FAST-MATCH: FAST AND ROBUST FEATURE MATCHING ON LARGE IMAGES

Jonas T. Arnfred and Stefan Winkler

Advanced Digital Sciences Center (ADSC), University of Illinois at Urbana-Champaign, Singapore

ABSTRACT

Today's cameras produce images that often exceed 10 megapixels. Yet computing and matching local features for images of this size can easily take 20 seconds or more using optimized matching algorithms. This is much too slow for interactive applications and much too expensive for large scale image operations. We introduce *Fast-Match*, an algorithm designed to match large images efficiently without compromising matching accuracy. It derives its speed from only computing features in those parts of the image that can be confidently matched. *Fast-Match* is an order of magnitude faster than the popular *Ratio-Match*, yet often doubles matching precision for difficult image pairs.

1. INTRODUCTION

Whenever we match local image features we are faced with a choice between performance and accuracy. On one hand SIFT features proposed by Lowe [1] have shown again and again to compare favorably to other local image descriptors, especially under unconstrained conditions [2–4]. On the other, SIFT keypoints and descriptors are slow to compute, the main raison d'être for the introduction of various alternative local image features. In many computer vision applications we want to 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.

In this paper we introduce the *Fast-Match* algorithm, designed to match features 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 variants of the algorithm: The *general* variant functions like a traditional matching algorithm and matches two unknown images. The *retrieval* variant on the other hand assumes that we know one of the images we intend to match beforehand; under this assumption, it can be a magnitude faster than existing matching methods without compromising on accuracy.

The problem that *Fast-Match* attempts to solve is twofold. 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 be a source of incorrect correspondences. At the same time, this enables us to improve computational speed by not computing keypoints and descriptors for large parts of the images and at the same time reducing the number of feature points we need to match.

Fast-Match makes use of an angular assumption to efficiently find new matches in the geometric neighborhood of already confirmed matches. However this constraint is only applied *locally* to increase the number of matches, making *Fast-Match* robust to outliers. In addition *Fast-Match* derives much of its speed from the fact that it does not require an initial set of local image features or matches.

2. RELATED WORK

Efforts to reduce the computational costs of finding nearest neighbors to feature points have largely focused on metric trees. Naïvely the set of nearest neighbors between features in two images can be computed by brute force in $O(n^2)$, where n is the total number of feature points in the two images. Lowe proposed using the best-bin-first method to approximately search for nearest neighbors [5, 6]. This reduces the computational complexity to $O(n \log n)$, but even approximate metric trees gain little compared 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 focused on improving approximate nearest neighbor searches by using several KD-Trees simultaneously while optimizing the tree structure using k-means to cluster similar features [7]. Recent work on knn-graphs shows a lot of promise for high-dimensional cases [8]. We later review these improvements and their effect on efficiently matching large images.

Much research has focused on increasing the efficiency of local feature matching. *Fast-Match* builds upon the foundation of *Ratio-Match* as originally introduced by Deriche et al. [9] and Baumberg [10], even though Lowe [1] is usually credited for it. The main idea is using the ratio of the similarity of the best to second best match of a given point to evaluate the uniqueness of the match. Their finding was later 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 neighbors [1–3, 11].

This work is supported by the research grant for ADSC's Human Sixth Sense Programme from Singapore's Agency for Science, Technology and Research (A*STAR).

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*, where matches are chosen from a pool of candidates according to how well they approximate a global epipolar geometry [12,13]. Similar global angular and distance constraints can be used to filter a set of matches [14, 15]. The problem can also be modeled as a graph matching problem where each feature is a vertex, and edge values correspond to a geometric relation between two features [16, 17].

While geometric constraints have been shown to work well, they are often susceptible to outliers and tend to be computationally demanding. Furthermore, 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 are two or three magnitudes slower than *Ratio-Match*.

3. INTRODUCING FAST-MATCH

If we set out to design a truly fast matching algorithm, we cannot rely just on optimizing the matching step. Finding and computing descriptors alone can easily account for 80% of the time spent for bigger images (cf. Fig. 4). For this reason *Fast-Match* is designed to only compute features for the part of the image we hope to match.

Fast-Match consists of 3 components: finding seeds, finding matches, and exploring for other places where matches might be, as outlined in Algorithm 1 (the *Fast-Match* source code is available for download at `https://github.com/arnfred/Fast-Match`).

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 the query and target images and find a set of matches. We save these matches and their confidence scores; for those 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. τ serves as a "thoroughness" parameter, i.e. how much time is spent matching, while the final precision and recall can be adjusted by thresholding the matches on their confidence scores at the end.

We propose a *general* and a *retrieval* variant of the algorithm. The latter assumes that we know one of the images ahead of time and can do some computations off-line. The *general* variant makes those computations on the fly instead.

3.1. Initializing Seeds

Several strategies can be used to obtain a set of initial seed matches. In practice we have found it efficient to resize both images to thumbnails and use *Ratio-Match* to obtain a set of matches and ratios that we then threshold with τ to obtain the initial seed matches. The thumbnail size needs to be chosen such that objects of interest are still detectable by a keypoint detector while remaining small enough to find matches efficiently. We found empirically that – for images larger than one megapixel – a thumbnail size of 300×300 pixels represents a good balance between speed and accuracy, independently of the original image size.

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}

3.2. 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, we are interested in collecting all matches between the regions R_q and R_t centered at p_q and p_t respectively. From each region we can extract a set of feature points between which we look for a set of matches M_{qt} and associated 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 [1–3, 11]. This ratio is more formally defined as:

$$r = \frac{d(f_q, f_t)}{d(f_q, f_b)}, \tag{1}$$

where f_q is a feature in the query image, f_t, f_b are the two nearest neighbors of f_q in the target image, and $d(f_i, f_j)$ is the dissimilarity between features (for SIFT this is the Euclidean distance). The lower r, the higher the confidence in a match. Using this ratio presumes that we expect each feature in the target image to have at most one true correspondence in the target image. 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 neighbors to be roughly equally well matched with f_i. On the other hand we attribute high confidence to matches where the closest neighbor is dramatically closer to f_i than the second closest.

When applying this technique to obtain the set of confidence scores C_{qt}, we are faced with the problem that for any match in M_{qt} we only know the nearest neighbors amongst the features of R_q and R_t. To get around this, we compute the features of one of the images, either offline (the *retrieval* variant) or online (the *general* variant). For a given match between features f_q and f_t we can now find the second closest neighbor f_b and calculate the confidence as per Eq. 1.

In the retrieval scenario we can further optimize this step, because we can approximate r by $d(f_t, f_q)/d(f_t, f_b)$. If we assume the target image is known in advance, this allows us to pre-compute the denominator for all features in the target image before we start matching.

3.3. 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 and confidence levels that we can use to predict whether the neighborhood of R_i is worth exploring.

There are many possible heuristics for predicting promising regions, including local and global epipolar assumptions and partial graph isomorphisms. However, for the sake of simplicity and speed we choose an approach based on weak angular assumptions, as illustrated in Fig. 1. In the target image we collect all features within a given radius R_t, 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. If a match is found between f_q and f_t in the collection step, we select three areas with potential for more matches based on the position of f_q in R_q. The center of each is matched with 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 computing the same matches or features twice, we need to make sure that results are properly cached. A matrix containing 'bins' of features can be used to store features from different image regions. Similarly a hash-set is suitable for keeping track of which regions have already been matched and which matches have already been found.

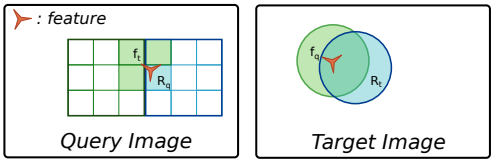

Fig. 1: Exploration of features based on a match. The blue areas were searched to obtain the match. The green areas are candidates for obtaining more matches.

4. EXPERIMENTAL VALIDATION

4.1. 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 our experiments we let $\tau = 0.9$ and create precision/recall plots by varying the confidence threshold over the final set of matches.

To achieve a good balance between speed and robustness we make the region in which we extract features a square of 90×90 pixels in size. We found this size to work well with the SIFT feature descriptor. The window is split into nine smaller squares (cf. Fig. 1). 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 by 25 pixels at all sides in order to capture feature points lying close to an edge. For the target image we find all features within a radius of 50 pixels of the seed match.

4.2. Database

The 3D Objects dataset by Moreels and Perona [3] allows us to experimentally compare matching algorithms over a large range of object and surface types under lighting and perspective changes. Sample images from the dataset are shown in Fig. 2. All photos were taken with a consumer camera in 3.1 megapixel resolution. We use images of 84 different objects photographed under 3 different lighting conditions at 12 angle intervals, conducting experiments with a total of 3024 image pairs.

Fig. 2: Sample images from the 3D Objects dataset [3].

To validate matches, Moreels and Perona proposed a method using epipolar constraints [3, p.266], which we also rely on here to generate the ground truth. To compute the total number of possible true 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.

We evaluate all algorithms by matching images at different angular intervals. For each object we pick the image taken at 10 degrees rotation as the query image. We then match this image with the same object rotated an additional Δ degrees, $\Delta \in \{5, 10, \ldots, 60\}$. For every angle difference we compare images taken under 3 different lighting conditions as provided by the dataset. We calculate the result for all these image

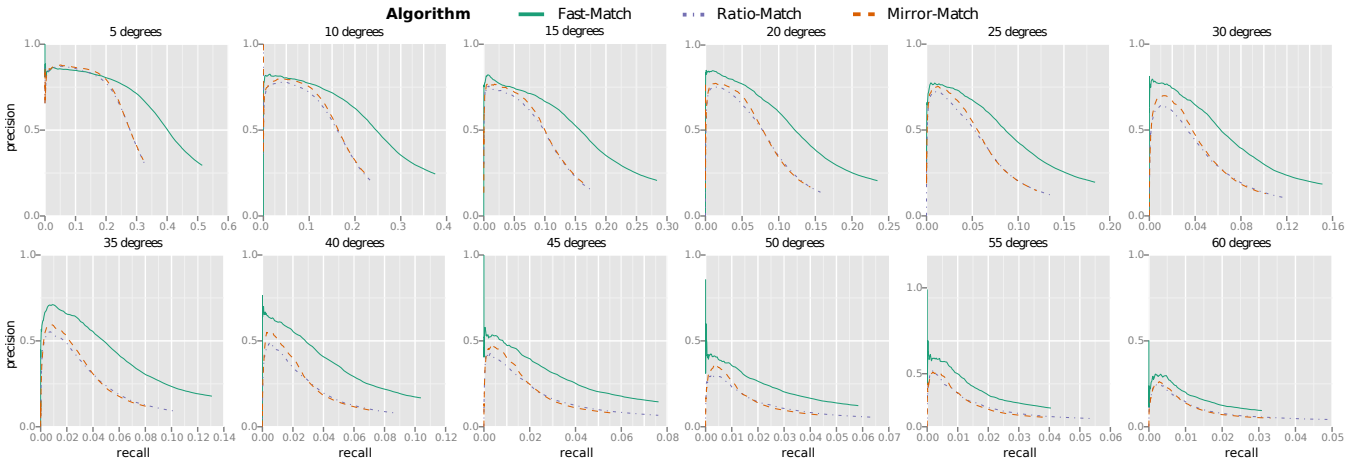

Fig. 3: Results for the 3D Objects dataset. Each plot shows the weighted average precision and recall over 84 objects photographed under 3 different lighting conditions.

pairs using a weighted average based on the number of actual correspondences for each pair, to ensure that each object contributes equally to the final result.

4.3. Results

We compare the algorithm to the standard *Ratio-Match* [1] as well as the newer *Mirror-Match* [18]. We do not include geometric algorithms because even the fastest are magnitudes slower than *Ratio-Match* and as such are too costly when matching features in large images under time constraints.

Fig. 3 shows the performance of the different matching methods in our proposed framework for 12 increasingly bigger angle differences. Each precision-recall plot shows the accumulated results over all 3D objects, weighted by the number of possible true correspondences for the individual objects.

At small angular differences all algorithms show similar performance at low recall, but *Fast-Match* is clearly superior to *Ratio-Match* and *Mirror-Match* at higher recall, more than doubling the precision at similar recall levels. At larger angular differences, the performance gap between *Fast-Match* and the other algorithms extends to low recall as well, although the overall precision of all algorithms declines.

Fig. 4 compares the speed of the two variants of *Fast-Match* to different variants of *Ratio-Match* on a 10-megapixel image, as is typical of photos taken with today's consumer cameras and phones. The retrieval variant of *Fast-Match* takes about one second, whereas a retrieval variant of *Ratio-Match* with precomputed features takes eight seconds. This is roughly equal to the time the general variant of *Fast-Match* spends matching the two images without any precomputations. The speed of nearest neighbor search depends on the specific algorithm, with the FLANN matcher [7] being vastly superior to brute force and KGraph.

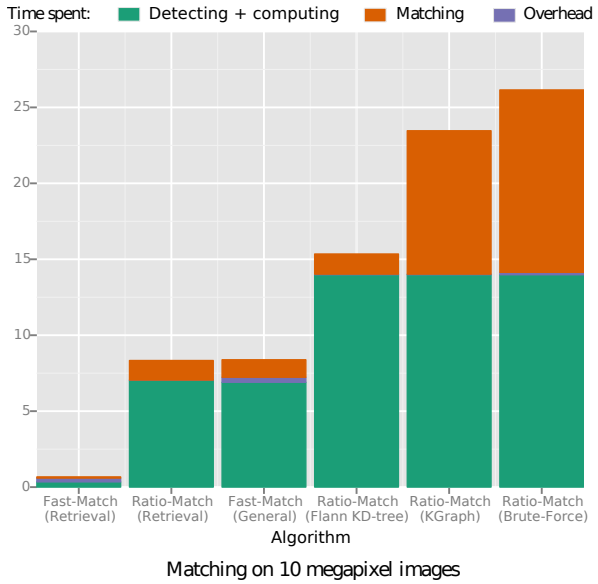

Fig. 4: Computation times of different variants of *Fast-Match* and *Ratio-Match* for a 10 megapixel image pair.

5. CONCLUSION

We have introduced *Fast-Match* in a general and a retrieval variant. We compared *Fast-Match* to *Ratio-Match* and *Mirror-Match* using 3024 image pairs of rotated 3D objects to demonstrate that *Fast-Match* outperforms the other algorithms significantly and in most cases doubles the matching precision at similar recall rates. At the same time *Fast-Match* can be nearly a magnitude faster than *Ratio-Match*. The retrieval variant of *Fast-Match* is particularly effective for matching a single given image to multiple large images. The *Fast-Match* source code is available for download at `https://github.com/arnfred/Fast-Match`.

6. REFERENCES

[1] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] Krystian Mikolajczyk and Cordelia Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[3] Pierre Moreels and Pietro Perona, "Evaluation of features detectors and descriptors based on 3D objects," *International Journal of Computer Vision*, vol. 73, no. 3, pp. 263–284, 2007.

[4] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm, "Comparative evaluation of binary features," in *Proc. European Conference on Computer Vision (ECCV)*, 2012, pp. 759–773.

[5] Jeffrey S. Beis and David G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1997, pp. 1000–1006.

[6] David G. Lowe, "Object recognition from local scale-invariant features," in *Proc. International Conference on Computer Vision (ICCV)*. IEEE, 1999, vol. 2, pp. 1150–1157.

[7] Marius Muja and David G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, 2009, pp. 331–340.

[8] Wei Dong, Charikar Moses, and Kai Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proc. International Conference on World Wide Web (IW3C2)*. ACM, 2011, pp. 577–586.

[9] Rachid Deriche, Zhengyou Zhang, Quang-Tuan Luong, and Olivier Faugeras, "Robust recovery of the epipolar geometry for an uncalibrated stereo rig," in *Proc. European Conference on Computer Vision (ECCV)*, 1994, pp. 567–576.

[10] Adam Baumberg, "Reliable feature matching across widely separated views," in *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2000, vol. 1, pp. 774–781.

[11] Julien Rabin, Julie Delon, and Yann Gousseau, "A statistical approach to the matching of local features," *SIAM Journal on Imaging Sciences*, vol. 2, no. 3, pp. 931–958, 2009.

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

[13] Philip H. S. Torr and Andrew 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.

[14] Jungho Kim, Ouk Choi, and In So Kweon, "Efficient feature tracking for scene recognition using angular and scale constraints," in *Proc. International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008, IEEE, pp. 4086–4091.

[15] Cordelia Schmid and Roger Mohr, "Local grayvalue invariants for image retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–535, 1997.

[16] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother, "Feature correspondence via graph matching: Models and global optimization," in *Proc. European Conference on Computer Vision (ECCV)*, 2008, pp. 596–609.

[17] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee, "Reweighted random walks for graph matching," in *Proc. European Conference on Computer Vision (ECCV)*, 2010, pp. 492–505.

[18] Jonas T. Arnfred, Stefan Winkler, and Sabine Süsstrunk, "Mirror Match: Reliable feature point matching without geometric constraints," in *Proc. Asian Conference on Pattern Recognition (ACPR)*. IAPR, 2013, pp. 256–260.