Multi-View Optimization of Local Feature Geometry

Mihai Dusmanu[1], Johannes L. Schönberger[2], and Marc Pollefeys[1,2]

[1] Department of Computer Science, ETH Zürich [2] Microsoft

Abstract. In this work, we address the problem of refining the geometry of local image features from multiple views without known scene or camera geometry. Current approaches to local feature detection are inherently limited in their keypoint localization accuracy because they only operate on a single view. This limitation has a negative impact on downstream tasks such as Structure-from-Motion, where inaccurate keypoints lead to large errors in triangulation and camera localization. Our proposed method naturally complements the traditional feature extraction and matching paradigm. We first estimate local geometric transformations between tentative matches and then optimize the keypoint locations over multiple views jointly according to a non-linear least squares formulation. Throughout a variety of experiments, we show that our method consistently improves the triangulation and camera localization performance for both hand-crafted and learned local features.

Keywords: 3D reconstruction, local features

1 Introduction

Local image features are one of the central blocks of many computer vision systems with numerous applications ranging from image matching and retrieval to visual localization and mapping. Predominantly, local feature extraction and matching are the first stages in these systems with high impact on their final performance in terms of accuracy and completeness [37]. The main advantages of local features are their robustness, scalability, and efficient matching, thereby enabling large-scale 3D reconstruction [18] and localization [21].

Handcrafted local feature approaches generally focus on low-level structures for detection [16,23]. Despite the typically accurate keypoint localization of these methods, they are easily perturbed by appearance variations such as day-to-night or seasonal changes, as shown by Sattler *et al.* [35]. To achieve a better robustness against viewpoint and appearance changes, recent methods turned to convolutional neural networks (CNNs) for local feature detection and description [27,10,12,30]. However, this comes at the cost of a poorer keypoint localization, mainly caused by relying on larger receptive fields and feature map down-sampling through pooling or strided convolutions.

Moreover, both traditional and CNN-based methods only exploit a single view, as feature detection and description is run independently on each image.

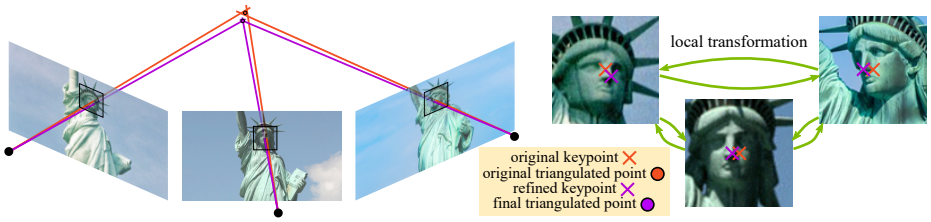


Fig. 1. Multi-view keypoint refinement. The proposed method estimates local transformations between multiple tentative views of a same feature and uses them to refine the 2D keypoint location, yielding more accurate and complete point clouds.

Even for low-level detectors, there is no inherent reason why detections would be consistent across multiple views, especially under strong viewpoint or appearance changes. While some recent works [15,17,44] consider multiple views to improve the feature matching step, to the best of our knowledge, no prior work exploits multiple views to improve the feature detection stage for more accurate keypoints.

In this paper, we propose a method for optimizing the geometry of local features by exploiting multiple views without any prior knowledge about the camera geometry or scene structure. Our proposed approach first uses a patch-alignment CNN between tentative matches to obtain an accurate two-view refinement of the feature geometry. The second stage aggregates all the two-view refinements in a multi-view graph of relative feature geometry constraints and then globally optimizes them jointly to obtain the refined geometry of the features. The proposed two-stage approach is agnostic to the type of local features and easily integrates into any application relying on local feature matching. Numerous experiments demonstrate the superior performance of our approach for various local features on the tasks of image matching, triangulation, camera localization, and end-to-end 3D reconstruction from unstructured imagery. The source code of our entire method and of the evaluation pipeline will be released as open source.

2 Related work

Our method is directly related to local features as well as patch description and matching. The two-view alignment network borrows concepts from recent advances in the field of image alignment and visual flow. In this section, we provide an overview of the state of the art in these research directions.

Local features. Traditional local feature extractors can be split into two main stages: first, feature detection finds interesting regions in the image using low-level statistics (*e.g.*, Difference-of-Gaussians [23] or the Harris score [16]), typically followed by the estimation of local feature geometry (*e.g.*, scale, orientation, affine shape) for the detected interest points to achieve viewpoint invariance. Second, feature description then normalizes the local image region around interest points to a canonical frame using the detected feature geometry and finally extracts an illumination invariant, compact numerical representation from the normalized patch (*e.g.*, SIFT [23], Root-SIFT [3], BRIEF [8]). More recently, researchers

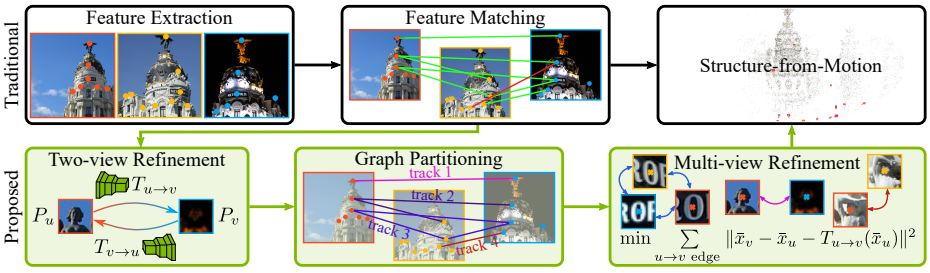


Fig. 2. Overview of the proposed method. Our method operates on the tentative matches graph (with patches as nodes P_u, P_v and matches as edges) without knowledge of scene and camera geometry. A neural network is used to annotate the edges of this graph with local geometric transformations ($T_{u\to v}, T_{v\to u}$). Next, the graph is partitioned into tracks, each track containing at most one patch from each image. Finally, the keypoint locations x_u, x_v are refined using a global optimization over all edges.

have developed trainable counterparts that either replace individual parts of the pipeline – learned detectors [43,36,6] and descriptors [5,26] – or reformulate the entire pipeline in an end-to-end trainable manner [45,29].

Lately, methods have moved away from the detect-then-describe methodology to a describe-then-detect approach, mainly due to the sensitivity of detections to changes in image statistics. These methods start by using a CNN as a dense feature extractor and afterwards either train a classifier for detection on top [27], use it as a shared encoder that splits into two decoders for detection and description respectively [10,30], or directly use non-maxima suppression on the deep feature maps [12]. However, these approaches have another issue: due to their large receptive field and feature map down-sampling, the obtained keypoints are generally not well localized when compared to their hand-crafted, low-level counterparts. This is the case even for methods [10] explicitly trained to detect corners. In this paper, we address the limited accuracy of feature detections for both hand-crafted as well as learned features. Our approach only requires images as input and achieves superior detection accuracy by considering multiple views jointly, which is in contrast to existing local feature approaches.

Patch description and matching. CNNs have been successfully used to learn local descriptors offering better robustness to viewpoint and illumination changes using different triplet losses [5], hard-negative mining techniques [26], and geometric similarity for training [25]. Likewise, in Multi-View Stereo, hand-crafted similarity metrics [14] and descriptors [42] traditionally used for patch matching were replaced by learned counterparts [24,15,17,44]. Closer to our approach are methods bypassing description and directly considering multiple views to decide whether two points correspond [46,47,15]. While these approaches focus on the second part of the local feature pipeline, we focus our attention on the detection stage. However, the intrinsic motivation is the same: exploiting multiple views facilitates a more informed decision for better results.

Geometric alignment and visual flow. Recent advances in semantic alignment [31,33] and image matching [33] as well as flow estimation [11] use a Siamese network followed by a feature matching layer. Our patch alignment network uses the correlation normalization introduced in [31]. The matching results are processed by a sequence of convolutional and fully connected layers for prediction. Contrary to visual flow, which is generally targeted at temporally adjacent video frames, where pixel displacements remain relatively low and appearance is similar, our method must handle large deformations and drastic illumination changes.

Refinement from known geometry or poses. Closer to our method, Eichhardt *et al.* [13] recently introduced an approach for local affine frame refinement. While they similarly formulate the problem as a constrained, multi-view least squares optimization, their method assumes known two-view camera geometries and does not consider visual cues from two views jointly to compute the patch alignment. Furthermore, they need access to ground-truth feature tracks (computed by an initial Structure-from-Motion process). In contrast, not requiring known camera geometry and feature tracks makes our approach amenable to a much wider range of practical applications, *e.g.*, Structure-from-Motion or visual localization. Moreover, the two methods are in fact complementary – our procedure can improve the quality of Structure-from-Motion, which can then be further refined using their approach.

3 Method

The generic pipeline for multi-view geometry estimation, illustrated in Figure 2, starts from a set of input images $\mathcal{I} = \{I_1, \ldots, I_N\}$ and first runs feature extraction on each image I_i independently yielding keypoints p_i with associated local descriptors d_i. Feature matching next computes tentative feature correspondences $\mathcal{M}_{i,j} = \{(k, l) \text{ such that } d_{i,k} \text{ matches } d_{j,l}\}$ between image pairs (I_i, I_j) based on nearest neighbors search in descriptor space (usually alongside filtering techniques). The output of this step can be interpreted as a tentative matches graph $G = (V, E)$ with keypoints as nodes ($V = \cup_i p_i$) and matches as edges ($E = \cup_{i,j} \mathcal{M}_{i,j}$), optionally weighted (*e.g.*, by the cosine similarity of descriptors). In the last step, the specific application (*e.g.*, a Structure-from-Motion [38] or visual localization pipeline [34]) takes the tentative matches graph as input and estimates camera or scene geometry as the final output.

In this paper, we propose a further geometric refinement of the nodes V in the tentative matches graph, as shown in the bottom part of Figure 2. This intermediate processing step naturally fits into any generic multi-view geometry pipeline. As demonstrated in experiments, our method significantly improves the geometric accuracy of the keypoints and thereby also the later processing steps, such as triangulation and camera pose estimation.

3.1 Overview

Our proposed method operates in a two-stage approach. First, for each edge, we perform a two-view refinement using a patch alignment network that, given local

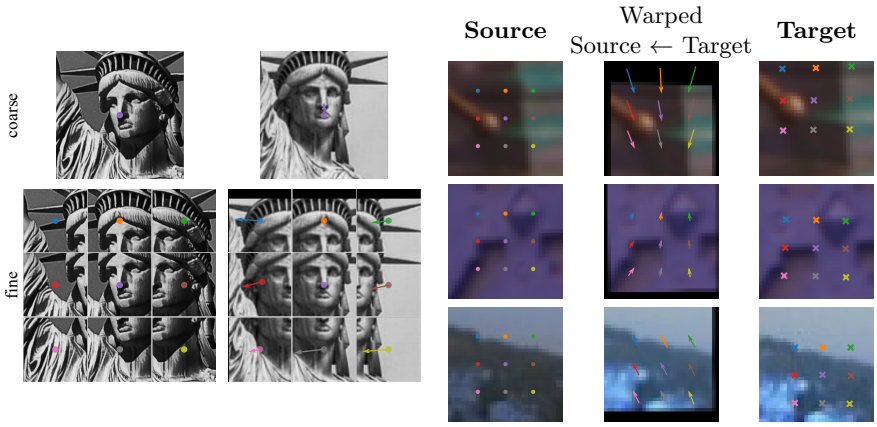


Fig. 3. Coarse-to-fine refinement and qualitative examples. *Left:* We start by a coarse alignment at feature extraction resolution taking into account only the central point, followed by a fine refinement on sub-patches corresponding to each grid point. *Right:* The first and last columns show the source and the target patch, respectively. The 3×3 regular grid is plotted with circles. For the target patch, we plot the deformed grid predicted by the coarse-to-fine refinement with crosses. The middle column shows the warped target patch using bisquare interpolation in between grid locations.

patches P_u, P_v around the corresponding initial keypoint locations $u, v \in \mathbb{R}^2$, predicts the flow $d_{u \to v}$ of the central pixel from one patch in the other and vice versa as $d_{v \to u}$. This network is used to annotate the edges of the tentative matches graph with geometric transformations $T_{u \to v}$, $T_{v \to u}$. In the second step, we partition the graph into components (*i.e.*, features tracks) and find a global consensus by optimizing a non-linear least squares problem over the keypoint locations, given the estimated two-view transformations.

3.2 Two-view refinement

Our method starts by computing a two-view refinement for every edge in the graph. Similarly to previous works in the field of CNNs for semantic alignment [31,32], image matching [33], and visual flow [11], we employ a Siamese architecture for feature extraction followed by a correlation layer. The final flow is predicted by a succession of convolutional and fully connected layers.

Feature extraction and correlation. The architecture first densely extracts features in both patches (P_u, P_v) with a standard CNN architecture. The output is two 3D tensors $F_u, F_v \in \mathbb{R}^{h \times w \times d}$, each of which can be interpreted as a set of d-dimensional descriptors associated to a $h \times w$ spatial grid in their corresponding patches $\mathbf{f}_u(i,j), \mathbf{f}_v(i,j) \in \mathbb{R}^d$. Before matching the descriptors using dot-product correlation, we perform L2-normalization as $\hat{\mathbf{f}}(i,j) = \frac{\mathbf{f}(i,j)}{\|\mathbf{f}(i,j)\|_2}$.

Dense matching can be implemented using a correlation layer yielding a 4D tensor $c \in \mathbb{R}^{h \times w \times h \times w}$ defined by $c(i_1, j_1, i_2, j_2) = \hat{\mathbf{f}}_u(i_1, j_1)^T \hat{\mathbf{f}}_v(i_2, j_2)$. This volume can be interpreted as a 3D tensor $m \in \mathbb{R}^{h \times w \times (h \cdot w)}$, where each channel

is associated to a different grid position in the opposite patch: $m(i_1, j_1)_k = c(i_1, j_1, i_2, j_2)$ where $k = i_2 \cdot w + j_2$.

Following the methodology proposed by [31], we use L2-normalization across the channel dimension to lower the values of ambiguous matches

$$\hat{m}(i,j) = \frac{\text{ReLU}(m(i,j))}{\|\text{ReLU}(m(i,j))\|_2} \ , \tag{1}$$

when the opposite patch contains more than one similar descriptor.

Regression. The final matching result $\hat{m}$ is post-processed by a CNN to aggregate local information. Finally, to enforce a patch-level consistency, a sequence of fully connected layers predicts the final output $d_{u \to v}$. Please refer to the supplementary material for more details regarding the architecture.

3.3 Multi-view refinement

In a two-view scenario, the network described in the previous section is sufficient: (u and $v + d_{u \to v}$) or ($u + d_{v \to u}$ and v) can directly be used as the refined keypoint locations. However, given that our final goal is to perform optimization over multiple views, there are several challenges we need to overcome.

Firstly, since corresponding features are generally observed from different viewpoints and looking at non-planar scene structures, the computed displacement vector is only valid for the central pixel and not constant within the patch (*i.e.*, $\frac{\delta}{\delta u} d_{u \to v} \neq \mathbf{0}_{2,2}$). Thus, when refining keypoint locations $u, v, w, \ldots$ over multiple views, consistent results can only be produced by forming displacement chains (*e.g.*, $d_{u \to v} + d_{(v + d_{u \to v}) \to w} + \ldots$) without loops. However, such an approach does not consider all possible edges in the graph and quickly accumulate errors along the chain. Another possible way to perform the refinement is to predict new displacements every time the keypoint locations are updated during the multi-view optimization. The main downside of this approach is its run-time, since the two-view network would have to be run for each edge after each optimization step. Therefore, to refine the keypoints over the entire graph and also achieve practical run-times, we use the two-view network to estimate local flow fields $T_{u \to v}$ prior to multi-view refinement and then efficiently interpolate displacements within the patch during the optimization. Some qualitative examples are shown in Figure 3 (right).

Secondly, the connected components of G generally contain feature tracks of different scene points, as the graph topology is purely based on appearance and feature matching is imperfect despite various filtering constraints – a single incorrect match can merge two tracks. As such, we partition the connected components into smaller, more reliable subsets based on the descriptor cosine similarity $s_{u,v}$ between patch pairs (u, v).

Thirdly, predicting the reverse flow or loops in the graph does not necessarily produce a consistent result (*e.g.*, $T_{v \to u} \circ T_{u \to v} \neq \mathbf{id}$, $T_{w \to u} \circ T_{v \to w} \circ T_{u \to v} \neq \mathbf{id}$) due to wrong matches or noisy network predictions. We tackle this by formulating a joint robust optimization of all tentatively matching keypoint locations considering

all the edges over multiple views, analogous to Pose Graph Optimization [28]. In the following paragraphs, we detail our solutions to the issues mentioned above.

Flow field prediction. To facilitate the multi-view optimization of the keypoint locations, we use repeated forward passes of the central flow network to predict a local flow field $T_{u\to v}$ around the initial keypoint location u. Note that this prediction is directionally biased and, as such, we always also predict the inverse flow field $T_{v\to u}$. For further space and time efficiency considerations, we approximate the full flow field between two patches by a 3×3 displacement grid and use bi-square interpolation with replicate padding in between the grid points. Assuming locally smooth flow fields, we can efficiently chain the transformations from any node u to another node w without any additional forward-passes of the two-view network. To obtain correspondences for all points of the 3×3 grid, we first predict a coarse alignment $d^c_{u\to v}$ using patches around matched features u, v at original keypoint extraction resolution. Subsequently, we further refine the coarse flow at a finer resolution using sub-patches around each 3×3 grid position g, $d^f_{u+g\to v+d^c_{u\to v}+g}$. The final transformation is given by: $T_{u\to v}(g) = d^c_{u\to v} + d^f_{u+g\to v+d^c_{u\to v}+g}$. This process is illustrated in Figure 3 (left).

Match graph partitioning. To address the second issue, our multi-view refinement starts by partitioning the tentative matches graph into disjoint components called tracks. A track is defined as a subset of the nodes V containing at most one node (patch) from each image. This is similar to a 3D feature track (*i.e.*, the set of 2D keypoints corresponding to the same 3D point). For each node $u \in V$, we denote t_u the track containing u. For a subset S of V, we define $\mathbf{I}_S$ as the set of images in which the features (nodes) of S were extracted (*i.e.*, $\mathbf{I}_S = \{I \in \mathcal{I} | \exists u \in S \text{ s.t. } u \in I\}$).

The proposed algorithm for track separation follows a greedy strategy and is closely related to Kruskal's minimum-spanning-tree algorithm [20]. The edges $(u \to v) \in E$ are processed in decreasing order of their descriptor similarity $s_{u\to v}$. Given an edge $u \to v$ linking two nodes from different tracks (*i.e.*, $t_u \neq t_v$), the two tracks are joined only if their patches come from different images (*i.e.*, $\mathbf{I}_{t_u} \cap \mathbf{I}_{t_v} = \emptyset$). The pseudo-code of this algorithm is defined in Figure 4 (left).

Another challenge commonly arising due to repetitive scene structures are very large connected components in the tentative matches graph. These large components are generally caused by a small number of low-similarity edges and lead to excessively large optimization problems. To prevent these large components from slowing down the optimization, we use recursive normalized graph-cuts (GC) on the meta-graph of tracks $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ until each remaining connected component has fewer nodes than the number of images N. The nodes of $\mathcal{G}$ correspond to tracks ($\mathcal{V} = \{t_u | u \in V\}$) and its edges aggregate over the edges of G, $\mathcal{E} = \{(t_u, t_v, w_{t_u,t_v}) | (u \to v) \in E, w_{t_u,t_v} = \sum_{(u'\to v')\in E \text{ s.t. } t_{u'}=t_u, t_{v'}=t_v} s_{u'\to v'}\}$. The G-cardinality of a subset $\mathcal{A} \subseteq \mathcal{V}$ is defined as: $|\mathcal{A}|_G = |\{u \in V | t_u \in \mathcal{A}\}|$. The pseudo-code is detailed in Figure 4 (right). This step returns a pair-wise disjoint family of sets $\mathcal{S}$ corresponding to the final connected components of $\mathcal{G}$.

Input: Graph $G = (V, E)$
Output: Track assignments $t_u, \forall u \in V$

for $u \in V$ **do**
  $t_u \leftarrow$ new track $\{u\}$;
end
$F \leftarrow E$ sorted by decreasing similarity;
for $(u, v) \in F$ **do**
  if $\mathbf{I}_{t_u} \cap \mathbf{I}_{t_v} = \emptyset$ **then**
    merge t_u and t_v;
  end
end

Input: Meta-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: Family of sets $\mathcal{S}$

$\mathcal{S} \leftarrow \{\}$;
for $\mathcal{C}$ *connected component of* $\mathcal{G}$ **do**
  RecursiveGraphCut($\mathcal{C}$);
end
Function *RecursiveGraphCut($\mathcal{C}$)*
  if $|\mathcal{C}|_G > N$ **then**
    $\mathcal{A}, \mathcal{B} \leftarrow$ NormalizedGC($\mathcal{C}$);
    RecursiveGraphCut($\mathcal{A}$);
    RecursiveGraphCut($\mathcal{B}$);
  else
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathcal{C}\}$;
  end

Fig. 4. Algorithms. *Left – track separation algorithm:* the tentative matches graph is partitioned into tracks following a greedy strategy. Each track contains at most one patch from each image. *Right – recursive graph cut:* we remove edges until having connected components of size at most N - the number of images. This algorithm yields a pair-wise disjoint family of sets $\mathcal{S}$, each set representing an ensemble of tracks.

Given the track assignments and a set of tracks $\mathcal{A} \in \mathcal{S}$, we define the set of intra-edges connecting nodes within a track as $E^{\mathcal{A}}_{\text{intra}} = \{(u \rightarrow v) \in E | t_u = t_v, t_u \in \mathcal{A}\}$ and the set of inter-edges connecting nodes of different tracks as $E^{\mathcal{A}}_{\text{inter}} = \{(u \rightarrow v) \in E | t_u \neq t_v, t_u \in \mathcal{A}, t_v \in \mathcal{A}\}$. In the subsequent optimization step, the intra-edges are considered more reliable and prioritized, since they correspond to more confident matches.

Graph optimization. Given the tentative matches graph augmented by differentiable flow fields T for all edges, the problem of optimizing the keypoint locations x_p can be formulated independently for each set of tracks $\mathcal{A} \in \mathcal{S}$ as the bounded non-linear least squares problem

$$\begin{aligned} \min_{\{x_p | t_p \in \mathcal{A}\}} & \sum_{(u \rightarrow v) \in E^{\mathcal{A}}_{\text{intra}}} s_{u \rightarrow v} \rho(\|\bar{x}_v - \bar{x}_u - T_{u \rightarrow v}(\bar{x}_u)\|^2) + \\ & \sum_{(u \rightarrow v) \in E^{\mathcal{A}}_{\text{inter}}} s_{u \rightarrow v} \psi(\|\bar{x}_v - \bar{x}_u - T_{u \rightarrow v}(\bar{x}_u)\|^2) \\ \text{s.t.} & \|\bar{x}_p\|_1 = \|x_p - x^0_p\|_1 \leq K, \forall p \ , \end{aligned} \tag{2}$$

where $x^0_\cdot$ are the initial keypoint locations, ρ is a soft, unbounded robust function for intra-edges, ψ is a stronger, bounded robust function for inter-edges, and K is the degree of liberty of each keypoint (in pixels). Finally, $s_{u \rightarrow v}$ is the cosine similarity between descriptors of nodes u and v; thus, closer matches in descriptor space are given more confidence during the optimization.

The inter-edges are essential since most features detectors in the literature sometimes fire multiple times for the same visual feature despite non-max suppression (at multiple scales or with different orientations). Without inter-edges, given our definition of a track as only containing at most one feature from each image, these detections would be optimized separately. With inter-edges, the optimization can merge different tracks for higher estimation redundancy if the deviations from the intra-track solutions are not too high.

Note that this problem can have multiple local minima corresponding to different scene points observed in all the patches of a track. For robust convergence of the optimization to a good local minimum, we fix the keypoint location of the node r_τ with the highest connectivity score[3] in each track τ, $r_\tau = \underset{\{u|t_u=\tau\}}{\arg\max}\, \gamma(u)$.

4 Implementation details

This section describes the loss and dataset used for training the patch alignment network in a supervised manner, as well as details regarding the graph optimization algorithm, hyperparameters, and runtime.

Training loss. For training the network, we use a squared L2 loss: $\mathcal{L} = \sum_{P_1,P_2} \|d_{1\to 2} - d^{\text{gt}}_{1\to 2}\|_2^2$, where d and d^{gt} are the predicted and ground-truth displacements for the central pixel from patch 1 to patch 2, respectively.

Training dataset. We use the MegaDepth dataset [22] consisting of 196 different scenes reconstructed from internet images using COLMAP [38,39] to generate training data. Given the camera intrinsics, extrinsics, and depth maps of each image, a random triangulated SIFT keypoint is selected as reference and reprojected to a matching image to generate a corresponding patch pair. We enforce depth consistency to ensure that the reference pixel is not occluded in the other view. We discarded 16 scenes due to inconsistencies between sparse and dense reconstructions. The extracted patch pairs are centered around the SIFT keypoint in the reference view and its reprojected correspondence in the target view respectively (*i.e.*, the ground-truth flow is $\mathbf{0}$). Random homographies are used on the target view to obtain varied ground-truth central point flow. While the MegaDepth dataset provides training data across a large variety of viewpoint and illumination conditions, the ground-truth flow is sometimes not perfectly sub-pixel accurate due to errors in the dense reconstruction. Therefore, we synthesize same-condition patch pairs with perfect geometric flow annotation using random warping of reference patches to generate a synthetic counterpart.

Feature extraction CNN. As the backbone architecture for feature extraction, we use the first two blocks of VGG16 [41] (up to `conv2_2`) pretrained on ImageNet [9]. To keep the features aligned with input patch pixels, we replace the 2×2 max-pooling with stride 2 by a 3×3 max-pooling with stride 2 and zero padding.

Training methodology. We start by training the regression head for 5 epochs. Afterwards, the entire network is trained end-to-end for 30 epochs, with the learning rate divided by 10 every 10 epochs. Adam [19] serves as the optimizer with an initial learning rate of 10^{-3} and a batch size of 32. To counter scene imbalance, 100 patch pairs are sampled from every scene during each epoch.

Graph optimization. During the optimization, keypoints are allowed to move a maximum of $K = 16$ pixels in any direction. We initialize x_p to the initial

[3] The connectivity score of a node u is defined as the similarity-weighted degree of the intra-edges $\gamma(u) = \sum_{\{(u\to v)|t_u=t_v\}} s_{u\to v}$.

keypoint locations x_p^0. Empirically, we model the soft robust function ρ as Cauchy scaled at 4 pixels, and the strong one ψ as Tukey scaled at 1 pixel. We solve the problems from Eq. 2 for each connected component $\mathcal{A} \in \mathcal{S}$ independently using Ceres [1] with sparse Cholesky factorization on the normal equations.

Runtime. The coarse-to-fine patch transformation prediction processes 1-4 image pairs per second on a modern GPU depending on the number of matches. The average runtime of the graph optimization across all methods on the ETH3D scenes is 3.0s (median runtime 1.0s) on a CPU with 16 logical processors.

5 Experimental evaluation

Despite being trained on SIFT keypoints, our method can be used with a variety of different feature detectors. To validate this, we evaluate our approach in conjunction with two well-known hand-crafted features (SIFT [23] and SURF [7]), one learned detector combined with a learned descriptor (Key.Net [6] with HardNet [26]), and three learned ones (SuperPoint [10] denoted SP, D2-Net [12], and R2D2 [30]). For all methods, we resize the images before feature extraction such that the longest edge is at most 1600 pixels (lower resolution images are kept unchanged). We use the default parameters as released by their authors in the associated public code repositories. Our refinement protocol takes exactly the same input as the feature extraction. The main objective is not to compare these methods against each other, but rather to show that each of them independently significantly improves when coupled with our refinement procedure.

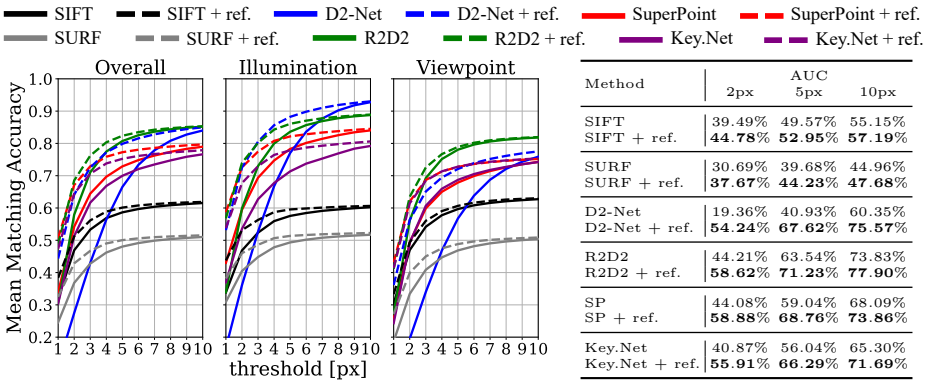
First, we evaluate the performance with and without refinement on a standard image matching task containing sequences with illumination and viewpoint changes. Then, we present results in the more complex setting of Structure-from-Motion. In particular, we demonstrate large improvements on the tasks of multi-view triangulation, camera localization, as well as their combination in an end-to-end image-based 3D reconstruction scenario.

For the Structure-from-Motion evaluations, we use the following matching protocol: for SIFT and SURF, we use a symmetric second nearest neighbor ratio test (with the standard threshold of 0.8) and mutual nearest neighbors filtering. For Key.Net+HardNet, we use the same protocol with a threshold of 0.9. For the remaining methods, we use mutual nearest neighbors filtering with different similarity thresholds - 0.755 for SuperPoint, 0.8 for D2-Net, and 0.9 for R2D2.[4]

5.1 Image matching

In this experiment, we evaluate the effect of our refinement procedure on the full image sequences from the well-known HPatches dataset [4]. This dataset consists of 116 sequences of 6 images with changes in either illumination or viewpoint. We follow the standard evaluation protocol introduced by [12] that discards 8 of

[4] The thresholds for the learned methods were determined following the methodology of [23]. Please refer to the supplementary material for more details.



Method	AUC 2px	AUC 5px	AUC 10px
SIFT	39.49%	49.57%	55.15%
SIFT + ref.	**44.78%**	**52.95%**	**57.19%**
SURF	30.69%	39.68%	44.96%
SURF + ref.	**37.67%**	**44.23%**	**47.68%**
D2-Net	19.36%	40.93%	60.35%
D2-Net + ref.	**54.24%**	**67.62%**	**75.57%**
R2D2	44.21%	63.54%	73.83%
R2D2 + ref.	**58.62%**	**71.23%**	**77.90%**
SP	44.08%	59.04%	68.09%
SP + ref.	**58.88%**	**68.76%**	**73.86%**
Key.Net	40.87%	56.04%	65.30%
Key.Net + ref.	**55.91%**	**66.29%**	**71.69%**

Fig. 5. Matching evaluation. We plot the mean matching accuracy on HPatches Sequences at different thresholds for illumination and viewpoint sequences, as well as overall. We also report the area under the overall curve (AUC) up to 2, 5, and 10 pixels. All methods have their performance improved by the proposed refinement procedure.

the sequences due to resolution considerations. The protocol reports the mean matching accuracy per image pair of a mutual nearest neighbors matcher while varying the pixel threshold up to which a match is considered to be correct.

Figure 5 shows the results for illumination-only, viewpoint-only, as well as overall for features with and without refinement. As expected, our method greatly improves upon learned features under either condition. Note that the evaluated learned methods represent the state of the art on this benchmark already and we further improve their results. For SIFT [23], while the performance remains roughly the same under viewpoint changes, our method significantly improves the results under illumination sequences, where low-level changes in image statistics perturb the feature detector. It is also worth noting that, especially in the viewpoint sequences for learned features, our refinement procedure improves the results for coarse thresholds by correcting wrong, far-away correspondences.

5.2 Triangulation

Next, we evaluate the triangulation quality with known ground-truth camera poses and intrinsics on the ETH3D benchmark [40]. Originally, this benchmark was proposed for multi-view stereo methods and provides highly accurate ground-truth camera poses and dense 3D point-clouds. Nevertheless, the same evaluation protocol also applies to our scenario – we want to evaluate the impact of refined keypoint locations on the completeness and accuracy of sparse multi-view triangulation. For each method, we run the multi-view triangulator of COLMAP [38] with fixed camera intrinsics and extrinsics. Given the sparse point cloud, we run the ETH3D evaluation code to report the accuracy (% of triangulated points) and completeness (% of ground-truth triangulated points) at different real-world thresholds. We refer to the original paper for more details about the evaluation.

Table 1 compares the different local feature approaches with their refined counterparts. Our proposed keypoint refinement procedure improves the results

Table 1. Triangulation evaluation. We report the accuracy (% of triangulated points) and completeness (% of ground-truth triangulated points) at 1cm, 2cm, and 5cm. The refined versions outperform their raw counterparts in both metrics.

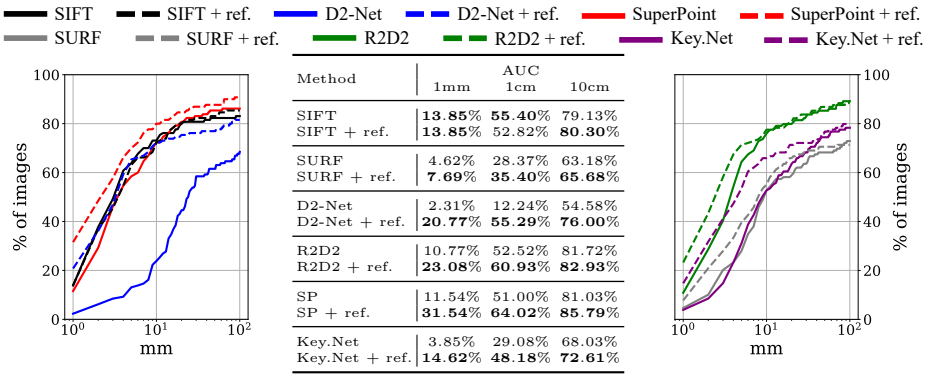
Dataset	Method	Comp. (%) 1cm	2cm	5cm	Accuracy (%) 1cm	2cm	5cm	Method	Comp. (%) 1cm	2cm	5cm	Accuracy (%) 1cm	2cm	5cm
Indoors 7 scenes	SIFT	0.20	0.86	3.61	75.74	84.77	92.26	SURF	0.08	0.41	1.97	66.37	79.05	89.61
	SIFT + ref.	**0.24**	**0.96**	**3.88**	**81.06**	**88.64**	**94.61**	SURF + ref.	**0.12**	**0.52**	**2.26**	**76.28**	**85.30**	**92.36**
	D2-Net	0.46	1.83	7.00	46.95	64.91	83.25	R2D2	0.53	2.04	8.53	66.70	79.26	90.04
	D2-Net + ref.	**1.44**	**4.53**	**12.97**	**78.53**	**86.46**	**93.05**	R2D2 + ref.	**0.66**	**2.32**	**9.08**	**77.56**	**85.74**	**92.54**
	SP	0.59	2.21	8.86	75.26	85.27	93.30	Key.Net	0.16	0.68	3.01	66.51	80.44	91.61
	SP + ref.	**0.71**	**2.51**	**9.55**	**86.03**	**91.91**	**95.83**	Key.Net + ref.	**0.21**	**0.81**	**3.36**	**80.51**	**89.24**	**94.73**
Outdoors 6 scenes	SIFT	0.06	0.34	2.44	58.31	73.13	86.24	SURF	0.03	0.17	1.22	44.21	63.11	79.71
	SIFT + ref.	**0.07**	**0.41**	**2.75**	**61.61**	**76.89**	**88.96**	SURF + ref.	**0.05**	**0.26**	**1.68**	**62.88**	**74.67**	**87.10**
	D2-Net	0.03	0.19	1.80	21.35	35.08	56.75	R2D2	0.11	0.55	3.61	48.75	65.74	82.81
	D2-Net + ref.	**0.21**	**1.09**	**6.13**	**59.07**	**72.34**	**85.62**	R2D2 + ref.	**0.16**	**0.71**	**4.08**	**63.85**	**78.10**	**90.09**
	SP	0.09	0.54	3.86	49.67	64.57	80.79	Key.Net	0.01	0.09	0.75	39.25	54.57	72.30
	SP + ref.	**0.15**	**0.77**	**4.91**	**65.23**	**77.50**	**88.37**	Key.Net + ref.	**0.02**	**0.13**	**0.91**	**55.62**	**69.41**	**85.56**

across the board for all methods. Once again, the learned keypoints that suffer from poor localization due to downsampling and large receptive field are drastically improved for both indoor and outdoor scenarios. Even though the performance gain is smaller in the case of SIFT, this experiment shows that exploiting multi-view information is beneficial for very well localized features as well. The increase in completeness for all local features shows that our approach does not trim the 3D models to only contain accurate points, but rather improves the overall quality by yielding more triangulated points which are also more precise. Please refer to the supplementary material for results on each dataset.

5.3 Camera localization

We also evaluate the camera localization performance under strict thresholds on the ETH3D dataset [40]. For each scene, we randomly sample 10 images that will be treated as queries (130 query images in total). For each query, a partial 3D model is built without the query image and its 2 closest neighbors in terms of co-visibility in the reference model (released with the dataset); 2D-3D correspondences are inferred from the tentative matches between the query image and all (partial) 3D model images; finally, absolute pose estimation with non-linear refinement from COLMAP is used to obtain the camera pose. The partial models are built independently, *i.e.*, multi-view optimization is only run on the views that are part of each partial model (without the query and holdout images). For the query keypoints, central point flow is predicted from the reprojected locations of 3D scene points in the matching views to the query view. To obtain a single 2D coordinate for each matching 3D point, we compute the similarity-weighted average of the flow for each track, which is equivalent to solving Eq. 2, where nodes of keypoints in the 3D model are connected through a single edge to matching query keypoints.

The results of this experiment are presented in Figure 6. The performance of SIFT [23] after refinement is on par with the unrefined version despite the increase in point-cloud accuracy and completeness; this suggests that the method has nearly saturated on this localization task. All the other features have their



Method	AUC 1mm	AUC 1cm	AUC 10cm
SIFT	**13.85%**	**55.40%**	79.13%
SIFT + ref.	**13.85%**	52.82%	**80.30%**
SURF	4.62%	28.37%	63.18%
SURF + ref.	**7.69%**	**35.40%**	**65.68%**
D2-Net	2.31%	12.24%	54.58%
D2-Net + ref.	**20.77%**	**55.29%**	**76.00%**
R2D2	10.77%	52.52%	81.72%
R2D2 + ref.	**23.08%**	**60.93%**	**82.93%**
SP	11.54%	51.00%	81.03%
SP + ref.	**31.54%**	**64.02%**	**85.79%**
Key.Net	3.85%	29.08%	68.03%
Key.Net + ref.	**14.62%**	**48.18%**	**72.61%**

Fig. 6. Camera localization evaluation. We report the percentage of localized images at different camera position error thresholds as well as the area under the curve (AUC) up to 1mm, 1cm and 10cm. The performance of SIFT remains similar on this task. All other features show greatly improved camera pose accuracy after refinement.

performance greatly improved by the proposed refinement. It is worth noting that the refined versions of SuperPoint [10] and R2D2 [30] drastically outperform SIFT especially on the finer thresholds (1mm and 1cm).

5.4 Structure-from-Motion

Finally, we evaluate our refinement procedure on the scenario of end-to-end 3D reconstruction from unstructured imagery on the benchmark introduced in [37]. For the internet datasets (Madrid Metropolis, Gendarmenmarkt, and Tower of London), instead of exhaustively matching all images, we use NetVLAD [2] to retrieve top 20 related views for each image and only match against these. Due to the wide range of resolutions in internet images, we impose the use of multi-scale features if available and not active by default (*i.e.*, for D2-Net [12]).

After matching and feature refinement, we run COLMAP [38] to obtain sparse 3D reconstructions. Finally, different reconstruction statistics taking into account only the images registered both with and without refinement are reported in Table 2. For independent results, please refer to the supplementary material.

Overall, the results with refined keypoints achieve significantly better statistics than their original counterparts. On the small datasets all refined methods apart from R2D2 have sub-pixel keypoint accuracy (*i.e.*, a reprojection error lower than 0.5). SuperPoint and Key.Net, despite being targeted at low-level features, are still largely behind SIFT in terms of reprojection error without refinement. The refinement lowers this gap while also improving their already significant track length. For SIFT, the main improvement is in terms of reprojection error showing that it is possible to refine even features with accurate, sub-pixel keypoint localization. For R2D2 and SURF, on the large datasets, we see a tendency to very slightly decrease the track length to improve the reprojection error. This points to the fact that loosely grouped features during SfM are split into multiple, but more accurate feature tracks. The results on the large internet datasets

Table 2. Local Feature Evaluation Benchmark. A 3D model is built for each method and different reconstruction statistics are reported. For the large datasets, we report the statistics on the common images only.

Dataset	Method	Reg. images	Num. obs.	Track length	Reproj. error	Method	Reg. images	Num. obs.	Track length	Reproj. error
Herzjesu 8 images	SIFT	8	15.9K	4.10	0.59	SURF	8	5.0K	3.64	0.70
	SIFT + ref.		**16.2K**	**4.16**	**0.29**	SURF + ref.		**5.2K**	**3.70**	**0.30**
	D2-Net	8	38.5K	3.36	1.32	R2D2	8	21.1K	5.84	1.08
	D2-Net + ref.		**47.7K**	**4.06**	**0.41**	R2D2 + ref.		**21.6K**	**6.04**	**0.57**
	SP	8	17.2K	4.54	1.00	Key.Net	8	5.0K	4.29	1.00
	SP + ref.		**17.9K**	**4.72**	**0.36**	Key.Net + ref.		**5.3K**	**4.46**	**0.42**
Fountain 11 images	SIFT	11	27.0K	4.51	0.55	SURF	11	5.6K	3.91	0.64
	SIFT + ref.		**27.4K**	**4.56**	**0.26**	SURF + ref.		**5.7K**	**3.95**	**0.30**
	D2-Net	11	62.0K	3.51	1.36	R2D2	11	33.0K	7.11	1.10
	D2-Net + ref.		**77.4K**	**4.47**	**0.40**	R2D2 + ref.		**33.6K**	**7.47**	**0.62**
	SP	11	21.5K	4.93	1.06	Key.Net	11	8.4K	5.53	1.00
	SP + ref.		**22.4K**	**5.19**	**0.43**	Key.Net + ref.		**8.7K**	**5.70**	**0.44**
Madrid Metropolis 1344 images	SIFT	379	187.2K	6.83	0.70	SURF	268	**116.0K**	**6.25**	0.76
	SIFT + ref.		**187.7K**	**6.86**	**0.66**	SURF + ref.		115.2K	**6.25**	**0.66**
	D2-Net	372	668.8K	6.00	1.47	R2D2	410	355.2K	**10.20**	0.90
	D2-Net + ref.		**752.5K**	**7.28**	**0.96**	R2D2 + ref.		**356.8K**	10.17	**0.76**
	SP	414	269.7K	7.64	0.98	Key.Net	304	111.9K	9.18	0.94
	SP + ref.		**277.7K**	**8.20**	**0.72**	Key.Net + ref.		**114.5K**	**9.31**	**0.75**
Gendarmen-markt 1463 images	SIFT	874	440.3K	6.33	0.82	SURF	472	163.9K	**5.45**	0.90
	SIFT + ref.		**441.4K**	**6.42**	**0.75**	SURF + ref.		**164.8K**	5.43	**0.78**
	D2-Net	858	1.479M	5.33	1.44	R2D2	929	**1.043M**	**10.09**	0.99
	D2-Net + ref.		**1.665M**	**6.37**	**1.04**	R2D2 + ref.		**1.043M**	10.05	**0.89**
	SP	911	626.9K	6.84	1.05	Key.Net	810	253.3K	7.08	0.99
	SP + ref.		**648.0K**	**7.10**	**0.89**	Key.Net + ref.		**258.6K**	**7.25**	**0.86**
Tower of London 1576 images	SIFT	561	447.8K	7.90	0.69	SURF	430	212.0K	**5.94**	0.70
	SIFT + ref.		**449.0K**	**7.96**	**0.59**	SURF + ref.		**212.7K**	5.92	**0.58**
	D2-Net	635	1.408M	5.96	1.48	R2D2	689	758.0K	13.44	0.92
	D2-Net + ref.		**1.561M**	**7.63**	**0.91**	R2D2 + ref.		**759.2K**	**13.74**	**0.76**
	SP	621	442.9K	8.06	0.95	Key.Net	495	186.5K	9.02	0.85
	SP + ref.		**457.6K**	**8.55**	**0.69**	Key.Net + ref.		**190.8K**	**9.18**	**0.65**

notably show the robustness of the multi-view refinement to incorrect matches, repeated structures, drastic illumination changes, and large, complex graphs with as much as 5 million nodes and more than 1 million tracks.

6 Conclusion

We have proposed a novel method for keypoint refinement from multiple views. Our approach is agnostic to the type of local features and seamlessly integrates into the standard feature extraction and matching paradigm. We use a patch alignment neural network for two-view flow prediction and formulate the multi-view refinement as a non-linear least squares optimization problem. The experimental evaluation demonstrates drastically improved performance on the Structure-from-Motion tasks of triangulation and camera localization. Throughout our experiments, we have shown that our refinement cannot only address the poor keypoint localization of recent learned feature approaches, but it can also improve upon SIFT – the arguably most well-known handcrafted local feature with accurate sub-pixel keypoint refinement.

Acknowledgements. This work was supported by the Microsoft Mixed Reality & AI Zürich Lab PhD scholarship.

References

1. Agarwal, S., Mierle, K., Others: Ceres solver. `http://ceres-solver.org`
2. Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. In: Proc. CVPR (2016)
3. Arandjelovic, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: Proc. CVPR (2012)
4. Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K.: HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In: Proc. CVPR (2017)
5. Balntas, V., Riba, E., Ponsa, D., Mikolajczyk, K.: Learning local feature descriptors with triplets and shallow convolutional neural networks. In: Proc. BMVC. (2016)
6. Barroso-Laguna, A., Riba, E., Ponsa, D., Mikolajczyk, K.: Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters. In: Proc. ICCV (2019)
7. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. In: Proc. ECCV (2006)
8. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: Binary robust independent elementary features. In: Proc. ECCV (2010)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: Proc. CVPR (2009)
10. DeTone, D., Malisiewicz, T., Rabinovich, A.: SuperPoint: Self-Supervised Interest Point Detection and Description. In: CVPR Workshops (2018)
11. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: FlowNet: Learning optical flow with convolutional networks. In: Proc. ICCV (2015)
12. Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T.: D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In: Proc. CVPR (2019)
13. Eichhardt, I., Barath, D.: Optimal multi-view correction of local affine frames. In: Proc. BMVC. (2019)
14. Goesele, M., Curless, B., Seitz, S.M.: Multi-view stereo revisited. In: Proc. CVPR (2006)
15. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: MatchNet: Unifying feature and metric learning for patch-based matching. In: Proc. CVPR (2015)
16. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proc. Alvey Vision Conf. (1988)
17. Hartmann, W., Galliani, S., Havlena, M., Van Gool, L., Schindler, K.: Learned multi-patch similarity. In: Proc. ICCV (2017)
18. Heinly, J., Schönberger, J.L., Dunn, E., Frahm, J.M.: Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In: Proc. CVPR (2015)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proc. ICLR (2015)
20. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society (1956)
21. Li, Y., Snavely, N., Huttenlocher, D., Fua, P.: Worldwide Pose Estimation using 3D Point Clouds. In: Proc. ECCV (2012)
22. Li, Z., Snavely, N.: MegaDepth: Learning single-view depth prediction from internet photos. In: Proc. CVPR (2018)
23. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV (2004)

24. Luo, W., Schwing, A.G., Urtasun, R.: Efficient deep learning for stereo matching. In: Proc. CVPR (2016)
25. Luo, Z., Shen, T., Zhou, L., Zhu, S., Zhang, R., Yao, Y., Fang, T., Quan, L.: GeoDesc: Learning local descriptors by integrating geometry constraints. In: Proc. ECCV (2018)
26. Mishchuk, A., Mishkin, D., Radenovic, F., Matas, J.: Working hard to know your neighbor's margins: Local descriptor learning loss. In: Advances in NeurIPS (2017)
27. Noh, H., Araujo, A., Sim, J., Weyand, T., Han, B.: Largescale image retrieval with attentive deep local features. In: Proc. ICCV (2017)
28. Olson, E., Leonard, J., Teller, S.: Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates. In: Proc. ICRA (2006)
29. Ono, Y., Trulls, E., Fua, P., Yi, K.M.: LF-Net: Learning local features from images. In: Advances in NeurIPS (2019)
30. Revaud, J., Weinzaepfel, P., de Souza, C.R., Humenberger, M.: R2D2: Repeatable and Reliable Detector and Descriptor. In: Advances in NeurIPS (2019)
31. Rocco, I., Arandjelović, R., Sivic, J.: Convolutional neural network architecture for geometric matching. In: Proc. CVPR (2017)
32. Rocco, I., Arandjelović, R., Sivic, J.: End-to-end weakly-supervised semantic alignment. In: Proc. CVPR (2018)
33. Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., Sivic, J.: Neighbourhood consensus networks. In: Advances in NeurIPS (2018)
34. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2D-to-3D matching. In: Proc. ICCV (2011)
35. Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J., Kahl, F., Pajdla, T.: Benchmarking 6DoF outdoor visual localization in changing conditions. In: Proc. CVPR (2018)
36. Savinov, N., Seki, A., Ladicky, L., Sattler, T., Pollefeys, M.: Quad-networks: unsupervised learning to rank for interest point detection. In: Proc. CVPR (2017)
37. Schönberger, J.L., Hardmeier, H., Sattler, T., Pollefeys, M.: Comparative evaluation of hand-crafted and learned local features. In: Proc. CVPR (2017)
38. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proc. CVPR (2016)
39. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: Proc. ECCV (2016)
40. Schöps, T., Schönberger, J.L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., Geiger, A.: A multi-view stereo benchmark with high-resolution images and multi-camera videos. In: Proc. CVPR (2017)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proc. ICLR (2015)
42. Tola, E., Lepetit, V., Fua, P.: Daisy: An efficient dense descriptor applied to wide-baseline stereo. IEEE PAMI (2009)
43. Verdie, Y., Yi, K., Fua, P., Lepetit, V.: TILDE: A temporally invariant learned detector. In: Proc. CVPR (2015)
44. Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: MVSNet: Depth Inference for Unstructured Multi-view Stereo. In: Proc. ECCV (2018)
45. Yi, K.M., Trulls, E., Lepetit, V., Fua, P.: LIFT: Learned invariant feature transform. In: Proc. ECCV (2016)
46. Zagoruyko, S., Komodakis, N.: Learning to compare image patches via convolutional neural networks. In: Proc. CVPR (2015)
47. Zbontar, J., LeCun, Y.: Stereo matching by training a convolutional neural network to compare image patches. Journal of Machine Learning Research (2016)