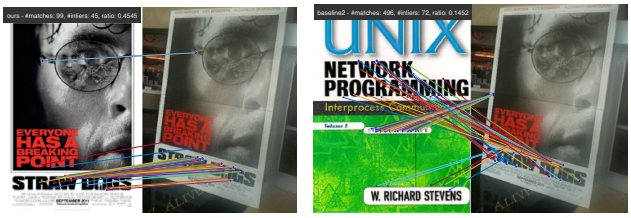
2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Unified Image Retrieval and Keypoint Matching by Local Geometric Consistency and Non-linear Diffusion

Sehyung Lee, Jongwoo Lim, and Il Hong Suh, *Fellow, IEEE*

***Abstract*— Feature-based image retrieval and feature matching have been used together in many applications, but they have been treated as two separate problems. We propose an unified approach which, for a query image, finds a set of candidate images together with feature matching results. By considering the local geometric consistency of neighboring features, we can find more and better feature matches even in challenging situations. Since the proposed forward/backward matching and non-linear diffusion run very efficiently, they can be used in the candidate image selection and improve the image retrieval performance significantly. Through quantitative comparisons we show that the proposed approach performs better than the recent state-of-the-art feature matching algorithms and image retrieval algorithms.**



(a) Ours (b) Baseline

Fig. 1: One example case of feature-based image retrieval test on the dataset [2]. The left image is database and the right is query. The proposed algorithm finds the correct image in spite of large perspective distortion, while the baseline fails to find the correct one.

I. Introduction

Local features have been used in many application areas, including robotics, augmented reality, and 3D modeling of objects or scenes. Appearance and geometric properties of image contents are captured by feature locations and descriptors, and they are used in finding correspondences in images and reconstructing geometry of the scene. Many geometric modeling tasks, specifically SLAM (simultaneous localization and mapping) or structure from motion, require fast and robust matching of features between multiple images. In addition, feature-based location recognition (loop closure detection) or object recognition need finding the images similar to the query image from many database images. For successful applications it is crucial to be able to retrieve the best matching image in a large collection of images and find good feature matches between the matched images.

Up to now image retrieval and feature matching have been solved as two separate problems. Image retrieval addresses the problem of finding a set of similar images to the given query image in a large set of database images. Vocabulary tree [1] is one of the most popular algorithms to this end. Feature matching algorithms find a set of correspondences between the features in two images, and their goal is to maximize the number or ratio of correct matches. The basic approaches compare the feature descriptors to find the best pairs and filter incorrect or uncertain ones, and recently the geometric relationship among the features in each image is considered, since it can be a good cue to distinguish ambiguous matches or to select weak but correct matches.

Although there have been huge advances in these areas, this step-by-step approach has several limitations. Since the goal of image retrieval and feature matching is different, often the retrieved images may not be geometrically consistent with the query image and result poor feature matching. Also when an image is retrieved, the feature detector needs to be run on the image or its features must be kept for further feature matching, and this cause overhead in storage and processing time. In fact the information built for image retrieval is similar to and useful for feature matching, but it is discarded after retrieval and recomputed in feature matching.

We propose to solve the image retrieval and feature matching in one unified framework, and actively utilize the local geometric relationship between features in matching step. On top of the feature correspondences searched by descriptor similarity from the vocabulary forest, we find more candidate feature matches by examining nearby features around the initial feature matches. The candidate matches are weighted by their descriptor and/or geometric similarity, and a connectivity graph on the correspondences are constructed for diffusion of the weights along the closely related correspondences. Finally for the query image, the candidate database images as well as the feature matches between the query and database images are returned.

Our algorithm has several advantages over the conventional stratified approach. First by considering the photometric and geometric properties of features and diffusing the scores of feature correspondences in the image retrieval stage it improves both feature matching and image retrieval performance. Second the proposed algorithm is designed to output the feature matching results for the retrieved images, thus additional feature matching for further processing is unnecessary. Also there is no need to keep the database

Sehyung Lee is with the Department of Systems Science, Kyoto University, Kyoto, Japan, sehyung@sys.i.kyoto-u.ac.jp
Jongwoo Lim is with the Division of Computer Science and Engineering, Hanyang university, Seoul, Korea, jlim@hanyang.ac.kr
Il Hong Suh is with the Department of Electronics and Computer Engineering, Hanyang University, Seoul, Korea, ihsuh@hanyang.ac.kr
All correspondences should be addressed to Il Hong Suh

images or feature descriptors for this purpose. Third the proposed algorithm runs very fast, enough to be used in online applications and generates high quality feature matching results in challenging scenarios. We extensively evaluate the proposed algorithm and quantitatively compare with other state-of-the-art algorithms.

The paper is organized as follows. The related work is discussed in Section II, and the proposed algorithm is presented in detail in Section III. The experimental results are shown in Section IV, then we conclude in Section V.

II. Related Work

After SIFT [3] and SURF [4], various local feature algorithms such as BRISK [5], BRIEF [5], ORB [6] and KAZE [7], were developed to improve the performance of feature detection and description. In these local feature algorithms, feature correspondences are established by comparing the descriptor vectors, with simple filtering of ambiguous matches using the descriptor distance ratios called nearest neighbor distance ratios (NNDR). However, in many real cases of large illumination or viewing-direction changes, the descriptors of same world points become significantly different and correct correspondences cannot be found by simply matching the feature descriptors. To overcome the limitation of the simple descriptor matching approach, many advanced feature matching algorithms utilizing other information have been proposed.

Leordeanu and Hebert [8] introduced pairwise geometric information to build the affinity matrix. Torresani *et al*. [9] designed a cost function based on the appearance and spatial proximities which can be efficiently optimized by dual decomposition. Cho *et al*. employed the agglomerative clustering [10] and random walk algorithm [11] instead of using spectral clustering [8]. Zaragoza *et al*. [12] proposed a technique to compute an as-projective-as-possible warping that aims for the feature locations to be globally projective. This method is developed for image stitching application in which the correspondence algorithm is based on the multiple-homography fitting using their previous work [13]. In Lin *et al*. [14], starting from a set of initial matches, their algorithm computes an affine motion field between two images by exploiting that each match defines a local affine transformation. Given the motion field, more correspondences are then recovered by finding the nearest matches in the descriptor space.

The above feature matching algorithms run on a single image pair. To find the candidate image pairs in multiple images, one may use the pairwise feature matching for all pairs, but it is very inefficient. The bag-of-visual-words approach is proposed to address this problem, and it is widely used. Nister *et al*. [1] proposed the vocabulary tree and the weighting method for matched visual words based on the term-frequency and inverted-document-frequency. Philbin *et al*. [15] applied geometric verification on the correspondences obtained by visual word matching, and the inlier count is used to re-rank the top-k retrieved images. In [16], Mikulik *et al*. proposed the method to reduce the quantization errors of partitioned feature space by learning the distance function between the leaf nodes.

These classical approaches only consider the appearance of input features and ignores geometric arrangements of features. There have been several attempts [17], [18], [19], [2] to embed geometric properties to the bag-of-visual-words approach. These algorithms mainly focused on developing good visual word weighting methods based on the geometric consistency between correspondences.

The proposed algorithm differs from the previous works in several ways. Our method focus on finding more and better feature correspondences, not giving weights on the matched visual words for better image retrieval performance. Also, unlike existing feature matching algorithms mentioned above, our algorithm can compute feature correspondence sets on multiple images without performing feature matching between pairs of images. By considering the local geometry of neighboring features in a very efficient manner, it can find more and higher quality feature matches in much faster speed.

III. Algorithm Description

For a query image, the proposed algorithm finds the best candidate images in the image database, and the feature matches between the images at the same time. The feature matches are found by constructing a graph between features in the query and database images, and selecting the most probable matches. The graph is built by forward and backward matching. The forward matching is similar to the standard vocabulary tree querying, and puts the edges from the query features to the database features. The backward matching considers the geometric proximity among features, and add the edges from the database features to the query features. The feature graph is then converted into an undirected graph whose nodes are possible feature matches with computed scores and edges represent the commonality relationship. The algorithm finds the diffused scores of the nodes and selects the final feature matches. The details of the proposed algorithm is described in this section.

A. Initial Forward Matching

In the proposed framework, the visual vocabulary forest [20], [21], which is a set of multiple different vocabulary trees, is used to find initial candidate feature matches. The quantization errors in using a single vocabulary tree can be mitigated by aggregating the outputs of the trees in the vocabulary forest.

Instead of storing the document IDs in the vocabulary forest, we store the individual feature IDs in the leaf nodes of the forest, as well as the geometric properties (position, scale, and orientation) of the features. Figure 2 illustrates the overview of initial matching process using the vocabulary forest. For a query image, its features are queried to the forest, and the matched features in the database can be quickly found. One query feature can be matched with many database features multiple times in different trees. Let $F = \{\mathbf{c}_i\}$ be the set of forward feature matches, $\mathbf{c}_i = (q_i, d_i)$,

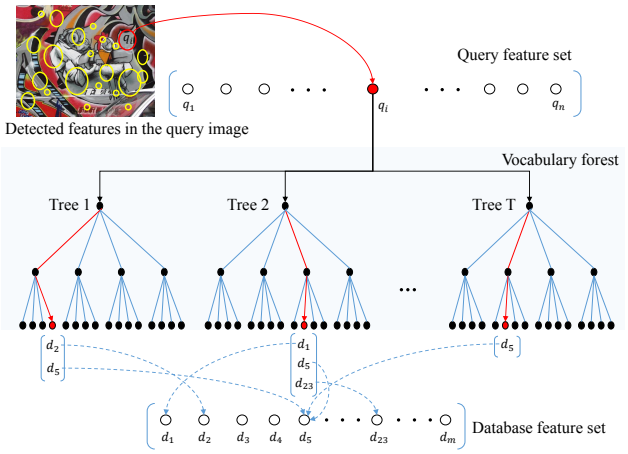


Fig. 2: The overview of initial forward matching. The initial matches are found by the vocabulary forest which is composed of multiple vocabulary trees. The database features stored in the leaf nodes matched with the input query features are considered as the forward matches.

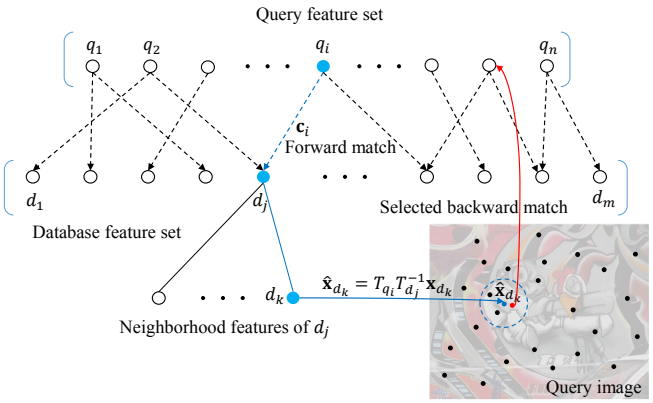


Fig. 3: The backward matching algorithm. The neighbor database features of the database features in a forward match are reprojected into the query image using the relative geometric transformation (Equation 3). The backward matches are created by examining the query features closely located in the reprojected position.

where q_i is the feature ID in the query image and d_i is the feature ID in the database. The forward match finds matched features across multiple database images, and they can be grouped as the database images. Without loss of generality we can divide F image-wise and assume that all d_i's belong to one database image. In our implementation, we start from the database image with most forward matches, and iterate on the next images.

The cost of a forward match $\mathbf{c}_i$ is computed by comparing the corresponding descriptor vectors as follows:

$$e_f(\mathbf{c}_i) = \left\| \mathbf{d}_{q_i} - \mathbf{d}_{d_i} \right\|, \tag{1}$$

where $\mathbf{d}_{q_i}$ and $\mathbf{d}_{d_i}$ are the feature descriptor vectors of the the query and database features. If the descriptors of database features are not stored, the matched database feature descriptor can be approximated by averaging the matched leaf nodes of the vocabulary trees,

$$\tilde{\mathbf{d}}_{d_i} = \frac{1}{N_{VT}} \sum_{k=1}^{N_{VT}} L_{VT_k}(d_i), \tag{2}$$

where N_{VT} is the number of vocabulary trees, and $L_{VT_k}(d_i)$ is the leaf nodes matched with database feature d_i.

B. Backward Feature Matching

The forward matches found by querying the vocabulary forest do not contain any information related to the arrangement of matched features in the 2D image space. Another problem is that the number of forward matches are not large enough for the next task after feature matching. In general structure-from-motion or loop closure requires at least hundreds of feature matches, but the result by vocabulary forest is much smaller, even though the ensemble of vocabulary trees reduces the quantization errors. To overcome this problem, we propose to find additional matches using local geometric consistency of neighboring features.

Let T_f be the transformation that maps a unit patch, which is centered at the origin, axis-aligned, and unit-sized, onto the patch of a feature f. In case of SIFT or SURF, each feature patch is represented as x-, y-position, scale, and orientation $(x_f, y_f, \sigma_f, \theta_f)$, and then T_f is written as a 2D similarity transform

$$T_f = \begin{bmatrix} \sigma_f \cos\theta_f & -\sigma_f \sin\theta_f & x_f \\ \sigma_f \sin\theta_f & \sigma_f \cos\theta_f & y_f \\ 0 & 0 & 1 \end{bmatrix}.$$

Depending on the geometric properties provided by local features, T_f can be any 2D transformation including translation, similarity, affine, or homography. For a feature correspondence $\mathbf{c} = (q, d)$, the warped position of a neighborhood feature d' near d can be written as

$$\hat{\mathbf{x}}_{d'} = T_q T_d^{-1} \mathbf{x}_{d'}, \tag{3}$$

where $\mathbf{x}$ is x-and y-position of the feature. The geometric properties of the forward matches guide backward matches of the neighboring database features to the query features. A query feature q' that is close to the transformed position of the neighbor feature d' is considered as a possible match, under the assumption that local geometry around the features q and d is preserved if the match is correct. The neighboring features can be efficiently found using Kd-trees of the 2D positions of the database and query features.

For a forward match (q, d), K nearest neighbors of d are found, their transformed positions in the query image are computed, then κ-nearest query features within the maximum radius to the computed positions are considered as the backward matches. Among the computed candidate backward matches for a neighbor database feature d', the backward correspondence $\mathbf{c}' = (q', d')$ is determined by choosing the match with the minimum cost,

$$e_b(\mathbf{c}') = \left\| \mathbf{d}_{q'} - \mathbf{d}_{d'} \right\| + \lambda \left\| \mathbf{x}_{q'} - \hat{\mathbf{x}}_{d'} \right\|, \tag{4}$$

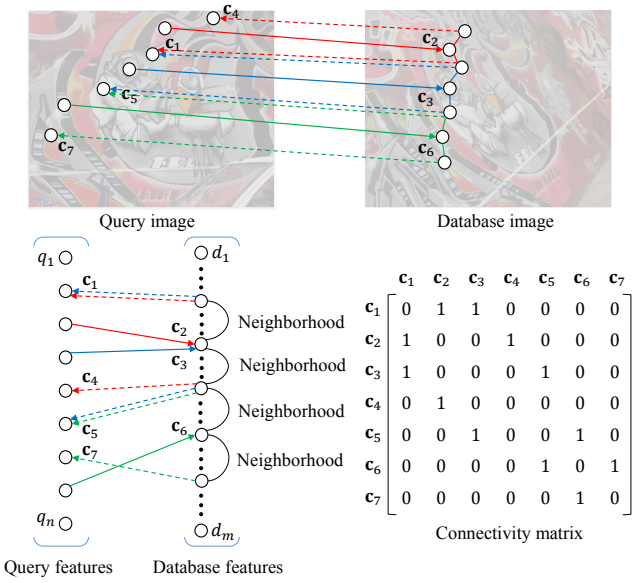


Fig. 4: This figure shows how the connectivity matrix is computed. The arrows from query to database features denote the forward matches, the dotted arrows from database to query features are the backward matches, and they are color-coded so that the backward matches have the same color as the corresponding forward match. The database features in multiple forward matches or those which are neighbors of multiple forward matches ($\mathbf{c}_1$ and $\mathbf{c}_5$) causes shared backward matches and forms the communities of matches.

where λ is the scaling coefficient. Figure 3 graphically shows how candidate backward matches are computed. Note that multiple forward matches can make one database feature linked to multiple query features. After computing the backward correspondence set $B = \{\mathbf{c}'\}$, the duplicated correspondences that connect the same query and database features are merged into the one correspondence.

C. Correspondence Selection by Non-linear Diffusion

The forward and backward correspondence sets F and B represents different aspects of features: descriptor similarity and geometric consistency. The forward matching looks for the features with similar appearance (descriptor) using the vocabulary forest. On the other hand, the backward matches are established by checking the geometric consistency of neighboring features near the forward matches. Now we describe the diffusion and selection algorithm which generates the final feature matches among the whole candidate feature correspondences $C = F \cup B$.

1) Correspondence Connectivity: To find good correspondences, we propose to construct a correspondence graph, diffuse the matching costs through nearby correspondences, and then select the correspondences above a threshold. The correspondence graph has all found correspondences C as the nodes, and the edges are added as follows. Note that each backward match has one forward match by which it is generated. A forward match $\mathbf{c}$ and a backward match $\mathbf{c}'$ are linked if $\mathbf{c}'$ is established by $\mathbf{c}$, i.e., $\mathbf{c}$ "guides" $\mathbf{c}'$. Figure 4 presents how the connectivity is created. The arrows and dotted arrows denote the forward and backward matches, respectively, and they are color-coded so that the guided backward matches have the same color as the guiding forward matches. There are 3 forward matches, $\mathbf{c}_2$, $\mathbf{c}_3$, and $\mathbf{c}_6$, and 4 backward matches $\mathbf{c}_1$, $\mathbf{c}_4$, $\mathbf{c}_5$, and $\mathbf{c}_7$. $\mathbf{c}_1$ and $\mathbf{c}_5$ are created by multiple forward matches, and these shared matches link related correspondences and form a community. In this example, $\mathbf{c}_1$ links blue and red match sets, and the blue and green sets are connected by $\mathbf{c}_5$. The final correspondence scores are computed using the constructed connectivity matrix.

2) Diffusion Model: Each node of the constructed correspondence graph has a cost representing the quality of the correspondence computed in the forward and backward matching. Due to the features sharing neighboring features, most correspondences are connected to other correspondences and form several communities. By diffusing the cost along the connections, the costs of highly correlated correspondences are mixed together. Let $\mathbf{G}$ and $\mathbf{s}$ be the connectivity matrix and the cost vector of the nodes. In diffusion we iteratively multiply $\mathbf{G}$ to $\mathbf{s}$,

$$\mathbf{s}^{t+1} = \bar{\mathbf{G}}\mathbf{s}^t, \tag{5}$$

where superscript t is the iteration, and $\bar{\mathbf{G}}$ is the normalized connectivity matrix. $\bar{\mathbf{G}}$ is set to $\mathbf{D}^{-1/2}\mathbf{G}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is a diagonal matrix whose diagonal elements are the sum of the rows of $\mathbf{G}$. This simple isotropic diffusion makes the costs of correspondences in the same groups averaged together. To consider the initial cost of the correspondences, we add a reweight term to the diffusion model, and it can be written as:

$$\mathbf{s}^{t+1} = (1-r)\bar{\mathbf{G}}\mathbf{s}^t + r\mathbf{s}^0, \tag{6}$$

where r is the reweight term for initial cost $\mathbf{s}^0$. It can be rewritten in terms of correspondence-wise costs as

$$\mathbf{s}_i^{t+1} = r\mathbf{s}_i^0 + \frac{(1-r)}{|N(i)|}\sum_{j\in N(i)}\mathbf{s}_j^t, \tag{7}$$

where $N(i)$ is the set of nodes directly connected to the i-th node.

To improve the convergence, we add two additional terms to this reweighted diffusion model,

$$\mathbf{s}_i^{t+1} = r\mathbf{s}_i^0 + \frac{(1-r)}{|N(i)|}\sum_{j\in N(i)}\left\{\mathbf{s}_j^t - \delta(\mathbf{s}_j^t) - \gamma(\mathbf{s}_j^t)\right\}, \tag{8}$$

where δ is the distinctive term and γ is the early acceptance term.

The function δ verifies whether the correspondence is distinctive among the matches from the database feature.

$$\delta(\mathbf{s}_j^t) = \begin{cases} \alpha & \text{if } \mathbf{s}_j^t/\varepsilon_j < \theta_\delta \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where ε_j is the second smallest cost among the correspondences from the database feature of the j-th correspondence,

and θ_δ is the ratio threshold. The early acceptance function γ is defined as

$$\gamma(\mathbf{s}_j^t) = \begin{cases} \beta & \text{if } \mathbf{s}_j^t < \theta_\gamma \\ 0 & \text{otherwise} \end{cases} . \quad (10)$$

γ function checks if the cost of the j-th correspondence is low enough that it is likely to be accepted as the final match, and in such case it lowers the cost. According to this anisotropic diffusion in the cost space, the costs of correspondences included in the same community are regularized. If distinctive or early accepted correspondences are in the same community, other similar correspondences are also more likely to be accepted. In the final step, we select the only correspondences satisfying the condition $\mathbf{s}^t < \theta_\gamma$, and eliminate the matches with larger cost than the threshold.

Overall algorithm flow.

The proposed algorithm runs as follows:

- Compute the initial forward matches by voting the database features using the vocabulary forest.
- Compute the backward matches by warping the matched database features into the query image.
- Build the connectivity matrix and iterate the anisotropic diffusion process.
- Select correspondences which satisfy $\mathbf{s}^t < \theta_\gamma$.
- Update the forward matches using selected correspondences, and iterate Steps 2 to 5 until new candidates are not found.

Once the feature correspondences are computed, further post-processing such as RANSAC can follow as in conventional approaches.

IV. Experimental Results

We conducted extensive experimental evaluation of the proposed algorithm on various datasets. In the first experiment, the quality of correspondences obtained by the proposed method is evaluated. It includes the quantitative comparison with the widely used NNDR-based algorithms, and the state-of-the-art feature matching methods [12], [14], [22], using ASIFT [24] feature and a well-known benchmark dataset [23]. In the second experiment we tested the proposed algorithm in the feature-based image retrieval task. In order to build the vocabulary forest, we use a standard implementation from VLFeat [25] using approximately 20 million descriptors extracted from various images. Total 4 vocabulary trees with 216,000 leaf nodes, the branching factor 60 and number of levels 3, are built. The experiments are conducted in MATLAB on a computer with i7 4.0 GHz CPU and 32GB memory, and some functions are implemented as MEX functions. The parameters are fixed for all experiments as $\lambda = 0.01$, $r = 0.45$, $\theta_\delta = 0.85$, $\alpha = 0.07$, $\theta_\gamma = 0.51$ and $\beta = 0.06$.

A. Feature Matching Performance

To measure feature matching performance, Heinly *et al.* [26] proposes three evaluation metrics: *putative match ratio*, *precision*, and *matching score*. The selectivity of algorithm can be tested by the putative match ratio, PMR = $\frac{\text{\#putative matches}}{\text{\#features}}$, which is the ratio of computed matches to all detections. For the given features in two images, some algorithm may output a large set of putative matches without much filtering, whereas another algorithm may generate only a small set of very certain matches. The precision, Precision = $\frac{\text{\#inlier matches}}{\text{\#putative matches}}$, represents the ratio of correct ones out of the putative matches (the inlier ratio), which measures the 'purity' of feature matches. The matching score, MS = $\frac{\text{\#inlier matches}}{\text{\#features}}$, shows how many true matches are selected - the 'recall' in detection literature. The algorithms can be quantitatively evaluated as how many features were matched (PMR), and how many are true in the output matches or in all possible matches (Precision and MS respectively).

We compare the proposed algorithm with four popular or state-of-the-art algorithms: NNDR with two different thresholds (0.8 for NNDR1 and 0.9 for NNDR2), Nearest, SCV [22], MDLT [12], and BMF [14]. For SCV, MDLT, and BMF, we used the author's original implementations[1]. For quantitative comparison, we use the dataset by Mikolajczyk *et al.* [23] as the ground-truth homographies of all image pairs are provided. It consists of eight sets of one reference and five target images (approximately 800 $\times$ 640 pixels). The level from L1 to L5 indicates the amount of perspective deformation or degradation. In this experiment, we use the ASIFT [24] feature since BMF works only with ASIFT, and the threshold for correct match is set to 5 pixels.

We tested two versions of our algorithms, Ours-VT and Ours-desc. Ours-desc uses the original database descriptors, and Ours-VT uses the approximated database descriptors by averaging the matched leaf nodes of vocabulary trees (Eqation 2). The proposed algorithm can handle one-to-many-image matching. Note that the results are computed by querying the reference image to the database with five L1-L5 images, while other algorithms performed one-to-one image matching.

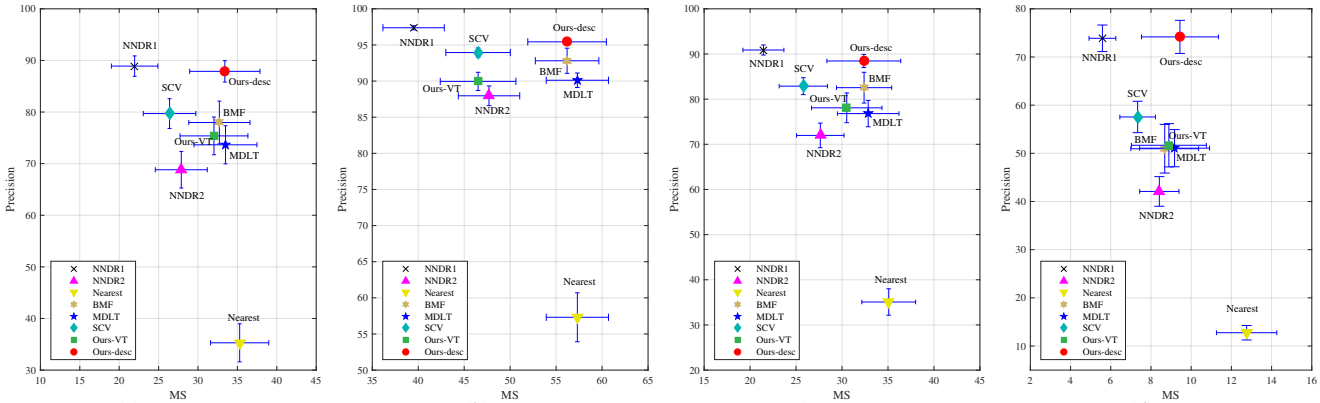
Table I and Figure 5 show the averaged performance in each level (L1–L5) and over all levels. The proposed method Ours-desc outperforms the others by a large margin in MS and Precision metrics, especially in the presence of large deformation (L4 and L5). Ours-VT performs similarly to the state-of-the-art algorithms but generates better results than simple algorithms (NNDR and Nearest).

In MS-vs-Precision performance shown in Figure 5, the proposed methods are located at the right-upper positions compared to the other algorithms. In many tasks, having a large number of inliers is as important as having a high inlier ratio. For example, in many feature-based applications such as structure-from-motion, location recognition, and 3D environment mapping, the number of RANSAC iterations is determined by the inlier ratio, and the quality and fidelity of the 3D reconstruction is largely affected by the number of inliers. Alto the number of inliers can affect stability or success ratio of algorithms since they usually accepts

[1]BMF [14]: http://mmcheng.net/bfun/
MDLT [12]: https://cs.adelaide.edu.au/~jzaragoza/doku.php?id=mdlt
SCV [22]: http://cmp.felk.cvut.cz/~cechj/SCV/

Algorithms		NNDR1	NNDR2	Nearest	SCV	MDLT	BMF	Ours-VT	Ours-desc
AVG	PMR	23.24	35.43	100	29.79	39.60	37.96	38.07	35.98
	Precision	88.90	68.82	35.29	79.71	73.65	77.97	75.38	87.89
	MS	21.96	27.88	35.29	26.39	33.49	32.71	32.03	33.41
L1	PMR	40.39	52.71	100	48.86	62.78	59.22	61.72	60.18
	Precision	97.38	87.96	57.31	93.95	90.11	92.81	89.96	95.43
	MS	39.50	47.70	57.31	46.51	57.31	56.19	56.20	57.69
L2	PMR	32.53	45.44	100	40.49	54.55	49.41	52.69	50.55
	Precision	95.54	82.40	48.08	90.19	86.54	91.23	85.36	93.61
	MS	31.21	38.93	48.08	37.23	47.81	46.13	45.42	47.21
L3	PMR	23.00	35.68	100	29.52	39.68	37.97	36.79	35.26
	Precision	90.88	71.97	35.08	82.88	76.80	82.54	78.08	88.45
	MS	21.45	27.64	35.08	25.79	32.85	32.40	30.51	32.37
L4	PMR	13.34	25.48	100	18.83	26.42	27.16	24.71	22.44
	Precision	86.83	59.67	23.21	73.97	63.72	72.32	71.82	87.76
	MS	12.05	16.72	23.21	15.08	20.32	20.17	19.12	20.32
L5	PMR	6.92	17.84	100	11.24	14.59	16.07	14.43	11.47
	Precision	73.88	42.09	12.76	57.55	51.07	50.95	51.68	74.17
	MS	5.59	8.42	12.76	7.34	9.17	8.68	8.89	9.44

TABLE I: PMR, Precision, and MS of SCV [22], MDLT [12], BMF [14], and the proposed methods on the test dataset (unit: %). The dataset [23] contains five different levels of geometric and photometric variations. From top to bottom, the total average and the average performance of each level (L1–L5) are shown.



(a) Overall (b) L1 (c) L3 (d) L5

Fig. 5: Comparisons of feature matching algorithms (unit: %). All figures show the average MS-vs-Precision performance with the bars for the standard deviation. The algorithm located at the right-upper position is better. (a) shows the average of all levels L1–L5, and (b-d) shows the performance of different levels.

the results only when the inliers are more than a certain threshold.

Note that both proposed algorithms runs several orders of magnitude faster than the state-of-the-art algorithms. The processing times of NNDR1, SCV, BMF, MDLT, and Ours-desc on 1087 and 966 ASIFT features, L2 pairs of 'bark' image set, are 0.019, 0.75, 2.06, 5.16, and 0.021 s respectively. In the larger-scale test with 4613 and 5663 ASIFT features, L5 pairs of 'graf' image set, the processing times are 0.48, 3.15, 3.44, 15.88, 0.065 s respectively. In matching the one-to-five-image matching of 'bark' set composed of 1087 query and 8222 database ASIFT features, Ours-desc takes 0.086 s. In execution time there is no meaningful difference between Ours-desc and Ours-VT.

Figure 6 shows representative examples of feature matching by different algorithms (please zoom in to see the details). It shows that the proposed method finds more inlier matches with higher precision. These experimental results validate that the proposed algorithm can efficiently compute better correspondence sets in matching the one-to-many-image matching.

B. Feature-based Image Retrieval

We extend the one-to-many-image matching experiment to a image retrieval experiment with more images in the database. In many robotics and augmented reality applications, it is common to find the matching image in hundreds or thousands of database images, and our algorithm is well suited for these applications. The proposed algorithm is not designed to handle millions of images as of now, but we plan to extend it to handle large-scale databases.

We build a standard feature-based image retrieval system [15] as a baseline comprised of KAZE [7] features, vocabulary tree [1], and geometric verification using RANSAC [27]. The test was performed using the image

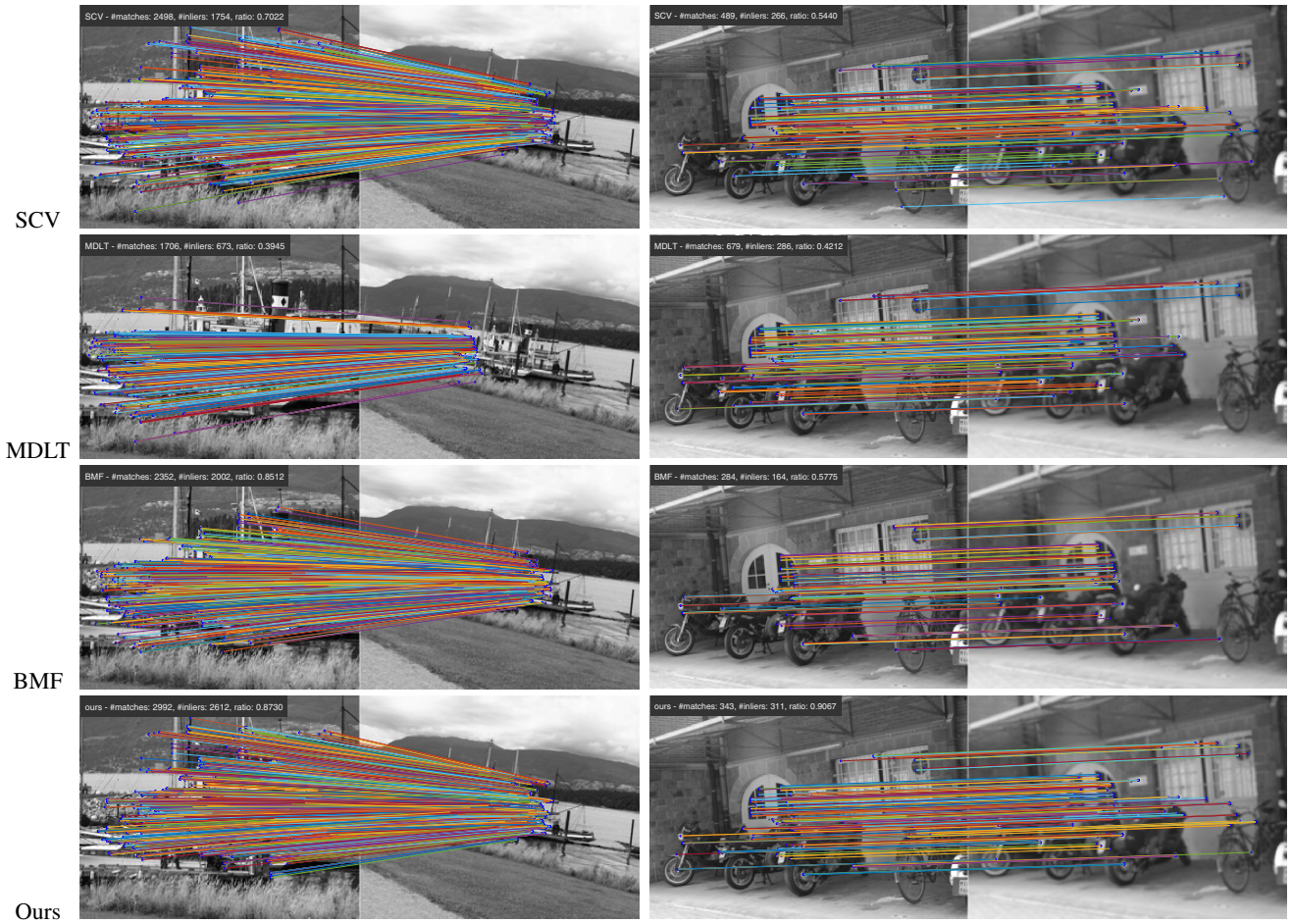
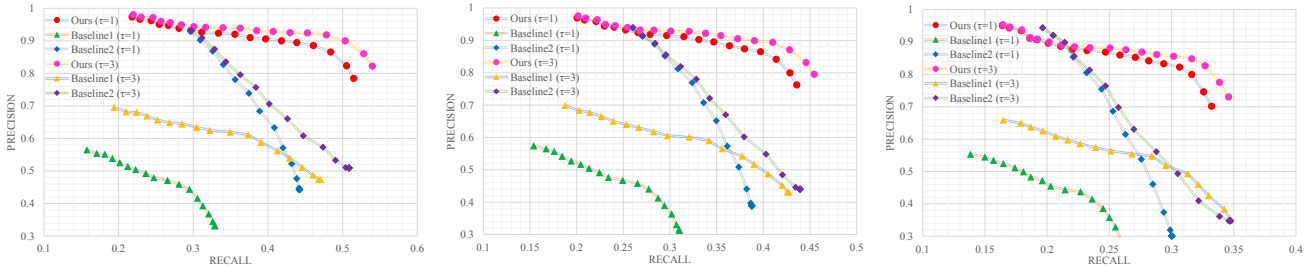


Fig. 6: From top to bottom, the feature matching results of SCV, MDLT, BMF, and our methods on L4 pairs of 'boat' and L5 pairs of 'bikes' in the dataset [12]. The proposed algorithm finds more and better feature matches than others with higher precision.



(a) Top-20 candidates (b) Top-10 candidates (c) Top-5 candidates

Fig. 7: Precision–recall curves on the dataset [2]. From left to right, the experimental results are shown with varying number of top-k candidates. It can be seen that the retrieval performance is improved by our matching algorithm. Also one can see that Ours Top-5 performs much worse than Top-10 or Top-20, which tells that the database images outside of Top-5 are re-ranked after backward matching and non-linear diffusion, thus verifies the effectiveness of the proposed algorithm.

retrieval dataset [2] comprising 400 database and 2,500 query images. For each query image, the extracted KAZE features are searched in the tree, and the top-k images in terms of the number of matched visual words in the database are processed for backward matching and non-linear diffusion. Baseline 1 uses the correspondences matched in the leaf nodes of the vocabulary tree as RANSAC input. In Baseline 2, NNDR1 is additionally performed between the query and top-k retrieved images to obtain more refined correspondences.

The feature matches established by individual algorithms are filtered by geometric verification using RANSAC with a fundamental matrix for final ranking of retrieved images. We varied top-k candidate images (from 20, 10 to 5), and threshold for the number of inliers (from 10 to 95 with 5 interval). The retrieved candidate images are reranked based on the number of inlier correspondences. If the correctly matched image is ranked within the acceptance parameter τ,

we count the retrieval is successful. We tested the experiment with two acceptance parameter: $\tau = 3$ and $\tau = 1$, and measured precision and recall rates of retrieval results.

As shown in the precision–recall curve of Figure 7, the image retrieval performance is significantly improved by adopting the proposed method. In the first graph in Figure 7, for a precision of approximately 90%, the recall rate increases from 30.93% to 41.82% compared to the second baseline. Also note that the performance of our algorithm in Top-5 is much worse than those of Top-10 or Top-20. This tells us that the database images outside of Top-5 forward matches are re-ranked after the backward matching and non-linear diffusion, thus verifies the effectiveness of the proposed algorithm. Figure 1 shows one example case where our algorithm can successfully finds the correct image whereas the baseline approach fails.

V. Conclusion

In this paper, we propose a unified framework of feature matching and image retrieval. Initial forward correspondences are computed by a set of vocabulary trees, and backward matches are established by considering the geometric arrangement of matched features. The final feature correspondences are selected among the collected forward and backward matches by combining the appearance and geometric similarities in the non-linear diffusion process. The experimental evaluation shows that the proposed algorithm efficiently computes much improved feature correspondences in one-to-many-image feature matching. We also demonstrate that the proposed algorithm solves the image retrieval and feature matching simultaneously, and yields enhanced performance in both directions.

ACKNOWLEDGMENT

This research was supported by the Industrial Strategic Technology Development Program(10044009) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea).

References

[1] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 2161–2168.

[2] X. Wang, M. Yang, T. Cour, S. Zhu, K. Yu, and T. X. Han, "Contextual weighting for vocabulary tree based image retrieval," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 209–216.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[5] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2548–2555.

[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[7] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "Kaze features," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 214–227.

[8] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1482–1489.

[9] L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph matching: Models and global optimization," in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 596–609.

[10] M. Cho and J. Lee, "Feature correspondence and deformable object matching via agglomerative correspondence clustering," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1280–1287.

[11] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 492–505.

[12] J. Zaragoza, T.-J. Chin, Q.-H. Tran, M. S. Brown, and D. Suter, "As-projective-as-possible image stitching with moving dlt," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 7, pp. 1285–1298, 2014.

[13] T.-J. Chin, J. Yu, and D. Suter, "Accelerated hypothesis generation for multistructure data via preference analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 4, pp. 625–638, 2012.

[14] W.-Y. D. Lin, M.-M. Cheng, J. Lu, H. Yang, M. N. Do, and P. Torr, "Bilateral functions for global motion modeling," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 341–356.

[15] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[16] A. Mikulík, M. Perdoch, O. Chum, and J. Matas, "Learning a fine vocabulary," in *European Conference on Computer Vision*. Springer, 2010, pp. 1–14.

[17] X. Li, M. Larson, and A. Hanjalic, "Pairwise geometric matching for large-scale object retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5153–5161.

[18] Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang, "Spatial-bag-of-features," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3352–3359.

[19] Y. Zhang, Z. Jia, and T. Chen, "Image retrieval with geometry-preserving visual phrases," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 809–816.

[20] T. Yeh, J. Lee, and T. Darrell, "Adaptive vocabulary forests br dynamic indexing and category learning," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[21] K. Mikolajczyk and H. Uemura, "Action recognition with motion-appearance vocabulary forest," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[22] J. Čech, J. Matas, and M. Perdoch, "Efficient sequential correspondence selection by cosegmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1568–1581, 2010.

[23] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *International journal of computer vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[24] J.-M. Morel and G. Yu, "Asift: A new framework for fully affine invariant image comparison," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 438–469, 2009.

[25] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1469–1472.

[26] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 759–773.

[27] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.