Learning of low-level feature keypoints for accurate and robust detection

Suwichaya Suwanwimolkul  Satoshi Komorita  Kazuyuki Tasaka
KDDI Research, Inc.
{su-suwanwimolkul, sa-komorita, ka-tasaka}@kdd-research.jp

Abstract

Joint learning of feature descriptor and detector has offered promising 3D reconstruction results; however, they often lack the low-level feature awareness, which causes low accuracy in matched keypoint locations. The others employed fixed operations to select the keypoints, but the selected keypoints may not correspond to the descriptor matching. To address these problems, we propose the supervised learning of keypoint detection with low-level features. Our detector is a single CNN layer extended from the descriptor backbone, which can be jointly learned with the descriptor for maximizing the descriptor matching. This results in a state-of-the-art 3D reconstruction, especially on improving reprojection error, and the highest accuracy in keypoint detection and matching on benchmark datasets. We also present a dedicated study on evaluation metrics to measure the accuracy of keypoint detection and matching.

1. Introduction

Accurately detecting and describing the interest points in images is crucial to many applications such as localization [29], Structure from Motion (SfM) [32], and 3D reconstruction [11]. Over the past few years, the joint learning of the keypoint description and detection has shown promising results on real applications. Nevertheless, these learning-based approaches often lack low-level feature keypoints. The low-level feature keypoints mark the essential information, i.e., shape, lines, and corners in the scene, which are robust to the geometric changes [10, 21, 19, 7]. The lack of robustness can cause the shift between matched keypoints, leading to the low accuracy in matched keypoint locations.

Traditionally, the process of keypoint detection is handcrafted. The keypoint detectors are designed to detect keypoints along the corners [10] and edges [15, 2, 20, 21]. These low-level features are robust against viewpoint and illumination changes. However, the performance of handcrafted approaches is limited to the prior knowledge that developers have in hand. Many researchers turn to use deep learning for the abundant descriptiveness and distinctiveness power of the semantic features from neural networks. The learned features for keypoint description [34, 38, 22, 17] have outperformed many handcrafted features, including SIFT [15]. However, a few learning-based approaches [26, 7, 12] focused only on learning of keypoint detector. To capture low-level features, [7] and [12] employed handcrafted keypoints [10, 15] in training. However, these works can fail when the handcrafted keypoints fail.

During the past few years, more studies focus on joint learning keypoint detection and description. Many algorithms encourage a certain level of low-level feature awareness in keypoint detector using unsupervised learning [6, 14, 31, 23, 30, 25]. The robustness against geometric changes was improved either by bootstrapping with multi-round adaptations on a base, pre-trained network [6, 14] or enforcing grid-wise peakiness to capture low-level features [25, 31, 40]. However, they do not employ any low-

Figure 1: Comparison between (a) R2D2 (before) versus (b) our LLF keypoints + R2D2 (after). Three point maximum error thresholds, 3px (top), 1px, and 0.6px (bottom), are used to filter out erroneous points. At the lowest threshold, our approach provides more accurate shape and 3D points.
level features to supervise the learning, which often leads to low accuracy in matched keypoint locations. To address the lack of low-level features, recent works [8, 19, 36] abandoned the learned detector. These methods derived the low-level feature keypoints from its neural network backbone and selected the keypoints with fixed softmax operations [8] or local peaks extraction [19, 36]. Although the keypoints are selected from the descriptor backbone, the fixed operations cannot guarantee to select the keypoints where descriptor matching is maximized. Thus, the accuracy of matched keypoint locations is low. Moreover, their keypoints densely cluster along edges and corners in images [19, 36].

To solve these problems, we propose to learn the low-level feature keypoint detector (LLF keypoints) with the multi-level features extracted from the neural network backbone. Our learning loss ensures the low-level feature awareness and sparse keypoint detection, which improves the accuracy of matched keypoint locations. We also propose a learning approach to jointly learn both our keypoint detector and descriptor to enforce LLF keypoints where the descriptor matching is maximized. Our learning can be applied on a single CNN layer, namely LLF detector. The LLF detector is extended from a descriptor backbone and can be trained with the descriptor backbone from scratch. In this paper, we present our application to improve the robustness and accuracy of repeatable keypoints of R2D2 [25], which results in reduced reprojection error while achieving a higher or similar number of registered images, sparse points, and dense points. Figure 1 demonstrates the effect of the LLF keypoints which provide sparse points with higher sub-pixel accuracy in 3D reconstruction (see section 4.1.1 for full report). We also present a dedicated study to evaluate the improved accuracy of keypoints matching and detection. The architecture and our learning approach are shown in Figure 2 and 3. Our contributions are as follows:

- New learning approach and learning loss to capture low-level features;
- Light-weight keypoint detection extension;
- A dedicated study on evaluation metric to measure the accuracy of keypoints detection and matching.

The rest of the paper is organized as follows. We review the related work in section 2. Our proposed method in Section 3. Experiments and results are discussed in Section 4.

2. Related works

**Handcrafted keypoints.** Most of the existing keypoint detectors are handcrafted to select particular low-level features such as blob, corners or edges [10, 15, 2, 20, 21]. Harris [10] and Hessian [5] detectors used first and second order derivatives to find corners or blobs in images, which have been extended for handling multi-scales and affine transformation [21] and acceleration [4]. While the corner detectors are robust and efficient, SIFT looks for blobs that capture regional information of the keypoints [15].

**Learned detectors.** The success of learning feature descriptions for general image classification has inspired the recent applications in keypoint detection. FAST [26] is the first approach that used machine learning for fusing point and line features for keypoint detection. Recent works [31, 40] focused on learning keypoint detection for repeatability by increasing the keypoints in repeatable areas between image pairs. QuadNet [31] employed ranking loss. Then, [40] added the grid-wise peakiness for the sparse detection. The keypoint repeatability of [31, 40] is high, but their matched keypoints have low accuracy since the ground truth of keypoint location is not well defined [12]. Meanwhile, [7, 12] resorted to using handcrafted keypoints such as [10, 15], in training, due to the consistent representation of handcrafted keypoints to the low-level features. However, these works can fail if handcrafted keypoints fail.

**Jointly learned descriptor and detector.** The joint learning description and detection ensures the keypoint detection where the descriptor matching is maximized, by optimizing the learning losses related to both terms [39, 37, 6, 14, 17, 23, 25]. Many works employed unsupervised learning [39, 6, 14, 25] and/or relied solely on high-level features for keypoint detection [25, 23], which enable the keypoints to be matched as much as possible, but at the cost of the low matched keypoint accuracy. DELF [23] proposed to supervise learning of an attention network with image-level class labels, which captures the relevance scores of high-level information in descriptors. SuperPoint [6] steps further by training a base network with annotated corners. Then, the base network is further trained by bootstrapping with multi-round homography adaptations. Without the pre-trained network, R2D2 [25] proposed to learn the repeatable and reliable keypoints by enforcing grid-wise peakiness associated with the descriptor matching area. However, these approaches do not use the low-level features in supervision, leading to the lack of robustness against viewpoint changes.

Unlike previous methods, recent methods [8, 19, 36] abandoned the learned detector and derived the low-level feature keypoints directly from descriptor backbones. The
keypoints are selected with fixed operations. D2-Net [8] applied detection scores similar to [10] and a softmax operation to the select keypoints. UR2KID [36] derived keypoints with a set of operations, i.e., activation norms, channel grouping, and self-distillation. ASLFeat [19] derived keypoints identified by local peakiness measure [40]. Because the keypoint selection is rather handcrafted, there is no guarantee that the selected keypoints are associated with matching the learned descriptors. Therefore, the matched keypoints do not always have high accuracy.

**Keypoint refinement.** Recently, [9] proposed a multiple views keypoint refinement method that solves the problem of low keypoint accuracy, similar to ours. [9] takes the matched keypoints and refines the geometry of keypoint detection and description from multiple views. However, our method improves the keypoint detection before matching, enabling other performance improvements aside from the keypoint accuracy. Also, [9] takes more computation to optimize the keypoint location under geometrical constraints. Meanwhile, our work shows the possibility of a lightweight extension that improves the matched keypoint accuracy. The comparison is provided in Section [4.2.2].

We take the inspiration from these works, especially [25, 19]. Our proposed method solves the low matched keypoint accuracy by learning with low-level features extracted from the descriptor backbone. In this paper, we apply our proposed method on R2D2 [25] as an example. In principle, our proposed scheme can be applied to other architectures, such as LIFT [59] or SuperPoint [6], that separate networks for feature detection and description. While it is possible to apply our concept to improve 3D keypoint, e.g. [35, 27], this work focuses on extracting keypoints in 2D coordinates.

### 3. Methods

Let us consider a setting where \( I \) and \( I' \) are the two images of the same scene, and \( U \in \mathbb{R}^{H \times W \times 2} \) denotes the ground truth correspondence between them. That is, if \((x, y)\) is a pixel coordinate in image \( I \) that corresponds to \((x', y')\) in image \( I' \), then \( U_{x,y} = (x', y') \). The \( Y \) and \( Y' \) denote the keypoint map associated with image \( I \) and \( I' \). Let \( Y'_U \) denote \( Y' \) warped according to \( U \). While it is natural to enforce the similarity between \( Y \) and \( Y'_U \) such as [24] with a least square loss, our proposal is to also learn for the keypoints that capture low-level features, which are stronger for geometric invariance. Thus, we aim to minimize

\[
L_{LLF} = ||Y - Y'_U||^2_2 + ||Y - \hat{F}'||^2_2 + ||Y'_U - \hat{F}'||^2_2
+ L_{peak}(Y) + L_{peak}(Y')
\]

(1)

where \( L_{peak}(\cdot) \) is the peakiness loss [25] for enforcing sparsity in keypoint detection, and \( \hat{F} \) and \( \hat{F}' \in \mathbb{R}^{H \times W} \) denote the multi-level features extracted from the descriptor backbone. The multi-level features capture the low-level information that is robust against geometric changes as proven by [8, 19, 36], but unlike these methods [8, 19, 36], we learn the detector with the multi-level features \( \hat{F} \). The resulting detector is the low-level feature (LLF) detector that predicts a set of sparse keypoint map \( Y \in [0,1]^{H \times W} \) with an emphasis on the low-level feature in an image. We present the architecture of our LLF detector in next. Then, we present the multi-level feature extraction for \( \hat{F} \) and our modification of Eq. (1) for joint learning with the descriptor of R2D2.

#### 3.1. Low-level feature detector

To enable our LLF detection with minimum extra runtime, we propose to attach the LLF detector after the network backbone and is in parallel with the existing detectors. The example of our application with R2D2 [25] is shown in Figure 2. Given an image \( I \) of size \( H \times W \), R2D2 backbone provides the dense descriptors \( X \in \mathbb{R}^{H \times W \times 128} \), and it is branched into two identical layers: (1) repeatable and (2) reliable detectors. The repeatable detector outputs the repeatability keypoint map \( S \in [0,1]^{H \times W} \) which indicates repeatable keypoint locations. The reliable detector outputs the reliability map \( R \in [0,1]^{H \times W} \) which estimates the reliability of the descriptor \( X_{x,y} \) for good matching.

To realize the lightweight computation, the LLF detector is a single CNN layer. Here, we choose the same architecture as the repeatable detector, i.e., an element-wise square operation followed by a \( 1 \times 1 \) convolutional layer and a softmax [25]. Unlike the repeatable detector, the LLF detector will be trained in supervised learning with the multi-level features \( \hat{F} \) for more robustness and accuracy. The LLF detector outputs the LLF keypoint map, which can be used in combination with R2D2 keypoints. Nevertheless, our study in Section [4.1.1] shows that the best performance is achieved by using 100% LLF keypoints without any repeatable keypoints. Thus, R2D2’s repeatable detector can be removed when running inference on the neural network. Our LLF detector can be used without costing extra runtime.

#### 3.2. Multi-level feature extraction

To extract the multi-level features \( \hat{F} \in \mathbb{R}^{H \times W} \) for supervising the LLF detector, we propose to use the weighted average of the dominating features from multi-channel, multi-level features for our learning. Given \( I \in \mathbb{R}^{H \times W} \) as the input image to the descriptor backbone, the multi-channel features \( F \in \mathbb{R}^{H \times W \times D} \) is extracted from \( f \)th level of the descriptor backbone with \( D \) feature channels. We extract the dominating features \( F^{(i)} \) from multi-channel features by

\[
\hat{F}^i = \sum_{c \in \mathcal{D}} \text{softmax}(F_{x,y,c})
\]

(2)

where \( c \in \mathcal{D} \) sorts each feature channel. softmax(\( F_{x,y,c} \)) = \( \text{exp}(F_{x,y,c}) / \sum_{x,y,c} \text{exp}(F_{x,y,c}) \). We, then, normalize the dominating features to maintain reasonable scales with spatial mean \( \mu_{x,y} \) and
variant $\sigma_y^2$, i.e., $\tilde{F} = \frac{x - \mu_y}{\sigma_y}$. To summarize the features extracted from multiple layers, we calculate the weighted sum of the dominating features from multiple layers:

$$ F = \sum_{i \in L} w_i F_i $$

where $w_i$ weights the important of each layer from $L$, the set of chosen layers from a descriptor backbone. For R2D2, we choose $L = \{2, 5, 8, 11, 14, 17\}$ which are the output from the relu of R2D2 backbone. The extracted features are used in supervised learning of $LLF$ detector.

### 3.3. Low-level feature loss

Although using the loss function Eq. (1) is possible, here we propose to modify Eq. (1) such that this loss function can be employed in the joint learning of R2D2 which is our example application. The learning losses in R2D2 are patch-based and are normalized with patch size. Thus, we modify the following terms for the joint learning. Given the image $I$, one can extract both keypoint map $Y$ and the low-level features $F$ from the backbone, where $F$ is calculated with Eq. (3).

The low-level feature loss $L_{feat}$ is defined as mean square distance between $Y$ and $\tilde{F}$:

$$ L_{feat}(Y) = \frac{1}{HW} \sum_{x,y} \|Y_{x,y} - \tilde{F}_{x,y}\|^2 $$

where $\tilde{F}$ is gradient detached. Using this loss alone results in dense keypoints. Thus, we employed the peakiness losses $L_{peak}$ to increase the gap between a local peak and the average value for sparsity. Let a patch of size $N \times N$ define the neighborhood of the local peak, and $P$ be the set of all overlapping patches. The peakiness loss is as follows:

$$ L_{peak}(Y) = 1 - \frac{1}{|P|} \sum_{p \in P} \max_{(x,y) \in P} Y_{x,y} - \min_{(x,y) \in P} Y_{x,y} $$

Instead of using the least square loss as in Eq. (1), we resort to the cosine loss $L_{cosim}$ similar to [25] for its bounded value $\in (0, 1)$. To ensure the consistent result to the peakiness loss, we calculate the cosine similarity of each patch $p \in P$. The cosine loss between keypoint maps defined as

$$ L_{cosim}(Y, Y', U) = 1 - \frac{1}{|P|} \sum_{p \in P} (\cos(Y[p], Y'[p])) $$

where $Y[p]$ is the flattened $N \times N$ patch extracted from $Y$. Finally, the learning loss of $LLF$ detector is defined as:

$$ L_{LLF} = L_{cosim}(Y, Y', U) + \frac{1}{2} (L_{feat}(Y) + L_{feat}(Y')) + \frac{1}{2} (L_{peak}(Y) + L_{peak}(Y')) $$

Next, we discuss the learning framework where the $LLF$ detector is jointly learned with the R2D2 descriptor.

### 3.4. Learning framework

We propose to enforce the $LLF$ keypoints where R2D2 descriptor matching is maximized, by optimizing the learning losses. We jointly learn our $LLF$ detector and R2D2 descriptor with separate losses, i.e., learning our $LLF$ detector with Eq. (7) and learning R2D2 descriptor with its own loss. R2D2 descriptor is learned by minimizing the loss at the reliable detector [25]. In addition, we found that joint learning of our $LLF$ detector and R2D2’s repeatable detector improves the quality of detection. The repeatable detector is learned in an unsupervised learning, which keeps the balance between discovering potential keypoints [39] [6], against our supervised learning that enforces similarity to the low-level features. Figure 3 demonstrates the joint learning of our $LLF$ detector, repeatable detector, and reliable detector in our learning framework.

The learning loss for each detector is as follows:

1. Our $LLF$ detector is learned with $L_{LLF}$ Eq. (7): $L_{Rep} = \frac{1}{2} (L_{peak}(S) + L_{peak}(S')) + L_{cosim}(S, S', T)$ (8)
2. R2D2’s repeatable detector is learned with Eq.(3) in (25):
3. R2D2 descriptor is learned with Eq.(4) in (25): $L_{APR} = \frac{1}{B} \sum_{p \in P} AP(p_{x,y})$ (9)
Table 1: The impact of the proposed modification, where our LLF detector is integrated with R2D2. The learning of LLF detector impacts the overall performance. As LLF keypoints increase, recalls, IoU, and MME, decreases. On the right, MME decreases as LLF keypoints increase. Our 100%LLF+R2D2 gives the lowest error.

Table 2: Impact on 3D reconstruction.

where AP() is computed for each of the B patches \(p_{x,y}\) in the batch. AP() is defined in [25]. These learning losses are used together when training the entire network.

4. Experimental results

In the following section, our method is evaluated across several practical scenarios, including keypoint detection and matching, 3D reconstruction, and visual localization. Visual results on keypoint detection and matching and 3D reconstruction are provided in Supplementary.

4.1. Keypoint detection and matching

Evaluation metrics. We evaluate the accuracy of the keypoint detection and matching using the following metrics from [8, 12, 13, 21]:

Intersection-over-Union error (\(\epsilon_{IoU}\)). To evaluate the error of keypoint detection, we measure \(\epsilon_{IoU}\) [12, 13] which is the compliment of the intersection over union areas between two candidates. We report the results of \(\epsilon_{IoU}\) using the keypoint scales and locations (SL), and using only the locations (L) where scales are detected by ground truth. Radius of size 30 is used to normalize the different keypoint scales, and we choose top 1000 keypoints in evaluation.

Repeatability. We measure the proportion of correct correspondences identified by \(\epsilon_{IoU}\). The correspondences between an image pair are correct, if \(\epsilon_{IoU} < 0.4\) [12]. We also report the number of correspondences where \(\epsilon_{IoU} < 0.4\) denoted as \(\#\text{Corr}_{IoU}\). Following [12, 21], the repeatability is the ratio between the correspondences (\(\#\text{Corr}_{IoU}\)) divided by the lower number of detected keypoints.

Mean Matching Accuracy (MMA). To demonstrate the performance of feature matching, we employed MMA [8], which is the average percentage of correct matches in an image pair across multiple error thresholds.

Mean Matched Error (MME). We extend MMA [8] to evaluate the matched keypoint sub-pixel error, similar to [16]. That is, we measure the average distance of the matched keypoints to the projected locations using the ground truth homography, thus, termed mean matched error (MME). We also report MMA across the error thresholds.

These evaluation metrics are presented in Table 1 and 2. We highlight the top two or top three and underline the best result. In addition, we report the average number of correct matches (\(\#\text{Inlie.}\)), total matches (\(\#\text{Matches}\)), and correct correspondences (\(\#\text{Corr.}\)) whose pixel error is < 3px.

Datasets. The full HPatches dataset [3] with ground-truth homography are used. The dataset contains 116 image sequence where 57 sequences have large illumination change, and 59 sequences with large viewpoint changes.

Baseline and training data. Our work is compared with the reproduced results of R2D2 [25]. We use the trained R2D2-WAF-N16 and R2D2-WASF-N16 released from the official site of [25]. We use the same training data and settings of WAF and WASF [25] for training our LLF detector and R2D2’s backbone from scratch, with batch size (B) = 4, number of epoch = 25, and patch size (N) = 16 (details provided in Supplementary). We use R2D2-WAF-N16-based development for every experiment. Except for visual localization, we use R2D2-WASF-N16-based development for the consistency with reported results in [25, 9].
Comparative methods. Our work is compared with 1) the standard handcrafted: SIFT [15] and SURF [15]; 2) learning-based detector: Key.Net [12]; 3) joint local feature learning methods: R2D2 [25], SuperPoint [6], DELF [23], D2-Net [8], and ASLFeat [19]. We compare with the learning methods: R2D2 [25], SuperPoint [6], DELF [23], Key.Net [12], and ASLFeat [19]. We report the proportion of keypoints leads to the higher accuracy in keypoint detection and matching. At first, without any LLF keypoints (0%LLF), the rest of keypoints provide higher #Corr.

### 4.1.1 Impact of low-level feature keypoints.

Table 1 shows the impact of our low-level feature (LLF) keypoints. We report the proportion of LLF keypoints as the percent to the total keypoints from LLF and R2D2’s repeatable detector. Generally, the higher percent of LLF keypoints leads to the higher accuracy in keypoint detection and matching. At first, without any LLF keypoints (0%LLF), the rest of keypoints provide higher #Corr. As the LLF keypoints increases (%LLF > 0), the error MME and $\epsilon_{\text{tot}}$ notably decrease, and MMA increases, which indicate the improved matched keypoint accuracy. We also report our max(LLF,Rep.) that chooses higher scored keypoints. On the right shows the MME across multiple error thresholds. The more LLF keypoints lead the lower MME, but this also cause the decrease in #Matches and #Corr.$\text{tot}$. The lower #Matches, but higher MMA, indicate that only the low error keypoints are matched. Meanwhile, the decrease in #Corr.$\text{tot}$, but higher #Corr., suggest the less clustered keypoints in repeatable area, leading to lower repeatability. Nevertheless, the impact is insignificant to the 3D reconstruction as in Table 2. As LLF keypoints increase, the reprojection error decreases, while the others are on par with R2D2. Examples of 3D reconstruction is in Figure 1.

### 4.1.2 Comparisons with the state-of-the-art methods.

Table 3 presents the comparison to state-of-the-art local feature extractors on the full HPatches dataset. The results are based on setting the error threshold to 3px. Overall,
our method shows the improved matched keypoints accuracy over the others. Our 100%LLF+R2D2 has the highest MMA, and max(LLF, Rep.)+R2D2 achieves the top three in MMA, $\epsilon_{\text{IoU}}$, both $L$ and $SL$, MMA, and #Corr. The low $\epsilon_{\text{IoU}}$ in $L$ and $SL$ validates the robustness of keypoints location and scales estimation. Meanwhile, R2D2 is the best in $\epsilon_{\text{IoU}}$ (SL). Our method has a slightly higher $\epsilon_{\text{IoU}}$ (SL) since LLF keypoints may not well aligned with scale detection of R2D2 layout. However, our method surpassed R2D2 on MMA and $\epsilon_{\text{IoU}}$ ($L$), which considers only the keypoint location. The repeatability is lower due to the less keypoint clustering, as discussed previously. SIFT and DELF have low accuracy. The repeatability is lower due to the less keypoint areas, and DELF has poor results in viewpoint. KeyNet and SuperPoint are the best in repeatability $L$ and $SL$. ASLFeat yields good results in many areas. However, KeyNet, SuperPoint, and ASLFeat have a much higher error, MMA, and $\epsilon_{\text{IoU}}$ (SL). The high MMA indicates misalignment between matched keypoints, while $\epsilon_{\text{IoU}}$ (SL) indicates the lack of robustness against the changed scales in viewpoint.

Figure 1 offers the broader view of the comparison on MMA [8] and MMA across multiple error thresholds (1-10 px). Following [8], we exclude eight high-resolution sequences and use the remaining 52 and 56 sequences with illumination or viewpoint changes, which causes a shift in the projector error, which suggests the possible future improvement.

4.2. 3D Reconstruction

Dataset. We employ the ETH benchmark [33] to evaluate the performance on 3D reconstruction. We follow most of the protocols in [33]. Except for our method and the baseline R2D2 [25], we use NetVLAD [1] to retrieve the top 20 nearby images of each image, and only match against them, for the images from Madrid Metropolis, Gendarmenmarkt, and Tower of London (instead of exhaustive matching). Then, we extract and match the local features with the matching ratio of 0.9. Sparse and dense reconstruction are performed by the SfM and MVS from COLMAP [32].

Evaluation protocols. We report the number of registered images (#Reg. Images), sparse points (#Sparse Points), mean tracking length (Track. Len.), and reprojection error (Reproj. Error) for sparse reconstruction. The number of dense points (#Dense Points) is for dense reconstruction. Our method is compared against the state-of-the-art local features and keypoint refinement in Section 4.2.1 and 4.2.2. We limit #kpts of R2D2 and ours to 10K in Section 4.2.1 and 5K in Section 4.2.2 max. scale size is set to 1600px.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Methods</th>
<th>#Reg. Images</th>
<th>#Sparse Track. Points</th>
<th>Reproj. Error</th>
<th>#Dense Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madrid</td>
<td>RootSIFT</td>
<td>1035</td>
<td>8.52</td>
<td>0.69</td>
<td>4.23M</td>
</tr>
<tr>
<td>Metropolis 1344 images</td>
<td>GeoDesc</td>
<td>1004</td>
<td>5.14</td>
<td>0.73</td>
<td>3.88M</td>
</tr>
<tr>
<td></td>
<td>D2-Net</td>
<td>965</td>
<td>5.55</td>
<td>1.28</td>
<td>3.15M</td>
</tr>
<tr>
<td></td>
<td>ASLFeat</td>
<td>1061</td>
<td>8.98</td>
<td>1.05</td>
<td>4.00M</td>
</tr>
<tr>
<td></td>
<td>SuperPoint</td>
<td>967</td>
<td>7.22</td>
<td>0.95</td>
<td>3.81M</td>
</tr>
<tr>
<td></td>
<td>R2D2</td>
<td>1000</td>
<td>9.42</td>
<td>0.96</td>
<td>4.35M</td>
</tr>
<tr>
<td></td>
<td>Our max(LLF, Rep.)+R2D2</td>
<td>1053</td>
<td>8.58</td>
<td>0.95</td>
<td>4.30M</td>
</tr>
<tr>
<td></td>
<td>Our 100%LLF+R2D2</td>
<td>1044</td>
<td>8.93</td>
<td>0.91</td>
<td>4.34M</td>
</tr>
<tr>
<td>Gendarmenmarkt 1463 images</td>
<td>RootSIFT</td>
<td>804</td>
<td>7.76</td>
<td>0.61</td>
<td>3.05M</td>
</tr>
<tr>
<td></td>
<td>GeoDesc</td>
<td>776</td>
<td>6.71</td>
<td>0.63</td>
<td>2.73M</td>
</tr>
<tr>
<td></td>
<td>D2-Net</td>
<td>708</td>
<td>5.20</td>
<td>1.34</td>
<td>2.86M</td>
</tr>
<tr>
<td></td>
<td>ASLFeat</td>
<td>846</td>
<td>13.16</td>
<td>0.95</td>
<td>3.08M</td>
</tr>
<tr>
<td></td>
<td>SuperPoint</td>
<td>631</td>
<td>8.47</td>
<td>0.96</td>
<td>2.77M</td>
</tr>
<tr>
<td></td>
<td>R2D2</td>
<td>763</td>
<td>13.35</td>
<td>0.89</td>
<td>3.11M</td>
</tr>
<tr>
<td></td>
<td>Our max(LLF, Rep.)+R2D2</td>
<td>806</td>
<td>11.58</td>
<td>0.86</td>
<td>3.19M</td>
</tr>
<tr>
<td></td>
<td>Our 100%LLF+R2D2</td>
<td>775</td>
<td>12.82</td>
<td>0.83</td>
<td>3.12M</td>
</tr>
</tbody>
</table>

Table 4: Comparison to the state-of-the-art methods on ETH benchmark [33] for 3D reconstruction.

4.2.1 Comparison with state-of-the-art local features.

From Table 4, our method, i.e., 100%LLF+R2D2 and max(LLF, Rep.)+R2D2, provides the complete sparse and dense reconstructions according to #Reg. Images and #Dense Points. Both of our works offer lower Reproj. Error than R2D2 [25]. Our 100%LLF+R2D2 has the second-best lowest Reproj. Error among the learning methods after GeoDesc [18]. This validates the impact of the proposed LLF detector for improving the accuracy of 3D reconstruction. Though GeoDesc has a very low Reproj. Error, its Track. Len. is much shorter than the others. Our method also performs better than R2D2 in #Reg. Images and #Sparse Points, while giving a shorter Track. Len. by 1.24 frame on average. This could be due to the mismatch between LLF keypoints and the scales detection from R2D2. However, our Track. Len. is still among the top three. Nevertheless, RootSIFT still has notably smaller reprojection error, which suggests the possible future improvement.
Table 5: Comparison with the keypoint refinement [9].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Madrid</td>
<td>R2D2 our rep.</td>
<td>439</td>
<td>10.56</td>
<td>0.92</td>
<td>366K</td>
</tr>
<tr>
<td>Metropolis</td>
<td>R2D2 by [9]</td>
<td>422</td>
<td>10.17</td>
<td>0.90</td>
<td>357K</td>
</tr>
<tr>
<td>1344 images</td>
<td>Ref [9]+R2D2</td>
<td>427</td>
<td>10.15</td>
<td>0.76</td>
<td>360K</td>
</tr>
<tr>
<td></td>
<td>Our max(LLF, Rep.)+R2D2</td>
<td>455</td>
<td>9.79</td>
<td>0.84</td>
<td>483K</td>
</tr>
<tr>
<td></td>
<td>Our 100%LLF+R2D2</td>
<td>463</td>
<td>10.36</td>
<td>0.83</td>
<td>474K</td>
</tr>
<tr>
<td>Gendarmenmarkt</td>
<td>R2D2 our rep.</td>
<td>993</td>
<td>9.83</td>
<td>0.97</td>
<td>1.13M</td>
</tr>
<tr>
<td>1463 images</td>
<td>R2D2 by [9]</td>
<td>988</td>
<td>9.94</td>
<td>0.98</td>
<td>1.10M</td>
</tr>
<tr>
<td></td>
<td>Ref [9]+R2D2</td>
<td>935</td>
<td>10.04</td>
<td>0.89</td>
<td>1.04M</td>
</tr>
<tr>
<td></td>
<td>Our max(LLF, Rep.)+R2D2</td>
<td>1030</td>
<td>8.29</td>
<td>0.93</td>
<td>1.33M</td>
</tr>
<tr>
<td></td>
<td>Our 100%LLF+R2D2</td>
<td>1028</td>
<td>8.78</td>
<td>0.91</td>
<td>1.29M</td>
</tr>
<tr>
<td>Tower of London</td>
<td>R2D2 our rep.</td>
<td>700</td>
<td>13.38</td>
<td>0.91</td>
<td>757K</td>
</tr>
<tr>
<td>1576 images</td>
<td>R2D2 by [9]</td>
<td>693</td>
<td>13.44</td>
<td>0.92</td>
<td>758K</td>
</tr>
<tr>
<td></td>
<td>Ref [9]+R2D2</td>
<td>700</td>
<td>13.73</td>
<td>0.76</td>
<td>760K</td>
</tr>
<tr>
<td></td>
<td>Our max(LLF, Rep.)+R2D2</td>
<td>746</td>
<td>11.10</td>
<td>0.85</td>
<td>964K</td>
</tr>
<tr>
<td></td>
<td>Our 100%LLF+R2D2</td>
<td>744</td>
<td>11.96</td>
<td>0.82</td>
<td>971K</td>
</tr>
</tbody>
</table>

4.2.2 Comparison with keypoint refinement.

Table 5 shows the comparison to the recent keypoint refinement method [9], which addresses the problem of refining the geometry of local features from multiple views. We denote this method as Ref [9]. For a fair comparison, we report the reproduced results of R2D2 by [9] as well as our reproduced results (our rep.). The similarity between R2D2 by [9] and our rep. indicates the two settings are similar. Firstly, from the Table 5 our method and Ref [9] provide less Reproj. Error than R2D2. The Reproj. Error of our 100%LLF+R2D2 is on par with Ref [9] on Gendarmenmarkt, but it does not achieve lower Reproj. Error than Ref [9] in Madrid Metropolis and Tower of London. Our Track. Len. is longer for Madrid Metropolis, but it is shorter in Gendarmenmarkt and Tower of London. Nevertheless, our method gives higher #Reg. Images and #Obs. Points in all cases. Though the goal of our method and Ref [9] are similar, Ref [9] refines the matched keypoints, which improves the accuracy in 3D reconstruction, but not #Reg. Images and #Obs. Points. Meanwhile, the LLF detector operates at the upstream of the 3D reconstruction pipeline; thus, it can improve the overall performance.

4.3. Visual Localization

Dataset. We employ the Aachen Day-Night dataset [30, 28] to demonstrate the effect on visual localization tasks, where the key challenge is to match images with extreme day-night changes for 98 queries.

Evaluation protocols. We use the evaluation protocols and tools provided by The Visual Localization Benchmark5.

Table 6: Evaluation on the Aachen Day-Night dataset [30, 28] for visual localization task.

which takes costumed features, performs image registration with COLMAP [32], and finally reports the percent of successfully localized images within 3 error tolerances: (0.25m, 2°), (0.5m, 5°) and (5m, 10°). We compare with the benchmark from official site. For our method and reproduced result of R2D2, we set max. scale size to 1600px and limit #kpts to 10K. We report results based on default image retrieval and feature matching supplied in the official site.

Results. From Table 6 our 100%LLF+R2D2 provides better performance than R2D2 (our rep.) at the error tolerances (0.5m, 5°) and (5m, 10°). Our 100%LLF+R2D2 achieves the highest percentage of successfully localized images at (0.5m, 5°) which is close to R2D2 (N = 8, 40K) that requires much more computational resources which are #kpts (40K vs. 10K) and #weights (1M vs. 0.5M+129). Because R2D2 (N = 8, 40K) employed smaller patch size (N = 8), the learned model is twice larger. ASLFeat v.2 also has competitive performance. It offers the highest percentages of successfully localized images at (0.25m, 2°) with lower #weights. Nevertheless, ASLFeat v.2 relies on using larger max. scale size (2048px) and an advanced database with depth information for training, i.e., blended images and rendered depths. However, our method improves the performance by new supervised learning technique, only.

5https://www.visuallocalization.net/
References


