Have your cake and eat it too: Fast robust matching with SIFT features

Anonymous ACCV 2014 submission

Paper ID ***

Abstract.

1 Introduction

Whenever we match local image features we are faced with a choice between performance and precision. On one hand SIFT features proposed by Lowe [1] have shown again and again to compare favorably to other local image descriptors especially under unrestrained conditions such as perspective change with non-planar scenes [2,3,4]. On the other, SIFT keypoints and descriptors are slow to compute, the main raison d'être for later local image features. In many applications of computer vision we would like the increase the computational performance in order to work on larger images, bigger image sets, at faster frame rates or with more limited hardware but we cannot give up the additional precision that SIFT affords us over other local features without negative effects for our application.

In this paper we introduce a matching algorithm designed to match features between two images only in image areas that are likely to correspond. This approach is much faster than traditional methods because there is no need to compute descriptors for areas in the image that are not matched. We provide two variations of the algorithm: a *general* and a *retrieval* approach. The *general* approach functions as a traditional matching algorithm and matches two unknown images albeit faster and more robustly than conventional methods. The *retrieval* approach on the other hand assumes that we know one of the images we intend to match beforehand but given this trade-off this approach offers performance a magnitude faster than existing matching methods without compromising on accuracy. We will unimaginatively refer to the proposed algorithm as *Fast-Match* in this paper and make clear from the context whether this refers to the *retrieval* or *general* variation. As an aside we adopt an artificial distinction between the words *match* and *correspondence* to make it clear when a match between two images is correct. The word *match* will be used to denote a result from a matching algorithm that may or may not be correct while a *correspondence* refers to an actual visual correspondence between two points in two images.

The problem *Fast-Match* attempts to solve is two-fold. By matching only image areas that are likely to correspond we hope to improve accuracy by entirely ignoring parts of the images that would otherwise likely be a source of incorrect correspondences, at the same time, this enables us to improve computational

speed by not having to compute keypoints and descriptors for large parts of the images and at the same time reducing the amount of feature points we need to match. Both problems have been addressed in part by past work.

As noted, the cost of computing SIFT keypoints and descriptors is addressed by other local image features such as SURF [5], BRIEF [6], BRISK [7] just to mention a few. Similarly efforts have been made to improve SIFT itself such as PCA-SIFT [8], and GLOH [2]. Both apply PCA to reduce descriptor lengths and improve distinctiveness but neither have been shown to consistently outperform SIFT [2,9].

Efforts to reduce the computational costs of finding nearest neighbours to feature points has largely been focused on metric trees. Naïvely the set of nearest neighbours between features in two images can computed by brute force in $O(n^2)$ where n is the number of features in the two images. When SIFT was originally published Lowe proposed using the Best-Bin-First method to approximately search for nearest neighbours [10,11]. This reduces the computational complexity to $O(nlog(n))$ but even approximate metric trees are hard pressed to compete with brute force due to the high dimensionality of SIFT and the constant costs incurred with constructing and searching in a metric tree. Later work by Muja and Lowe has focused on improving approximate nearest neighbour searches by using several KD-Trees simultaneously while optimizing the tree structure using K-Means to cluster similar features [12]. Recent work on knn-graphs shows a lot of promise for high dimensional cases [13]. We later review these improvements and their effect on efficiently matching large scale images.

A wealth of matching methods have focused instead on increasing the efficiency of local feature matching. *Fast-Match* builds upon the foundation of *Ratio-Match* introduced originally by Deriche et al. [14] and Baumberg [15] even though Lowe is usually credited for introducing it [1]. They both propose using the ratio of the similarity of the best to second best correspondence of a given point to evaluate how unique it is. Their finding has later been tested by several independent teams, all concluding that thresholding based on this ratio is generally superior to thresholding based on similarity or returning all nearest neighbours [1,2,3,16].

Brown and Lowe [17] extend ratio match to deal with a set of images by using not the ratio of the best and second best correspondence, but the average ratio of the best and the average of second best correspondences across a set of images. Rabin et al. [16] try to enhance descriptor matching by looking at the statistical distribution of local features in the matched images, and only return a match when such a correspondence would not occur by mere chance. Finally, Arnfred and Winkler generalize *Ratio-Match* to make use of both of the matched images to provide a more accurate evaluation of the uniqueness of a match [18].

Another inspiration for the design of *Fast-Match* is *Patch-Match* as introduced by Barnes et al. [19]. Like *Fast-Match*, *Patch-Match* is an iterative algorithm for fast image matching, but unlike *Fast-Match* which is focused on sparse local image features, *Patch-Match* is designed for dense image features. It works

by randomly creating a set of matches from both images and then iteratively improving it.

For sparse local image features many solutions have combined *Ratio-Match* with various geometric constraints to improve matching. These constraints are based on assumptions regarding the transformation between the *query* and *target images*. A commonly used example is *RANSAC* applied to feature matching where matches are chosen from a pool of candidates according to how well they approximate a global epipolar geometry [20,21,22]. Similar global angular and distance constraints can be used to filter a set of matches as shown in [23,24]. Finally the problem of feature matching can be modelled as a graph matching problem where each feature is a vertex, and edge values correspond to a geometric relation between two features as demonstrated in [25,26,27]. Pairwise constraints have shown to be a popular alternative for cases that require more flexibility such as scenes with moving elements and non-planar perspective change. Introduced by Leordeanu and Herbert [28] pairwise constraints work by applying a geometric constraint across correspondences on a pairwise basis, attempting to minimize the pairwise error. This approach has later been adopted by Pang et al. but made scale invariant and more efficient by using a voting scheme [29,30]. Similarly we can cluster matches together based on a geometric constraint and treat each cluster as a separate case [31,32]. An interesting application of this principle is demonstrated by Chen et al. who iteratively use a Hugh transform to cluster matches using angular constraints [33].

While geometric constraints have been shown to work well, they are often susceptible to outliers and tend to be computationally demanding. All of the above geometric methods require a set of initial matches usually provided by *Ratio-Match* which acts as a lower bound on their running time. In practice even fast graph matching methods such as the covering trees method proposed by Yarkony et al. are two or three magnitudes slower than *Ratio-Match* [26].

Fast-Match makes use of a angular assumption to efficiently find new matches in the geometric neighbourhood of already confirmed matches. However this constraint is only locally applied to increase the number of matches, making *Fast-Match* robust to outliers. In addition *Fast-Match* does not rely on an initial set of matches and derives much of its speed from the fact that not even an initial set of local image features is required, making it faster than *Ratio-Match* on most occasions.

This paper is structured as follows: In Section 2 we introduce the *Fast-Match* algorithm. Section 3 outlines the experimental setup. In Section 4 we discuss the results before concluding in Section 5.

2 Matching Fast and Slow

In this chapter we will introduce the fundamentals of *Fast-Match* and motivate the design choices as we go. The algorithm is made of 3 components; finding seeds, finding matches, and exploring for other places where matches might be. After discussing a few design choices in Section 2.1 we discuss each component

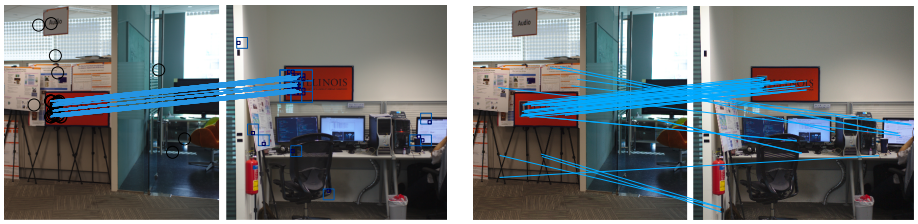


Fig. 1: The 50 best matches by the proposed *Fast-Match* algorithm (left) and the *Ratio-Match* algorithm [1] (right). The lines between the image pairs denote a match as proposed by the algorithm. For *Fast-Match* the zones of the right image where features were computed has been marked with a blue square.

individually. Seed matches are discussed in Section 2.2, collecting matches in Section 2.3 and exploring for new matches in Section 2.4.

2.1 Considerations

If we set out to design a truly fast matching algorithm, we cannot rely just on optimizing the matching step once the descriptors from both images have been found and computed. As illustrated in Figure **??** finding and computing descriptors can easily account for 80% of the time spent matching for bigger images. For this reason *Fast-Match* is designed to only compute features for the part of the image we hope to match.

The *Fast-Match* algorithm is demonstrated in Algorithm 1. Given a *query image* and a *target image* that we intend to match and a confidence threshold τ, we obtain a set of seed matches from the two images. For each seed match we look at the matched position in *query image* and *target image* and find a set of matches. We save these matches and their confidence scores and for those of them that pass the confidence threshold τ we obtain another set of seed matches. In this way we iterate until we have no more seed matches and return the matches and their confidence scores. In an intuitive sense τ serves as a parameter directing the thoroughness of the algorithm, i.e. how much time we spend matching, while the final precision and recall can be adjusted by thresholding the matches on their confidence scores at the end.

We introduce two variations of the algorithm, the a *general* and the *retrieval* approach. The *retrieval* approach assumes that we know one of the images that we intend to match ahead of time and can do some off-line computations. For the *general* approach we make those computations on the fly instead and do not assume that anything can be pre-computed. in In Figure 1 we illustrate the result of matching *Fast-Match* on two images compared with the result of *Ratio-Match* for the same images.

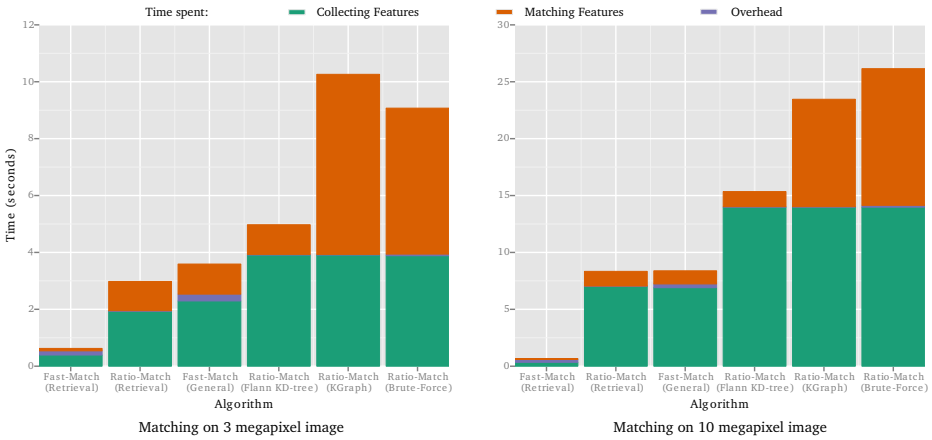


Fig. 2: The duration in seconds of different variations of *Fast-Match* and *Ratio-Match*. The top plot shows results for two 10 MPixel images two 3 Mpixel images is used for the comparison at the bottom. In both plots from left to right: 1. The proposed *Fast-Match* algorithm with one image pre-cached. 2. The proposed *Fast-Match* algorithm with no pre-cached images. 3-5. *Ratio-Match* using respectively Flann KD-Tree, KGraph and Brute-Force to find nearest neighbours

2.2 Initiating Seeds

To obtain a set of initial seed matches several strategies can be used depending on the use case. For the case of matching a pair of two images we can get a rough group of seed matches by matching the thumbnails of the two images with for example *Ratio-Match*. However if we were matching images in a series, such as frames from a movie we could instead make use of a subset of the matches from the last frame to seed *Fast-Match* on the next.

2.3 Collecting Matches and Computing Confidence

If a seed match yields a connection between two points, p_q in the *query image* and p_t in the *target image*, then we are interested in collecting all matches between the regions R_q and R_t centered around p_q and p_t respectively. From each region we can extract a set of feature points between which we try to find a set of matches M_{qt} and a set of confidence scores, C_{qt}.

Lowe and others have shown that the distance between two SIFT descriptors is much less indicative of a true correspondence than the ratio between the best and second best match [1,2,3,16]. This ratio is more formally defined as follows: If we let f_q be a feature in the *query image* and f_t, f_b be the two nearest neighbours

Algorithm 1 Fast-Match

Require: I_{query}, I_{target} : images, maxiter $\in \mathbb{N}$, $\tau \in [0, 1]$
  $M_{seed} \leftarrow$ seed_matches(I_{query}, I_{target})
  $M_{final} \leftarrow \varnothing$
  $C_{final} \leftarrow \varnothing$
  $M_{seen} \leftarrow \varnothing$
  while $M_{seed} \neq \varnothing \wedge i <$maxiter **do**
    $M_{round} \leftarrow$ get_matches(M_{seed})
    $C_{round} \leftarrow$ get_confidence(M_{round})
    $M_{seed} \leftarrow$ get_seeds$(M_{round} \setminus M_{seen}, C_{round}, \tau)$
    $M_{seen} \leftarrow M_{seen} \cup M_{seed}$
    $M_{final} \leftarrow M_{final} \cup M_{round}$
    $C_{final} \leftarrow C_{final} \cup C_{round}$
  end while
  return M_{final}, C_{final}

of f_q in the *target image* then the ratio r is defined as follows:

$$\begin{aligned} r &= \mathrm{r}(f_q, (f_t, f_b)) \\ &= \frac{d(f_q, f_t)}{d(f_q, f_b)}. \end{aligned}$$

Here $d(f_i, f_j)$ is the distance between the features f_i and f_j. For SIFT this is the euclidean distance. Using r as measure for match confidence presumes that we expect a one-to-one matching of the features between the two images. Intuitively if we try to find a match for a feature f_i in an image that does not have any true correspondences, then we would expect the two closest neighbours to be roughly equally well matched with f_i. For this reason we attribute high confidence to matches where the closest neighbour is dramatically closer to f_i than the second closest neighbour and discard the rest.

We are faced with a problem when applying this technique to obtain the set of confidence scores C_{qt} since for any match in M_{qt} we only know the nearest neighbours amongst the features of R_q and R_t. To get around this problem we either assume to know one of the images beforehand (the *retrieval* approach) or in case we cannot make that assumption we compute the features of one of the images (the *general* approach). For a given match between features f_q and f_t we can now find the second closest neighbour neighbour f_b and calculate the confidence as $r = \frac{d(f_q, f_t)}{d(f_q, f_b)}$.

In the case that we know the image beforehand, we can further optimize this step. If we assume the *target image* is known in advance we can approximate the ratio by letting $\hat{r} = \frac{d(f_t, f_q)}{d(f_t, f_b)}$. Here the feature f_b is also part of the target image, which means we can pre-calculate $d(f_t, f_b)$ for all features in the cached *target image* before we start matching.

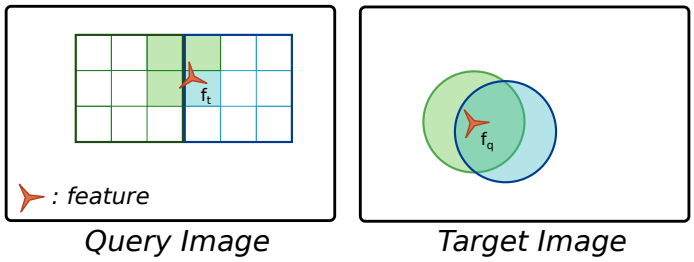


Fig. 3: Exploration of features based on match. The areas shaded blue are the areas that were searched to obtain the match. The areas shaded green are candidate areas for obtaining more matches.

2.4 Exploring for Matches

In each iteration we compute a new set of seed matches, i.e. positions that might yield more matches in the image. For each region R_i evaluated during the collection step we now have a set of matches, M_i and a set of confidences, C_i that we can make use to predict whether the neighbourhood of R_i is worth exploring.

There are many possible heuristics for predicting possible matching regions including using local and global epipolar assumptions and partial graph isomorphisms. However for the sake of simplicity and speed we have chosen a straightforward approach based on weak angular assumptions and illustrated in Figure 3. Assuming that the *target image* has been cached either in advance or before the matching step, the blue shaded areas show R_q and R_t for a given seed match. In the *target image* we collect all features in a given radius while we compute features in the rectangular R_q in the *query image*. For performance reasons we compute all features in the blue square but match only the features inside the shaded area. Next a match is found in the collection step between f_q and f_t. Based on the position of f_q in R_q we select three areas with potential for more matches. The center of each is matched with the center position f_t to produce three seed matches for the next iteration.

In practice it is necessary that these squares overlap in order to detect features lying close to the edges. This incurs a bit of overhead which is minimized by only collecting features for groups of 9 squares. In order to avoid doing double work, quite a bit of care has to be expended making sure that results are properly cached. In Figure 1 the areas across all iterations are indicated to illustrate a typical result.

2.5 Computational Complexity

The difference between the *general* and *retrieval* approach of *Fast-Match* consists in whether we compute the features in the *target image* (general) or if we presume the features are computed offline (retrieval). Computing the features is linear in

the amount of features, while finding $d(f_q, f_b)$ for all n features in the *target image* can be done in $O(nlogn)$ using metric trees.

Once the features and their distances have been computed, the algorithm iteratively finds new seed matches based on previous sets of seed matches. For each seed match we collect new matches, calculate confidence scores and obtain new seed matches. Each of these steps varies only with the local region size which is constant. As a consequence the running time is linear in the amount of possible seed matches.

We will show that the amount of possible seed matches is on average linear in the amount of image features and provide an upper bound for the probability that it is not. We assume that outside of true correspondences, features have better or at least equally good matches in terms of confidence in the image they come from. More rigorously put: For any feature in the target image f_t let f_i be the best matching feature from either image which is not a true correspondence. Let A_{ti} be the event that f_i is found in the *query image*, then $\mathbb{P}(A_{ti}) \leq 0.5$.

For a given feature in the *target image* f_t and its nearest neighbour in the *target image*, f_b, we can let define the stochastic variable X_t as:

$$X_t = |\{\, f_q \,|\, f_q \in F_{query},\, d(f_q, f_t) < d(f_q, f_b)\}| \tag{1}$$

Here F_{query} is the set of features in the *query image*. That is, X_t is the number of features in the query image closer to f_q than f_b. For each feature f_t, X_t is an i.i.dštochastic variable.

In the worst case, each feature in the *target image* has a true correspondence in the *query image* and $\mathbb{P}(A_{ti} = 0.5)$, in which case we find can find $\mathbb{P}(X_t = n) = \mathbb{P}(A_{ti})^n = 0.5^n$, as well as the expected value $\mathbb{E}(X_t) = \sum_{i=1}^{\infty} 0.5^i i = 2$, and the variance $Var(X_t) = \sum_{i=1}^{\infty} 0.5^i (i-2)^2 = 6$.

For $\tau = 1$ we accept a seed match when $\tau < r$, that is, when $d(f_t, f_q) \leq d(f_t, f_b)$. That means that for any feature in the *query image*, we accept at most X_t matches. To find the total amount of seed matches given n, we let $S_n = X_1 + \ldots + X_n$, in which case $\mathbb{E}(S_n) = 2n$ which shows that the expected amount of seed matches is linear in the number of target features.

We can provide an upper bound on the probability that a particular pair of images will exhibit a super-linear behaviour by observing that:

$$\mathbb{P}\left(\left|\frac{S_n}{n} - \mathbb{E}(X_t)\right| \leq f(n)\right) \leq \frac{Var\left(\frac{S_n}{n}\right)}{(f(n))^2} \qquad (chebyshev) \tag{2}$$

$$\leq \frac{6}{n(f(n))^2} \tag{3}$$

For larger values of n it be becomes increasingly likely that *Fast-Match* at worst will show linear performance in terms of the amount of target features.

The noteworthy part of this result is that for the *retrieval* approach we can match two images in $O(n)$. Since there are no large constants involved this makes it possible to rapidly match very large images. In particular *Fast-Match* is suited for cases where we are looking to match several large *query images* to one *target*

image. In this case we can compute the features and distances of the *target image* and then match each *query image* in linear time.

3 Experimental Setup

We evalute *Fast-Match* over 3024 image pairs featuring various 3D objects at different angles. We compare the algorithm to the standard *Ratio-Match*[1] as well as the newer *Mirror-Match*[18]. These two algorithms were selected because they are magnitudes faster than the fastest geometric algorithms while at the same time providing robust matches.

3.1 Evaluation of Fast-Match on 3D Objects

The 3D objects dataset by Moreels and Pietro [3] allows us to experimentally compare matching algorithms over a large range of object and surface types rotated on a turnstile and photographed from every 5 degree turn. 15 objects from the dataset are shown in Figure 4. We use images of 84 different objects under three different lighting conditions at 12 different angle intervals, conducting experiments with a total of 3024 image pairs.

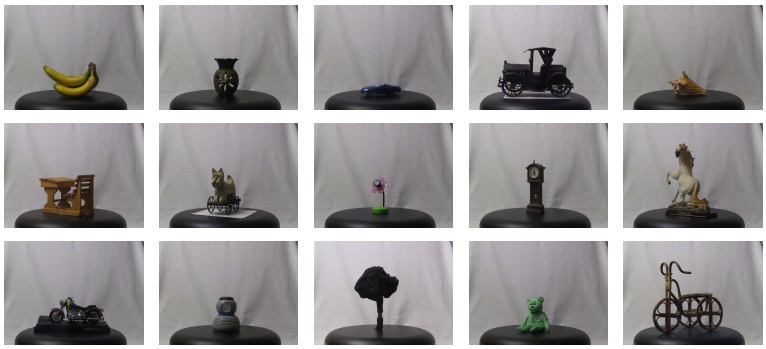
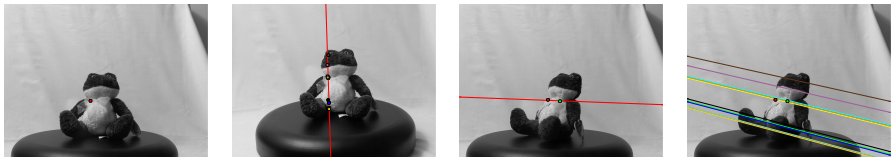


Fig. 4: 15 objects from the 3D Objects dataset by Moreels and Pietro [3].

To validate matches, Moreels and Pietro propose a method using epipolar constraints [3, p.266], which is outlined in Figure 5. According to their experiments, these constraints are able to identify true correspondences with an error rate of 2%. We use their proposed method to generate the ground truth for the evaluation of our framework.

To compute the total number of possible correspondences, we take each feature in a *query image* and count how many of them have a feature in the *target image* which would satisfy the epipolar constraints outlined above. When using this dataset, features with no correspondences were not included in the set of



(a) Query Image (b) Auxiliary Image (c) Target Image (d) Target Image with epipolar constraints

Fig. 5: Creating epipolar constraints based on three source images [3]. (a) *Query image* marked with the position of the feature we are attempting to match. (b) Auxiliary image, taken at the same rotation as the *query image* but from a higher elevation angle. The line going through the image is the epipolar line of the feature point in the *query image*. The markers indicate all feature points in the image found near the epipolar line. In c) we show the *target image* which in this case is rotated 45 degrees from the *query image*. The line overlaid on the image represents the epipolar line corresponding to the feature point shown in the *query image*. The markers indicate all feature points in the image found near the epipolar line. In d) we show the *target image* again, this time overlaid with the epipolar lines corresponding to all the features shown in the *auxiliary image*. A true correspondence should be found within a small distance of one of the intersections of the line in c) and the lines in d). In this particular case both feature points shown in c) and d) could possibly be a correct correspondence.

features for testing, so as to avoid matching non-moving background and foreground objects.

We evaluate all matching algorithms from our framework on the 3D Objects dataset by matching images at different angular intervals. For each object we pick the *query* image as the image taken at 10 degrees rotation for calibration stability. We then match this image with the same object turned an additional Δ degrees, $\Delta \in \{5, 10, \ldots, 60\}$. For every angle interval we compare images taken under 3 different lighting conditions as provided by the dataset. We include all objects in the database for which photos at 5 degree angle intervals are available except for the "Rooster" and "Sponge" objects due to image irregularities.

3.2 Configuration of *Fast-Match*

The central parameters of *Fast-Match* are the confidence threshold, τ for selecting seed matches and the final confidence threshold applied to the total set of matches. For the experiments we have let $\tau = 0.9$ and created precision/recall plots by varying the confidence threshold over the final set of matches.

To achieve a good balance between speed and robustness we let the region in which we extract features be a square with a side length of 90 pixels. On empirical tests we found that anything smaller would result in decreased performance while much bigger Regions would decrease speed. The 90 pixel window is split in to

nine smaller squares. When a seed match falls in any of these squares we match only the features within the smaller square. We let both the regions and the smaller squares overlap each other at all edges with 25 pixels in order to capture feature points lying close to an edge. For the image we have already cached we find all features within a radius of 50 pixels of the seed match.

While these parameters have worked well in practice, we are convinced that there exists other and better ways to implement *Fast-Match* in particular for specific applications. We encourage experimentation and will happily assist with advice.

4 Results

5 Conclusion

References

1. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal on Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
2. K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.
3. P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3D objects," *International Journal on Computer Vision*, vol. 73, no. 3, pp. 263–284, 2007.
4. J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in *Proc. European Conference on Computer Vision (ECCV)*, 2012, pp. 759–773.
5. H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. European Conference on Computer Vision (ECCV)*, 2006, pp. 404–417.
6. M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in *Proc. European Conference on Computer Vision (ECCV)*, 2010, pp. 778–792.
7. S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2548–2555.
8. Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. IEEE, 2004, pp. II–506.
9. J. Li and N. M. Allinson, "A comprehensive review of current local features for computer vision," *Neurocomputing*, vol. 71, no. 10, pp. 1771–1787, 2008.
10. J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997, pp. 1000–1006.
11. D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.
12. M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." in *VISAPP (1)*, 2009, pp. 331–340.

13. W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 577–586.
14. R. Deriche, Z. Zhang, Q.-T. Luong, and O. Faugeras, "Robust recovery of the epipolar geometry for an uncalibrated stereo rig," in *Proc. European Conference on Computer Vision (ECCV)*, 1994, pp. 567–576.
15. A. Baumberg, "Reliable feature matching across widely separated views," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2000, pp. 774–781.
16. J. Rabin, J. Delon, and Y. Gousseau, "A statistical approach to the matching of local features," *SIAM Journal on Imaging Sciences*, vol. 2, no. 3, pp. 931–958, 2009.
17. M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2005, pp. 510–517.
18. J. T. Arnfred, S. Winkler, and S. Süsstrunk, "Mirror Match: Reliable feature point matching without geometric constraints," in *Proc. IAPR Asian Conference on Pattern Recognition (ACPR)*, 2013, pp. 256–260.
19. C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: a randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics-TOG*, vol. 28, no. 3, p. 24, 2009.
20. 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.
21. P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.
22. R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.
23. J. Kim, O. Choi, and I. S. Kweon, "Efficient feature tracking for scene recognition using angular and scale constraints," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008, pp. 4086–4091.
24. C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–535, 1997.
25. L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph matching: Models and global optimization," in *Proc. European Conference on Computer Vision (ECCV)*. Springer, 2008, pp. 596–609.
26. J. Yarkony, C. Fowlkes, and A. Ihler, "Covering trees and lower-bounds on quadratic assignment," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 887–894.
27. M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 492–505.
28. M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, San Diego, CA, 2005, pp. 1482–1489.
29. Y. Yuan, Y. Pang, K. Wang, and M. Shang, "Efficient image matching using weighted voting," *Pattern Recognition Letters*, vol. 33, no. 4, pp. 471–475, 2012.
30. Y. Pang, M. Shang, Y. Yuan, and J. Pan, "Scale invariant image matching using triplewise constraint and weighted voting," *Neurocomputing*, vol. 83, pp. 64–71, 2012.

31. M. Cho, J. Lee, and K. M. Lee, "Feature correspondence and deformable object matching via agglomerative correspondence clustering," in *Proc. International Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 1280–1287.
32. L. Wu, Y. Niu, H. Zhang, H. Zhu, and L. Shen, "Robust feature point matching based on local feature groups (LFGs) and relative spatial configuration," *Journal of Computational Information Systems*, vol. 7, no. 9, pp. 3235–3244, 2011.
33. H.-Y. Chen, Y.-Y. Lin, and B.-Y. Chen, "Robust feature matching with alternate hough and inverted hough transforms," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2762–2769.