

Learning to Find Good Correspondences

Kwang Moo Yi^{1,*} Eduard Trulls^{2,*} Yuki Ono³ Vincent Lepetit⁴ Mathieu Salzmann² Pascal Fua²

¹Visual Computing Group, University of Victoria ²Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne

³Sony Imaging Products & Solutions Inc. ⁴Institute for Computer Graphics and Vision, Graz University of Technology
 kyi@uvic.ca, {firstname.lastname}@epfl.ch, yuki.ono@sony.com, lepetit@icg.tugraz.at

Abstract

We develop a deep architecture to learn to find good correspondences for wide-baseline stereo. Given a set of putative sparse matches and the camera intrinsics, we train our network in an end-to-end fashion to label the correspondences as inliers or outliers, while simultaneously using them to recover the relative pose, as encoded by the essential matrix. Our architecture is based on a multi-layer perceptron operating on pixel coordinates rather than directly on the image, and is thus simple and small. We introduce a novel normalization technique, called Context Normalization, which allows us to process each data point separately while embedding global information in it, and also makes the network invariant to the order of the correspondences. Our experiments on multiple challenging datasets demonstrate that our method is able to drastically improve the state of the art with little training data.

1. Introduction

Recovering the relative camera motion between two images is one of the most basic tasks in Computer Vision, and a key component of wide-baseline stereo and Structure from Motion (SfM) pipelines. However, it remains a difficult problem when dealing with wide baselines, repetitive structures, and illumination changes, as depicted by Fig. 1. Most algorithms rely on sparse keypoints [19, 2, 24] to establish an initial set of correspondences across images, then try to find a subset of reliable matches—inliers—which conform to a given geometric model, and use them to recover the pose [11]. They rely on combinations of well-established techniques, such as SIFT [19], RANSAC [10], and the 8-point algorithm [18], which have been in use for decades.

With the advent of deep learning, there has been a push towards reformulating local feature extraction using neural networks [35, 26]. However, while these algorithms outper-

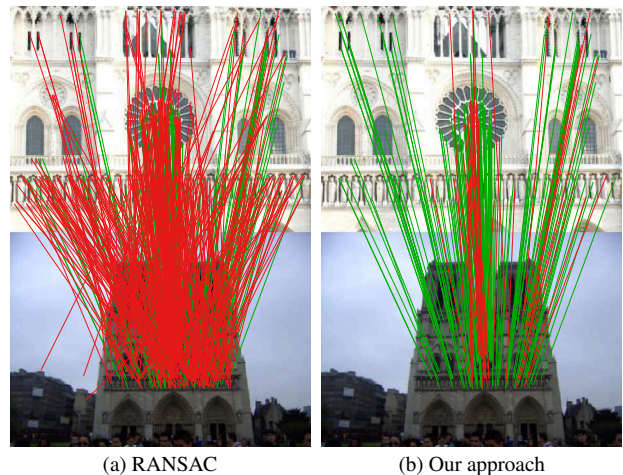


Figure 1. We extract 2k SIFT keypoints from a challenging image pair and display the inlier matches found with RANSAC (left) and our approach (right). We draw them in green if they conform to the ground-truth epipolar geometry, and in red otherwise.

form earlier ones on point-matching benchmarks, incorporating them into pose estimation pipelines may not necessarily translate into a performance increase, as indicated by two recent studies [25, 3]. This suggests that the limiting factor may not lie in *establishing* the correspondences as much as in *choosing* those that are best to recover the pose.

This problem has received comparatively little attention and most algorithms still rely on non-differentiable hand-crafted techniques to solve it. DSAC [4] is the only recent attempt we know of to tackle sparse outlier rejection in a differentiable manner. However, this method is designed to mimic RANSAC rather than outperform it. Furthermore, it is specific to 3D to 2D correspondences, rather than stereo.

By contrast, we propose a novel approach to finding geometrically consistent correspondences with a deep network. Given feature points in both images, and potential correspondences between them, we train a deep network that simultaneously solves a classification problem—retain or reject each correspondence—and a regression problem—retrieving the camera motion—by exploiting epipolar constraints. To do so we introduce a *differentiable* way to com-

*First two authors contributed equally. K.M. Yi was at EPFL during the development of this project. This work was partially supported by EU FP7 project MAGELLAN under grant ICT-FP7-611526 and by systems supplied by Compute Canada.

pute the pose through a simple weighted reformulation of the 8-point algorithm, with the weights predicting the likelihood of a correspondence being correct. In practice, we assume the camera intrinsics to be known, which is often true because they are stored in the image meta-data, and express camera motion in terms of the essential matrix [11].

Our method has the following advantages: (i) it can double the performance of the state of the art; (ii) being keypoint-based, it generalizes better than image-based dense methods to unseen scenes, which we demonstrate with a *single model* that outperforms current methods on drastically different indoors and outdoors datasets; (iii) it requires only weak supervision through essential matrices for training; (iv) it can work effectively with very little training data, *e.g.*, we can still outperform the state of the art on challenging outdoor scenes with only 59 training images.

Traditional handcrafted methods. The traditional approach for estimating the relative camera motion between two images is to use sparse keypoints, such as SIFT [19], to establish an initial set of correspondences, and reject outliers with RANSAC [10], using the 5-point algorithm [20] or the 8-point algorithm [18] to retrieve the essential matrix.

A major drawback of these approaches is that they rely on small subsets of the data to generate the hypotheses, *e.g.*, the 5-point algorithm considers only five correspondences at a time. This is sub-optimal, as image pairs with large baselines and imaging changes will contain a large percentage of outliers, thus making most of the hypotheses useless.

Figure 2. Our deep network takes N correspondences between two 2D points ($4 \times N$ values) as input, and produces one weight for each correspondence, encoding its likelihood to be an inlier. Each correspondence is processed independently by weight-sharing Perceptrons (P), rendering the network invariant to permutations of the input. Global information is embedded by Context Normalization, a novel technique we detail in Section 3.2.

Learning-based methods. Solving image correspondence with deep networks has received a lot of interest recently. In contrast with traditional methods, many of these new techniques [32, 31, 37, 36] are dense, using the raw image as input. They have shown promising results on constrained stereo problems, such as matching two frames from a video sequence, but the general problem remains far from solved. As evidenced by our experiments, this approach can be harmful on scenes with occlusions or large baselines, which are commonplace in photo-tourism applications.

In [9], a dual architecture with separate networks for extracting sparse keypoints and forming the correspondences, assuming a homography model, was proposed. Because of its requirement for ground-truth annotations of object corners, this work was only demonstrated on synthetic, non-realistic images. By contrast, our method only requires essential matrices for training and thus, as demonstrated by our experiments, is effective in real-world scenarios.

Our goal is to establish valid and geometrically consistent correspondences across images and use them to retrieve the camera motion. We rely on local features, which can be

unreliable and require outlier rejection. Traditional methods approach this problem by iteratively sampling small subsets of matches, which makes it difficult to exploit the global context. By contrast, we develop a deep network that can leverage all the available information in a single shot.

Specifically, we extract local features over two images, create an overcomplete set of putative correspondences, and feed them to the network depicted by Fig. 2. It assigns a weight to each correspondence, which encodes their likelihood of being an inlier. We further use it to set the influence of each correspondence in a weighted reformulation of the eight-point algorithm [18]. In other words, our method performs joint inlier/outlier classification and regression to the essential matrix. The relative rotation and translation between cameras can then be recovered with a singular value decomposition of the resulting essential matrix [11].

In the remainder of this section, we formalize the problem and describe the architecture of Fig. 2. We then reformulate the eight-point algorithm for weighted correspondences, and discuss our learning strategy with a hybrid loss.

3.1. Problem Formulation

Formally, let us consider a pair of images $(\mathbf{I}, \mathbf{I}')$. We extract local features $(\mathbf{k}_i, \mathbf{f}_i)_{i \in [1, N]}$ from image \mathbf{I} , where $\mathbf{k}_i = (u_i, v_i)$ contains keypoint locations, and \mathbf{f}_i is a vector encoding a descriptor for the image patch centered at (u_i, v_i) . The keypoint orientation and scale are used to extract these descriptors and discarded afterwards. Similarly, we obtain $(\mathbf{k}'_j, \mathbf{f}'_j)_{j \in [1, N']}$ from \mathbf{I}' , keeping $N = N'$ for simplicity. Our formulation is thus amenable to traditional features such as SIFT [19] or newer ones such as LIFT [35].

We then generate a list of N putative correspondences by matching each keypoint in \mathbf{I} to the nearest one in \mathbf{I}' based on the descriptor distances. We denote this list as

$$\begin{aligned} \mathbf{x} &= [\mathbf{q}_1, \dots, \mathbf{q}_N], \\ \mathbf{q}_i &= [u_i, v_i, u'_i, v'_i]. \end{aligned} \quad (1)$$

More complex strategies [19, 5] could be used, but we found this one sufficient. As many traditional algorithms, we use the camera intrinsics to normalize the coordinates to $[-1, 1]$, which makes the optimization numerically better-behaved [11]. We discard the descriptors, so that the list of N location quadruplets is the only input to our network.

As will be discussed in Section 3.3, it is straightforward to extend the eight-point algorithm to take as input a set of correspondences with accompanying weights

$$\mathbf{w} = [w_1, \dots, w_N], \quad (2)$$

where $w_i \in [0, 1]$ is the score assigned to correspondence \mathbf{q}_i , with $w_i = 0$ standing for \mathbf{q}_i being an outlier. Then, let g be a function that takes as input \mathbf{x} and \mathbf{w} and returns the essential matrix \mathbf{E} . Now, given a set of P image pairs $(\mathbf{I}_k, \mathbf{I}'_k)_{k \in [1, P]}$ and their corresponding essential matrices \mathbf{E}_k , we extract the set of correspondences \mathbf{x}_k of Eq. (1),

and our problem becomes that of designing a deep network that encodes a mapping f with parameters Φ , such that

$$\begin{aligned} \forall k, 1 \leq k \leq P, \mathbf{w}_k &= f_\Phi(\mathbf{x}_k), \\ \mathbf{E}_k &\approx g(\mathbf{x}_k, \mathbf{w}_k). \end{aligned} \quad (3)$$

3.2. Network Architecture

We now describe the network that implements the mapping of Eq. (3). Since the order of the correspondences is arbitrary, permuting \mathbf{x}_k should result in an equivalent permutation of $\mathbf{w}_k = f_\Phi(\mathbf{x}_k)$. To achieve this, inspired by PointNet [21], a deep network designed to operate on unordered 3D point clouds, we exploit Multi-Layer Perceptrons (MLP). As MLPs operate on *individual* correspondences, unlike convolutional or dense networks, incorporating information from other perceptrons, *i.e.*, the context, is *indispensable* for it to work properly. The distinguishing factor between PointNet and our method is how this is done.

In PointNet, the point-wise features are explicitly combined by a separate network that generates a global context feature, which is then concatenated to each point-wise output. By contrast, as shown in Fig. 2, we introduce a simple strategy to normalize the feature maps according to their distribution, after every perceptron. This lets us process each correspondence separately while framing it in the global context defined by the rest, which encodes camera motion and scene geometry. We call this non-parametric operation *Context Normalization* (CN).

Formally, let $\mathbf{o}_i^l \in \mathbb{R}^{C^l}$ be the output of layer l for correspondence i , where C^l is the number of neurons in l . We take the normalized version of \mathbf{o}_i^l to be

$$\text{CN}(\mathbf{o}_i^l) = \frac{(\mathbf{o}_i^l - \boldsymbol{\mu}^l)}{\boldsymbol{\sigma}^l}, \quad (4)$$

where

$$\boldsymbol{\mu}^l = \frac{1}{N} \sum_{i=1}^N \mathbf{o}_i^l, \quad \boldsymbol{\sigma}^l = \sqrt{\frac{1}{N} \sum_{i=1}^N (\mathbf{o}_i^l - \boldsymbol{\mu}^l)^2}. \quad (5)$$

This operation is mechanically similar to other normalization techniques [14, 1, 30], but is applied to a different dimension and plays a different role. We normalize each perceptron's output across correspondences, but separately for each image pair. This allows the distribution of the feature maps to encode scene geometry and camera motion, embedding contextual information into context-agnostic MLPs.

By contrast, the other normalization techniques primarily focus on convergence speed, with little impact on how the networks operate. Batch Normalization [14] normalizes the input to each neuron over a mini-batch, so that it follows a consistent distribution while training. Layer Normalization [1] transposes this operation to channels, thus being independent on the number of observations for each neuron. They do *not*, however, add contextual information to the input. Batch Normalization assumes that every sample

follows the same global statistics, and reduces to subtracting and dividing by fixed scalar values at test time. Layer Normalization is applied independently for each neuron, *i.e.*, it considers the features for one correspondence at a time.

More closely related is Instance Normalization [30], which applies the same operation we do to full images to normalize their contrast, for image stylization. However, it is also limited to enhancing convergence, as in their application context is already captured by spatial convolutions, which are not amenable to the sparse data in our problem.

Note however that our technique is compatible with Batch Normalization, and we do in fact use it to speed up training. As shown in Fig. 2, our network is a 12-layer ResNet [12], where each layer contains two sequential blocks consisting of: a Perceptron with 128 neurons sharing weights for each correspondence, a Context Normalization layer, a Batch Normalization layer, and a ReLU. After the last perceptron, we apply a ReLU followed by a tanh to force the output in the range $[0, 1]$. We use this truncated tanh instead of, *e.g.*, a sigmoid, so that the network can easily output $w_i = 0$ to completely remove an outlier.

3.3. Computing the Essential Matrix

We now define the function g of Eq. (3) that estimates the essential matrix from a weighted set of correspondences. If we are to integrate it into a Deep Learning framework, it must be differentiable with respect to the weights. We first outline the 8-point algorithm [18] that tackles the unweighted scenario, and then extend it to the weighted case.

Traditional Formulation. Let \mathbf{x} be a set of N correspondences $\mathbf{q}_i = [u_i, v_i, u'_i, v'_i]$. When $N \geq 8$, the essential matrix $\mathbf{E} \in \mathbb{R}^{3 \times 3}$ can be computed by solving a singular value problem [11] as follows. We construct a matrix $\mathbf{X} \in \mathbb{R}^{N \times 9}$ each row of which is computed from a different correspondence and has the form $[uu', uv', u, vu', vv', v, u', v', 1]$, and we reorganize the coefficients of \mathbf{E} into a column vector $\text{Vec}(\mathbf{E})$. This vector has to be the unit vector that minimizes $\|\mathbf{X} \text{Vec}(\mathbf{E})\|$, or equivalently $\|\mathbf{X}^\top \mathbf{X} \text{Vec}(\mathbf{E})\|$. Therefore, $\text{Vec}(\mathbf{E})$ is the eigenvector associated to the smallest eigenvalue of $\mathbf{X}^\top \mathbf{X}$. Additionally, as the essential matrix needs to have rank 2, we find the rank-2 matrix $\hat{\mathbf{E}}$ that minimizes the Frobenius norm $\|\mathbf{E} - \hat{\mathbf{E}}\|_F$ [11].

Weighted reformulation. In practice, the 8-point algorithm can be severely affected by outliers and is used in conjunction with an outlier rejection scheme. In our case, given the weights \mathbf{w} produced by the network, we can simply minimize $\|\mathbf{X}^\top \text{diag}(\mathbf{w}) \mathbf{X} \text{Vec}(\mathbf{E})\|$ instead of $\|\mathbf{X}^\top \mathbf{X} \text{Vec}(\mathbf{E})\|$, where diag is the diagonalization operator. This amounts to giving weight w_i to the i -th row in \mathbf{X} , representing the contribution of the i -th correspondence.

Since $\mathbf{X}^\top \text{diag}(\mathbf{w}) \mathbf{X}$ is symmetric, the function g that computes $\text{Vec}(\mathbf{E})$, and therefore \mathbf{E} , from \mathbf{X} is a self-adjoint

eigendecomposition, which has differentiable closed-form solution with respect to \mathbf{w} [15]. Note that $g(\mathbf{x}, \mathbf{w})$ only depends on the inliers because any correspondence with zero weight has strictly no influence on the result. To compute the final essential matrix $\hat{\mathbf{E}}$, we follow the same procedure as in the unweighted case, which is also differentiable.

3.4. Learning with Classification and Regression

We train the network f_Φ with a hybrid loss function

$$\mathcal{L}(\Phi) = \sum_{k=1}^P (\alpha \mathcal{L}_x(\Phi, \mathbf{x}_k) + \beta \mathcal{L}_e(\Phi, \mathbf{x}_k)), \quad (6)$$

where Φ are the network parameters and \mathbf{x}_k is the set of putative correspondences for image pair k . \mathcal{L}_x is a classification loss aggregated over each individual correspondence, and \mathcal{L}_e is a regression loss over the essential matrix, obtained from the weighted 8-point algorithm of Section 3.3 with the weights produced by the network. α and β are the hyper-parameters weighing the two loss terms.

Classification loss \mathcal{L}_x : rejecting outliers. Given a set of N putative correspondences \mathbf{x}_k and their respective labels $\mathbf{y}_k = [y_k^1, \dots, y_k^N]$, where $y_k^i \in \{0, 1\}$ and $y_k^i = 1$ denotes that the i th correspondence is an inlier, we define

$$\mathcal{L}_x(\Phi, \mathbf{x}_k) = \frac{1}{N} \sum_{i=1}^N \gamma_k^i H(y_k^i, S(o_k^i)), \quad (7)$$

where o_k^i is the linear output of the last layer for the i -th correspondence in training pair k , S is the logistic function used in conjunction with the binary cross entropy H , and γ_k^i is the per-label weight to balance positive and negative examples.

To avoid exhaustive annotation, we generate the labels \mathbf{y} by exploiting the epipolar constraint. Given a point in one image, if the corresponding point in the other image does not lie on the epipolar line, the correspondence is spurious. We can quantify this with the epipolar distance [11]

$$d(\mathbf{p}, \mathbf{E}\mathbf{p}') = \frac{\mathbf{p}'^T \mathbf{E} \mathbf{p}}{\sqrt{(\mathbf{E}\mathbf{p})_{[1]}^2 + (\mathbf{E}\mathbf{p})_{[2]}^2}}, \quad (8)$$

where $\mathbf{p} = [u, v, 1]^T$ and $\mathbf{p}' = [u', v', 1]^T$ indicate a putative correspondence between images \mathbf{I} and \mathbf{I}' in homogeneous coordinates, and $\mathbf{v}_{[j]}$ denotes the j th element of vector \mathbf{v} . In practice, we use the *symmetric epipolar distance* $d(\mathbf{p}, \mathbf{E}\mathbf{p}') + d(\mathbf{p}', \mathbf{E}^T \mathbf{p})$, with a threshold of 10^{-2} in normalized coordinates to determine valid correspondences.

Note that this is a weak supervisory signal because outliers can still have a small epipolar distance if they lie close to the epipolar line. However, a fully-supervised labeling would require annotating correspondences for every point in both images, which is prohibitively expensive, or depth data, which is not available in most scenarios. Our experiments show that, in practice, weak supervision is sufficient.



Figure 3. Matching pairs from our training sets. Left to right: ‘Buckingham’, ‘Notredame’, ‘Sacre Coeur’, ‘St. Peter’s’ and ‘Reichstag’, from [28, 13]; and ‘Brown 1’ and ‘Harvard 1’ from [34] (please see the appendix for details). Note that we crop the images for presentation purposes, but our algorithm is based on sparse correspondences and is thus not restricted in any way by image size or aspect ratio.

Regression loss \mathcal{L}_e : predicting the essential matrix.

The weak supervision applied by the classification loss proves robust in our experience, but it may still let outliers in. We propose to improve it by using the weighted set of correspondences to extract the essential matrix and penalize deviations from the ground-truth. We therefore write

$$\mathcal{L}_e(\Phi, \mathbf{x}_k) = \min \left\{ \|\mathbf{E}_k^* \pm g(\mathbf{x}_k, \mathbf{w}_k)\|^2 \right\}, \quad (9)$$

where \mathbf{E}_k^* is the ground-truth essential matrix, and g is the function defined in Section 3.3 that predicts an essential matrix from a set of weighted correspondences. We have to compute either the difference or the sum because the sign of $g(\mathbf{x}, \mathbf{w})$ can be flipped with respect to that of \mathbf{E}^* . Note that here we use the non-rank-constrained estimate of the essential matrix \mathbf{E} , instead of $\hat{\mathbf{E}}$, as defined in Section 3.3, because minimizing \mathcal{L}_e already aims to get the correct rank.

Optimization. We use Adam [16] to minimize the loss $\mathcal{L}(\Phi)$, with a learning rate of 10^{-4} and mini-batches of 32 image pairs. The classification loss can train accurate models by itself, and we observed that using the regression loss early on can actually harm performance. We find empirically that setting $\alpha = 1$ and enabling the regression loss after 20k batches with $\beta = 0.1$ works well. This allows us to increase relative performance by 5 to 20%.

3.5. RANSAC at Test Time

The loss function must be differentiable with respect to the network weights. However, this requirement disappears at test time. We can thus apply RANSAC on the correspondences labeled as inliers by our network to weed out any remaining outliers. We will show that this performs much better stand-alone RANSAC, which confirms our intuition that sampling is a sub-optimal way to approach “needle-in-the-haystack” scenarios with a large ratio of outliers.

4. Experimental Results

We first present the datasets and evaluation protocols, and then justify our implementation choices. Finally, we benchmark our approach against state-of-the-art techniques.

4.1. Datasets

Sparse feature matching is particularly effective in outdoor scenes, and we will see that this is where our approach shines. By contrast, keypoint-based methods are less suitable for indoor scenes. Nevertheless, recent techniques have tackled this problem, and we show that our approach still outperforms the state of the art in this scenario.

Outdoor scenes. To evaluate our method in outdoor settings, we aim to leverage many images featuring similar scenes seen from different viewpoints. As such, photo-tourism datasets are natural candidates. We rely on Yahoo’s YFCC100M dataset [28], a collection of 100 million publicly accessible Flickr images with accompanying metadata, which were later curated [13] into 72 image collections suitable for Structure from Motion (SfM). We pick five sequences, depicted in Fig. 3. We use VisualSfM [33] to recover the camera poses and generate the ground-truth.

We also consider the ‘Fountain’ and ‘HerzJesu’ sequences from [27]. They are very small and we only use them for testing, to show that our networks trained on photo-tourism datasets can successfully generalize.

Indoor scenes. We use the SUN3D dataset [34], which comprises a series of indoor videos captured with a Kinect, with 3D reconstructions. They feature office-like scenes with few distinctive features, many repetitive elements, and substantial self-occlusions, which makes them extremely challenging for sparse keypoint methods. We select 9 sequences for training and testing, and use 15 sequences previously chosen by [31] for testing only, to provide a fair comparison. We subsample the videos by a factor of 10.

For every sequence we train on, we randomly split the images into disjoint subsets for training (60%), validation (20%) and test (20%). To select valid image pairs on the ‘Outdoors’ subset, we sample two images randomly and check if they have a minimum number of 3D points in common from the SfM reconstruction, indicating that the problem is solvable. For the ‘Indoors’ set, we use the depth maps to determine visibility. We use 1k image pairs for validation and testing, and as many as possible for training.

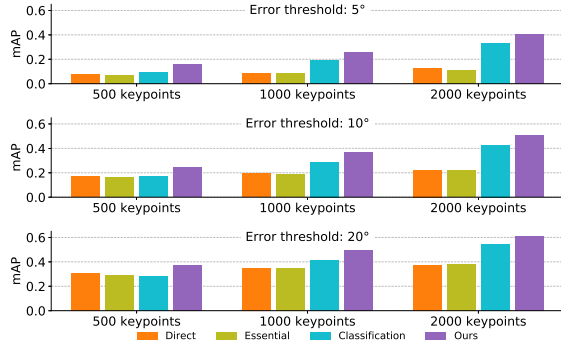


Figure 4. mAP for multiple error thresholds and number of keypoints, using the four optimization strategies of Section 4.3.1.

4.2. Evaluation Protocols

Keypoint-based methods. They include well-established algorithms, RANSAC [10], MLESAC [29], and LMEDS [23], as well as the very recent GMS [3]. For GMS, we incorporate an additional RANSAC step as described for our method in Section 3.5, which we empirically found mandatory to obtain good performance.

Note that GMS operates with a large (10k) pool of ORB features [24], and thus behaves similarly to dense methods. For all others, ours included, we evaluate both SIFT [19] and LIFT [35] features. For SIFT we use the OpenCV library, and for LIFT the publicly available models, which were trained on photo-tourism data different from ours.

Dense methods. We consider G3DR [36] and DeMoN [31]. For G3DR, we implement their architecture and train it using only the pose component of their loss function, as they argue that pose estimation is more accurate without the classification loss [36]. For DeMoN, we use the publicly available models, which were trained on SUN3D sequences and on SfM reconstructions of outdoors sets.

Metrics. Given two images, it is possible to estimate rotation exactly—in theory—and translation only up to a scale factor [11]. We thus use the angular difference between the estimated and ground-truth vectors, *i.e.*, the closest arc distance, in degrees, for both, as our error metric. We do so as follows. First, we generate a curve by classifying each pose as accurate or not, *i.e.* we compute the precision, given a threshold (0 to 180°), and build a normalized cumulative curve as in [8, 3]. Second, we compute the area under this curve (AUC) up to a maximum threshold of 5, 10 or 20°, because after a point it does not matter how inaccurate pose estimates are. As the curve measures precision, its AUC is equivalent to mean average precision (mAP). We apply the same threshold over rotation and translation, for simplicity.

4.3. Ablation Study

At the heart of our approach are two key ideas: to label the correspondences as inliers or outliers while simul-

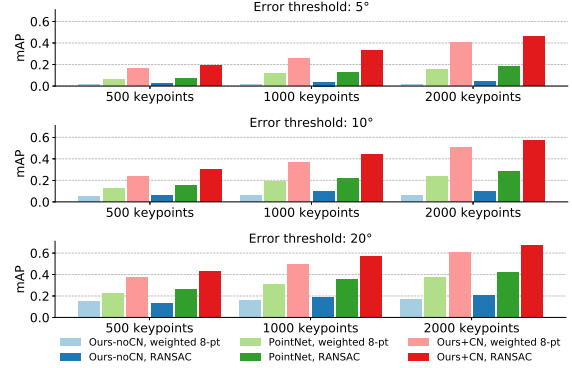


Figure 5. mAP for multiple error thresholds and number of keypoints, comparing PointNet with our approach, with and without Context Normalization (CN), as explained in Section 4.3.2.

taneously using them to estimate the essential matrix, and to leverage a PointNet-like architecture, replacing its global context feature with our Context Normalization, introduced in Section 3.2. We examine the impact of these two choices.

4.3.1 Hybrid Approach vs Classification vs Regression

To show the benefits of our hybrid approach, we compare four different settings of our algorithm:

- **Ours:** Our complete formulation with α and β in Eq. (6) set to 1 and 0 initially and then to 1 and 0.1 after 20k batches. In other words, we first seek to assign reasonable weights to the correspondences before also trying to make the essential matrix accurate.
- **Essential:** We disable the classification loss by setting $\alpha = 0$ and $\beta = 1$ in Eq. (6). This amounts to direct regression from correspondences to the essential matrix by assigning weights to the correspondences and performing least-squares fitting.
- **Classification:** We disable the essential loss, setting $\alpha = 1$ and $\beta = 0$ in Eq. (6). The network then tries to classify correspondences into inliers and outliers.
- **Direct:** We regress the essential matrix directly by average-pooling the output of our last ResNet block and adding a fully-connected layer that outputs nine values, which we take to be the output of g in Eq. (9). In other words, we directly predict the coefficients of the essential matrix without resorting to a weighted version of the 8-point algorithm, as in **Essential**.

We ran these four variants on the ‘Sacre Coeur’ collection from YFCC100M, using SIFT keypoints, and report the results in Fig. 4. **Essential** and **Direct**, which do direct regression without classification, perform worse than **Classification**, as long as the number of keypoints is sufficient. **Ours** outperforms all the others by combining both classification and regression, by a margin of 12-24%. Note that

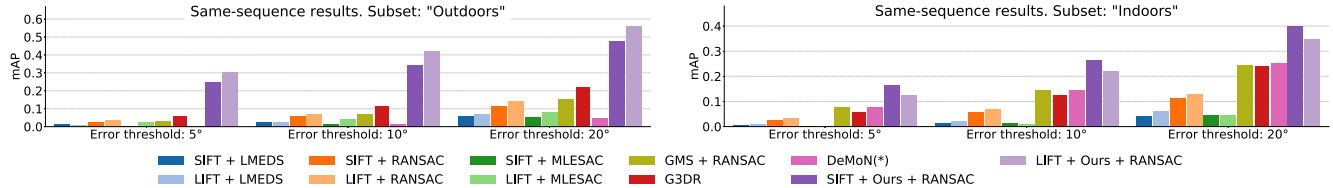


Figure 6. Results on known scenes: ‘Outdoors’ (left) and ‘Indoors’ (right). We split the images for each set with 60% for training, 20% for validation and 20% for testing, and report results over 1000 image pairs (or every possible combination for smaller sets) for the test split. We mark DeMoN with an asterisk as we use the pre-trained models provided by the authors.

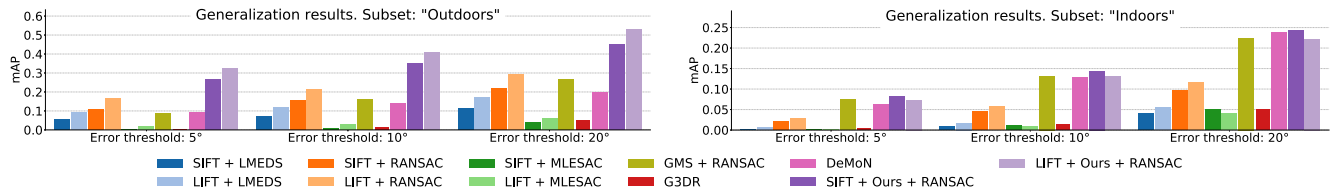


Figure 7. Generalization results with models trained and tested on different scenes: ‘Outdoors’ (left) and ‘Indoors’ (right). We train a *single model* combining one sequence from the ‘Indoors’ set and one from the ‘Outdoors’ set. For ‘Outdoors’, we test on every other available sequence and average the results. For ‘Indoors’, we test on the 15 sequences chosen by [31] for this purpose and average the results.

the difference is larger for smaller error thresholds, suggesting that both **Essential** and **Direct** are learning the general trend of the dataset without providing truly accurate poses.

4.3.2 Context Normalization vs Context Feature

For comparison purposes, we reformulate our approach using the PointNet architecture [21]. The specificity of PointNet is that it extracts a global context feature from the input features, which is then concatenated to each individual feature and re-injected into the network. By contrast, we simply embed it into each point-wise feature through Context Normalization. For a fair comparison, we use an architecture of similar complexity to ours, with 12 MLPs each for the extraction of point features, global features, and the final output. We replace max pooling by average pooling to extract the global feature, which gave better results. We refer to this architecture as **PointNet**, and train it using the hybrid loss of Eq. (6), as we do for **Ours**. For completeness, we also try our approach without Context Normalization.

We report results on the ‘Sacre Coeur’ sequence in Fig. 5. In all three cases, we present results without and with the final RANSAC stage of Section 3.5. As expected, our approach without Context Normalization performs poorly, whereas, with it, it does much better than **PointNet**.

4.4. Post-processing with RANSAC

Note that traditional keypoint-based methods do not work at all without RANSAC. By contrast, our network outperforms RANSAC in a *single forward pass*, using only the 8-point algorithm. However, at test time we can drop the differentiability constraints, as explained in Section 3.5, and apply RANSAC on our surviving inliers to further boost performance. We do this for the remainder of this paper.

Moreover, our method is much faster than stand-alone

RANSAC because it can throw away most bad matches in a single step. Given 2k matches, a forward pass through our network takes 13 ms on GPU (or 25 ms on CPU) and returns on average ~300 inliers—RANSAC then removes a further ~100 matches in 9 ms. By contrast, a RANSAC loop with the full 2k matches needs 373 ms and returns ~300 inliers. Our method, including RANSAC, is thus not only more accurate but also 17x faster (please refer to the supplementary material for comprehensive numbers).

4.5. Comparison to the Baselines

We now turn to comparing our method against the baselines discussed above. Given the results presented in Section 4.3, we use **Ours** with context normalization turned on, and apply RANSAC post-processing to *all* of the sparse methods. We average the results over multiple sets—please refer to the supplementary material for additional results.

We first consider networks trained and tested on images from the same scene, and then on different scenes. Note that we split the images into disjoint sets for training and testing, as explained in Section 4.1, so there is no overlap between both sets in either case. For keypoint-based methods other than GMS, we consider both SIFT and LIFT.

4.5.1 Performance on Known Scenes

Consider the 5 collections of outdoor images from YFCC100M and the 9 indoor sequences from SUN3D, described in Section 4.1 and depicted by Fig. 3. We report our comparative results in Figs. 6 and 8. The training and test sets are always disjoint, but drawn from the *same* collection.

For the five ‘Outdoors’ collections, whose images are feature-rich, we achieve our best results using LIFT features trained on photo-tourism datasets. **Ours** then delivers an mAP that is more than twice that of previous state-of-the-

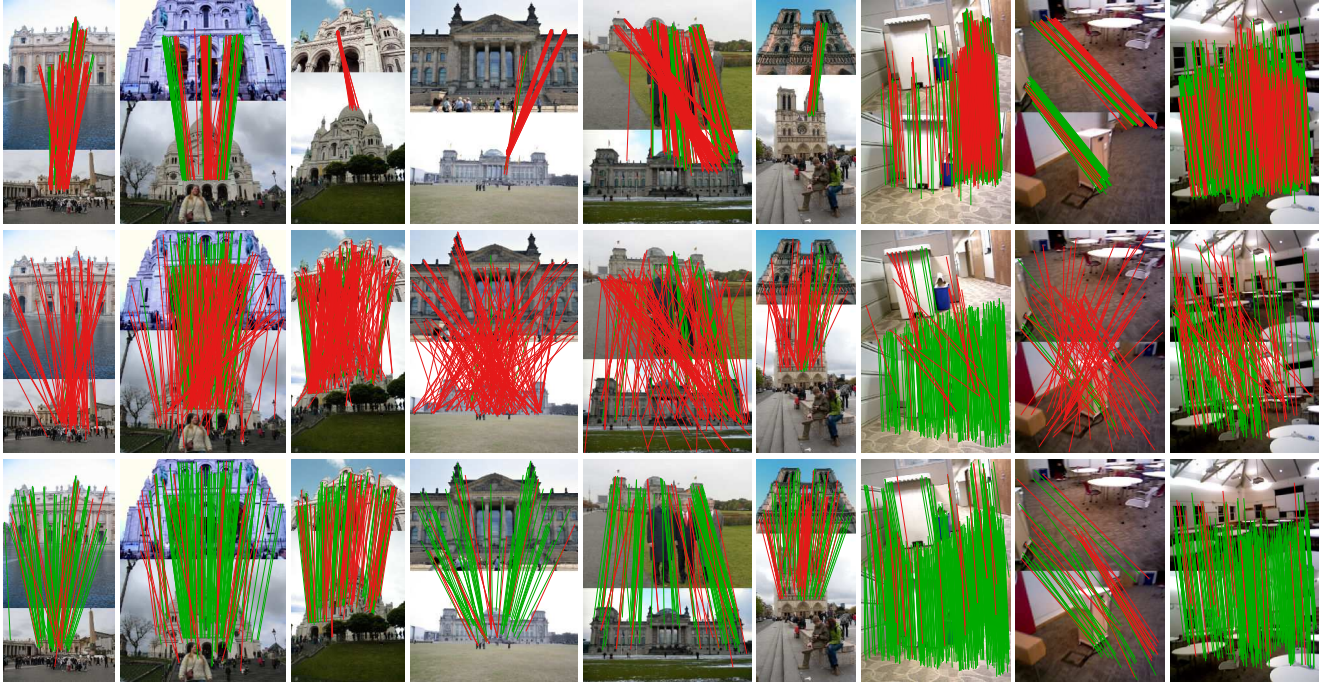


Figure 8. Matches from **(top)** GMS with 10k ORB features, **(middle)** RANSAC, and **(bottom)** our approach, with the same 2k SIFT features. Matches are in green if their symmetric epipolar distance in normalized coordinates is below 0.01, and red otherwise.

art methods. However, even when using the more popular SIFT, it still outperforms the other methods. Furthermore, the gap grows wider for small error thresholds, indicating that our approach performs better the more strict we are.

By contrast, the two ‘Indoors’ collections feature poorly textured images with repetitive patterns, static environments, and consistent scales, which makes them ill-suited for keypoint-based methods and more amenable to dense methods. Nevertheless, our method significantly outperforms all compared methods, including dense ones. On these images, **Ours** performs better with SIFT than LIFT, which is consistent with the fact that LIFT was trained on photo-tourism images. Note that for the numbers reported in Fig. 6 for DeMoN, we used their pre-trained models as it is not possible to re-train it due to the lack of depth information for the ‘Outdoors’ data. Note also that their training sets include not one but many sequences from SUN3D [31].

4.5.2 Generalization to Unknown Scenes

Here, we evaluate the generalization capability of our method by training and testing on different scenes. Fig. 7 reports results for the model trained with the combination of the ‘Saint Peter’s’ sequence from ‘Outdoors’ and the ‘Brown 1’ sequence from ‘Indoors’. For ‘Outdoors’ we report the average result for all sets excluding ‘Saint Peter’s’. For ‘Indoors’, we report the average result for the 15 test-only sequences selected by [31], for fair comparison.

We outperform all baselines on ‘Outdoors’ by a significant margin. Comparing Figs. 6 and 7 shows that our

models generalize very well to unknown scenes. Note that the jump in the performance of several baselines between Figs. 6 and 7 is solely due to the addition of two easy sequences, ‘Fountain’ and ‘Herzjesu’, for testing in Fig. 7.

On ‘Indoors’ we outperform the state of the art by a small margin and lose some generalization power, probably limited by the capabilities of SIFT and LIFT for indoor scenes. Nevertheless, we outperform the state of the art on *both* subsets with a *single model* trained on only 2000 outdoors and 500 indoors images and tested on completely different scenes. Further results are given in the supplementary material, where we show that we can still outperform the state of the art with much smaller training sets, *e.g.*, 59 images.

5. Conclusion

We have proposed a single-shot technique to recover the relative motion between two images with a deep network. In contrast with current trends, our method is sparse, indicating that keypoint-based robust estimation can still be relevant in the age of dense, black-box formulations. Our approach outperforms the state of the art by a significant margin with few images and limited supervision.

Our solution requires known intrinsics. In the future, we plan to investigate using the fundamental matrix instead of the essential. While the formalism would remain largely unchanged, we expect numerical stability problems in the regression component of the hybrid loss, which may require additional normalization layers or regularization terms.

References

- [1] J. Ba, J. Kiros, and G. Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016. [3](#)
- [2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *CVIU*, 10(3):346–359, 2008. [1](#)
- [3] J. Bian, W. Lin, Y. Matsushita, S. Yeung, T. Nguyen, and M. Cheng. GMS: Grid-Based Motion Statistics for Fast, Ultra-Robust Feature Correspondence. In *CVPR*, 2017. [1](#), [2](#), [6](#)
- [4] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. DSAC – Differentiable RANSAC for Camera Localization. *ARXIV*, 2016. [1](#), [2](#)
- [5] M. Cho, J. Sun, O. Duchenne, and J. Ponce. Finding Matches in a Haystack: A Max-Pooling Strategy for Graph Matching in the Presence of Outliers. In *CVPR*, 2014. [3](#)
- [6] S. Choi, T. Kim, and W. Yu. Performance Evaluation of RANSAC Family. In *BMVC*, 2009. [2](#)
- [7] O. Chum and J. Matas. Matching with Prosac - Progressive Sample Consensus. In *CVPR*, pages 220–226, June 2005. [2](#)
- [8] A. Crivellaro, M. Rad, Y. Verdie, K. Yi, P. Fua, and V. Lepetit. Robust 3D Object Tracking from Monocular Images Using Stable Parts. *PAMI*, 2017. [6](#)
- [9] D. Detone, T. Malisiewicz, and A. Rabinovich. Toward Geometric Deep SLAM. *arXiv preprint arXiv:1707.07410*, 2009. [2](#)
- [10] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications ACM*, 24(6):381–395, 1981. [1](#), [2](#), [6](#)
- [11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. [1](#), [2](#), [3](#), [4](#), [6](#)
- [12] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. [4](#)
- [13] J. Heinly, J. Schoenberger, E. Dunn, and J.-M. Frahm. Reconstructing the World in Six Days. In *CVPR*, 2015. [5](#)
- [14] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015. [3](#)
- [15] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for Deep Networks with Structured Layers. 2015. [4](#)
- [16] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimisation. In *ICLR*, 2015. [5](#)
- [17] W.-Y. Lin, M.-M. Cheng, J. Lu, H. Yang, M. Do, and P. Torr. Bilateral Functions for Global Motion Modeling. In *ECCV*, 2014. [2](#)
- [18] H. Longuet-Higgins. A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature*, 293:133–135, 1981. [1](#), [2](#), [3](#), [4](#)
- [19] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 20(2), 2004. [1](#), [2](#), [3](#), [6](#)
- [20] D. Nister. An Efficient Solution to the Five-Point Relative Pose Problem. In *CVPR*, June 2003. [2](#)
- [21] C. Qi, H. Su, K. Mo, and L. Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. [2](#), [3](#), [7](#)
- [22] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm. USAC: A Universal Framework for Random Sample Consensus. *PAMI*, 35(8):2022–2038, 2013. [2](#)
- [23] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection*. Wiley, 1987. [2](#), [6](#)
- [24] E. Rublee, V. Rabaud, K. Konolidge, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *ICCV*, 2011. [1](#), [6](#)
- [25] J. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys. Comparative Evaluation of Hand-Crafted and Learned Local Features. In *CVPR*, 2017. [1](#)
- [26] E. Simo-serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. moreno-noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *ICCV*, 2015. [1](#)
- [27] C. Strecha, W. Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery. In *CVPR*, 2008. [5](#)
- [28] B. Thomee, D. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L. Li. YFCC100M: the New Data in Multimedia Research. In *CACM*, 2016. [5](#)
- [29] P. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *CVIU*, 78:138–156, 2000. [2](#), [6](#)
- [30] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance Normalization: the Missing Ingredient for Fast Stylization. *arXiv Preprint*, 2016. [3](#), [4](#)
- [31] 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. [2](#), [5](#), [6](#), [7](#), [8](#)
- [32] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-Net: Learning of Structure and Motion from Video. *arXiv Preprint*, 2017. [2](#)
- [33] C. Wu. Towards Linear-Time Incremental Structure from Motion. In *3DV*, 2013. [5](#)
- [34] J. Xiao, A. Owens, and A. Torralba. SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In *ICCV*, 2013. [5](#)
- [35] K. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. In *ECCV*, 2016. [1](#), [3](#), [6](#)
- [36] A. R. Zamir, T. Wekel, P. Agrawal, J. Malik, and S. Savarese. Generic 3D Representation via Pose Estimation and Matching. In *ECCV*, 2016. [2](#), [6](#)
- [37] T. Zhou, M. Brown, N. Snavely, and D. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *CVPR*, 2017. [2](#)