ConvMatch: Rethinking Network Design for Two-View Correspondence Learning

Shihua Zhang, Jiayi Ma [*]

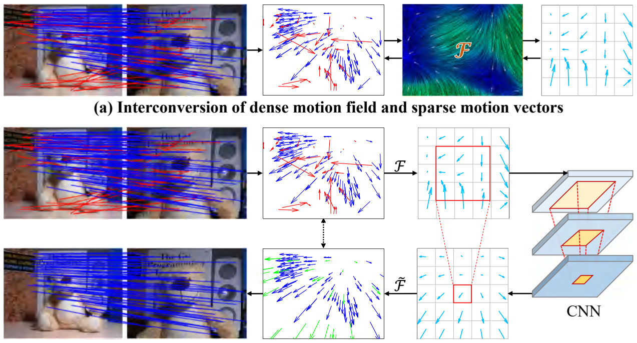
Electronic Information School, Wuhan University, Wuhan 430072, China
suhzhang001@gmail.com, jyma2010@gmail.com

Abstract

Multilayer perceptron (MLP) has been widely used in two-view correspondence learning for only unordered correspondences provided, and it extracts deep features from individual correspondence effectively. However, the problem of lacking context information limits its performance and hence, many extra complex blocks are designed to capture such information in the follow-up studies. In this paper, from a novel perspective, we design a correspondence learning network called ConvMatch that for the first time can leverage convolutional neural network (CNN) as the backbone to capture better context, thus avoiding the complex design of extra blocks. Specifically, with the observation that sparse motion vectors and dense motion field can be converted into each other with interpolating and sampling, we regularize the putative motion vectors by estimating dense motion field implicitly, then rectify the errors caused by outliers in local areas with CNN, and finally obtain correct motion vectors from the rectified motion field. Extensive experiments reveal that ConvMatch with a simple CNN backbone consistently outperforms state-of-the-arts including MLP-based methods for relative pose estimation and homography estimation, and shows promising generalization ability to different datasets and descriptors.

Introduction

Two-view corresponding is a fundamental problem in computer vision. It aims to establish sparse feature correspondences/matches between two-view images and estimate geometry relationship, serving as a premise for many complex vision problems such as structure from motion (Snavely, Seitz, and Szeliski 2008), simultaneous location and mapping (Mur-Artal, Montiel, and Tardos 2015), visual localization (Philbin et al. 2010), and image fusion (Xu et al. 2022). The most classical geometry matching pipeline starts from feature extraction and matching, and great efforts have been spent on handcrafted or learning-based detectors and descriptors, *e.g.*, SIFT (Lowe 2004) and SuperPoint (DeTone, Malisiewicz, and Rabinovich 2018). Outlier rejection is then applied to preserve correct matches (*a.k.a.* inliers) and reject false ones (*a.k.a.* outliers) caused by sensitive detector-descriptor methods, estimating relative pose robustly. This paper also focuses on removing outliers to obtain two-view correspondences and geometry relationship accurately.



(a) Interconversion of dense motion field and sparse motion vectors

(b) Framework of ConvMatch for two-view correspondence learning

Figure 1: Illustrations of the proposed framework for correspondence learning. Lines or arrows colored with blue are inliers, with red are outliers, and with green are rectified outliers. The head and tail of each arrow correspond to the positions of feature points in two images. (a) Interconversion of dense motion field and sparse motion vectors, where motion field can act as the bridge between unordered and ordered motion vectors. From left to right: putative correspondences, sparse unordered motion vectors, dense motion filed, sparse ordered motion vectors. (b) Our framework with three steps: regularizing the unordered motion vectors to ordered ones with motion field $\mathcal{F}$, rectifying errors of ordered motion vectors with CNN, recovering the unordered motion vectors from rectified ordered ones with rectified motion field $\widetilde{\mathcal{F}}$.

Among the traditional methods, RANSAC (Fischler and Bolles 1981) and its variants (Raguram et al. 2012; Barath, Matas, and Noskova 2019) obtain the smallest consistent inlier set to fit a given transformation model. The locality consistency-based methods such as LPM (Ma et al. 2019) and GMS (Bian et al. 2017) are also well developed, together with another active topic based on motion field consensus, *e.g.*, CRC (Fan et al. 2021) and VFC (Ma et al. 2014). However, handcrafted methods often fail in case of heavy outliers due to extreme viewpoint changes or repetitive structures, which often occur in the feature matching problem. Hence, researchers seek help from more powerful learning-based techniques. After PointCN (Yi et al. 2018) first consider-

[*] Corresponding author

ing the outlier rejection as a binary classification problem under the multilayer perceptron (MLP) framework, many other algorithms are proposed with the similar MLP-based network, *e.g.*, PointACN (Sun et al. 2020), OANet (Zhang et al. 2019), LMCNet (Liu et al. 2021), and CLNet (Zhao et al. 2021). The essential reason for such a unified network design is the sparse and unordered nature of point correspondences, and only MLP can extract deep features stably in this case. However, the individual feature-extraction property of MLP leads to a huge defect, *i.e.*, lack of context information, which is indispensable for two-view correspondence and geometry learning. Therefore, the MLP-based methods have to design extra blocks to capture context as a remedy. Although promising performance has been achieved using this pipeline, shortcomings still exist. For example, in PointCN and PointACN, normalization operation only captures global context information. In OANet and LMCNet, permutation-invariance pooling-like operation increases the difficulty of training. While in CLNet, sorting operation exacerbates instability easily. Clearly, all these problems in extra blocks have negative impacts on context information incorporation. In this case, one may wonder how to further solve this problem: *continuing to improve the context capturing block, or just substituting context-agnostic MLP with a context-perception network?*

It is widely known that convolutional neural network (CNN) can incorporate local information directly and global information gradually such as in image recognition or segmentation (He et al. 2016; Minaee et al. 2021). If we can use CNN instead of MLP to extract deep features for correspondences, the lack of local and global context information can be naturally solved with the larger receptive field of CNN. However, CNN operates only on ordered data such as image and feature map while point correspondences are totally unordered.

Fortunately, with the observation that dense motion field can be interpolated with sparse motion vectors and the sparse vectors can be sampled on the dense field in turn, the motion field can act as the bridge between unordered correspondences and ordered motion vectors as shown in Figure 1(a). In particular, after calculating the motion vector of each correspondence, we interpolate a dense motion field $\mathcal{F}$, and then sample at equal intervals to obtain ordered data. Clearly, the reverse process is also available. In this way, CNN is applicable in two-view correspondence learning.

Based on the above analysis, we propose a new framework for two-view correspondence learning, called ConvMatch, as shown in Figure 1(b). We first regularize the unordered motion vectors into an image structure-like data with motion field $\mathcal{F}$ estimated by an interpolated operation implicitly. Then we rectify the error of each ordered motion vector with the context information by CNN. From another perspective, the errors caused by outliers are regarded as the noise on motion field, and CNN can filter out it naturally with its low-pass character, thus the rectification is analogous to the denoising operation. Finally, with a similar interpolated operation to estimate the rectified motion field $\widetilde{\mathcal{F}}$, we obtain correct motion vectors for all correspondences from the rectified ordered vectors, and the inliers and outliers can be distinguished accordingly.

In summary, our main contributions are as follows:

- Rather than the MLP-based network together with extra context capturing blocks for two-view correspondence learning, we design a new framework with CNN to incorporate context information, avoiding the complex design and various shortcomings of additional blocks. To our best knowledge, it is the first time leveraging CNN as the backbone to solve the outlier rejection problem.
- With the observation that sparse motion vectors and dense motion field can be converted into each other, we regularize the sparse unordered motion vectors by estimating motion field implicitly, so that they can be processed by CNN rather than being restricted to MLP-based equivalent permutation network.
- We implement a specific network of the new framework named as ConvMatch. We demonstrate its effectiveness on relative pose estimation and homography estimation, which consistently outperforms the current state-of-the-arts. We also further analyze the effects of CNN, regularizing and rectifying processes.

Methodology

The key innovation of our network lies in using CNN to capture context information while extracting deep features instead of MLP with extra blocks. To this end, we convert the unordered motion vectors into ordered ones that can be processed by CNN, in which the conversion is achieved with a motion field estimated implicitly. Then, we incorporate context information to rectify the ordered motion vectors with CNN analogous to the denoising problem. As shown in Figure 2, given N putative correspondences $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)|i = 1, \dots, N, \boldsymbol{x}_i \in \mathbb{R}^2, \boldsymbol{y}_i \in \mathbb{R}^2\}$, the input of the network is the putative motion vectors $\{\boldsymbol{m}_i = (\boldsymbol{x}_i, \boldsymbol{d}_i)|i = 1, \dots, N, \boldsymbol{m}_i \in \mathbb{R}^4\}$, where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are the coordinates of two corresponding keypoints, and $\boldsymbol{d}_i = \boldsymbol{y}_i - \boldsymbol{x}_i$ is the displacement. The output is the logits for inlier/outlier classification $\{\hat{z}_i|i = 1, \dots, N\}$. Specifically, we first initialize the motion vectors in high dimensional space, and the coordinates of ordered motion vectors as well. Then regularize the unordered motion vectors followed by rectification on the ordered ones with CNN. Finally, we recover the unordered motion vectors from the rectified ordered ones in turn. The inlier/outlier classification results are then predicted by comparing the ultimate unordered motion vectors with the original ones. Built on the operations of Regularize, Rectify and Recover, ConvMatch is stacked L times totally. In the following, we present the new framework for outlier rejection and describe the specific network ConvMatch in detail.

Motion Vector Initialization

We use the sparse motion vectors $\{\boldsymbol{m}_i\}$ as the input because the motion field served as the bridge in Figure 1(a) is interpolated by them. However, the dimension of motion vector is too low to extract the deep features, hence as LMCNet (Liu et al. 2021) has done, we convert it to high dimensional motion vector $\boldsymbol{f}_i \in \mathbb{R}^C$ for input layer (layer 0):

$$^{(0)}\boldsymbol{f}_i = \mathcal{E}(\boldsymbol{m}_i), \quad i = 1, \dots, N, \tag{1}$$

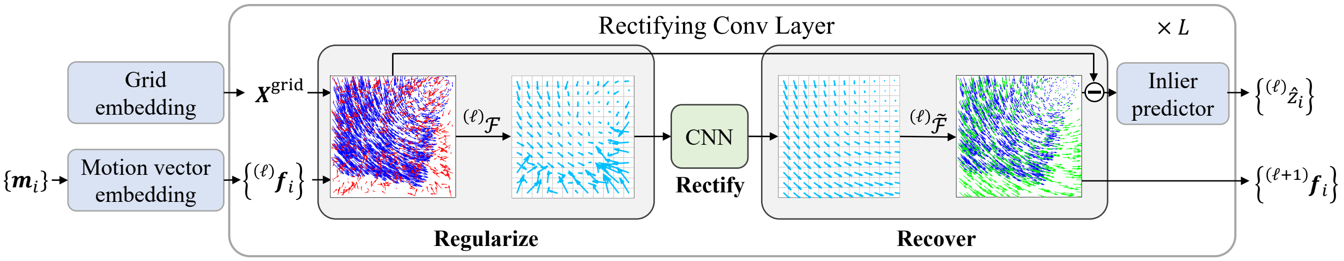

Rectifying Conv Layer
× L
Grid embedding
$\boldsymbol{X}^{\text{grid}}$
$\{\boldsymbol{m}_i\}$
Motion vector embedding
$\{{}^{(\ell)}\boldsymbol{f}_i\}$
${}^{(\ell)}\mathcal{F}$
CNN
Rectify
${}^{(\ell)}\widetilde{\mathcal{F}}$
Inlier predictor
$\{{}^{(\ell)}\hat{z}_i\}$
$\{{}^{(\ell+1)}\boldsymbol{f}_i\}$
Regularize
Recover

Figure 2: Architecture of ConvMatch. The Rectifying Conv Layer includes Regularize, Rectify and Recover blocks. Specifically, the Regularize block generates ordered motion vectors ${}^{(\ell)}\boldsymbol{F}^{\text{grid}}$ from the unordered ones $\{{}^{(\ell)}\boldsymbol{f}_i\}$ by sampling on implicitly estimated motion field ${}^{(\ell)}\mathcal{F}$. The Rectify block rectify the errors in ${}^{(\ell)}\boldsymbol{F}^{\text{grid}}$ with CNN and obtain ${}^{(\ell)}\widetilde{\boldsymbol{F}}^{\text{grid}}$. The Recover block outputs the rectified unordered motion vectors $\{{}^{(\ell+1)}\boldsymbol{f}_i\}$ by sampling on the rectified motion field ${}^{(\ell)}\widetilde{\mathcal{F}}$ estimated from ${}^{(\ell)}\widetilde{\boldsymbol{F}}^{\text{grid}}$. For each Rectifying Conv Layer, the input is the embedded unordered motion vectors $\{{}^{(\ell)}\boldsymbol{f}_i\}$ and a fixed embedded grid $\boldsymbol{X}^{\text{grid}}$. The output logits $\{{}^{(\ell)}\hat{z}_i\}$ are used to calculate losses and classify inliers/outliers.

where $\mathcal{E}(\cdot)$ means upgrading the dimension of motion vector to C by conducting positional embedding (Vaswani et al. 2017), here we choose $C = 128$.

Then, due to the image structure-like data required by CNN, we hope to sample on the motion field at the same intervals. To this end, we divide bounded 2D space of image into a $K \times K$ grid, sampling at the center of the grid $\boldsymbol{x}_{j,k}^{\text{grid}}$ to obtain the ordered motion vector. Similarly, to better sample with network, we embed $\boldsymbol{x}_{j,k}^{\text{grid}}$ to high dimensional space:

$$\boldsymbol{X}_{j,k}^{\text{grid}} = \text{Up}(\boldsymbol{x}_{j,k}^{\text{grid}}), \quad j,k = 1,\dots,K, \tag{2}$$

where $\text{Up}(\cdot)$ is a simple MLP that maps variable from low dimensional space to high dimensional space, and $\boldsymbol{X}_{j,k}^{\text{grid}} \in \mathbb{R}^C$ with $C = 128$.

Ordered Motion Vector Generation

As mentioned above, to satisfy the requirement of CNN for input data, we should interpolate a dense motion field and then sample at the equal-interval location to obtain ordered motion vectors. In traditional methods, interpolating motion field is normally realized by regularization (Baldassarre et al. 2012). The process can be expressed briefly as:

$$\mathcal{F} = \phi(\{\boldsymbol{m}_i\}), \quad i = 1,\dots,N, \tag{3}$$

where $\phi(\cdot)$ is interpolating function, and $\mathcal{F}$ is the estimated motion field. Then the ordered motion vector $\boldsymbol{m}_{j,k}^{\text{grid}}$ can be sampled easily from $\mathcal{F}$ at the center of the grid $\boldsymbol{x}_{j,k}^{\text{grid}}$:

$$\begin{aligned}\boldsymbol{m}_{j,k}^{\text{grid}} = \left(\boldsymbol{x}_{j,k}^{\text{grid}}, \boldsymbol{d}_{j,k}^{\text{grid}}\right) &= \left(\boldsymbol{x}_{j,k}^{\text{grid}}, \mathcal{F}(\boldsymbol{x}_{j,k}^{\text{grid}})\right) \\ &= \left(\boldsymbol{x}_{j,k}^{\text{grid}}, \phi(\{\boldsymbol{m}_i\})|_{\boldsymbol{x}_{j,k}^{\text{grid}}}\right).\end{aligned} \tag{4}$$

However, the function $\phi(\cdot)$ depends on the handcrafted kernel and regularization parameter, and is hard to process the high dimensional expressions in our network. Therefore, we use the GAT network (Veličković et al. 2018) $\mathcal{G}(\cdot,\cdot)$ instead to interpolate motion field $\mathcal{F}$ implicitly and generate ordered motion vector directly:

$$\boldsymbol{m}_{j,k}^{\text{grid}} = \mathcal{G}\left(\{\boldsymbol{m}_i\}, \boldsymbol{x}_{j,k}^{\text{grid}}\right). \tag{5}$$

Re-write it in high dimensional space according to Eqs. (1) and (2) with matrix form:

$${}^{(\ell)}\boldsymbol{F}^{\text{grid}} = \mathcal{G}\left(\{{}^{(\ell)}\boldsymbol{f}_i\}, \{\boldsymbol{X}_{j,k}^{\text{grid}}\}\right) = \mathcal{G}\left({}^{(\ell)}\boldsymbol{F}, \boldsymbol{X}^{\text{grid}}\right), \tag{6}$$

where ${}^{(\ell)}\boldsymbol{F}^{\text{grid}} = \{{}^{(\ell)}\boldsymbol{f}_{j,k}^{\text{grid}}\}$ and ${}^{(\ell)}\boldsymbol{f}_{j,k}^{\text{grid}}$ is the high dimensional embedding of $\boldsymbol{m}_{j,k}^{\text{grid}}$. By simplifying the variable notations of ${}^{(\ell)}\boldsymbol{F}$ and $\boldsymbol{X}^{\text{grid}}$, we further define $\mathcal{G}(\boldsymbol{F}, \boldsymbol{X})$ as:

$$\mathcal{G}(\boldsymbol{F}, \boldsymbol{X}) = \text{Comb}(\boldsymbol{X}, \text{Aggr}(\boldsymbol{X}, \boldsymbol{F})), \tag{7}$$

where $\text{Aggr}(\cdot,\cdot)$ tries to estimate the motion field at the position of $\boldsymbol{X}$ by aggregating all known motion vectors $\boldsymbol{F}$, and $\text{Comb}(\cdot,\cdot)$ tries to combine the motion filed information and location information on grid center points to obtain the new motion vectors $\boldsymbol{F}^{\text{grid}}$. Specifically:

$$\begin{gathered}\text{Aggr}(\boldsymbol{X}, \boldsymbol{F}) = \text{Softmax}(\boldsymbol{Q}\boldsymbol{K}^T)\boldsymbol{V}, \\ \boldsymbol{Q} = \boldsymbol{W}_1\boldsymbol{X} + \boldsymbol{b}_1, \\ \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} \boldsymbol{W}_2 \\ \boldsymbol{W}_3 \end{bmatrix} \boldsymbol{F} + \begin{bmatrix} \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{bmatrix},\end{gathered} \tag{8}$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3$ are learnable weights and $\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$ are learnable biases, and

$$\text{Comb}(\boldsymbol{X}, \boldsymbol{\mathcal{A}}) = \boldsymbol{X} + \text{MLP}(\boldsymbol{X} \| \boldsymbol{\mathcal{A}}), \tag{9}$$

where $\boldsymbol{\mathcal{A}}$ indicates the result of function $\text{Aggr}(\cdot,\cdot)$, and $\|$ means concatenating by channels.

With ordered motion vector generation, we transfer the putative motion vectors to the ordered ones. The transformation is achieved by interpolating a dense motion field implicitly then sampling, *i.e.*, Eq. (6). In the following, we aim to reconstruct the ordered motion vectors into an image structure-like data, and handle it with CNN to rectify the errors caused by outliers.

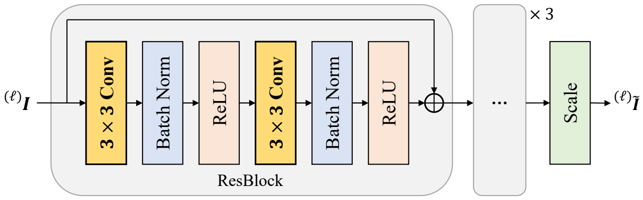


Figure 3: Architecture details of CNN backbone. It is stacked with 3 Resblocks consisting of 3×3 convolutional layer, batch normalization, Relu activation function and shortcut connection, and finally scales the result adaptively. Note that we do not use the Context Normalization which is heavily reused in MLP-based methods.

Rectify Motion Vectors with CNN

The ordered motion vectors $^{(\ell)}\boldsymbol{F}^{\text{grid}} = \{^{(\ell)}\boldsymbol{f}^{\text{grid}}_{j,k}\}$ obtained according to Eq. (6) can represent the dense motion field $\mathcal{F}$ due to the local consistent of the motion field. Thus, by selecting the motion vectors in sequence we reshape $\{^{(\ell)}\boldsymbol{f}^{\text{grid}}_{j,k} \in \mathbb{R}^C, j,k = 1,\ldots,K\}$ to an image structure-like data $^{(\ell)}\boldsymbol{I} \in \mathbb{R}^{K\times K\times C}$, which can be considered as the digital form of dense analogue motion field. Due to the fact that the motion field built by inliers are smoother and more local consistent than that with outliers (Ma et al. 2014), once obtaining a motion field with heavy outliers, we can smooth it to get the more consistent one $^{(\ell)}\widetilde{\mathcal{F}}$. By doing so, the contaminated motion field are fixed to filter out outliers. Therefore, we further consider outlier rejection as a denoising problem where outliers are noise signals, rectifying the errors caused by outliers of the motion field with CNN:

$$^{(\ell)}\widetilde{\boldsymbol{I}} = \text{CNN}\left(^{(\ell)}\boldsymbol{I}\right). \tag{10}$$

Note that the CNN not only smooths the motion field to reduce the influence of outliers, but also incorporates local and global context information from neighboring up to all motion vectors. This information is conducive to rectify the vector far different from its neighbors into similar to them, which cannot be achieved with MLP.

We choose a simple Resblock (He et al. 2016) that contains only two convolutional layers followed by batch normalization (Ioffe and Szegedy 2015) and Relu (Glorot, Bordes, and Bengio 2011) activation function with shortcut connection, as shown in Figure 3. We simply stack the Resblock three times in a layer, which is sufficient to get satisfying results, proving the effectiveness of the CNN backbone.

By Eq. (10), the ordered digital motion field is smoothed and rectified by a CNN block, means that original motion field represented by $^{(\ell)}\mathcal{F}$ is transferred to $^{(\ell)}\widetilde{\mathcal{F}}$, then $^{(\ell)}\widetilde{\boldsymbol{I}}$ can be re-expanded to sequence $^{(\ell)}\widetilde{\boldsymbol{F}}^{\text{grid}} = \{^{(\ell)}\widetilde{\boldsymbol{f}}^{\text{grid}}_{j,k}\}$, where $^{(\ell)}\widetilde{\boldsymbol{f}}^{\text{grid}}_{j,k}$ describes the motion vector of the corresponding grid center in the new motion field $^{(\ell)}\widetilde{\mathcal{F}}$.

Unordered Motion Vector Recovery

Ideally, the errors on motion vectors caused by outliers are rectified after obtaining $^{(\ell)}\widetilde{\boldsymbol{F}}^{\text{grid}}$. However, a single Rectifying Conv Layer may not work well, and it is typically necessary to filter out outliers progressively with multiple layers (Ma et al. 2019). To this end, we convert the rectified ordered motion vectors back to the unordered ones. As the same as Eq. (5), we estimate rectified motion field $\widetilde{\mathcal{F}}$ implicitly and obtain the new motion vector of each correspondence:

$$\widetilde{\boldsymbol{m}}_i = \left(\boldsymbol{x}_i, \widetilde{\mathcal{F}}(\boldsymbol{x}_i)\right) = \mathcal{G}\left(\{^{(\ell)}\widetilde{\boldsymbol{m}}^{\text{grid}}_{j,k}\}, \boldsymbol{x}_i\right), \tag{11}$$

Re-write it in high dimensional space with matrix form as well:

$$^{(\ell+1)}\boldsymbol{F} = {}^{(\ell)}\widetilde{\boldsymbol{F}} = \mathcal{G}\left(^{(\ell)}\widetilde{\boldsymbol{F}}^{\text{grid}}, {}^{(\ell)}\boldsymbol{X}\right), \tag{12}$$

where $^{(\ell)}\boldsymbol{X} = \{^{(\ell)}\boldsymbol{X}_i\}$ should be the high dimensional embedding of $\{\boldsymbol{x}_i\}$ to provide the location information of original motion vectors. Particularly, such information is available in original high dimensional motion vectors $^{(l)}\boldsymbol{F}$, we can replace $^{(\ell)}\boldsymbol{X}$ with it, which also helps our network to retrieve the possible information missing in the regularizing processes:

$$^{(\ell+1)}\boldsymbol{F} = {}^{(\ell)}\widetilde{\boldsymbol{F}} = \mathcal{G}\left(^{(\ell)}\widetilde{\boldsymbol{F}}^{\text{grid}}, {}^{(\ell)}\boldsymbol{F}\right). \tag{13}$$

Note that $^{(0)}\boldsymbol{F}$ is obtained from Eq. (1), and $\mathcal{G}$ is defined as the same as Eq. (7). The new high dimensional motion vectors $^{(\ell)}\widetilde{\boldsymbol{F}} = \{^{(\ell)}\widetilde{\boldsymbol{f}}_i\}$ are the input of next layer as $^{(\ell+1)}\boldsymbol{F} = \{^{(\ell+1)}\boldsymbol{f}_i\}$.

Inlier Predictor

With Eq. (13), the new motion vector $^{(\ell)}\widetilde{\boldsymbol{f}}_i$ substitutes $^{(\ell)}\boldsymbol{f}_i$ after a complete Rectifying Conv Layer. To judge whether the correspondence $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is an inlier or not, a common approach is comparing the similarity of $^{(\ell)}\boldsymbol{f}_i$ and $^{(\ell)}\widetilde{\boldsymbol{f}}_i$ using Euclidean distance with the role that motion vector of inlier should not change a lot after rectifying but outlier changes significantly, classifying inliers and outliers with a threshold. However, calculating similarity makes training process unstable easily. Refer to other learning-based methods (Sun et al. 2020; Chen et al. 2021), we add extra inlier predictor in each layer while training but only preserve the last one at inference. The input of inlier predictor is $^{(\ell)}\widetilde{\boldsymbol{f}}_i - {}^{(\ell)}\boldsymbol{f}_i$, and the predictor maps it to one dimension, outputting logit $^{(\ell)}\hat{z}_i$ for classification. Note that only the output of the last layer $\hat{z}_i = {}^{(L-1)}\hat{z}_i$ is used to classify the inlier at inference. The structure of inlier predictor is shown in Figure 4.

Loss Functions

Our ConvMatch outputs the logits $^{(\ell)}\hat{\boldsymbol{z}} = \{^{(\ell)}\hat{z}_i\}$ similar to PointCN, OANet, *etc.*, so we use the same classification loss function and regression loss function:

$$\mathcal{L} = \sum_{\ell=0}^{L-1} \mathcal{L}_{cls}\left(\boldsymbol{z}, {}^{(\ell)}\hat{\boldsymbol{z}}\right) + \lambda\mathcal{L}_{reg}\left(\boldsymbol{E}, {}^{(\ell)}\widehat{\boldsymbol{E}}\right), \tag{14}$$

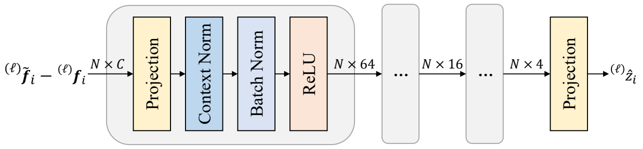


Figure 4: Architecture details of inlier predictor. The input is the difference between each motion vector before and after the Rectifying Conv Layer. With projection blocks, the predictor reduces the dimension of feature difference and outputs the logit $^{(\ell)}\hat{z}_i$ for classification and loss calculation.

where superscript (ℓ) means the ℓ-th layer, and we sum up the classification and regression losses of all layers.

The classification loss function $\mathcal{L}_{cls}$ is a simple binary cross entropy loss, and $\boldsymbol{z}$ is the weakly supervised label judged by a threshold of 10^{-4} after calculating the Sampson distance (Hartley and Zisserman 2003). $\mathcal{L}_{reg}$ is also obtained with the Sampson distance:

$$\mathcal{L}_{\text{reg}}\left(\boldsymbol{E}, \widehat{\boldsymbol{E}}\right) = \sum_{i=1}^{N} \frac{\left(\boldsymbol{y}_i^T \widehat{\boldsymbol{E}} \boldsymbol{x}_i\right)^2}{\|\boldsymbol{E}\boldsymbol{x}_i\|_{[1]}^2 + \|\boldsymbol{E}\boldsymbol{x}_i\|_{[2]}^2 + \|\boldsymbol{E}^T\boldsymbol{y}_i\|_{[1]}^2 + \|\boldsymbol{E}^T\boldsymbol{y}_i\|_{[2]}^2}, \tag{15}$$

where $\widehat{\boldsymbol{E}}$ is calculated by the weighted eight-point algorithm (Yi et al. 2018), $\|\boldsymbol{v}\|_{[m]}$ denotes the m-th element of vector $\boldsymbol{v}$. λ is the hyper-parameter to balance two loss functions.

Implementation Details

In our implementation, coordinates of correspondences are normalized to the range $[-1, 1]$ using the image size and camera intrinsic if available. Our ConvMatch contains the same Rectifying Conv Layer stacked 6 times which means $L = 6, \ell \in \{0, \ldots, 5\}$, and we set $K = 16$ in regularizing process for performance and time balance, more details are discussed in parameter analysis. Putative correspondences are established with SIFT features and nearest neighbor method, and we extract up to 2000 correspondences for each image pair. We use Adam (Kingma and Ba 2015) optimizer during training with a learning rate of 10^{-4} during the first $80k$ iterations and then 5×10^{-5} in the rest $420k$ iterations, and batch size 32. Weight λ is 0 during the first $20k$ iterations and then 0.5.

Experiment Results

We conduct experiments to evaluate ConvMatch with the tasks of relative pose estimation and homography estimation. In addition, we discuss the parameter setting and analyze our method comprehensively to demonstrate the effectiveness of the CNN backbone, regularizing and rectifying processes.

Relative Pose Estimation

The relative pose estimation tries to estimate the positional relationship (rotation and translation) between the cameras that capture the image pair with the predicted inliers. Following the same settings in OANet, we exploit the outdoor YFCC100M (Thomee et al. 2016) and indoor SUN3D (Xiao, Owens, and Torralba 2013) datasets. The input putative correspondences up to 2000 are detected with SIFT features and matched with nearest neighbor (NN) method. We calculate the **AUC** (Yi et al. 2018; Zhang et al. 2019) of the maxima pose error of rotation and translation at the thresholds $(5°, 10°, 20°)$ to evaluate pose estimation accuracy. In addition, we report the F-score which considers precision and recall all together to evaluate inlier/outlier classification performance. We consider that the correspondence whose epipolar distance is smaller than a certain threshold (*e.g.* 10^{-4}) is correct.

Table 1: Outdoor relative pose estimation with RANSAC. All learning-based methods are trained on YFCC100M dataset with SIFT features following OANet (Zhang et al. 2019), except for SuperGlue (Sarlin et al. 2020) of which we use the results reported in their supplementary material.

Method	**AUC**		
	@5°	@10°	@20°
CRC	12.05	23.17	36.53
GMS	13.29	24.38	37.83
LPM	15.99	28.25	41.76
VFC	17.43	29.98	43.00
PointCN	26.73	44.01	60.49
OANet	27.26	45.93	63.17
CLNet	31.45	51.06	68.40
LMCNet	30.48	49.84	66.94
MS^{2}DGNet	31.55	50.94	68.34
ConvMatch (Ours)	**31.69**	**51.41**	**68.45**
SuperGlue*	30.49	51.29	69.72

Table 2: Outdoor relative pose estimation without RANSAC. Additional row for OANet with RANSAC to show the competitive performance of ConvMatch even without RANSAC.

Method	**AUC**		
	@5°	@10°	@20°
PointCN	10.16	24.43	43.31
OANet	15.92	35.93	57.11
CLNet	24.34	44.69	63.61
LMCNet	22.35	43.57	63.34
MS^{2}DGNet	20.61	42.90	64.26
ConvMatch (Ours)	**26.83**	**49.14**	**67.91**
OANet w. RANSAC	27.26	45.93	63.17

We compare ConvMatch with both classical outlier rejection methods such as CRC (Fan et al. 2021), GMS (Bian et al. 2017), LPM (Ma et al. 2019), VFC (Ma et al. 2014), and learning-based methods such as PointCN (Yi et al. 2018), OANet (Zhang et al. 2019), CLNet (Zhao et al. 2021), LMCNet (Liu et al. 2021), MS^{2}DGNet (Dai et al. 2022). For outdoor relative pose estimation, we compare our method with the feature matching method SuperGlue (Sarlin et al. 2020) additionally.

Table 3: Indoor relative pose estimation with RANSAC. All learning-based methods are trained on SUN3D dataset with SIFT features following OANet (Zhang et al. 2019).

Method	AUC		
	@5°	@10°	@20°
CRC	4.07	10.44	20.87
GMS	4.12	10.53	20.82
LPM	4.80	12.28	23.77
VFC	5.26	13.05	24.84
PointCN	6.09	15.43	29.74
OANet	6.78	17.10	32.41
LMCNet	6.84	17.62	33.43
MS^{2}DGNet	7.13	17.80	33.47
ConvMatch (Ours)	**7.32**	**18.45**	**34.41**

Table 4: Indoor relative pose estimation without RANSAC.

Method	AUC		
	@5°	@10°	@20°
PointCN	3.05	10.00	24.06
OANet	5.93	16.91	34.32
LMCNet	7.08	19.09	37.15
MS^{2}DGNet	5.88	16.83	34.28
ConvMatch (Ours)	**8.76**	**22.23**	**40.49**

For outdoor data, the estimation results with RANSAC (Raguram et al. 2012) as robust essential matrix estimator are shown in Table 1. We also report the results without it in Table 2, where the relative pose is calculated from the essential matrix predicted by network directly with the weighted eight-point algorithm (Yi et al. 2018). The threshold used to classify the correspondence is set to 0 for all experiments (the same in other sections, unless otherwise specified), which means that $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is inlier if $\hat{z}_i > 0$. However, we find that the value of threshold does affect the estimation results, especially for learning-based methods. Thus we test them under different thresholds and draw the results of **AUC**@5° and F-score in Figure 5. Combining Table 1 and Figure 5, we can conclude that ConvMatch performs better than state-of-the-arts no matter with a fixed threshold for its higher **AUC** metric, or with changeable thresholds for its higher peak value in **AUC** and F-score metrics. For indoor data, we also report the results with/without RANSAC in Tables 3 and 4, respectively. ConvMatch still outperforms others, delivering the best **AUC**. We further illustrate the qualitative results of outlier rejection and relative pose estimation, as shown in Figure 6, ConvMatch can capture stronger context so that reversing more inliers and less outliers, and still estimating good relative pose while some other methods fail for the viewpoint and scale changing dramatically.

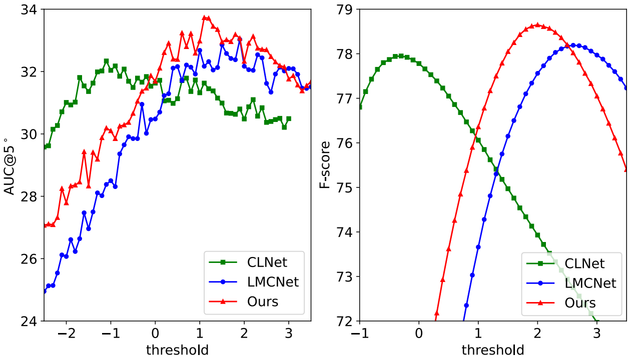
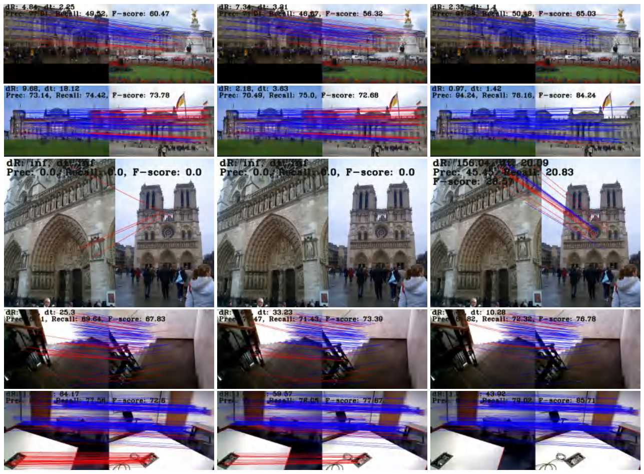


Figure 5: Outdoor relative pose estimation results on **AUC**@5° and F-score with inlier classification threshold changing. Note that CLNet (Zhao et al. 2021) and LMCNet (Liu et al. 2021) are the most recent well-performed learning-based methods.



OANet LMCNet ConvMatch (Ours)

Figure 6: Qualitative illustration of outlier rejection. Image pairs from left to right are outlier rejection results of OANet, LMCNet and ConvMatch, respectively. The false matches are marked with red while correct matches are marked with blue. The relative pose estimation result is provided in the top left corner of each image pair.

Homography Estimation

Homography estimation is a basic yet critical task in computer vision. We perform homography estimation experiment on HPatches benchmark (Balntas et al. 2017) with RANSAC. There are 116 scenes with 696 images in HPatches, where 57 scenes are taken under different illumination and others undergo viewpoint changes. Each scene consists of 6 images, one for reference and others for targets with ground-truth homography. We detect keypoints up to 4000 with SIFT and match them with NN method. For metric, we follow the suggestion of SuperPoint (DeTone, Malisiewicz, and Rabinovich 2018), adopting the homography error to classify an estimate to be accurate or not by a threshold as 3 pixels here. We calculate mean accurate of all image pairs as **Acc.**, and F-score (**F.**) for inlier/outlier classification. We also report the mean time cost of each image pair.

We compare ConvMatch with almost the same learning-based methods as relative pose estimation. The models we used are trained with SIFT features on YFCC100M dataset.

Table 5: Homography estimation results. We report the mean time of inference for all methods additionally.

Method	**Acc.**	**F.**	Time (ms)
PointCN	67.93	81.74	**7.26**
OANet	69.66	81.87	13.36
CLNet	57.07	64.16	25.96
LMCNet	72.93	**85.44**	318.99
MS^{2}DGNet	72.07	81.01	18.51
ConvMatch (Ours)	**73.45**	83.80	22.09

Table 6: Generalization ability test. We repeat relative pose estimation on different datasets with multiple descriptors exploiting the same model. **AUC**@10° without RANSAC is reported.

Method	YFCC100M		SUN3D		
	RootSIFT	SuperPoint	SIFT	RootSIFT	SuperPoint
PointCN	24.71	14.94	1.56	1.71	3.52
OANet	36.43	20.49	3.37	3.64	3.42
CLNet	44.71	24.05	2.66	2.84	3.45
LMCNet	44.42	23.63	5.41	5.59	4.15
MS^{2}DGNet	43.60	25.24	5.49	5.75	3.93
ConvMatch (Ours)	**50.03**	**29.31**	**7.34**	**7.78**	**6.42**

The results are shown in Table 5. ConvMatch outperforms other methods in homography estimation for its best **Acc.**, although only achieves competitive performance on F-score. This is because ConvMatch can make use of context information more comprehensively and evenly in the image with CNN, which is more friendly to homography estimation even in difficult scenes while other methods fail. In addition, our method takes exciting inference time especially compared with the most recent methods (CLNet, LMCNet, and MS^{2}DGNet).

Analysis

We further analyze ConvMatch in this section, including the generalization ability on different datasets with multiple descriptors while using the same parameter model, the parameter analysis to determine the network structure, and the ablation studies to reveal the effectiveness of our framework and CNN backbone.

Generalization Ability. To demonstrate the good generalization ability of ConvMatch, we repeat the relative pose estimation for learning-based methods on YFCC100M with RootSIFT or SuperPoint, and on SUN3D with SIFT, RootSIFT or SuperPoint, employing the models trained on YFCC100M with SIFT only. Note that we generate RootSIFT keypoints up to 2000 as the same as SIFT, but SuperPoint keypoints up to 1000, and use NN method to obtain putative correspondences. As shown in Table 6, ConvMatch achieves the best performance in all cases, which proves the robustness context incorporation of CNN in our method.

Parameter Analysis. The parameter values of layer number L and grid number K affect a lot for a larger L leads to more rectifying processes and a larger K makes ordered motion vectors finer, which ideally contribute to better performance but more consumption. After multiple attempts, we choose $L = 6, K = 16$ to achieve performance and consumption balance. And the results of outdoor relative pose estimation in Table 7 with different L and K can support our choice, where a larger L or K achieves similar performance but more costs while the smaller one causes obvious degradation for pose estimation.

Table 7: Parameter analysis of L and K. We finally choose $L = 6, K = 16$ marked in bold. And the metric of performance is **AUC**@20° without RANSAC. Note that one of the parameters is fixed while another changes.

Metric	$L=4$	$\boldsymbol{L=6}$	$L=8$	$K=8$	$\boldsymbol{K=16}$	$K=24$
AUC@20°	63.79	67.91	68.89	66.64	67.91	67.94
Time (ms)	16.90	23.49	29.04	23.43	23.49	23.59

Table 8: The results of ablation studies. All models are trained on YFCC100M with SIFT features and evaluated with exactly the same settings followed by RANSAC.

Method	**AUC**		
	@5°	@10°	@20°
ConvMatch-Classify	29.88	48.49	65.92
ConvMatch-MLP	30.79	49.43	66.21
ConvMatch	31.69	51.41	68.45
ConvMatch-UNet	**33.30**	**53.02**	**69.96**

Ablation Studies. Moreover, to demonstrate the effectiveness of the regularizing and rectifying processes in Figure 1(b), we directly classify the unordered correspondences with GAT network completely as the same as SuperGlue (Sarlin et al. 2020), named as ConvMatch-Classify. The result is worse than ConvMatch that rectifies the errors of the motion field with CNN, which demonstrates the important role the regularizing and rectifying processes play. To demonstrate the positive influence of CNN which incorporates context information more comprehensively and rectify the motion field more naturally, we exchange the CNN backbone with MLP followed by Context Normalization (Yi et al. 2018) to capture context, named as ConvMatch-MLP. Our full ConvMatch performs better than it, which reveals that rectifying the motion field can reject more outliers and the CNN does capture context information better. In addition, we exchange the simple backbone with a sophisticated CNN architecture UNet, named as ConvMatch-UNet. The result reveals that our new framework in Figure 1(b) has great potential in performance boost with a well-designed CNN backbone. All the results are shown in Table 8.

Conclusion

In this paper, we design a new network called ConvMatch with CNN acting as the backbone instead of MLP to capture better context information. With the character that dense motion field and sparse motion vectors can be converted into each other, we regularize the putative motion vectors, so that rectifying the errors of outliers with CNN. Extensive experiments demonstrate the superiority of our method over the state-of-the-arts and the competitive inference time as well, and that more powerful backbone network can achieve further enhancement.

References

Baldassarre, L.; Rosasco, L.; Barla, A.; and Verri, A. 2012. Multi-output learning via spectral filtering. *Mach. Learn.*, 87(3): 259–301.

Balntas, V.; Lenc, K.; Vedaldi, A.; and Mikolajczyk, K. 2017. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 5173–5182.

Barath, D.; Matas, J.; and Noskova, J. 2019. MAGSAC: marginalizing sample consensus. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 10197–10205.

Bian, J.; Lin, W.-Y.; Matsushita, Y.; Yeung, S.-K.; Nguyen, T.-D.; and Cheng, M.-M. 2017. Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 4181–4190.

Chen, H.; Luo, Z.; Zhang, J.; Zhou, L.; Bai, X.; Hu, Z.; Tai, C.-L.; and Quan, L. 2021. Learning to match features with seeded graph matching network. In *IEEE Int. Conf. Comput. Vis.*, 6301–6310.

Dai, L.; Liu, Y.; Ma, J.; Wei, L.; Lai, T.; Yang, C.; and Chen, R. 2022. MS2DG-Net: Progressive Correspondence Learning via Multiple Sparse Semantics Dynamic Graph. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 8973–8982.

DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2018. Superpoint: Self-supervised interest point detection and description. In *IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 224–236.

Fan, A.; Jiang, X.; Ma, Y.; Mei, X.; and Ma, J. 2021. Smoothness-driven consensus based on compact representation for robust feature matching. *IEEE Trans. Neural Networks Learn. Syst.*

Fischler, M. A.; and Bolles, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6): 381–395.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Int. Conf. Artif. Intell. Stat.*, 315–323. JMLR Workshop and Conference Proceedings.

Hartley, R.; and Zisserman, A. 2003. *Multiple view geometry in computer vision*. Cambridge university press.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 770–778.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. Mach. Learn.*, 448–456. PMLR.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *Int. Conf. Learn. Represent.*

Liu, Y.; Liu, L.; Lin, C.; Dong, Z.; and Wang, W. 2021. Learnable motion coherence for correspondence pruning. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 3237–3246.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2): 91–110.

Ma, J.; Zhao, J.; Jiang, J.; Zhou, H.; and Guo, X. 2019. Locality preserving matching. *Int. J. Comput. Vision*, 127(5): 512–531.

Ma, J.; Zhao, J.; Tian, J.; Yuille, A. L.; and Tu, Z. 2014. Robust point matching via vector field consensus. *IEEE Trans. Image Process.*, 23(4): 1706–1721.

Minaee, S.; Boykov, Y. Y.; Porikli, F.; Plaza, A. J.; Kehtarnavaz, N.; and Terzopoulos, D. 2021. Image segmentation using deep learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*

Mur-Artal, R.; Montiel, J. M. M.; and Tardos, J. D. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. Rob.*, 31(5): 1147–1163.

Philbin, J.; Isard, M.; Sivic, J.; and Zisserman, A. 2010. Descriptor learning for efficient retrieval. In *Eur. Conf. Comput. Vis.*, 677–691. Springer.

Raguram, R.; Chum, O.; Pollefeys, M.; Matas, J.; and Frahm, J.-M. 2012. USAC: A universal framework for random sample consensus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8): 2022–2038.

Sarlin, P.-E.; DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2020. Superglue: Learning feature matching with graph neural networks. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 4938–4947.

Snavely, N.; Seitz, S. M.; and Szeliski, R. 2008. Modeling the world from internet photo collections. *Int. J. Comput. Vision*, 80(2): 189–210.

Sun, W.; Jiang, W.; Trulls, E.; Tagliasacchi, A.; and Yi, K. M. 2020. Acne: Attentive context normalization for robust permutation-equivariant learning. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 11286–11295.

Thomee, B.; Shamma, D. A.; Friedland, G.; Elizalde, B.; Ni, K.; Poland, D.; Borth, D.; and Li, L.-J. 2016. YFCC100M: The new data in multimedia research. *Commun. ACM*, 59(2): 64–73.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Adv. Neural Inf. Process. Syst.*, volume 30.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Int. Conf. Learn. Represent.*

Xiao, J.; Owens, A.; and Torralba, A. 2013. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *IEEE Int. Conf. Comput. Vis.*, 1625–1632.

Xu, H.; Ma, J.; Yuan, J.; Le, Z.; and Liu, W. 2022. RFNet: Unsupervised Network for Mutually Reinforcing Multi-Modal Image Registration and Fusion. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 19679–19688.

Yi, K. M.; Trulls, E.; Ono, Y.; Lepetit, V.; Salzmann, M.; and Fua, P. 2018. Learning to find good correspondences. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2666–2674.

Zhang, J.; Sun, D.; Luo, Z.; Yao, A.; Zhou, L.; Shen, T.; Chen, Y.; Quan, L.; and Liao, H. 2019. Learning two-view correspondences and geometry using order-aware network. In *IEEE Int. Conf. Comput. Vis.*, 5845–5854.

Zhao, C.; Ge, Y.; Zhu, F.; Zhao, R.; Li, H.; and Salzmann, M. 2021. Progressive correspondence pruning by consensus learning. In *IEEE Int. Conf. Comput. Vis.*, 6464–6473.