC [24]	GC-RSC [4]	MSC [6]	MSC++ [5]	EAS [23]	OANet [78]	CLNet [79]	NG-RSC [10]	∇-RANSAC
Avg. time (ms) ↓	**21.00**	30.67	73.25	281.3	318.13	325.83	**21.00**	34.83	25.85	28.66
Buckingham Palace	23.26	24.95	26.48	27.23	26.28	28.28	24.70	27.46	28.06	**33.05**
Brandenburg Gate	31.74	39.70	43.01	42.02	42.43	34.84	42.49	39.69	43.19	**47.66**
Colosseum Exterior	43.32	48.25	50.86	51.76	51.56	50.49	40.50	47.12	52.42	**55.80**
Grand Place Brussels	27.45	31.42	33.31	32.60	33.08	32.40	29.10	31.80	32.13	**35.61**
Notre Dame Front Facade	30.39	37.20	40.48	39.35	39.00	39.98	33.17	38.20	40.59	**46.10**
Palace of Westminster	22.20	28.29	33.15	31.94	31.56	32.42	31.33	32.96	33.54	**41.15**
Pantheon Exterior	54.22	59.23	62.26	61.86	61.60	60.54	49.89	56.81	61.31	**64.48**
Prague Old Town Square	26.61	30.14	32.48	32.39	31.30	34.35	34.05	37.58	33.13	**37.80**
Sacre Coeur	41.58	49.03	56.36	53.23	53.06	45.10	41.30	45.09	56.61	**61.52**
Taj Mahal	38.43	48.44	51.51	50.71	50.43	51.63	50.04	52.17	54.71	**58.58**
Trevi Fountain	29.85	31.67	34.99	33.94	34.28	33.75	27.69	30.82	34.61	**39.11**
Westminster Abbey	50.97	52.27	55.99	55.15	54.91	53.07	42.10	48.37	53.61	**56.55**
Avg. over all scenes ↑	35.00	40.10	43.41	42.68	42.46	42.91	36.91	40.67	43.66	**48.12**

Table 1. The average run-time (*first row; in milliseconds*) and F1 scores (*each, average in last row*) for **F** matrix estimation on 12K image pairs from the PhotoTourism dataset [37]. We use the threshold tuned in [7] for RANSAC, GC-RANSAC, MAGSAC, and MAGSAC++. We tuned the parameters of EAS, and retrained OANet, CLNet, NG-RANSAC on the same data as training ∇-RANSAC. The results with the pre-trained models provided by the authors are in Tab. 2. Best results are **bold**, the second best underlined.

most commonly used feature detector and matcher, *i.e.*, SuperPoint [22] with SuperGlue [60]. For these two cases, we use 0.75 and 3.0 pixels as the inlier-outlier thresholds for robust estimators, respectively. *Second*, a clearer setup of connecting ∇-RANSAC directly to learnable feature matching method, *i.e.*, LoFTR [66], to improve matching prediction with reliable confidence scores (Section 4.5). All the experiments were conducted on Ubuntu 20.04 with GTX 3090Ti, OpenCV 4.5.5, and PyTorch 1.11.1 with Cuda 11.3.1. We re-implemented RANSAC components in PyTorch and connected them with other modules for training.

4.1. Fundamental Matrix Estimation

Benefiting from the proposed ∇-RANSAC, we trained the model parameters jointly with the prediction network to learn the statistical features of tentative matches and predict inlier probabilities. We adopt one consensus learning block from [79] but without filtering the points with their predicted probabilities. The model is trained for 10 epochs and optimized by Adam [39] with a learning rate of $1e^{-5}$. For **F** estimation, the coordinates are normalized by the image sizes before training. For testing, we use 1K randomly chosen image pairs from each of the remaining 12 scenes. Thus, the methods are tested on 12K image pairs in total. To measure the quality of the estimated **F** matrices, we use the F1 score in percentage (%) and median epipolar errors in pixel (px)following [10]. We use the normalized 8PC algorithm for training as it leads to more stable solutions than the 7PC solver in our experiments.

The average F1 scores and the run-time of the robust estimation (in milliseconds) are reported in Tab. 1 on each scene of the dataset and also averaged overall in the last row. The proposed ∇-RANSAC achieves the most accurate results on all but one scene, where it is the second-best by a small margin. On average, it improves by ∼5% compared to the second best method. ∇-RANSAC runs at a comparable speed to less accurate alternatives. The most efficient method is LMeDS achieving an 11.61% lower F1 score than

Metric / Method	OANet [78]	CLNet [79]	NeFSAC [12]	NG-RSC [10]	∇-RANSAC
F1 Score (%) ↑	42.29	38.61	43.17	45.80	**48.12**
Med. epi. error (px) ↓	2.50	8.73	1.92	1.51	**0.86**
Time (ms) ↓	**21.75**	27.83	33.58	25.90	28.66

Table 2. Comparison of ∇-RANSAC and the models provided by the authors of NG-RANSAC, OANet, CLNet and the recent work, NeFSAC for **F** matrix estimation on PhotoTourism. CLNet and OANet were trained on more than 541K image pairs from YFCC [69]. Note that we train on 4950 image pairs of one specific scene and show good generalization on real-world data. NG-RANSAC uses twice as many imag pairs as us.

∇-RANSAC. Note that these timings do not include the inference time, around 2 ms on average using GPU. Also, we show the comparison of the proposed method with the given pre-trained models of the state-of-the-art learning-based robust estimators in Tab. 2. We perform better in terms of F1 scores and errors, with comparable run-time, even though we trained on the least data among the methods in the table. Also, we test on the provided models by the recent work, NeFSAC [12]. The proposed method leads to significant improvements compared to NeFSAC. It is important to note, however, that the contributions of ∇-RANSAC are orthogonal to that of NeFSAC. Thus, they can be straightforwardly combined together.

Furthermore, we show the cumulative distribution functions (CDF) of the epipolar errors, run-times, and F1 scores calculated from the 12K test image pairs in Fig. 3. In the two left plots of epipolar error and run times, being close to the top-left corner indicates better performance. The epipolar error curve of the proposed ∇-RANSAC is above the other methods on the entire plot. While not the fastest, it finishes in 0.1 seconds in 100% of the test cases. This confirms that ∇-RANSAC is applicable in time-sensitive applications. The right figure shows the CDFs of the F1 scores. In contrast to the other plots, being close to the bottom-right corner is preferred. ∇-RANSAC has a better score on the entire range. It coincides with other methods only at the end of the range, as supposed to end up with 1.

4.2. Essential Matrix Estimation

We evaluate **E** matrix estimation both on RootSIFT features of PhotoTourism [64] (outdoors) as used for **F** estimation, and SuperPoint features matched with SuperGlue on ScanNet [20] (indoors). The correspondences are normalized by the intrinsic matrices. **E** matrix estimation is trained with our differentiable 5PC solver. for 10 epochs, where the iteration number of robust estimation is fixed to 100. For evaluation, we decompose the **E** matrix to rotation and translation, calculate their errors $\epsilon_{\mathbf{R}}$, $\epsilon_{\mathbf{t}}$ and report the maximum of rotation and translation errors $\max(\epsilon_{\mathbf{R}}, \epsilon_{\mathbf{t}})$. We calculate the Area Under the Recall curve (AUC) thresholded at 5°, 10°, and 20° following the previous work [76, 10].

PhotoTourism [37]. The AUC scores averaged over 12 testing scenes from PhotoTourism are reported in Tab. 3. The highest AUC scores, at all thresholds, are achieved by ∇-RANSAC. For example, its AUC@5° score is higher than that of the second best methods (*i.e.*, MAGSAC and MAGSAC++) by AUC 3 points. It has comparable run-time to other less accurate alternatives.

Method	AUC@5° ↑	AUC@10° ↑	AUC@20° ↑	Time (ms) ↓
LMEDS[59]	0.24	0.30	0.37	**27**
RANSAC [24]	0.26	0.32	0.40	88
GC-RANSAC [4]	0.33	0.37	0.42	175
MAGSAC [6]	0.37	0.42	0.47	239
MAGSAC++ [5]	0.37	0.42	0.47	113
EAS [23]	0.24	0.28	0.34	325
OANet [78]	0.29	0.33	0.39	49
CLNet [79]	0.34	0.40	0.47	58
NeFSAC [12]	0.34	0.40	0.45	374
NG-RSC [10]	0.35	0.41	0.47	80
∇-RANSAC	**0.41**	**0.45**	**0.50**	117

Table 3. The average AUC scores of ∇-RANSAC and comparison methods over 12 scenes from PhotoTourism, under different thresholds. We are the most accurate method for **E** estimation.

Method	train data	AUC@10° ↑	med. **R** error ↓	med. **t** error ↓
pretr. OANet [78]	YFCC [69], 541K	0.67	2.12	5.26
pretr. CLNet [79]	YFCC [69], 541K	0.69	1.75	4.34
∇-RANSAC	St. Peter's, 55K	**0.77**	**1.26**	**2.91**

Table 4. **E** estimation performance of the proposed method and the pretrained CLNet and OANet, both finishing with a RANSACas a post processing procedure, on the testing scenes used in [2].

Instead of comparing with retrained models, we ran the pretrained CLNet [79] and OANet [78] models provided by the authors for **E** estimation on the test scenes as used in [2]. Both methods finish with a RANSAC on their inliers. The results in Tab. 4 demonstrate that ∇-RANSAC achieves considerably better results even when trained on a fraction of data used for other methods.

11 Other Scenes from PhotoTourism as [2]. In [2], the authors compare on the test set that consists of entirely different scenes from the last paragraph. To be comparable to MQNet [2], we report results on this split in Tab. 5. The proposed ∇-RANSAC leads to substantial improvements compared to MQNet [2] and MAGSAC++ [5], both in terms of rotation (**R**) and translation (**t**) matrix estimation accuracy.

Task	AUC@10° ↑	MAGSAC++ [5]	MQNet [2]	∇-RANSAC
E est.	**R / t**	0.71 / 0.47	0.79 / 0.62	**0.84 / 0.74**
F est.	**R / t**	0.64 / 0.31	0.70 / 0.36	**0.79 / 0.53**

Table 5. Rotation and translation estimation performance on the testing scenes from PhotoTourism [64] as used in MQNet [2].

ScanNet [20]. Following the matching and evaluation process in SuperGlue [60], we trained and tested ∇-RANSAC with the most popular indoor 3D point cloud dataset, *i.e.* ScanNet, consisting of 1201 scans for training and 312 scans for validation. The tentative correspondences are detected and matched using SuperPoint [22] and SuperGlue. We train ∇-RANSAC using 16790 randomly selected pairs from 1201 scans and 4680 for validation. The 1500 test-

Method	Confidence	AUC@5° ↑	AUC@10° ↑	AUC@20° ↑	run-time (μs) ↓
RANSAC	-	16.02	33.53	51.84	110.2
MAGSAC++	-	17.70	35.15	51.75	58.9
MAGSAC++	SuperGlue	18.67	35.85	52.60	51.5
MAGSAC++	∇-trained	**19.15**	**36.40**	**53.47**	**49.3**

Table 6. **E** matrix evaluation on 1500 test pairs of ScanNet used in SuperGlue [60]. The last two rows are tested on MAGSAC++ with PROSAC sampler guided by the confidence predicted from the provided SuperGlue model and ∇-RANSAC trained on SuperGlue matches. Note the run-time is in *microseconds*.

ing pairs are the same ones used in the SuperGlue paper. We evaluate ∇-RANSAC by comparing the AUC scores of RANSAC and MAGSAC++, PROSAC sampling with either SuperGlue confidence or ∇-RANSAC prediction. As shown in Tab. 6, our trained weights work better in guided sampling than the confidence given by SuperGlue.

4.3. 3D Point Rigid Registration

To extend the application of the proposed generalized differentiable RANSAC, we apply ∇-RANSAC on point-cloud rigid registration. As the differentiable minimal solver, we implemented the standard Kabsch algorithm [38] in PyTorch. We trained ∇-RANSAC on GeoTransformer [56] correspondences detected in the challenging indoor benchmarks, *i.e.*, 3DMatch [77] and 3DLoMatch [31]. We use 75 scenes from 3DMatch dataset, 14k image pairs in total for training, also, 8 scenes for validation (1331 pairs), and 8 for testing (1623 pairs) following [56]. In addition, another 8 scenes from 3DLoMatch are used for testing, 1781 image pairs included. Note that 3DLoMatch is more challenging due to low overlap, *i.e.*, below 30%.

To measure the error of the registration, we calculate the Relative Rotation Error (RRE), Relative Translation Error (RTE), and Root Mean Squared Error (RMSE), Registration Recall (RR). We optimize the CLNet [79] inlier probability predictor by the proposed ∇-RANSAC. We then feed MAGSAC++ [5] with PROSAC the original GeoTransformer match confidence predictions and the ones optimized by the proposed method.

As shown in Tab. 7, 8, the inlier priors predicted by ∇-RANSAC better guide the PROSAC sampler than using the original confidences directly from GeoTransformer.

Inlier prob. predictor	# iters	RRE (°) ↓	RTE (cm) ↓	RMSE (cm) ↓	RR (%) ↑
GeoTransformer [56]	1K	27.91	66.64	28.36	76.55
	10K	27.78	68.48	28.41	75.78
	50K	27.76	67.38	28.91	76.36
∇-RANSAC	1K	26.25	64.57	**27.21**	**77.23**
	10K	**25.72**	**62.34**	27.55	76.94
	50K	25.99	63.81	27.45	76.94

Table 7. Point cloud registration performance of our trained ∇-RANSAC on GeoTransformer matches of 3DLoMatch [31] compared with matching confidence, with the best in **bold**.

Inlier prob. predictor	# iters	RRE (°) ↓	RTE (cm) ↓	RMSE (cm) ↓	RR (%) ↑
GeoTransformer [56]	1K	5.76	15.80	6.67	96.90
	10K	6.31	16.99	6.97	96.43
	50K	6.27	16.42	6.80	96.55
∇-RANSAC	1K	5.55	14.68	6.53	**97.01**
	10K	**5.44**	**14.24**	**6.47**	96.90
	50K	**5.44**	14.45	6.57	**97.01**

Table 8. Point cloud registration performance of our trained ∇-RANSAC on GeoTransformer matches of 3DMatch [77] compared with matching confidence, with the best in **bold**.

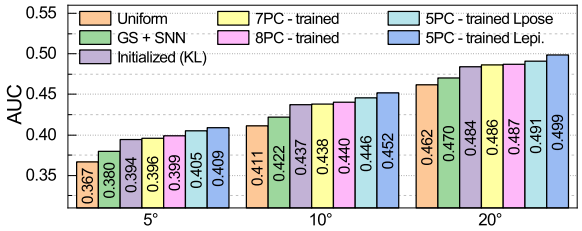

Figure 4. Ablation Studies of Gumbel Softmax (GS) sampler. AUC scores for **E** estimation using Uniform sampler, GS with SNN ratio, and learned weights (Initialized, 7PC-trained, 8PC-trained, and 5PC-trained) on 12K images from PhotoTourism.

4.4. Ablation Studies

Objective functions. Tab. 9 compares training objectives optimized by ∇-RANSAC running the same components (including Gumbel Sampler) in *all* cases. The first row shows the results with the proposed Eq. 2 as the objective. The next two rows replace Eq. 2 with other objectives, *e.g.*, the probabilistic selection approach of DSAC [9]. In the last row, we use REINFORCE for gradient calculation and backpropagation as [10] does.

For **F** estimation, ∇-RANSAC trained with Eq. 2 performs the best in terms of median epipolar error. SoftAM achieves a comparable F1 score but low efficiency. In addition, ∇-RANSAC is the best in terms of **E** estimation accuracy with the second-best run-time, while the fastest REINFORCE [10] achieves lower accuracy. Note that we cannot directly compare with DSAC, but the learning objective of DSAC can be integrated in our training as an option.

Differentiable Sampler. The sampling procedures are tested using MAGSAC++ on 12K image pairs from PhotoTourism in Fig. 4. Uniform and GS + SNN [44] are uniform random sampling and GS guided by SNN ratios, respectively. The other tested methods are GS with different weights: initialized (with KL divergence), trained with 7PC, 8PC, and 5PC with different losses. Epipolar error works better than pose error for **E** estimation. Compared with a uniform sampler and non-learnable weights for the GS sampler, ∇-RANSAC weights perform better.

Differentiable Solvers. For **F** matrix estimation, the average results when ∇-RANSAC is trained with the 7PC and norm. 8PC solvers are shown in Tab. 10. For **F** estimation, the 8PC solver leads to the most accurate results while be-

Learning Objectives	**F** matrix estimation			**E** matrix estimation	
	F1 score ↑	med. epi. error ↓	run-time (ms) ↓	AUC@10° ↑	run-time (ms) ↓
Eq. 2	<u>48.12</u>	**0.86**	**28.66**	**0.45**	**116.90**
Prob. selection [9]	40.95	4.92	<u>34.21</u>	0.43	134.48
SoftAM [9]	**48.17**	<u>0.88</u>	47.46	<u>0.44</u>	147.88
REINFORCE [10]	42.77	4.06	34.75	<u>0.44</u>	<u>125.65</u>

Table 9. Performance of ∇-RANSAC trained with different learning objectives: SoftAM and probabilistic model selection (DSAC) from [9]; and the REINFORCE gradient approximation [10] for **F** and **E** estimation. Best results in **bold**, the second best <u>underlined</u>.

F matrix estimation	Initialized	7PC [28]-trained	8PC [28]-trained
F1 score (%) ↑	45.71	46.04	**47.21**
Med. epi. error (px) ↓	1.73	1.54	**1.00**

Table 10. Performance of ∇-RANSAC trained with **differentiable solvers** on **F** estimation, evaluated by the proposed GS sampler. The same 12 testing scenes are used from PhotoTourism [37].

ing the fastest. As shown in Fig. 4, the highest AUC scores at all thresholds for **E** matrix estimation are achieved by the 5PC method, confirming the necessity of using better minimal solvers for training rather than the 8PC algorithm.

4.5. Learning Feature Matching with ∇-RANSAC

In this section, we tune an end-to-end feature matcher, LoFTR [66], on the epipolar error using ∇-RANSAC. An advantage of our method is that it allows the RANSAC pipeline to be optimized for test-time metrics such as pose and epipolar errors. Even though they are differentiable themselves, their input comes from RANSAC. If RANSAC is non-differentiable, these metrics cannot be directly used as a loss function. Additionally, the entire feature computation and matching module can be directly optimized on such metrics if the RANSAC is differentiable.

Following [51, 52], LoFTR was initialized with the weights from the standard LoFTR and trained. Training of LoFTR with ∇-RANSAC for **F** matrix estimation was performed on scene St. Peter's Square of the PhotoTourism dataset, while the remaining 12 scenes were used for testing as the main experiments. The training used AdamW optimizer [43] for 10 epochs, with a learning rate of $1e^{-6}$. Note that CLNet was not used for this setup as LoFTR directly predicts the filtered correspondences and their confidence scores. Using the ∇-RANSAC, the gradients of the loss are obtained with respect to both the correspondences and confidence scores. Unlike the main experiments, here we use the top-30% of the estimated models for the training.

The evaluation is performed using three different inference protocols, namely OpenCV-RANSAC [24], OpenCV-PROSAC [15], and MAGSAC-PROSAC [5], the results are presented in Tab. 11. These evaluations use a threshold of 3 pixels to determine a prediction as an inlier based on the epipolar error, different from the other experiments in the paper that use a threshold of 0.75 px. The results show that the proposed ∇-RANSAC can be used to improve learning-based feature-matching approaches. This observation indicates the robustness of ∇-RANSAC as a pre-trained model that can be used without any adaptation plug-and-play.

Inference Protocol	LoFTR [66]	F1 score (%) ↑	avg. epi. error (px) ↓	run-time (ms) ↓
OpenCV-RANSAC [24]	Standard	64.07	13.16	2.83
	∇-trained	**64.43**	**11.90**	**2.81**
OpenCV-PROSAC [15]	Standard	66.34	12.53	3.26
	∇-trained	**67.22**	**10.76**	**2.99**
MAGSAC-PROSAC [5]	Standard	69.09	13.06	163.51
	∇-trained	**69.94**	**10.31**	**128.97**

Table 11. LoFTR performance before (standard) and after (∇-trained) trained with ∇-RANSAC. Evaluating **F** matrix estimation uses three different inference methods on 12K image pairs from PhotoTourism. ∇-RANSAC improves LoFTR predicting accurate tentative matches and reliable confidence.

5. Conclusion

In this paper, we propose the differentiable ∇-RANSAC, *i.e.*, the first trainable randomized robust estimator with every component differentiable. It predicts the inlier probabilities of the input data points, exploits the predictions in a guided sampler, and estimates the model (*e.g.*, fundamental matrix) and its quality while propagating the gradients through the entire procedure. ∇-RANSAC leads to the most accurate epipolar geometries compared to state-of-the-art robust estimators on real-world datasets from outdoors and indoors. Also, it is one of the fastest algorithms. To demonstrate its potential in unlocking the training of geometric pipelines, we train ∇-RANSAC together with a recent detector-free feature matcher, LoFTR [66], with which we achieve improved confidence prediction and accurate robust estimation. We believe that the community will benefit from the algorithm.

Acknowledgements

This research was supported by Research Center for Informatics (project CZ.02.1.01/0.0/0.0/16_019/0000765 funded by OP VVV), by the Czech Technical University in Prague grant No. SGS23/173/OHK3/3T/13, by Centers for BMK, BMAW, and the State of Upper Austria within the SCCH competence center INTEGRATE (grant no. 892418) part of the FFG COMET Competence Excellent Technologies Programme and by the ETH Postdoc Fellowship.

References

[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. 5

[2] Daniel Barath, Luca Cavalli, and Marc Pollefeys. Learning to find good models in RANSAC. In *CVPR*, 2022. 4, 7

[3] Daniel Barath., Tat-Jun Chin., Ondřej Chum., Dmytro Mishkin., René Ranftl, and Jiří Matas. RANSAC in 2020 tutorial. In *CVPR*, 2020. 2

[4] Daniel Barath and Jiří Matas. Graph-cut RANSAC. In *CVPR*, 2018. 1, 5, 6, 7

[5] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiří Matas. Magsac++, a fast, reliable and accurate robust estimator. In *CVPR*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 9

[6] Daniel Barath, Jana Noskova, and Jiří Matas. MAGSAC: marginalizing sample consensus. In *CVPR*, 2019. https://github.com/danini/magsac. 1, 4, 5, 6, 7

[7] Daniel Barath, Jana Noskova, and Jiří Matas. Marginalizing sample consensus. *TPAMI*, 2021. 6

[8] Dhruv Batra, Bart Nabbe, and Martial Hebert. An alternative formulation for five point relative pose problem. In *Workshop on Motion and Video Computing*, 2007. 4

[9] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC - differentiable ransac for camera localization. In *CVPR*, 2017. 1, 2, 4, 8, 9

[10] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *ICCV*, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 9

[11] Martin Bujnak, Zuzana Kukelova, and Tomas Pajdla. Making minimal solvers fast. In *CVPR*, 2012. 4

[12] Luca Cavalli, Marc Pollefeys, and Daniel Barath. NeFSAC: Neurally filtered minimal samples. *ECCV*, pages 351–366, 2022. 5, 7

[13] Xiangyi Chen, Steven Z Wu, and Mingyi Hong. Understanding gradient clipping in private sgd: A geometric perspective. *NeurIPS*, 2020. 5

[14] Ondřej Chum and Jırı Matas. Randomized RANSAC with tdd test. In *BMVC*, 2002. 1

[15] Ondřej Chum and Jiří Matas. Matching with PROSAC-progressive sample consensus. In *CVPR*, 2005. 1, 2, 9

[16] Ondřej Chum and Jiří Matas. Optimal randomized RANSAC. *TPAMI*, 2008. 1

[17] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, 2003. 1

[18] Ondřej. Chum, Tomas. Werner, and Jırı. Matas. Epipolar geometry estimation via RANSAC benefits from the oriented epipolar constraint. In *ICPR*, 2004. 1

[19] Ondřej Chum, Tomas Werner, and Jiří Matas. Two-view geometry estimation unaffected by a dominant plane. In *CVPR*, 2005. 1, 4

[20] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 5, 7

[21] Luanyuan Dai, Yizhang Liu, Jiayi Ma, Lifang Wei, Taotao Lai, Changcai Yang, and Riqing Chen. MS2DG-Net: Progressive correspondence learning via multiple sparse semantics dynamic graph. In *CVPR*, 2022. 1, 2

[22] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Workshops*, 2018. 6, 7

[23] Aoxiang Fan, Jiayi Ma, Xingyu Jiang, and Haibin Ling. Efficient deterministic search with robust loss functions for geometric model fitting. *TPAMI*, 2022. 5, 6, 7

[24] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 1, 5, 6, 7, 9

[25] Artyom Gadetsky, Kirill Struminsky, Christopher Robinson, Novi Quadrianto, and Dmitry Vetrov. Low-variance black-box gradient estimates for the plackett-luce distribution. In *AAAI*, 2020. 4

[26] Laureano Gonzalez, Henri Lombardi, Tomás Recio, and M-F Roy. Sturm-habicht sequence. In *ACM-SIGSAM*, 1989. 4

[27] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *ICLR*, 2019. 2, 4

[28] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 4, 5, 9

[29] Richard I Hartley. In defense of the eight-point algorithm. *TPAMI*, 1997. 4

[30] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 1977. 1

[31] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *CVPR*, pages 4267–4276, 2021. 5, 8

[32] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 1988. 1

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

[34] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *IJCV*, 2012. 1

[35] Maksym Ivashechkin, Daniel Barath, and Jiří Matas. VSAC: Efficient and accurate estimator for H and F. In *ICCV*, 2021. 1

[36] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 2, 3

[37] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiří Matas, Pascal Fua, Kwang Yi, and Eduard Trulls. Image matching challenge. In *CVPR*, 2020. 5, 6, 7, 9

[38] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 1976. 2, 8

[39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[40] Karel Lebeda, Jirı Matas, and Ondřej Chum. Fixing the locally optimized RANSAC. In *BMVC*, 2012. 5

[41] Hongdong Li. Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *ICCV*, 2009. 1

[42] Hongdong Li and Richard Hartley. Five-point motion estimation made easy. In *ICPR*, 2006. 4

[43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 9

[44] David G Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 2, 5, 8

[45] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012. 3

[46] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR*, 2017. 3

[47] Jiří Matas, Ondřej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 2004. 1

[48] Dmytro Mishkin, Jiří Matas, and Michal Perdoch. MODS: Fast and robust method for two-view matching. *Computer Vision and Image Understanding*, 2015. 1

[49] Jorge J Moré. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis*. 1978. 5

[50] David Nistér. An efficient solution to the five-point relative pose problem. *TPAMI*, 2004. 2, 4

[51] Yash Patel, Tomáš Hodaň, and Jiří Matas. Learning surrogates via deep embedding. In *ECCV*, 2020. 9

[52] Yash Patel and Jiří Matas. FEDS-filtered edit distance surrogate. In *ICDAR*, 2021. 9

[53] Trung Thanh Pham, Tat-Jun Chin, Konrad Schindler, and David Suter. Interacting geometric priors for robust multimodel fitting. *IEEE Transactions on Image Processing*, 2014. 1

[54] Philip Pritchett and Andrew Zisserman. Wide baseline stereo matching. In *ICCV*, 1998. 1

[55] 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. 2

[56] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *CVPR*, 2022. 5, 8

[57] Rene Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 2

[58] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *WACV*, 2020. 4

[59] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005. 5, 6, 7

[60] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 5, 6, 7, 8

[61] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1

[62] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 4

[63] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 1

[64] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3D. In *SIGGRAPH*. 2006. 7

[65] Henrik Stewénius, David Nistér, Fredrik Kahl, and Frederik Schaffalitzky. A minimal solution for relative pose with unknown focal length. *Image and Vision Computing*, 2008. 4

[66] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *CVPR*, 2021. 1, 2, 6, 9

[67] Weiwei Sun, Wei Jiang, Andrea Tagliasacchi, Eduard Trulls, and Kwang Moo Yi. Attentive context normalization for robust permutation-equivariant learning. In *CVPR*, 2020. 1, 2

[68] Chris Sweeney. Theia multiview geometry library: Tutorial & reference. http://theia-sfm.org. 4

[69] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 2016. 7

[70] Philip HS Torr and David W Murray. Outlier detection and motion segmentation. In *Sensor Fusion VI*, 1993. 1

[71] Philip HS Torr and Andrew Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 2000. 1, 4

[72] Eduard Trulls, Yuhe Jin, Kwang Moo Yi, Dmytro Mishkin, and Jiří Matas. Image matching challenge 2022. https://www.kaggle.com/competitions/image-matching-challenge-2022, 2022. 1

[73] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2014. 4

[74] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992. 2, 4

[75] Guobao Xiao, Hanzi Wang, Taotao Lai, and David Suter. Hypergraph modelling for geometric model fitting. *Pattern Recognition*, 2016. 1

[76] Kwang Moo Yi*, Eduard Trulls*, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 1, 2, 4, 7

[77] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, pages 1802–1811, 2017. 5, 8

[78] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. *ICCV*, 2019. 1, 2, 5, 6, 7

[79] Chen Zhao, Yixiao Ge, Feng Zhu, Rui Zhao, Hongsheng Li, and Mathieu Salzmann. Progressive correspondence pruning by consensus learning. In *ICCV*, 2021. 1, 2, 3, 4, 5, 6, 7, 8