This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Reinforced Feature Points: Optimizing Feature Detection and Description for a High-Level Task

Aritra Bhowmik[1], Stefan Gumhold[1], Carsten Rother[2], Eric Brachmann[2]
[1] TU Dresden, [2] Heidelberg University

Abstract

We address a core problem of computer vision: Detection and description of 2D feature points for image matching. For a long time, hand-crafted designs, like the seminal SIFT algorithm, were unsurpassed in accuracy and efficiency. Recently, learned feature detectors emerged that implement detection and description using neural networks. Training these networks usually resorts to optimizing low-level matching scores, often pre-defining sets of image patches which should or should not match, or which should or should not contain key points. Unfortunately, increased accuracy for these low-level matching scores does not necessarily translate to better performance in high-level vision tasks. We propose a new training methodology which embeds the feature detector in a complete vision pipeline, and where the learnable parameters are trained in an end-to-end fashion. We overcome the discrete nature of key point selection and descriptor matching using principles from reinforcement learning. As an example, we address the task of relative pose estimation between a pair of images. We demonstrate that the accuracy of a state-of-the-art learning-based feature detector can be increased when trained for the task it is supposed to solve at test time. Our training methodology poses little restrictions on the task to learn, and works for any architecture which predicts key point heat maps, and descriptors for key point locations.

1. Introduction

Finding and matching sparse 2D feature points across images has been a long-standing problem in computer vision [19]. Feature detection algorithms enable the creation of vivid 3D models from image collections [21, 56, 45], building maps for robotic agents [31, 32], recognizing places [44, 24, 35] and precise locations [25, 52, 41] as well as recognizing objects [26, 34, 38, 1, 2]. Naturally, the design of feature detection and description algorithms, subsumed as feature detection in the following, has received tremendous attention in computer vision research since its early days. Although invented three decades ago, the seminal SIFT algorithm [26] remains the gold standard feature detection pipeline to this day.

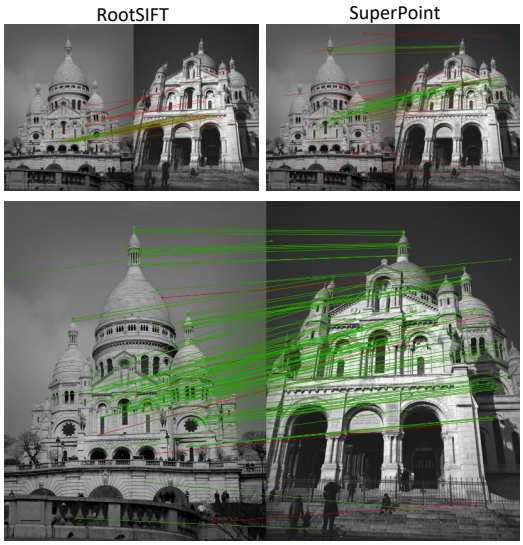


Figure 1. We show the results of estimating the relative pose (essential matrix) between two images using RootSIFT [1] (top left) and SuperPoint [14] (top right). Our Reinforced SuperPoint (bottom), utilizing [14] within our proposed training schema, achieves a clearly superior result. Here the inlier matches wrt. the ground truth essential matrix are drawn in green, outliers in red.

With the recent advent of powerful machine learning tools, some authors replace classical, feature-based vision pipelines by neural networks [22, 53, 4]. However, independent studies suggest that these learned pipelines have not yet reached the accuracy of their classical counterparts [46, 42, 59, 43], due to limited generalization abilities. Alternatively, one prominent strain of current research aims to keep the concept of sparse feature detection but replaces hand-crafted designs like SIFT [26] with data-driven, learned representations. Initial works largely focused on learning to compare image patches to yield expressive feature descriptors [18, 50, 54, 29, 27, 51]. Fewer works attempt to learn feature detection [12, 5] or a complete architecture for feature detection and description [58, 35, 14].

Training of these methods is usually driven by optimizing low-level matching scores inspired by metric learning [54] with the necessity to define ground truth correspondences between patches or images. When evaluated on low-

level matching benchmarks like H-Patches [3], such methods regularly achieve highly superior scores compared to a SIFT baseline. H-Patches [3] defines sets of matching image patches that undergo severe illumination and viewpoint changes. However, the increased accuracy in such matching tasks does not necessarily translate to increased accuracy in high-level vision pipelines. For example, we show that the state-of-the-art learned SuperPoint detector [14], while highly superior to SIFT [26] on H-Patches [3], does not reach SIFT's capabilities when estimating an essential matrix for an image pair. Similar observations were reported in earlier studies, where the supposedly superior learned LIFT detector [58] failed to produce richer reconstructions than SIFT [26] in a structure-from-motion pipeline [46].

Some authors took notice of the discrepancy between low-level training and high-level performance, and developed training protocols that mimic properties of high-level vision pipelines. Lua *et al*. [27] perform hard negative mining of training patches in a way that simulates the problem of self-similarity when matching at the image level. Revaud *et al*. [39] train a detector to find few but reliable key points. Similarly Cieslewski *et al*. [12] learn to find key points with high probability of being inliers in robust model fitting.

In this work, we take a more radical approach. Instead of hand-crafting a training procedure that emulates aspects of high-level vision pipelines, we embed the feature detector in a complete vision pipeline during training. Particularly, our pipeline addresses the task of relative pose estimation, a central component in camera re-localization, structure-from-motion or SLAM. The pipeline incorporates key point selection, descriptor matching and robust model fitting. We do not need to pre-define ground truth correspondences, dispensing with the need for hard-negative mining. Furthermore, we do not need to speculate whether it is more beneficial to find many matches or few, reliable matches. All these aspects are solely guided by the task loss, *i.e*. by minimizing the relative pose error between two images.

Key point selection and descriptor matching are discrete operations which cannot be directly differentiated. However, since many feature detectors predict key point locations as heat maps, we can reformulate key point selection as a sampling operation. Similarly, we lift feature matching to a distribution where the probability of a match stems from its descriptor distance. This allows us to apply principles from reinforcement learning [48] to directly optimize a high-level task loss. Particularly, all operations after the feature matching stage, *e.g*. robust model fitting, do not need to be differentiable since they only provide a reward signal for learning. In summary, our training methodology puts little restrictions on the feature detection architecture or the vision task to be optimized for.

We demonstrate our approach using the SuperPoint detector [14], which regularly ranks among top methods in independent evaluations [16, 5, 39]. We train SuperPoint for the task of relative pose estimation by robust fitting of the essential matrix. For this task, our training procedure closes the gap between SuperPoint and a state-of-the-art SIFT-based pipeline, see Fig. 1 for a comparison of results. We summarize our main contributions:

- A new training methodology which allows for learning a feature detector and descriptor, embedded in a complete vision pipeline, to optimize its performance for a high-level vision task.
- We apply our method to a state-of-the-art architecture, Superpoint [14], and train it for the task of relative pose estimation.
- After training, SuperPoint [14] reaches, and slightly exceeds, the accuracy of SIFT [26] which previously achieved best results for this task.

2. Related Work

Of all hand-crafted feature detectors, SIFT [26] stands out for its long lasting success. SIFT finds key point locations as a difference-of-Gaussian filter response in the scale space of an image, and describes features using histograms of oriented gradients [13]. Arandjelovic and Zisserman [1] improve the matching accuracy of SIFT by normalizing its descriptor, also called RootSIFT. Other hand-crafted feature detectors improve efficiency for real-time applications while sacrificing as little accuracy as possible [6, 40].

MatchNet [18] is an early example of learning to compare image patches using a patch similarity network. The reliance on a network as a similarity measure prevents the use of efficient nearest neighbor search schemes. L2-Net [50], and subsequent works, instead learn patch descriptors to be compared using the Euclidean distance. Balntas *et al*. [54] demonstrated the advantage of using a triplet loss for descriptor learning over losses defined on pairs of patches only. A triplet combines two matching and one non-matching patch, and the triplet loss optimizes relative distances within a triplet. HardNet [29] employs a "hardest-in-batch" strategy when assembling triplets for training, *i.e*. for each matching patch pair, they search for the most similar non-matching patch within a mini-batch. GeoDesc [27] constructs mini-batches for training that contain visually similar but non-matching patch pairs to mimic the problem of self-similarity when matching two images. SOSNet [51] uses second order similarity regularization to enforce a structure of the descriptor space that leads to well separated clusters of similar patches.

Learning feature *detection* has also started to attract attention recently. ELF [7] shows that feature detection can be implemented using gradient tracing within a pre-trained neural network. Key.Net [5] combines hand-crafted and learned filters to avoid overfitting. The detector is trained using a repeatability objective, *i.e*. finding the same points

in two related images, synthetically created by homography warping. SIPs [12] learns to predict a pixel-wise probability map of inlier locations as key points, inlier being a correspondence which can be continuously tracked throughout an image sequence by an off-the-shelf feature tracker.

LIFT [58] was the first, complete learning-based architecture for feature detection *and* description. It rebuilds the main processing steps of SIFT with neural networks, and is trained using sets of matching and non-matching image patches extracted from structure-from-motion datasets. DELF [35] learns detection and description for image retrieval, where coarse key point locations emerge by training an attention layer on top of a dense descriptor tensor. D2-Net [16] implements feature detection and description by searching for local maxima in the filter response map of a pre-trained CNN. R2D2 [39] proposes a learning scheme for identifying feature locations that can be matched uniquely among images, avoiding repetitive patterns.

All mentioned learning-based works design training schemes that emulate difficult conditions for a feature detector when employed for a vision task. Our work is the first to directly embed feature detection and description in a complete vision pipeline for training where all real-world challenges occur, naturally. On a similar note, KeypointNet [49] describes a differentiable pipeline that automatically discovers category-level key points for the task of relative pose estimation. However, [49] does not consider feature description nor matching. In recent years, Brachmann *et al*. described a differentiable version of RANSAC (DSAC) [8, 9] to learn a camera localization pipeline end-to-end. Similar to DSAC, we derive our training objective from policy gradient [48]. However, by formulating feature detection and matching via sampling we do not require gradients of RANSAC, and hence we do not utilize DSAC.

We realize our approach using the SuperPoint [14] architecture, a fully convolutional CNN for feature detection and description, pre-trained on synthetic and homography-warped real images. In principle, our training scheme can be applied to architectures other than SuperPoint, like LIFT [58] or R2D2 [39], and also to separate networks for feature detection and description.

3. Method

As an example of a high-level vision task, we estimate the relative transformation $T = (R, \mathbf{t})$, with rotation R and translation $\mathbf{t}$, between two images I and I'. We solve the task using sparse feature matching. We determine 2D key points $\mathbf{x}_i$ indexed by i, and compute a descriptor vector $\mathbf{d}(\mathbf{x}_i)$ for each key point. Using nearest neighbor matching in descriptor space, we establish a set of tentative correspondences $\mathbf{m}_{ij} = (\mathbf{x}_i, \mathbf{x}'_j)$ between images I and I'. We solve for the relative pose based on these tentative correspondences by robust fitting of the essential matrix [20]. We apply a robust estimator like RANSAC [17] with a 5-point solver [33] to find the essential matrix which maximises the inlier count among all correspondences. An inlier is defined as a correspondence with a distance to the closest epipolar line below a threshold [20]. Decomposition of the essential matrix yields an estimate of the relative transformation $\hat{T}$.

We implement feature detection using two networks: a detection network and a description network. In practice, we use a joint architecture, SuperPoint[14], where most weights are shared between detection and description. The main goal of this work is to optimize the learnable parameters $\mathbf{w}$ of both networks such that their accuracy for the vision task is enhanced. For our application, the networks should predict key points and descriptors such that the relative pose error between two images is minimized. Key point selection and feature matching are discrete, non-differentiable operations. Therefore, we cannot directly propagate gradients of our estimated transformation $\hat{T}$ back to update the network weights, as in standard supervised learning. Components of our vision pipeline, like the robust estimator (*e.g*. RANSAC [17]) or the minimal solver (*e.g*. the 5-point solver [33]) might also be non-differentiable. To optimize the neural network parameters for our task, we apply principles from reinforcement learning [48]. We formulate feature detection and matching as probabilistic actions where the *probability* of taking an action, *i.e*. selecting a key point, or matching two features, depends on the output of the neural networks. During training, we sample different instantiations of key points and their matchings based on the probability distributions predicted by the neural networks. We observe how well these key points and their matching perform in the vision task, and adjust network parameters $\mathbf{w}$ such that an outcome with low loss becomes more probable. We show an overview of our approach in Fig. 2.

In the following, we firstly describe how to reformulate key point selection and feature matching as probabilistic actions. Thereafter, we formulate our learning objective, and how to efficiently approximate it using sampling.

3.1. Probabilistic Key Point Selection

We assume that the detection network predicts a key point heat map $f(I; \mathbf{w})$ for an input image, as is common in many architectures [58, 14, 39, 12]. Feature locations are usually selected from $f(I; \mathbf{w})$ by taking all local maxima combined with local non-max suppression.

To make key point selection probabilistic, we instead interpret the heat map as a probability distribution over key point locations $f(I; \mathbf{w}) = P(\mathbf{x}; \mathbf{w})$ parameterized by the network parameters $\mathbf{w}$. We define a set of N key points for image I as $\mathcal{X} = \{\mathbf{x}_i\}$ sampled independently according to

$$P(\mathcal{X}; \mathbf{w}) = \prod_{i=1}^{N} P(\mathbf{x}_i; \mathbf{w}), \tag{1}$$

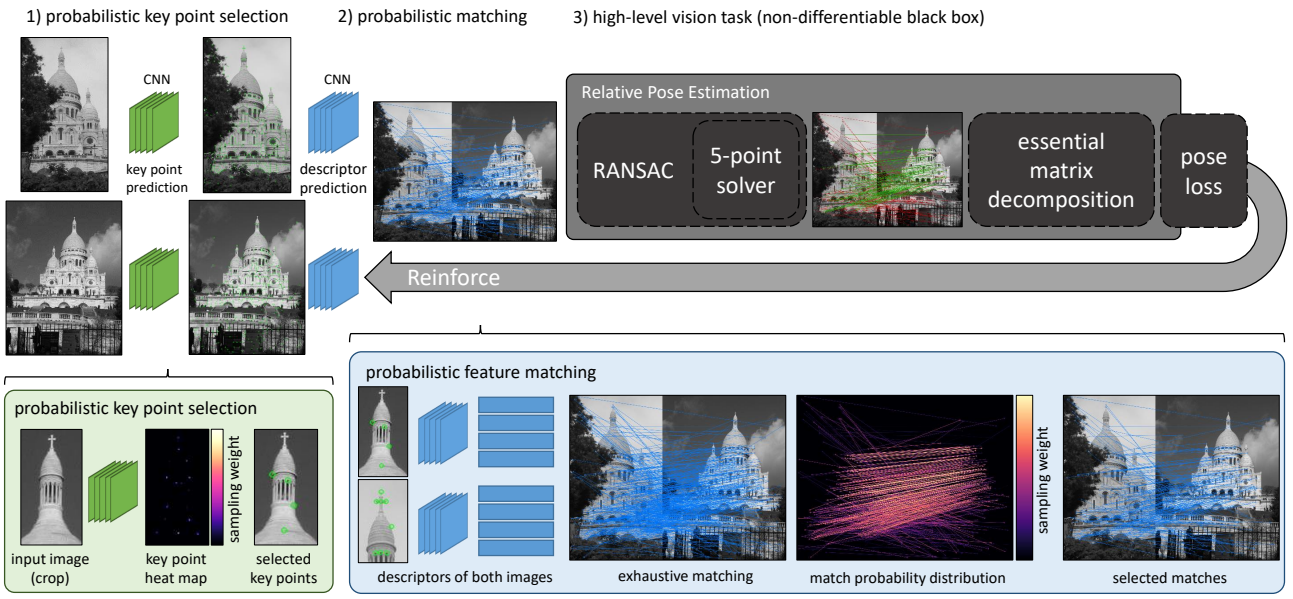


Figure 2. **Method. Top:** Our training pipeline consists of probabilistic key point selection and probabilistic feature matching. Based on established feature matches, we solve for the relative pose and compare it to the ground truth pose. We treat the vision task as a (potentially non-differentiable) black box. It provides an error signal, used to reinforce the key point and matching probabilities. Based on the error signal both CNNs (green and blue) are updated, making a low loss more likely. **Bottom Left:** We sample key points according to the heat map predicted by the detector. **Bottom Right:** We implement probabilistic matching by, firstly, doing an exhaustive matching between all key points, secondly, calculating a probability distribution over all matches depending on their descriptor distance, and, thirdly, sampling a subset of matches. We pass only this subset of matches to the black box estimator.

see also Fig. 2, bottom left. Similarly, we define $\mathcal{X}'$ for image I'. We give the joint probability of sampling key points independently in each image as

$$P(\mathcal{X}, \mathcal{X}'; \mathbf{w}) = P(\mathcal{X}; \mathbf{w}) P(\mathcal{X}'; \mathbf{w}). \quad (2)$$

3.2. Probabilistic Feature Matching

We assume that a second description network predicts a feature descriptor $\mathbf{d}(\mathbf{x}; \mathbf{w})$ for a given key point $\mathbf{x}$. To simplify notation, we use $\mathbf{w}$ to denote the learnable parameters associated with feature detection *and* description. We define a feature match as a pairing of one key point from image I and image I', respectively: $\mathbf{m}_{ij} = (\mathbf{x}_i, \mathbf{x}'_j)$. We give the probability of a match between two key points $\mathbf{x}_i$ and $\mathbf{x}'_j$ as a function of their descriptor distance,

$$P(\mathbf{m}_{ij}|\mathcal{X}, \mathcal{X}'; \mathbf{w}) = \frac{\exp[-||\mathbf{d}(\mathbf{x}_i; \mathbf{w}) - \mathbf{d}(\mathbf{x}'_j; \mathbf{w})||]}{\sum_{\mathbf{m}_{kk'}} \exp[-||\mathbf{d}(\mathbf{x}_k; \mathbf{w}) - \mathbf{d}(\mathbf{x}'_{k'}; \mathbf{w})||]}. \quad (3)$$

Note that the matching probability is conditioned on the sets of key points which we selected in an earlier step. The matching distribution is normalized using all possible matches $\mathbf{m}_{kk'} = (\mathbf{x}_k, \mathbf{x}'_{k'})$ with $\mathbf{x}_k \in \mathcal{X}$ and $\mathbf{x}'_{k'} \in \mathcal{X}'$. The matching distribution assigns a low probability to a match, if the associated key points have very different descriptors. In turn, if the network wants to increase the probability of a (good) match during training, it has to reduce the descriptor distance of the associated key points relative to all other matches for the same image pair.

We define a complete set of M matches $\mathcal{M} = \{\mathbf{m}_{ij}\}$ between I and I' sampled independently according to

$$P(\mathcal{M}|\mathcal{X}, \mathcal{X}'; \mathbf{w}) = \prod_{\mathbf{m}_{ij} \in \mathcal{M}} P(\mathbf{m}_{ij}|\mathcal{X}, \mathcal{X}'; \mathbf{w}). \quad (4)$$

3.3. Learning Objective

We learn network parameters $\mathbf{w}$ in a supervised fashion, *i.e.* we assume to have training data of the form (I, I', T^*) with ground truth transformation T^*. Note that we do *not* need ground truth key point locations $\mathcal{X}$ or ground truth image correspondences $\mathcal{M}$.

Our learning formulation is agnostic to the implementation details of how the vision task is solved using tentative image correspondences $\mathcal{M}$. We treat the exact processing pipeline after the feature matching stage as a black box which produces only one output: a loss value $\ell(\mathcal{M}, \mathcal{X}, \mathcal{X}')$, which depends on the key points $\mathcal{X}$ and $\mathcal{X}'$ that we selected for both images, and the matches $\mathcal{M}$ that we selected among the key points. For relative pose estimation, calculating ℓ entails robust fitting of the essential matrix, its decomposition to yield an estimated relative camera transformation $\hat{T}$, and its comparison to a ground truth transformation T^*. We only require the loss value itself, not its gradients.

Our training objective aims at reducing the expected task loss when sampling key points and matches according to the probability distributions parameterized by the learnable

parameters $\mathbf{w}$:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \mathbb{E}_{\mathcal{M},\mathcal{X},\mathcal{X}' \sim P(\mathcal{M},\mathcal{X},\mathcal{X}';\mathbf{w})} \left[\ell(\mathcal{M},\mathcal{X},\mathcal{X}')\right] \\ &= \mathbb{E}_{\mathcal{X},\mathcal{X}' \sim P(\mathcal{X},\mathcal{X}';\mathbf{w})} \mathbb{E}_{\mathcal{M} \sim P(\mathcal{M}|\mathcal{X},\mathcal{X}';\mathbf{w})} \left[\ell(\cdot)\right], \end{aligned} \tag{5}$$

where we abbreviate $\ell(\mathcal{M},\mathcal{X},\mathcal{X}')$ to $\ell(\cdot)$. We split the expectation in key point selection and match selection. Firstly, we select key points $\mathcal{X}$ and $\mathcal{X}'$ according to the heat map predictions of the detection network $P(\mathcal{X},\mathcal{X}';\mathbf{w})$ (see Eq. 1 and 2). Secondly, we select matches among these key points according to a probability distribution $P(\mathcal{M}|\mathcal{X},\mathcal{X}';\mathbf{w})$ calculated from descriptor distances (see Eq. 3 and 4).

Calculating the expectation and its gradients exactly would necessitate summing over all possible key point sets, and all possible matchings, which is clearly infeasible. To make the calculation tractable, we assume that the network is already initialized, and makes sensible predictions that we aim at optimizing further for our task. In practice, we take an off-the-shelf architecture, like SuperPoint [14], which was trained on a low-level matching task. For such an initialized network, we observe the following properties:

1. Heat maps predicted by the feature detector are sparse. The probability of selecting a key point is zero at almost all image pixels (see Fig. 2 bottom, left). Therefore, only few image locations have an impact on the expectation.
2. Matches among unrelated key points have a large descriptor distance. Such matches have a probability close to zero, and no impact on the expectation.

Observation 1) means, we can just sample from the key point heat map, and ignore other image locations. Observation 2) means that for the key points we selected, we do not have to realise a complete matching of all key points in $\mathcal{X}$ to all key points in $\mathcal{X}'$. Instead, we rely on a k-nearest-neighbour matching with some small k. All nearest neighbours beyond k likely have large descriptor distances, and hence near zero probability. In practice, we found no advantage in using a $k > 1$ which means we can do a normal nearest neighbour matching during training when calculating $P(\mathcal{M}|\mathcal{X},\mathcal{X}';\mathbf{w})$ (see Fig. 2 bottom, right).

We update the learnable parameters $\mathbf{w}$ according to the gradients of Eq. 5, following the classic REINFORCE algorithm [55] of Williams:

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) = \\ &\quad \mathbb{E}_{\mathcal{X},\mathcal{X}'} \left[\mathbb{E}_{\mathcal{M}|\mathcal{X},\mathcal{X}'} \left[\ell(\cdot)\right] \frac{\partial}{\partial \mathbf{w}} \log P(\mathcal{X},\mathcal{X}';\mathbf{w}) \right] \\ &+ \mathbb{E}_{\mathcal{X},\mathcal{X}'} \left[\mathbb{E}_{\mathcal{M}|\mathcal{X},\mathcal{X}'} \left[\ell(\cdot) \frac{\partial}{\partial \mathbf{w}} \log P(\mathcal{M}|\mathcal{X},\mathcal{X}';\mathbf{w}) \right] \right] \end{aligned} \tag{6}$$

Note that we only need to calculate the gradients of the log probabilities of key point selection and feature matching. We approximate the expectations in the gradient calculation by sampling. We approximate $\mathbb{E}_{\mathcal{X},\mathcal{X}'}$ by drawing $n_{\mathcal{X}}$ samples $\hat{\mathcal{X}},\hat{\mathcal{X}}' \sim P(\mathcal{X},\mathcal{X}';\mathbf{w})$. For a given key point sample, we approximate $\mathbb{E}_{\mathcal{M}|\mathcal{X},\mathcal{X}'}$ by drawing $n_{\mathcal{M}}$ samples $\hat{\mathcal{M}} \sim P(\mathcal{M}|\hat{\mathcal{X}},\hat{\mathcal{X}}';\mathbf{w})$. For each sample combination, we run the vision pipeline and observe the associated task loss ℓ. To reduce the variance of the gradient approximation, we subtract the mean loss over all samples as a baseline [48]. We found a small number of samples for $n_{\mathcal{X}}$ and $n_{\mathcal{M}}$ sufficient for the pipeline to converge.

4. Experiments

We train the SuperPoint [14] architecture for the task of relative pose estimation, and report our main results in Sec. 4.1. Furthermore, we analyse the impact of reinforcing SuperPoint for relative pose estimation on a low-level matching benchmark (Sec. 4.2), and in a structure-from-motion task (Sec. 4.3).

4.1. Relative Pose Estimation

Network Architecture. SuperPoint [14] is a fully-convolutional neural network which processes full-sized images. The network has two output heads: one produces a heat map from which key points can be picked, and the other head produces 256-dimensional descriptors as a dense descriptor field over the image. The descriptor output of SuperPoint fits well into our training methodology, as we can look up descriptors for arbitrary image locations without doing repeated forward passes of the network. Both output heads share a common encoder which processes the image and reduces its dimensionality, while the output heads act as decoders. We use the network weights provided by the authors as an initialization.

Task Description. We calculate the relative camera pose between a pair of images by robust fitting of the essential matrix. We show an overview of the processing pipeline in Fig. 2. The feature detector produces a set of tentative image correspondences. We estimate the essential matrix using the 5-point algorithm [33] in conjunction with a robust estimator. For the robust estimator, we conducted experiments with a standard RANSAC [17] estimator, as well as with the recent NG-RANSAC [10]. NG-RANSAC uses a neural network to suppress outlier correspondences, and to guide RANSAC sampling towards promising candidates for the essential matrix. As a learning-based robust estimator, NG-RANSAC is particularly interesting in our setup, since we can refine it in conjunction with SuperPoint during end-to-end training.

Datasets. To facilitate comparison to other methods, we follow the evaluation protocol of Yi *et al*. [59] for relative pose estimation. They evaluate using a collection of 7 outdoor and 16 indoor datasets from various sources

[47, 21, 57]. One outdoor scene and one indoor scene serve as training data, the remaining 21 scenes serve as test set. All datasets come with co-visibility information for the selection of suitable image pairs, and ground truth poses.

Training Procedure. We interpret the output of the detection head of SuperPoint as a probability distribution over key point locations. We sample 600 key points for each image, and we read out the descriptor for each key point from the descriptor head output. Next, we perform a nearest neighbour matching between key points, accepting only matches of mutual nearest neighbors in both images. We calculate a probability distribution over all the matches depending on their descriptor distance (according to Eq. 3). We randomly choose 50% of all matches from this distribution for the relative pose estimation pipeline. We fit the essential matrix, and estimate the relative pose up to scale. We measure the angle between the estimated and ground truth rotation, as well as, the angle between the estimated and the ground truth translation vector. We take the maximum of both angles as our task loss ℓ. For difficult image pairs, essential matrix estimation can fail, and the task loss can be very large. To limit the influence of such large losses, we apply a square root soft clamping [10] of the loss after a value of $25°$, and a hard clamping after a value of $75°$.

To approximate the expected task loss $\mathcal{L}(\mathbf{w})$ and its gradients in Eq. 5 and Eq. 6, we draw key points $n_{\mathcal{X}} = 3$ times, and, for each set of key points, we draw $n_{\mathcal{M}} = 3$ sets of matches. Therefore, for each training iteration, we run the vision pipeline 9 times, which takes 1.5s to 2.1s on a single Tesla K80 GPU, depending on the termination of the robust estimator. We train using the Adam [23] optimizer and a learning rate of 10^{-7} for 150k iterations which takes approximately 60 hours. Our training code is based on PyTorch [37] for SuperPoint [14] integration and learning, and on OpenCV [11] for estimating the relative pose. We will make our source code publicly available to ensure reproducibility of our approach.

Test Procedure. For testing, we revert to a deterministic procedure for feature detection, instead of doing sampling. We select the strongest 2000 key points from the detector heat map using local non-max suppression. We remove very weak key point with a heat map value below 0.00015. We do a nearest neighbor matching of the corresponding feature descriptors, and keep all matches of mutual nearest neighbors. We adhere to this procedure for SuperPoint before and after our training, to ensure comparability of the results.

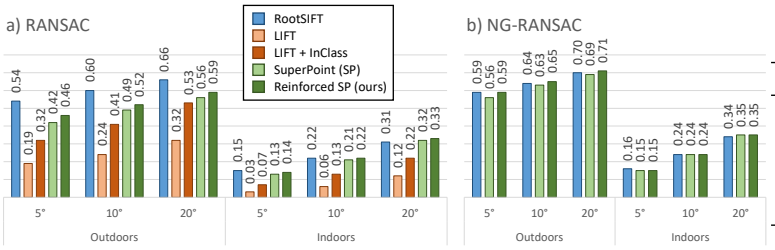
Discussion. We report test accuracy in accordance to Yi *et al*. [59], who calculate the pose error as the maximum of rotation and translation angular error. For each dataset, the area under the cumulative error curve (AUC) is calculated and the mean AUC for outdoor and indoor datasets are reported separately.

Firstly, we train and test our pipeline using a standard RANSAC estimator for essential matrix fitting, see Fig. 3 a). We compare to a state-of-the-art SIFT-based [26] pipeline, which uses RootSIFT descriptor normalization [1]. For RootSIFT, we apply Lowe's ratio criterion [26] to filter matches where the distance ratio of the nearest and second nearest neighbor is above 0.8. We also compare to the LIFT feature detector [58], with and without the learned inlier classification scheme of Yi *et al*. [59] (denoted *InClass*). Finally, we compare the results of SuperPoint [14] before and after our proposed training (denoted *Reinforced SP*).

Reinforced SuperPoint exceeds the accuracy of SuperPoint across all thresholds, proving that our training scheme indeed optimizes the performance of SuperPoint for relative pose estimation. The effect is particularly strong for outdoor environments. For indoors, the training effect is weaker, because large texture-less areas make these scenes difficult for sparse feature detection, in principle. SuperPoint exceeds the accuracy of LIFT by a large extent, but does not reach the accuracy of RootSIFT. We found that the excellent accuracy of RootSIFT is largely due to the effectiveness of Lowe's ratio filter for removing unreliable SIFT matches. We tried the ratio filter also for SuperPoint, but we found no ratio threshold value that would consistently improve accuracy across all datasets.

To implement a similarly effective outlier filter for SuperPoint, we substitute the RANSAC estimator in our vision pipeline with the recent learning-based NG-RANSAC [10] estimator. We train NG-RANSAC for SuperPoint using the public code of Brachmann and Rother [10], and with the initial weights for SuperPoint by Detone *et al*. [14]. With NG-RANSAC as a robust estimator, SuperPoint almost reaches the accuracy of RootSIFT, see Fig. 3, b). Finally, we embed both, SuperPoint and NG-RANSAC in our vision pipeline, and train them jointly and end-to-end. After our training schema, Reinforced SuperPoint matches and slightly exceeds the accuracy of RootSIFT. Fig. 3, c) shows an ablation study where we either update only NG-RANSAC, only SuperPoint or both during end-to-end training. While the main improvement comes from updating SuperPoint, updating NG-RANSAC as well allows the robust estimator to adapt to the changing matching statistics of SuperPoint throughout the training process.

Analysis. We visualize the effect of our training procedure on the outputs of SuperPoint in Fig. 4. For the key point heat maps, we observe two major effects. Firstly, many key points seem to be discarded, especially for repetitive patterns that would result in ambiguous matches. Secondly, some key points are kept, but their position is adjusted, presumably to achieve a lower relative pose error. For the descriptor distribution, we see a tendency of reducing the descriptor distance for correct matches, and increas-



c) Ablation Study

	Outdoors (5°/10°/20°)	Indoors (5°/10°/20°)
RootSIFT + NG-RANSAC	0.59/0.64/0.70	0.16/0.24/0.34
SuperPoint (init) + NG-RANSAC (init)	0.56/0.63/0.69	0.15/0.24/0.35
SuperPoint (init) + NG-RANSAC (e2e)	0.56/0.63/0.70	0.15/0.24/0.35
SuperPoint (e2e) + NG-RANSAC (init)	0.58/0.64/0.70	0.15/0.24/0.36
SuperPoint (e2e) + NG-RANSAC (e2e)	0.59/0.65/0.71	0.15/0.24/0.35

Figure 3. **Relative Pose Estimation. a)** AUC of the relative pose error using a RANSAC estimator for the essential matrix. Results of RootSIFT as reported in [10], results of LIFT as reported in [59]. **b)** AUC using NG-RANSAC [10] as robust estimator. **c)** For our best result, we show the impact of training SuperPoint vs. NG-RANSAC end-to-end. *Init.* for SuperPoint means weights provided by Detone *et al*. [14], *init.* for NG-RANSAC means training according to Brachmann and Rother [10] for SuperPoint. We show results worse than the RootSIFT baseline in red, and results better than or equal to RootSIFT in green.

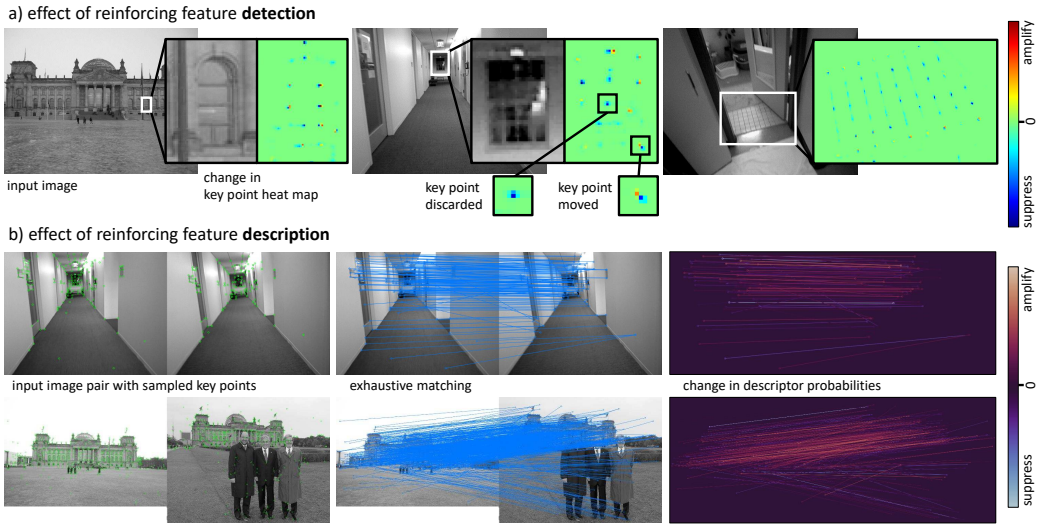


Figure 4. **Effect of Training. a)** We visualize the *difference* in key point heat maps predicted by SuperPoint before and after our end-to-end training. Key points which appear blue were discarded, key points with a gradient from blue to red were moved. **b)** We create a fixed set of matches using (initial) SuperPoint, and visualize the difference in matching probabilities for these matches before and after our end-to-end training. The probability of red matches increased by reducing their descriptor distance relative to all other matches.

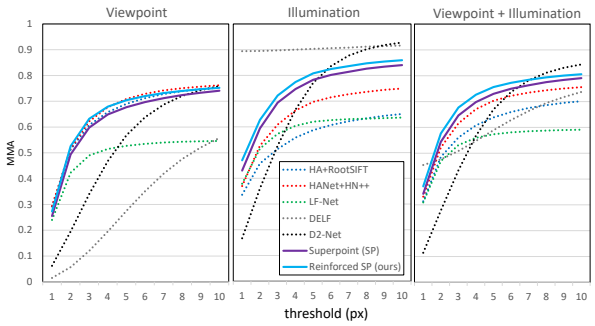
ing the descriptor distance for wrong matches. Quantitative analysis confirms these observations, see Table 1. While the number of key points reduces after end-to-end training, the overall matching quality increases, measured as the ratio of estimated inliers, and ratio of ground truth inliers.

4.2. Low-Level Matching Accuracy

We investigate the effect of our training scheme on low-level matching scores. Therefore, we analyse the performance of Reinforced SuperPoint, trained for relative pose estimation (see previous section), on the H-Patches [3] benchmark. The benchmark consists of 116 test sequences showing images under increasing viewpoint and illumination changes. We adhere to the evaluation protocol of Dusmanu *et al*. [16]. That is, we find key points and matches between image pairs of a sequence, accepting only matches of mutual nearest neighbours between two images. We calculate the reprojection error of each match using the ground truth homography. We measure the average percentage of correct matches for thresholds ranging from 1px to 10px for the reprojection error. We compare to a RootSIFT [1] baseline with a hessian affine detector [28] (denoted *HA+RootSIFT*) and several learned detectors, namely HardNet++ [29] with learned affine normalization [30] (denoted *HANet+HN++*), LF-Net [36], DELF [35] and D2-Net [16]. The original SuperPoint [14] beats all competitors in terms of AUC when combining illumination and viewpoint sequences. In particular, SuperPoint significantly exceeds the matching accuracy of RootSIFT on H-Patches, although RootSIFT outperforms SuperPoint in the task of relative pose estimation. This confirms that low-level matching accuracy does not necessarily translate to accuracy in a high-level vision task, see our earlier discussion. As for Reinforced SuperPoint, we observe an increased matching accuracy compared to SuperPoint, due to having fewer but more reliable and precise key points.

4.3. Structure-from-Motion

We evaluate the performance of Reinforced SuperPoint, trained for relative pose estimation, in a structure-from-



	Viewpoint		Illumination		Viewpoint+Illumination	
	AUC @5px	AUC @10px	AUC @5px	AUC @10px	AUC @5px	AUC @10px
HA+RootSIFT	55.2%	64.4%	49.1%	56.1%	52.1%	60.2%
HANet+HN++	**56.4%**	**65.6%**	57.3%	65.4%	56.9%	65.5%
LF-Net	43.9%	49.0%	53.8%	58.5%	48.9%	53.8%
DELF	13.2%	29.9%	**89.8%**	**90.5%**	51.5%	60.2%
D2-Net	32.7%	51.8%	49.9%	69.5%	41.3%	60.7%
Superpoint (SP)	53.5%	62.8%	65.0%	73.6%	59.2%	68.2%
Reinforced SP (ours)	56.3%	65.1%	68.0%	76.2%	**62.2%**	**70.6%**

Figure 5. **Evaluation on H-Patches [3]. Left:** We show the mean matching accuracy for SuperPoint before and after being trained for relative pose estimation. Results of competitors as reported in [16]. **Right:** Area under the curve (AUC) for the plots on the left.

	Kps	Matches	Inliers	GT Inl.
Outdoors				
SuperPoint (SP)	1993	1008	24.8%	21.9%
Reinf. SP (ours)	1892	955	28.4%	25.3%
Indoors				
SuperPoint (SP)	1247	603	13.4%	9.6%
Reinf. SP (ours)	520	262	16.4%	11.1%

Table 1. Average number of key points and matches found by SuperPoint before and after our training. We also report the estimated ratio of inliers, and the ground truth ratio of inliers.

Dataset	Method	# Sparse Points	Track Len.	Repr. Error
Fountain (11 img.)	DSP-SIFT	15k	4.79	**0.41**
	GeoDesc	17k	**4.99**	0.46
	SuperPoint	**31k**	4.75	0.97
	Reinf. SP	9k	4.86	0.87
Herzjesu (8 img.)	DSP-SIFT	8k	4.22	**0.46**
	GeoDesc	9k	**4.34**	0.55
	SuperPoint	**21k**	4.10	0.95
	Reinf. SP	7k	4.32	0.82
South Building (128 img.)	DSP-SIFT	113k	5.92	**0.58**
	GeoDesc	**170k**	5.21	0.64
	SuperPoint	160k	7.83	0.92
	Reinf. SP	102k	**7.86**	0.88

Table 2. Effect of our end-to-end training on a structure-from-motion benchmark. *Reinf. SP* denotes SuperPoint after being trained for relative pose estimation. Reprojection error is in px.

motion (SfM) task. We follow the protocol of the SfM benchmark of Schönberger *et al*. [46]. We select three of the smaller scenes from the benchmark, and extract key points and matches using SuperPoint and Reinforced SuperPoint. We create a sparse SfM reconstruction using COLMAP [45], and report the number of reconstructed 3D points, the average track length of features (indicating feature stability across views), and the average reprojection error (indicating key point precision). We report our results in Table 2, and confirm the findings of our previous experiments. While the number of key points reduces, the matching quality increases, as measured by track length and reprojection error. For reference, we also show results for DSP-SIFT [15] the best of all SIFT variants on the benchmark [46], and GeoDesc [27], a learned descriptor which achieves state-of-the-art results on the benchmark. Note that SuperPoint only provides pixel-accurate key point locations, compared to the sub-pixel accuracy of DSP-SIFT and GeoDesc. Hence, the reprojection error of SuperPoint is higher.

5. Conclusion

We have presented a new methodology for end-to-end training of feature detection and description which includes key point selection, feature matching and robust model estimation. We applied our approach to the task of relative pose estimation between two images. We observe that our end-to-end training increases the pose estimation accuracy of a state-of-the-art feature detector by removing unreliable key points, and refining the locations of remaining key points. We require a good initialization of the network, which might have a limiting effect in training. In particular, we observe that the network rarely discovers new key points. Key point locations with very low initial probability will never be selected, and cannot be reinforced. In future work, we could combine our training schema with importance sampling, for biased sampling of interesting locations.

Acknowledgements: This project has received funding from the European Social Fund (ESF) and Free State of Saxony under SePIA grant 100299506, DFG Cluster of Excellence CeTI (EXC2050/1 Project ID 390696704), DFG grant 389792660 as part of TRR 248, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 647769), and DFG grant COVMAP: Intelligente Karten mittels gemeinsamer GPS- und Videodatenanalyse (RO 4804/2-1). The computations were performed on an HPC Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

References

[1] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013.

[3] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, 2017.

[4] V. Balntas, S. Li, and V. A. Prisacariu. Relocnet: Continuous metric learning relocalisation using neural nets. In *ECCV*, 2018.

[5] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk. Key.Net: Keypoint detection by handcrafted and learned CNN filters. In *ICCV*, 2019.

[6] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *ECCV*, 2006.

[7] A. Benbihi, M. Geist, and C. Pradalier. ELF: Embedded localisation of features in pre-trained CNN. In *ICCV*, 2019.

[8] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. DSAC: Differentiable RANSAC for camera localization. In *CVPR*, 2017.

[9] E. Brachmann and C. Rother. Learning less is more - 6D camera localization via 3D surface regression. In *CVPR*, 2018.

[10] E. Brachmann and C. Rother. Neural-Guided RANSAC: Learning where to sample model hypotheses. In *ICCV*, 2019.

[11] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[12] T. Cieslewski, K. G. Derpanis, and D. Scaramuzza. SIPs: Succinct interest points from unsupervised inlierness probability learning. In *3DV*, 2019.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[14] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPR Workshops*, 2018.

[15] J. Dong and S. Soatto. Domain-size pooling in local descriptors: Dsp-sift. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[16] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-Net: A trainable CNN for joint detection and description of local features. In *CVPR*, 2019.

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

[18] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015.

[19] C. Harris and M. Stephens. A combined corner and edge detector. In *AVC*, 1988.

[20] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[21] J. Heinly, J. L. Schönberger, E. Dunn, and J.-M. Frahm. Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In *CVPR*, 2015.

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

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[24] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010.

[25] Y. Li, N. Snavely, D. P. Huttenlocher, and P. Fua. Worldwide pose estimation using 3D point clouds. In *ECCV*, 2012.

[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[27] Z. Luo, T. Shen, L. Zhou, S. Zhu, R. Zhang, Y. Yao, T. Fang, and L. Quan. GeoDesc: Learning local descriptors by integrating geometry constraints. In *ECCV*, 2018.

[28] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 2004.

[29] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbors margins: Local descriptor learning loss. In *NeurIPS*, 2017.

[30] D. Mishkin, F. Radenovic, and J. Matas. Repeatability is not enough: Learning affine regions via discriminability. In *ECCV*, 2018.

[31] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *T-RO*, 2015.

[32] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *T-RO*, 2017.

[33] D. Nistér. An efficient solution to the five-point relative pose problem. *TPAMI*, 2004.

[34] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[35] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, 2017.

[36] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. LF-Net: Learning local features from images. In *NeurIPS*, 2018.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NeurIPS-W*, 2017.

[38] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[39] J. Revaud, P. Weinzaepfel, C. R. de Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019.

[40] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011.

[41] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *TPAMI*, 2016.

[42] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. Benchmarking 6DOF outdoor visual localization in changing conditions. In *CVPR*, 2018.

[43] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. Understanding the limitations of CNN-based absolute camera pose regression. In *CVPR*, 2019.

[44] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007.

[45] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016.

[46] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *CVPR*, 2017.

[47] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, 2008.

[48] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[49] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi. Discovery of latent 3D keypoints via end-to-end geometric reasoning. In *NeurIPS*, 2018.

[50] Y. Tian, B. Fan, and F. Wu. L2-Net: Deep learning of discriminative patch descriptor in Euclidean space. In *CVPR*, 2017.

[51] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas. SOSNet: Second order similarity regularization for local descriptor learning. In *CVPR*, 2019.

[52] C. Toft, E. Stenborg, L. Hammarstrand, L. Brynte, M. Pollefeys, T. Sattler, and F. Kahl. Semantic match consistency for long-term visual localization. In *ECCV*, 2018.

[53] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *CVPR*, 2017.

[54] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *BMVC*, 2016.

[55] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

[56] C. Wu. Towards linear-time incremental structure from motion. In *3DV*, 2013.

[57] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*, 2013.

[58] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned invariant feature transform. In *ECCV*, 2016.

[59] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. Learning to find good correspondences. In *CVPR*, 2018.