Learnable Motion Coherence for Correspondence Pruning

Yuan Liu[1] Lingjie Liu[2] Cheng Lin[1] Zhen Dong[3] Wenping Wang[1,4*]
[1]The University of Hong Kong [2]MPI Informatics, Saarland Informatics Campus
[3]Wuhan University [4]Texas A&M University

Abstract

Motion coherence is an important clue for distinguishing true correspondences from false ones. Modeling motion coherence on sparse putative correspondences is challenging due to their sparsity and uneven distributions. Existing works on motion coherence are sensitive to parameter settings and have difficulty in dealing with complex motion patterns. In this paper, we introduce a network called Laplacian Motion Coherence Network (LMCNet) to learn motion coherence property for correspondence pruning. We propose a novel formulation of fitting coherent motions with a smooth function on a graph of correspondences and show that this formulation allows a closed-form solution by graph Laplacian. This closed-form solution enables us to design a differentiable layer in a learning framework to capture global motion coherence from putative correspondences. The global motion coherence is further combined with local coherence extracted by another local layer to robustly detect inlier correspondences. Experiments demonstrate that LMCNet has superior performances to the state of the art in relative camera pose estimation and correspondences pruning of dynamic scenes[1].

1. Introduction

Estimating correspondences between two images is a fundamental problem in computer vision tasks such as Structure-from-Motion (SfM) [19], visual localization [45], image stitching [10] and visual SLAM [32]. A standard pipeline of correspondence estimation relies on local feature matching to establish a set of putative correspondences, which contains numerous false correspondences (i.e., outliers). To prevent outliers from affecting downstream tasks, a correspondence pruning algorithm is usually applied to select a reliable subset consisting of true correspondences (i.e., inliers). The most prevalent correspondence pruning methods are RANSAC [18] and its variants [52, 14, 2, 12], which detect true correspondences by finding the largest subset conforming to a task-specific geometric model such as epipolar geometry or homography. However, due to large outlier ratios of putative correspondences, multiple plausible geometric models may exist, which makes it difficult for RANSAC and its variants to identify the correct geometric model.

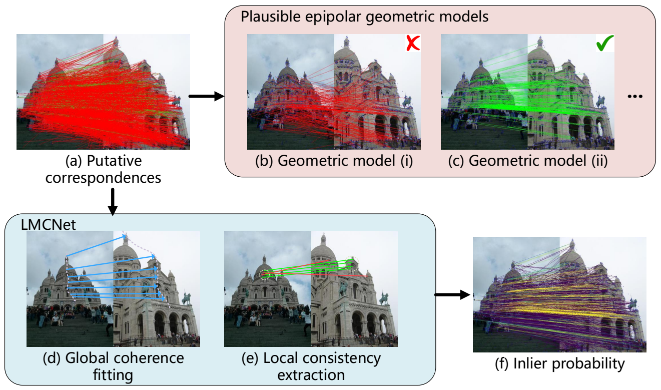

Plausible epipolar geometric models
✗
✓
...
(a) Putative correspondences
(b) Geometric model (i)
(c) Geometric model (ii)
LMCNet
(d) Global coherence fitting
(e) Local consistency extraction
(f) Inlier probability

Figure 1. Given a set of putative correspondences (a), multiple plausible epipolar geometries exist, e.g. (b) and (c). However, true correspondences (c) are usually motion coherent while false ones (b) are not. In this paper, we design a network called LMCNet to explicitly utilize the motion coherence of correspondences via a global coherence fitting (d) and a local consistency extraction (e). Hence, LMCNet is able to robustly predict their inlier probabilities (f), where brighter color means higher inlier probability.

Besides the task-specific geometric model, true correspondences also conform to a more general motion model called *motion coherence*, which means that neighboring pixels share similar motions, while false correspondences are usually randomly scattered, as illustrated in Fig. 1 (a-c). Motion coherence supplements the task-specific geometric model and is key to determining true correspondences when multiple plausible geometric models exist.

To model motion coherence, existing works propose local [5, 29] or global handcrafted rules [23, 22] to find coherent correspondences. However, modeling motion coherence on sparse correspondences generated by local descriptors is challenging. First, unlike in tasks of dense correspondence

*Corresponding author
[1]Code and supplementary material can be found in the project page: https://liuyuan-pal.github.io/LMCNet/

estimation such as optical flow estimation [26, 6, 7], putative correspondences generated by local descriptors are discrete and sparse, which makes it much more difficult to estimate the underlying smooth motion field. Second, putative correspondences are usually distributed unevenly over an image because there are often many detected keypoints in textured regions and few keypoints in textureless regions. This uneven distribution makes it hard to find a uniform coherence constraint on the correspondences. Third, an observed scene may have a complex structure, such as an abrupt change of depth, so that the underlying motion fields are only piece-wise smooth, which brings about difficulty in finding motion boundaries [22]. Due to these challenges, existing works either need careful parameter tuning for different datasets [22, 23] or may fail when motion patterns are complex [29, 5].

We address these problems by proposing a neural network to learn the motion coherence property for correspondence pruning. Compared to handcrafted rules of motion coherence, neural networks have more powerful and flexible representational ability to learn more complex motion patterns from data and reliably detect motion boundaries.

Designing differentiable layers to capture the motion coherence property is the key to adopting a learning-based approach. Traditional global motion coherence models [22, 23] usually involve a non-differentiable iterative convex optimization solver, which cannot be used for training a network end-to-end. To address this issue, we propose a novel formulation of the motion coherence property via smooth function fitting on a graph of correspondences. We call this formulation **Laplacian Motion Fitting** (LMF) and further show that the proposed LMF has a simple closed-form solution by decomposition of graph Laplacian, which enables us to design a differentiable **Coherence Residual Layer**, called CR-Layer for abbreviation, to capture global motion coherence from putative correspondences.

Besides the global coherence model, true correspondences also have motion-consistent supporting correspondences in their local neighborhoods [5]. Based on this observation, we design a **Local Coherence Layer**, called LC-Layer for abbreviation, to extract local motion coherence from these neighboring supporting correspondences. By integrating both the global and local motion coherence layers, we design a network called **Laplacian Motion Coherence Network** (LMCNet), which takes coordinates or other optional features of correspondences as inputs and outputs the probability of each correspondence being an inlier.

We conducted extensive experiments to demonstrate the effectiveness of the proposed neural network in two tasks: relative camera pose estimation, and correspondence pruning of dynamic scenes. On both tasks, LMCNet achieves superior performance than other baseline methods, demonstrating its ability to robustly select inliers and potential to enhance object tracking or video object recognition.

Our contributions are as follows.

1. We proposed a novel formulation of motion coherence on sparse correspondences which has a simple closed-form solution by decomposition of graph Laplacian.
2. We proposed differentiable layers, which work together to robustly capture motion coherence of sparse putative correspondences.
3. We designed a neural network for correspondence pruning and demonstrate its effectiveness on the relative pose estimation problem and correspondence pruning of dynamic scenes.

2. Related Works

2.1. RANSAC-related correspondence pruning

RANSAC [18] and its variants [38, 52, 43, 1, 37, 2, 3, 14, 51, 12] are the standard methods to select true correspondences from putative correspondences by finding the largest subset which conforms to a task-specific geometric model. However, these methods may fail when multiple plausible geometric models all have a large amount of supporting correspondences. Some works address this problem by using techniques like marginalization [2, 3], degeneration detection [14], learning to sample hypotheses [9] or density estimation [51]. In this paper, we resort to another useful motion coherence property of inlier correspondences rather than solely relying on a task-specific geometric model.

2.2. Deep networks for correspondence pruning

With the advance of deep learning methods, pioneering works such as DSAC [8], PointCN [31] and DFE [39] demonstrate the feasibility of classifying correspondences by neural networks with coordinates as inputs. Follow-up works improve the architecture by inserting global clustering layer [60], attention mechanism [47, 13] or introducing new neighborhood definition [61]. These methods mainly focus on designing a permutation-equivariant operator on correspondences and treat the learning process as a black-box. In contrast, we explicitly incorporate the motion coherence property in a deep neural network to ensure that such property is learned during the training process.

2.3. Motion coherence

Motion coherence [59, 33] has been explored for decades in computer vision. There are many works [26, 6, 7, 11, 54, 34, 41, 42, 20, 33] focusing on applying motion coherence constraints on dense correspondence estimation tasks such as optical flow. However, fitting a smooth motion field on sparse correspondences is much harder. BF [23] and CODE [22] proposed a global motion coherence model on sparse correspondences. Some other works [29, 5] use local consistency to find motion-coherent correspondences.

These handcrafted rules or models achieve impressive performances but still have difficulty in handling complex motion patterns and need careful parameter tuning for different datasets. In our method, we propose a novel formulation of motion coherence by fitting a smooth function via graph Laplacian. We further design both global and local differentiable layers to capture motion coherence, which enable our network to learn more complex motion patterns from data than those handcrafted methods.

2.4. Image matching

Previously, works about image matching mainly focus on learning repetitive detectors [25, 58, 16, 40, 4, 17, 46] or discriminative descriptors [28, 27, 24, 49, 50, 30, 53, 15]. However, recent works [31] show that the performance bottleneck of image matching may be how to match these descriptors. Several deep learning-based models are proposed to learn to prune correspondences [31, 39, 60] or to match descriptors [44, 56]. Our method belongs to the category of deep learning-based correspondences pruners.

3. Method

We propose a novel architecture LMCNet for correspondence pruning. Given N putative correspondences $\{\boldsymbol{c}_i = (x_i, y_i, u_i, v_i)|i = 1, ..., N\}$ and their optional d_0-dim features $\{\boldsymbol{f}_i \in \mathbb{R}^{d_0}\}$, where (x_i, y_i) and (u_i, v_i) are the image coordinates of two corresponding keypoints, our goal is to estimate the probability $\{p_i\}$ that a putative correspondence $\boldsymbol{c}_i$ is a true correspondence. In the rest of this section, we first introduce our new formulation of motion coherence, called *Laplacian Motion Fitting* in Sec. 3.1. Then, the key components of LMCNet, i.e. Coherence Residual Layer and Local Coherence Layer, are elaborated in Sec. 3.2 and Sec. 3.3. In the end, we describe the whole architecture in Sec. 3.4 and some implementation details in Sec. 3.5.

3.1. Laplacian Motion Fitting

Motion coherence means that true correspondences have similar motions to each other, while false correspondences scatter randomly. Most commonly-used models [22, 21, 59] utilize this property by first recovering the underlying continuous smooth motion field from putative correspondences. Then, true and false correspondences can be distinguished according to their deviations from the recovered motion field. However, in order to recover the underlying motion field, such formulations usually involve an iterative convex optimization solver which is non-differentiable. To address this, we propose a novel formulation of motion coherence by estimating a set of smooth discrete motions on a graph that encodes the adjacency of the putative correspondences. We show that this formulation allows a simple closed-form solution via decomposition of graph Laplacian, which can be used for constructing a differentiable layer in a network.

We construct a graph $G = \{V, E\}$ where the nodes in V represent all the putative correspondences, and E includes the edges from every correspondence to its k-nearest neighbors according to their coordinate distance $d_{i,j} = \|\boldsymbol{c}_i - \boldsymbol{c}_j\|_2$. We compute the associated weights on the edges by $w_{i,j} = \exp(-d_{i,j}^2/\sigma^2)$, where σ is a predefined constant, and $w_{i,i} = 0$ for all nodes. Then, we define the adjacency matrix as $\boldsymbol{A} = [w_{i,j}]$, the degree matrix as $\boldsymbol{D} = diag([d_i = \sum_j w_{i,j}])$ and the Laplacian matrix as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$. Here, we define a matrix or a vector $\boldsymbol{v}$ by $\boldsymbol{v} = [v_{i,j}]$ whose components are $v_{i,j}$ and use $diag(\boldsymbol{v})$ to denote a diagonal matrix whose diagonal elements are the components of $\boldsymbol{v}$.

For every correspondence, we compute its motion by $\{\boldsymbol{m}_i = (m_{x,i}, m_{y,i}) = (u_i - x_i, v_i - y_i)\}$. Our goal is to estimate a set of *smooth* motions $\{\boldsymbol{s}_i = (s_{x,i}, s_{y,i})\}$ which are as consistent with the input motions $\{\boldsymbol{m}_i\}$ as possible. We formulate the problem as follows,

$$\underset{\{\boldsymbol{s}_i|i=1,...,N\}}{minimize} \sum_i^N \|\boldsymbol{s}_i - \boldsymbol{m}_i\|_2^2 + \frac{1}{2}\eta \sum_{i,j} w_{i,j} \|\boldsymbol{s}_i - \boldsymbol{s}_j\|_2^2, \quad (1)$$

where $\|\boldsymbol{s}_i - \boldsymbol{m}_i\|_2^2$ penalizes the deviation of the estimated motion $\boldsymbol{s}_i$ from the input motion $\boldsymbol{m}_i$, η is a predefined constant, and $w_{i,j}\|\boldsymbol{s}_i - \boldsymbol{s}_j\|_2^2$ is a smoothness cost which penalizes the motion variation between two neighboring correspondences $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ according to the weight $w_{i,j}$.

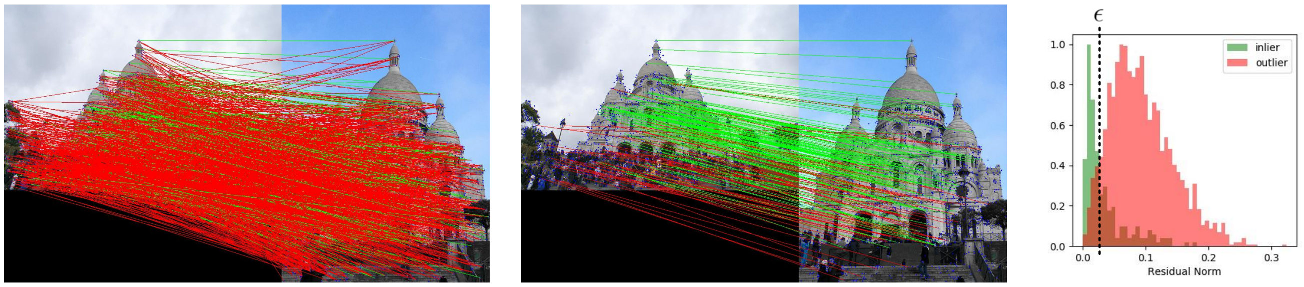
By aggregating $\boldsymbol{s}_i$ and $\boldsymbol{m}_i$ into a matrix form $\boldsymbol{s} = [\boldsymbol{s}_i] \in \mathbb{R}^{N\times 2}$ and $\boldsymbol{m} = [\boldsymbol{m}_i] \in \mathbb{R}^{N\times 2}$, we can rewrite Problem (1) as follows,

$$\underset{\boldsymbol{s}}{minimize}\ \mathrm{Tr}((\boldsymbol{s} - \boldsymbol{m})^\intercal(\boldsymbol{s} - \boldsymbol{m})) + \eta\,\mathrm{Tr}(\boldsymbol{s}^\intercal \boldsymbol{L}\boldsymbol{s}), \quad (2)$$

where $\boldsymbol{s}^\intercal \boldsymbol{L}\boldsymbol{s}$ is used as a regularization term because it measures the smoothness of the graph signal $\boldsymbol{s}$ [35]. Problem (2) has a closed-form solution as stated in the following proposition; we leave the proof and its connection to the previous motion coherence theories [59, 33, 22] in the supplementary material.

Proposition 1. *Let the eigenvalue decomposition of the Laplacian matrix $\boldsymbol{L}$ be $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal$, where $\boldsymbol{\Lambda} = diag([\lambda_i])$ is a diagonal matrix of eigenvalues λ_i and columns of $\boldsymbol{U}$ are the associated eigenvectors. Then the solution to Problem (2) is $\boldsymbol{s} = \boldsymbol{U}diag([1/(1+\eta\lambda_i)])\boldsymbol{U}^\intercal\boldsymbol{m}$.*

Denote $\boldsymbol{R}(\eta) \equiv \boldsymbol{U}diag([1/(1+\eta\lambda_i)])\boldsymbol{U}^\intercal$. Then, the residuals between the smoothed motions and the input motions are computed by $\boldsymbol{R}(\eta)\boldsymbol{m} - \boldsymbol{m}$. Since only true correspondences can be well fitted by the smooth motions $\boldsymbol{R}(\eta)\boldsymbol{m}$ while false correspondences cannot, the residuals of true correspondences will be significantly smaller than those of false correspondences. Hence, the true correspondences can be distinguished from the false ones by thresholding on the L2-norms of the residual motions. The whole



(a) Input correspondences (b) Output of LMF (c) Histogram of residuals

Figure 2. (a) 2000 putative correspondences. Green correspondences are true while red ones are false. (b) Output correspondences of LMF. (c) Distribution of motion residual norms of inliers (green) and outliers (red). Histograms are normalized with 1 as the max values.

process, called *Laplacian Motion Fitting* (LMF), is summarized in Algorithm 1, and an example is shown in Fig. 2.

```
Algorithm 1: Laplacian Motion Fitting
  Data: Input correspondences $\{\boldsymbol{c}_i = (x_i, y_i, u_i, v_i)\}$,
        smooth strength $\eta$ and inlier threshold $\epsilon$
  Result: Probability of being inliers $\{p_i\}$
1 Compute Laplacian matrix $\boldsymbol{L}$ on $\{\boldsymbol{c}_i\}$;
2 Eigen decomposition of $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\intercal$;
3 Compute motions $\boldsymbol{m} = [\boldsymbol{m}_i] = [(u_i - x_i, v_i - y_i)]$;
4 Compute smoothed motions $\boldsymbol{s} = [\boldsymbol{s}_i] = \boldsymbol{R}(\eta)\boldsymbol{m}$;
5 Find inliers $\{p_i = 1\ if\ \|\boldsymbol{s}_i - \boldsymbol{m}_i\|_2 \leq \epsilon\ else\ 0\}$;
```

Computational complexity. For a graph Laplacian matrix, its eigenvalues λ_i are non-negative. As λ_i increases, the $1/(1+\eta\lambda_i)$ becomes sufficiently small and so negligible. In light of this, we can just use the k_e smallest eigenvalues and their associated eigenvectors for the computation of $\boldsymbol{R}(\eta)$, which means changing $\boldsymbol{U} \in \mathbb{R}^{N\times N}$ to $\mathbb{R}^{N\times k_e}$. This simplification essentially lowers the computational complexity of $\boldsymbol{R}(\eta)$ from $O(n^2)$ to $O(n)$.

Graph construction. In our implementation, the graph is constructed by connecting every correspondence with its k-nearest neighbors in the coordinate space $\boldsymbol{c}_i \in \mathbb{R}^4$, which is the "bilateral space" proposed in CODE [22]. This bilateral space allows finding a piece-wise smooth motion field rather than a global smooth motion field. Meanwhile, the graph construction can be quite flexible. We can also adopt the affine compatibility proposed in [61] for graph construction to utilize detected affine transformations.

3.2. Coherence Residual Layer

So far, we have presented the LMF algorithm on motions for correspondences pruning. However, due to the sparsity and uneven distributions of putative correspondences, expert knowledge is needed for carefully tuning the threshold ϵ and the smoothness strength η in order to achieve a better performance. To avoid this, we incorporate the LMF within a learning framework so that we can utilize the powerful representational ability of neural networks to automatically learn complex motion patterns from training data.

Instead of directly thresholding on residual motions, we apply the LMF on the features $\boldsymbol{f}_l \in \mathbb{R}^{N\times d}$ of correspondences, which are extracted by a neural network, to find a set of smoothed features $\boldsymbol{f}'_l \in \mathbb{R}^{N\times d}$. Then, discriminative features can be extracted from the residuals $\boldsymbol{f}_l - \boldsymbol{f}'_l$ for the classification of correspondences. We consider the problem of estimating smooth features by,

$$\underset{\boldsymbol{f}'_l}{minimize}\ \mathrm{Tr}((\boldsymbol{f}'_l - \boldsymbol{f}_l)^\intercal(\boldsymbol{f}'_l - \boldsymbol{f}_l)) + \eta\,\mathrm{Tr}(\boldsymbol{f}'^\intercal_l \boldsymbol{L}\boldsymbol{f}'_l), \quad (3)$$

which is similar to Problem (2) by replacing the input motions $\boldsymbol{m}$ with the features $\boldsymbol{f}_l$. However, the features $\boldsymbol{f}_l$ are extracted by the neural network via multiple layers of abstractions and thus Problem (3) is much more generalized than fitting a single smooth motion field in Problem (2). These features can simply be motions if necessary or any other more complex coherent attributes like local affine transformations [23, 21], which are implicitly learned by the neural network during training.

The solution to Problem (3) is also given by $\boldsymbol{f}'_l = \boldsymbol{R}(\eta)\boldsymbol{f}_l$ according to Proposition 1. Based on this, we formulate a new layer called *Coherence Residual Layer* (CR-Layer) by,

$$\boldsymbol{f}_{l+1} = \mathrm{ContextNorm}(\boldsymbol{f}_l - \boldsymbol{R}(\eta)\boldsymbol{f}_l), \quad (4)$$

where the $\boldsymbol{f}_{l+1}$ are output features and ContextNorm [31] contains a fully-connected operator and an instance normalization operator for feature extraction. The forward pass of CR-Layer implicitly solves Problem (3) by $\boldsymbol{R}(\eta)\boldsymbol{f}_l$ and extracts features from the residuals $\boldsymbol{f}_l - \boldsymbol{R}(\eta)\boldsymbol{f}_l$. Since the whole process only involves matrix multiplication, CR-Layer is differentiable and therefore can be incorporated in a network. Meanwhile, we also learn the smoothness strength η from data by making it a trainable parameter.

3.3. Local Coherence Layer

Besides fitting a global smooth function on correspondences, another important observation of motion coherence

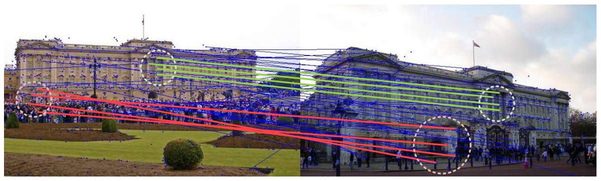


Figure 3. True correspondences (green) have motion-consistent supporting neighbors while neighbors of false correspondences (red) scatter randomly.

is that true correspondences tend to have motion-consistent supporting correspondences in their neighborhoods, while false correspondences do not have such supporting correspondences. A typical example is shown in Fig. 3. Based on this observation, we introduce a new layer called *Local Coherence Layer* (LC-Layer).

Specifically, given the feature $\boldsymbol{f}_{l,i}$ of the i-th correspondence, we first compute the feature differences between the correspondence and its neighbors $(i,j) \in E$ by $\boldsymbol{f}_{l,i} - \boldsymbol{f}_{l,j}$. The feature differences measure the local consistency in the neighborhood of the i-th correspondence. Then, the LC-Layer is defined by,

$$\boldsymbol{f}_{l+1,i} = \underset{(i,j)\in E}{\mathrm{MaxPool}} \left(\{ \mathrm{MLP}(\boldsymbol{f}_{l,i} - \boldsymbol{f}_{l,j}) \} \right), \tag{5}$$

where MLP is a multi-layer perceptron, MaxPool is a max-pooling operator which pools on all neighboring features to get a single feature vector, and $\boldsymbol{f}_{l+1,i}$ is the output feature of this correspondence which contains information about the local coherence in its neighborhood.

3.4. Architecture

Overview. The architecture of our network is illustrated in Fig. 4. Given the input correspondences, the Geometry Embedding is a ContextNorm layer which processes the input coordinates $\boldsymbol{c} \in \mathbb{R}^{N\times 4}$ to produce d-dimensional features $\boldsymbol{f}_1 \in \mathbb{R}^{N\times d}$. If there are optional features associated with correspondences, we process these features with another ContextNorm layer (Feature Embedding), and we add it back to the output of Geometry Embedding. Then, the features are processed by 4 blocks called LMCBlocks, which are the main feature extraction module of LMCNet. Finally, the output features of LMCBlocks $\boldsymbol{f}_{out} \in \mathbb{R}^{N\times d}$ are processed by a probability predictor, which simply consists of a fully connected layer and a sigmoid function, to produce the inlier probability $\boldsymbol{p} = [p_i] \in \mathbb{R}^N$.

LMCBlock. The structure of a LMCBlock is illustrated in Fig. 5. The LC-Layer is placed on the top to extract some useful information from the neighborhoods of correspondences and the CR-Layer is placed after several other layers so that its inputs are at a higher level of abstraction thus more flexible. Except for the proposed LC-Layers and CR-Layers, a LMCBlock also includes two ContextNorm layers and a clustering layer, which are used for extracting other information such as the underlying epipolar geometry. The clustering layer used here is proposed in [60], which is implemented by differentiable pooling to k_c clusters (DiffPool), order-aware filtering among clusters (OAFilter) and unpooling to original correspondences (DiffUnpool). All layers in the LMCBlock take d-dimensional features as inputs and also output d-dimensional features. Thus, a skip connection is applied to add the inputs to the output features on all layers. These skip connections are important because both the CR-Layer and the LC-Layer only retain relative information between correspondences while such skip connections preserve the absolute information. We also use a bottleneck structure in LC-Layers, which encodes the input features into a lower dimension d_l, then extracts the local consistency features on the low-dimensional features, and finally lift the dimension back to d.

3.5. Implementation Details

In our implementation, coordinates of correspondences are normalized by camera intrinsic matrices if available, or are normalized to the range $[-1, 1]$ using the input image size. For the construction of the adjacency matrix, we use $k = 8$ neighbors and $\sigma = 0.1$. We use the normalized graph Laplacian matrix $\hat{\boldsymbol{L}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$ to compute $\boldsymbol{R}(\eta)$ and only the smallest $k_e = 32$ eigenvalues and the associated eigenvectors are selected. In all the CR-Layers, η is initialized to 10.0. Both the feature dimension d and the cluster number n_c in LMCBlock are 128, and the bottleneck feature dimension d_l used in LC-Layers is 8. A correspondence is determined as an inlier if its predicted inlier probability is larger than 0.95. More details about the architecture and the training process can be found in the supplementary material. When implementing the LMF algorithm, we use $k_e = 128$ eigenvalues, the smoothness strength $\eta = 10.0$ and the threshold $\epsilon = 0.025$.

Loss. For image pairs of dynamic scenes, we use the binary cross entropy loss ℓ_{cls} for training. For image pairs in relative pose estimation, we use an additional geometric loss ℓ_{geom} [60, 39, 19], in which we estimate the essential matrix by the weighted 8 points algorithm [19] and compute distances from ground-truth correspondences to estimated epipolar lines as loss.

4. Experiments

4.1. Evaluation protocols

To demonstrate the effectiveness of our methods, we evaluate three models, which are the LMF algorithm, the LMCNet with only coordinates as input and the LMCNet with both coordinates and local descriptors as inputs (LMCNet-F). We report performance on relative pose estimation and correspondence pruning of dynamic scenes.

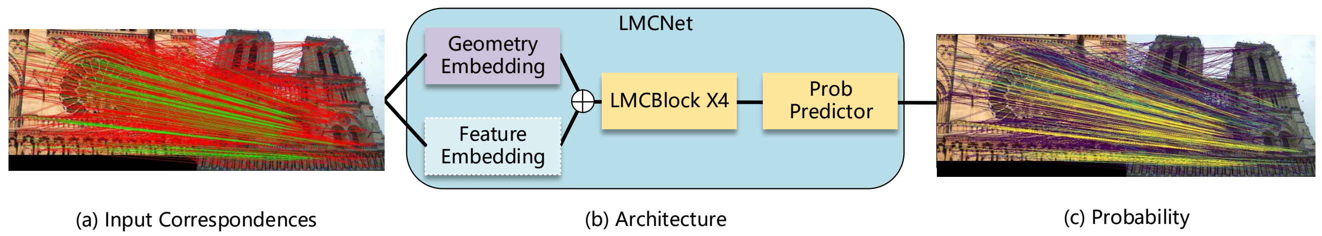


Figure 4. (a) Input correspondences. Red represents false correspondences while green represents true ones. (b) Architecture of LMCNet. The feature embedding is optional. (c) Output probability of being inliers. Brighter color means higher probability.

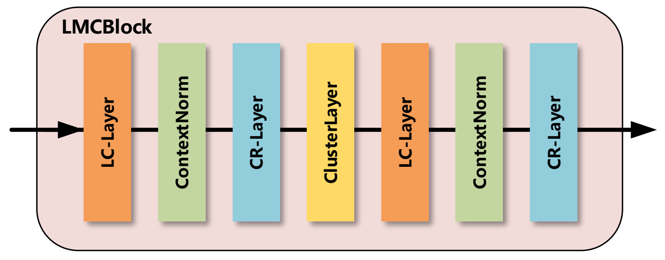


Figure 5. A LMCBlock consists of 7 layers.

Method	AUC@5°	AUC10°	AUC20°
Ratio test	24.09	40.71	58.14
MAGSAC [2]	28.24	44.86	61.53
LPM [29]	10.48	18.91	29.26
GMS [5]	19.05	32.35	46.79
CODE [22]	16.99	30.23	43.85
LMF	16.91	29.49	43.44
PointCN [31]	27.39	44.61	61.22
AttenCN [47]	29.08	48.13	65.49
OANet [60]	29.12	48.28	65.37
SuperGlue [44]	30.49	51.29	69.72
LMCNet	34.62	53.86	70.53
LMCNet-F	**35.91**	**55.68**	**72.35**

Table 1. Pose AUCs on the YFCC100M dataset. All methods use the putative correspondences generated by nearest neighborhood matching of SIFT descriptors.

Method	AUC@5°	AUC10°	AUC20°
Ratio test	4.51	11.62	23.02
LPM [29]	2.81	7.40	15.36
GMS [5]	4.36	11.08	21.68
CODE [22]	3.52	8.91	18.32
LMF	3.34	8.85	18.04
PointCN [31]	5.64	14.88	29.32
AttenCN [47]	5.97	15.69	30.98
OANet [60]	5.94	15.79	31.03
LMCNet	6.77	17.14	32.55
LMCNet-F	**8.86**	**19.64**	**34.96**

Table 2. Pose AUCs of LMCNet and other baseline methods on the indoor SUN3D dataset.

Relative pose datasets. The outdoor YFCC100M [48] dataset and the indoor SUN3D [57] dataset are used for relative pose estimation. We use the same train-test split as [60]. Input putative correspondences are generated from nearest neighborhood matching of 2000 SIFT [25] descriptors on every image. We regard correspondences with small distances (≤ 0.01 in normalized image coordinates) to their ground-truth epipolar lines as true correspondences. For a image pair, an essential matrix is estimated by RANSAC on the predicted true correspondences and is then decomposed to a rotation and a translation.

Dynamic scene datasets. The dynamic scene dataset contains images with dynamic objects. We use the DETRAC dataset [55] for evaluation. The DETRAC dataset contains images in traffic monitoring, so the background is fixed and the main dynamic objects are cars. We extract 2048 SuperPoint [16] features on every image and extract putative correspondences by nearest neighborhood matching. Since there are only annotated bounding boxes of different car instances in the dataset, we regard a correspondence as a true correspondence if it connects two bounding boxes of the same instance while a correspondence is false if it connects different instances or it connects an instance with background, as shown in Fig. 7. We use the provided train-test split with 60 sequences (sample 30k image pairs) for training and 40 sequences (sample 4k image pairs) for testing. We also include some qualitative results on the DAVIS [36] dataset in the supplementary material.

Metrics. On the task of relative pose estimation, we compute the Area Under the Curve (AUC) of the pose accuracy curve at thresholds 5°, 10° and 20°, the same as used in [44]. On the task of correspondence pruning on dynamic image pairs, we report the precision, recall and F1 scores.

4.2. Results on relative pose estimation

Baselines. We consider both the traditional handcrafted pruners, including LPM [29], GMS [5] and CODE [22], and learning-based pruners, including PointCN [31], AttenCN [47] and OANet [60] as baseline methods. We also include the results of SIFT+SuperGlue [44] on the

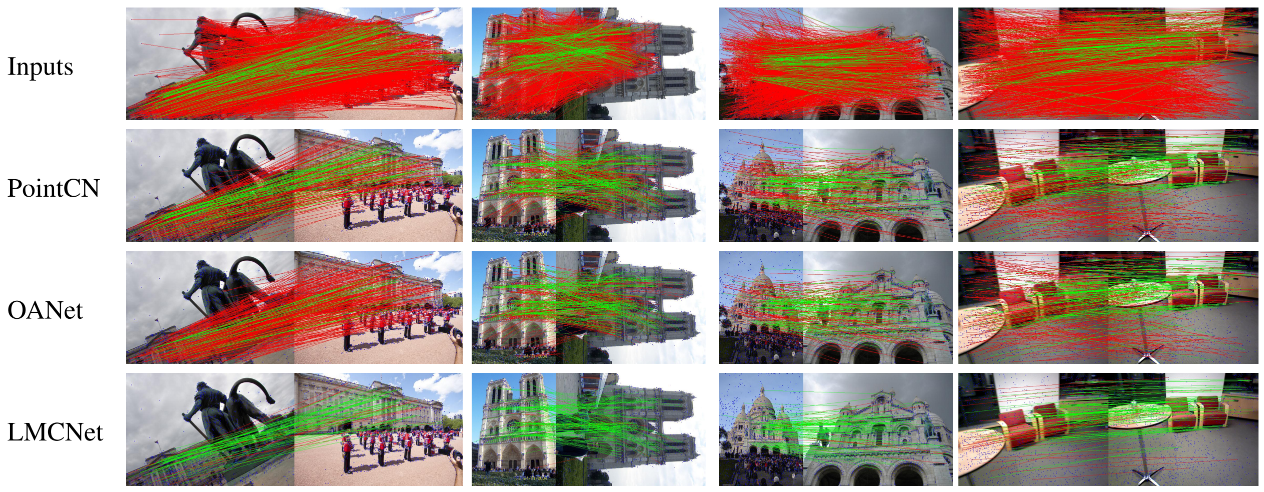


Figure 6. Input correspondences (Row 1), output of PointCN [31] (Row 2), OANet [60] (Row 3) and LMCNet (Row 4). Green correspondences are correct (small distances to true epipolar lines) while red correspondences are incorrect.

YFCC100M dataset. For implementation, we directly use the released code of GMS [5] and LPM [29], and we re-implement CODE [22] which takes only coordinates of correspondences as inputs. For learning-based methods, we directly use the released code and these methods are trained on the same training set with the same learning rates, except for SuperGlue [44], of which we directly use the results reported in their supplementary material [44].

Results. The quantitative results are reported in Table 1 and Table 2. Some qualitative results are provided in Fig. 6. The results show that LMCNet outperforms all the baseline methods on all pose metrics. Further adding descriptors as inputs (LMCNet-F) leads to about 1.5%-2% improvements on both datasets. Among the handcrafting methods, the results of LMF are similar to the global method CODE [22] but are inferior to the local method GMS [5]. This is due to the unevenly-distributed correspondences which make it hard for global methods to determine a uniform global threshold for pruning. However, learning the motion coherence via CR-Layers enables the network to handle complex motion patterns and thus achieves a better performance.

4.3. Results on dynamic scenes

On the DETRAC dataset, we compare LMCNet with the same baseline methods as used for relative pose estimation. The quantitative results in Table 3 show that LMCNet outperforms all the baseline methods in all the metrics. From qualitative results in Fig. 7, we can see that LMCNet is able to robustly find motion-coherent correspondences while baseline methods may include some false correspondences or neglect some sparse true correspondences. Note the displacement of motion is up to several hundreds pixels and we are unable to produce reasonable results using a optical-flow based method for this task.

Method	Precision	Recall	F1-Score
LPM [29]	85.12	52.27	63.47
GMS [5]	84.89	76.92	80.01
CODE [22]	83.63	77.86	79.95
LMF	82.84	78.66	79.35
PointCN [31]	84.27	89.34	86.26
AttenCN [47]	85.34	88.53	86.89
OANet [60]	82.70	87.66	84.53
LMCNet	**87.23**	**91.16**	**88.79**

Table 3. Precision, recall and F1 score of LMCNet and other baseline models on the DETRAC dataset.

#Block	#CR-Layer	#LC-Layer	k_e	AUC@ 5°	AUC@ 10°	AUC@ 20°
3	0	0	32	29.38	48.00	64.86
3	2	0	32	32.72	51.66	69.49
3	2	2	32	34.62	53.86	70.53
3	1	1	32	31.49	50.73	67.65
1	2	2	32	33.18	52.32	68.92
5	2	2	32	34.31	54.22	71.07
3	2	2	16	32.58	51.72	69.51
3	2	2	32	34.62	53.86	70.53
3	2	2	64	34.71	53.79	70.63

Table 4. Ablation studies of LMCNet on the YFCC100M dataset. "#Block" means the number of LMCBlock used in the model. "#CR-Layer" and "#LC-Layer" means the number of these two layers used in every LMCBlock.

4.4. Analysis

Ablation studies. To demonstrate the effectiveness of LC-Layers and CR-Layers, we perform ablation studies on the YFCC100M dataset and report the results in Table 4.

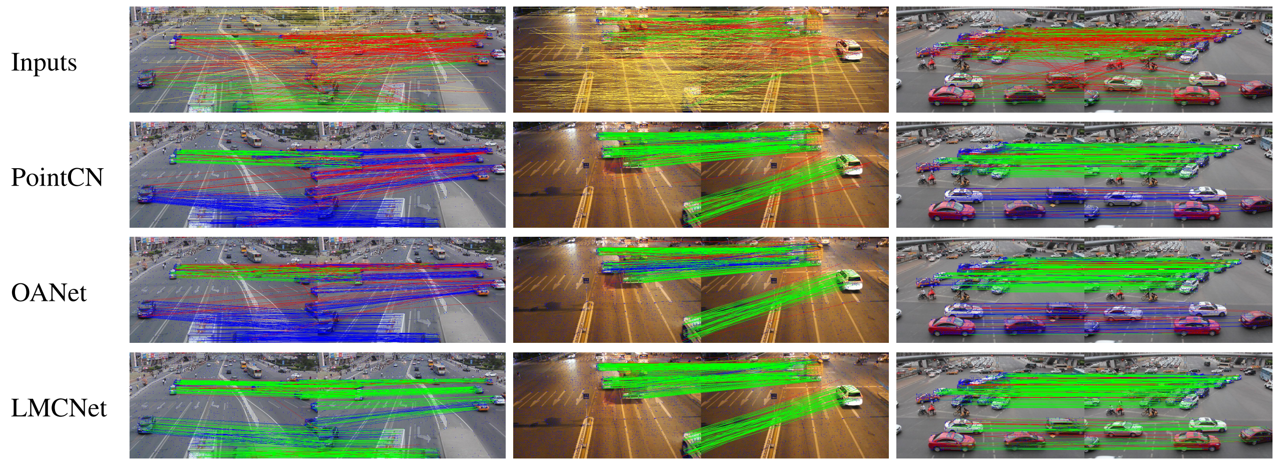


Figure 7. Input correspondences (Row 1), output correspondences of PointCN [31] (Row 2), OANet [60] (Row 3) and LMCNet (Row 4) on the DETRAC dataset. Yellow means that correspondences are in the background, Green means correct, Red means false positive and blue color means false negative. Note the inputs include both foreground and background correspondences. For clear visualization, we randomly draw 512 correspondences on all image pairs and neglect the background correspondences in Row 2, 3 and 4.

The baseline model replaces all CR-Layers and LC-Layers with ContextNorm layers with almost equivalent computational complexity and parameter numbers. The results show that adding CR-Layers brings about 2-5% improvements to all pose AUCs, and further adding LC-Layers results in 1-2% improvements. We also show how the number of block and number of layers used in every block affect the performance of LMCNet. Adding more blocks results in slight improvements while reducing the number of CR-Layers and LC-Layers degrades the performance significantly.

Different numbers of eigenvectors in CR-Layers. As mentioned in Sec. 3.1, we can use k_e smallest eigenvalues and their associated eigenvectors in the computation of CR-Layers. To show how the number of eigenvectors k_e affects performance, we train LMCNet with different k_e and report the results in Table 4. The results show that using only 16 eigenvectors decreases the performance remarkably compared to using 32 eigenvectors. However, increasing the number from 32 to 64 does not bring about a significant performance improvement. In light of this, we use 32 eigenvectors in LMCNet for computational efficiency.

Compatibility with learning-based descriptors and matchers. In Table 5, we report performance with or without LMCNet on the SUN3D dataset and the ScanNet dataset using SuperPoint [16] as the local descriptor and SuperGlue [44] as the matcher. In this experiment, SuperGlue and SuperPoint only uses 1024 keypoints but still achieve a better performance than using 2000 SIFT features. On the ScanNet dataset, we use exactly the same experimental setting as used in SuperGlue [44] and finetune the LMCNet model with 20k image pairs from the ScanNet training set. Results show that applying LMCNet as a pruner can further improve the accuracy of estimated poses in all cases on both datasets. Qualitative results and details about this experiment can be found in the supplementary material.

Dataset	Desc	Matcher	Pruner	AUC@ 5°	AUC@ 10°	AUC@ 20°
SUN3D	SP	NN	/	4.66	12.49	25.39
	SP	NN	LMCNet	6.76	17.25	32.90
	SP	SG	/	7.09	17.82	33.26
	SP	SG	LMCNet	8.13	20.36	37.55
ScanNet	SP	NN	/	9.43	21.53	36.40
	SP	NN	LMCNet	12.32	28.15	47.13
	SP	SG	/	16.16	33.81	51.84
	SP	SG	LMCNet	16.41	35.33	54.98

Table 5. Pose AUCs of LMCNet on the indoor SUN3D dataset with the SuperPoint (SP) [16] descriptor and the SuperGlue (SG) [44] matcher. NN means the nearest neighbor matcher.

5. Conclusion

In this paper, we designed a novel architecture LMCNet to learn the motion coherence property for correspondence pruning. We proposed a novel formulation LMF of the motion coherence by fitting a smooth function via decomposition of graph Laplacian, which enables us to design a differentiable CR-Layer to capture the global motion coherence in a neural network. Furthermore, we also designed a LC-Layer to extract local coherence information from neighborhoods of correspondences. Integrating these two coherence layers, the proposed LMCNet achieves superior performances on relative pose estimation and correspondence pruning of dynamic scenes.

Acknowledgement. This work was supported by the General Research Fund (GRF) of the Research Grant Council of Hong Kong (17210718).

References

[1] Daniel Barath and Jiří Matas. Graph-cut ransac. In *CVPR*, 2018.

[2] Daniel Barath, Jiri Matas, and Jana Noskova. Magsac: Marginalizing sample consensus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10197–10205, 2019.

[3] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. Magsac++, a fast, reliable and accurate robust estimator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1304–1312, 2020.

[4] Axel Barroso-Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key.net: Keypoint detection by handcrafted and learned cnn filters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5836–5844, 2019.

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

[6] Michael J Black and Padmanabhan Anandan. A framework for the robust estimation of optical flow. In *ICCV*, 1993.

[7] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1):75–104, 1996.

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

[9] Eric Brachmann and Carsten Rother. Neural- Guided RANSAC: Learning where to sample model hypotheses. In *ICCV*, 2019.

[10] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.

[11] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *Transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2010.

[12] Luca Cavalli, Viktor Larsson, Martin Ralf Oswald, Torsten Sattler, and Marc Pollefeys. Adalam: Revisiting handcrafted outlier detection. *arXiv preprint arXiv:2006.04250*, 2020.

[13] Zhi Chen, Fan Yang, and Wenbing Tao. Gla-net: An attention network with guided loss for mismatch removal. *ArXiv*, 2019.

[14] Ondrej Chum, Tomas Werner, and Jiri Matas. Two-view geometry estimation unaffected by a dominant plane. In *CVPR*, 2005.

[15] François Darmon, Mathieu Aubry, and Pascal Monasse. Learning to guide local feature matches. *arXiv preprint arXiv:2010.10959*, 2020.

[16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[17] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. *arXiv preprint arXiv:1905.03561*, 2019.

[18] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

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

[20] Xinghui Li, Kai Han, Shuda Li, and Victor Adrian Prisacariu. Dual-resolution correspondence networks. *arXiv preprint arXiv:2006.08844*, 2020.

[21] Wen-Yan Lin, Siying Liu, Yasuyuki Matsushita, Tian-Tsong Ng, and Loong-Fah Cheong. Smoothly varying affine stitching. In *CVPR 2011*, pages 345–352. IEEE, 2011.

[22] Wen-Yan Lin, Fan Wang, Ming-Ming Cheng, Sai-Kit Yeung, Philip HS Torr, Minh N Do, and Jiangbo Lu. Code: Coherence based decision boundaries for feature correspondence. *Transactions on pattern analysis and machine intelligence*, 40(1):34–47, 2017.

[23] Wen-Yan Daniel Lin, Ming-Ming Cheng, Jiangbo Lu, Hongsheng Yang, Minh N Do, and Philip Torr. Bilateral functions for global motion modeling. In *ECCV*, 2014.

[24] Yuan Liu, Zehong Shen, Zhixuan Lin, Sida Peng, Hujun Bao, and Xiaowei Zhou. Gift: Learning transformation-invariant dense visual descriptors via group cnns. In *Advances in Neural Information Processing Systems*, pages 6992–7003, 2019.

[25] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91—110, 2004.

[26] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings DARPA image Understanding*, page 121430, 1981.

[27] Zixin Luo, Tianwei Shen, Lei Zhou, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Contextdesc: Local descriptor augmentation with cross-modality context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2527–2536, 2019.

[28] Zixin Luo, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang, and Long Quan. Geodesc: Learning local descriptors by integrating geometry constraints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 168–183, 2018.

[29] Jiayi Ma, Ji Zhao, Junjun Jiang, Huabing Zhou, and Xiaojie Guo. Locality preserving matching. *International Journal of Computer Vision*, 127(5):512–531, 2019.

[30] Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems*, pages 4826–4837, 2017.

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

[32] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *Transactions on robotics*, 31(5):1147–1163, 2015.

[33] Andriy Myronenko, Xubo Song, and Miguel A Carreira-Perpinán. Non-rigid point set registration: Coherent point drift. In *NeurIPS*, 2007.

[34] Manjunath Narayana, Allen Hanson, and Erik Learned-Miller. Coherent motion segmentation in moving camera videos using optical flow orientations. In *CVPR*, 2013.

[35] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[36] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *ArXiv*, 2017.

[37] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. Usac: a universal framework for random sample consensus. *Transactions on pattern analysis and machine intelligence*, 35(8):2022–2038, 2012.

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

[39] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018.

[40] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: Repeatable and reliable detector and descriptor. *arXiv preprint arXiv:1906.06195*, 2019.

[41] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. Efficient neighbourhood consensus networks via submanifold sparse convolutions. *arXiv preprint arXiv:2004.10566*, 2020.

[42] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *Advances in Neural Information Processing Systems*, pages 1651–1662, 2018.

[43] Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

[44] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020.

[45] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *CVPR*, 2018.

[46] Yafei Song, Ling Cai, Jia Li, Yonghong Tian, and Mingyang Li. Sekd: Self-evolving keypoint detection and description. *arXiv preprint arXiv:2006.05077*, 2020.

[47] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. Acne: Attentive context normalization for robust permutation-equivariant learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11286–11295, 2020.

[48] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[49] Yurun Tian, Axel Barroso-Laguna, Tony Ng, Vassileios Balntas, and Krystian Mikolajczyk. Hynet: Local descriptor with hybrid similarity measure and triplet loss. *arXiv preprint arXiv:2006.10202*, 2020.

[50] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. Sosnet: Second order similarity regularization for local descriptor learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11016–11025, 2019.

[51] Lokender Tiwari and Saket Anand. Dgsac: Density guided sampling and consensus. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 974–982. IEEE, 2018.

[52] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156, 2000.

[53] Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *arXiv preprint arXiv:2006.13566*, 2020.

[54] Sebastian Volz, Andres Bruhn, Levi Valgaerts, and Henning Zimmer. Modeling temporal coherence for optical flow. In *CVPR*, 2011.

[55] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 2020.

[56] Olivia Wiles, Sebastien Ehrhardt, and Andrew Zisserman. D2d: Learning to find good correspondences for image matching and manipulation. *arXiv preprint arXiv:2007.08480*, 2020.

[57] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013.

[58] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.

[59] Alan L Yuille and Norberto M Grzywacz. A mathematical analysis of the motion coherence theory. *International Journal of Computer Vision*, 3(2):155–175, 1989.

[60] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *CVPR*, 2019.

[61] Chen Zhao, Zhiguo Cao, Chi Li, Xin Li, and Jiaqi Yang. Nm-net: Mining reliable neighbors for robust feature correspondences. In *CVPR*, 2019.