

# 000                   Fast-Match: Fast and robust feature matching on 001                   large images

004                   Anonymous ACCV 2014 submission

006                   Paper ID \*\*\*

009                   **Abstract.** Both consumer cameras and camera phones produce images  
010                   that often exceed 10 megapixels. Yet computing and matching local fea-  
011                   tures in images of this size can easily take more than twenty seconds  
012                   using fast matching algorithms. This is much too slow for interactive ap-  
013                   plications and much too expensive for large scale image operations. We  
014                   introduce *Fast-Match*, an algorithm designed to swiftly match large im-  
015                   ages without compromising on matching precision or recall. *Fast-Match*  
016                   derives its speed from only computing features in those parts of the image  
017                   that can be confidently matched. We show that *Fast-Match* is an order  
018                   of magnitude faster than *Ratio-Match*, yet often doubles the precision  
019                   on difficult to match image pairs at equal recall. In addition we prove  
020                   that when one image is known in advance, *Fast-Match* can achieve linear  
021                   performance  $O(n)$  in the number of feature points  $n$ .

## 023                   1 Introduction

025                   Whenever we match local image features we are faced with a choice between per-  
026                   formance and precision. On one hand SIFT features proposed by Lowe [1] have  
027                   shown again and again to compare favorably to other local image descriptors,  
028                   especially under unconstrained conditions such as perspective change with non-  
029                   planar scenes [2–4]. On the other, SIFT keypoints and descriptors are slow to  
030                   compute, the main raison d’être for the introduction of various alternative local  
031                   image features. In many applications of computer vision we would like the in-  
032                   crease the computational performance in order to work on larger images, bigger  
033                   image sets, at faster frame rates or with more limited hardware, but we cannot  
034                   give up the additional precision that SIFT affords us over other local features  
035                   without negative effects for our application.

036                   In this paper we introduce a matching algorithm designed to match features  
037                   between two images only in image areas that are likely to correspond. This  
038                   approach is much faster than traditional methods because there is no need to  
039                   compute descriptors for areas in the image that are not matched. We provide  
040                   two variations of the algorithm: The *general* variation functions as a traditional  
041                   matching algorithm and matches two unknown images albeit faster and more  
042                   robustly than conventional methods. The *retrieval* variation on the other hand  
043                   assumes that we know one of the images we intend to match beforehand; under  
044                   this assumption, it can be a magnitude faster than existing matching methods

2 ACCV-14 submission ID \*\*\*

045 without compromising on accuracy. We will unimaginatively refer to the proposed algorithm as *Fast-Match* in this paper and make clear from the context  
 046 whether this refers to the *retrieval* or *general* variation.  
 047

048 The problem that *Fast-Match* attempts to solve is two-fold. By matching only  
 049 image areas that are likely to correspond we hope to improve accuracy by entirely  
 050 ignoring parts of the images that would otherwise likely be a source of incorrect  
 051 correspondences. At the same time, this enables us to improve computational  
 052 speed by not having to compute keypoints and descriptors for large parts of the  
 053 images and at the same time reducing the amount of feature points we need to  
 054 match. Both problems have been addressed in part by past work.

055 *Fast-Match* makes use of an angular assumption to efficiently find new matches  
 056 in the geometric neighbourhood of already confirmed matches. However this con-  
 057 straint is only applied *locally* to increase the number of matches, making *Fast-*  
 058 *Match* robust to outliers. In addition *Fast-Match* does not rely on an initial set  
 059 of matches and derives much of its speed from the fact that not even an initial  
 060 set of local image features is required.

061 This paper is structured as follows: Section 2 discuss related work. Section 3  
 062 introduces the *Fast-Match* algorithm. Section 4 outlines the experimental setup.  
 063 In Section 5 we discuss the results before concluding in Section 6.

## 065 2 Related work

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

074 Efforts to reduce the computational costs of finding nearest neighbours to  
 075 feature points have largely focused on metric trees. Naïvely the set of near-  
 076 est neighbours between features in two images can computed by brute force in  
 077  $O(n^2)$ , where  $n$  is the total number of feature points in the two images (a typical  
 078 three megapixel image may contain anywhere from 500 to 5000 feature points).  
 079 When SIFT was originally published Lowe, proposed using the Best-Bin-First  
 080 method to approximately search for nearest neighbours [10, 11]. This reduces  
 081 the computational complexity to  $O(n \log n)$ , but even approximate metric trees  
 082 are hard pressed to compete with brute force due to the high dimensionality  
 083 of SIFT and the constant costs incurred with constructing and searching in a  
 084 metric tree. Later work by Muja and Lowe [12] focused on improving approxi-  
 085 mate nearest neighbour searches by using several KD-Trees simultaneously while  
 086 optimizing the tree structure using K-Means to cluster similar features. Recent  
 087 work on knn-graphs shows a lot of promise for high dimensional cases [13]. We  
 088 later review these improvements and their effect on efficiently matching large  
 089 scale images.

090 A wealth of matching methods have focused instead on increasing the efficiency of local feature matching. *Fast-Match* builds upon the foundation of  
 091 *Ratio-Match* introduced originally by Deriche et al. [14] and Baumberg [15] even  
 092 though Lowe is usually credited for introducing it [1]. They both propose using  
 093 the ratio of the similarity of the best to second best match of a given point to evaluate  
 094 how unique it is. Their finding has later been tested by several independent  
 095 teams, all concluding that thresholding based on this ratio is generally superior  
 096 to thresholding based on similarity or returning all nearest neighbours [1–3,16].  
 097

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

106 Another inspiration for the design of *Fast-Match* is *Patch-Match* as introduced  
 107 by Barnes et al. [19]. Like *Fast-Match*, *Patch-Match* is an iterative algorithm  
 108 for fast image matching, but unlike *Fast-Match*, which uses sparse local  
 109 image features, *Patch-Match* is designed for dense image features. It works by  
 110 randomly creating a set of matches from both images and then iteratively im-  
 111 proving it.

112 For sparse local image features many solutions have combined *Ratio-Match*  
 113 with various geometric constraints to improve matching. These constraints are  
 114 based on assumptions regarding the transformation between the *query* and *tar-*  
*115 get images*. A commonly used example is *RANSAC* applied to feature matching  
 116 where matches are chosen from a pool of candidates according to how well they  
 117 approximate a global epipolar geometry [20–22]. Similar global angular and dis-  
 118 tance constraints can be used to filter a set of matches as shown in [23,24]. Finally  
 119 the problem of feature matching can be modelled as a graph matching problem  
 120 where each feature is a vertex, and edge values correspond to a geometric re-  
 121 lation between two features as demonstrated in [25–27]. For cases that require  
 122 more flexibility such as scenes with moving elements and non-planar perspective  
 123 change Pairwise constraints provide an alternative to the more rigid global con-  
 124 straints. Introduced by Leordeanu and Herbert [28], pairwise constraints work by  
 125 applying a geometric constraint across matches on a pairwise basis, attempting  
 126 to minimize the pairwise error. This approach has later been adopted by Pang et  
 127 al. but made scale invariant and more efficient by using a voting scheme [29,30].  
 128 Similarly we can cluster matches together based on a geometric constraint and  
 129 treat each cluster as a separate case [31,32]. An interesting application of this  
 130 principle is demonstrated by Chen et al. who iteratively use a Hough transform  
 131 to cluster matches using angular constraints [33].

132 While geometric constraints have been shown to work well, they are often  
 133 susceptible to outliers and tend to be computationally demanding. All of the  
 134 above geometric methods require a set of initial matches usually provided by

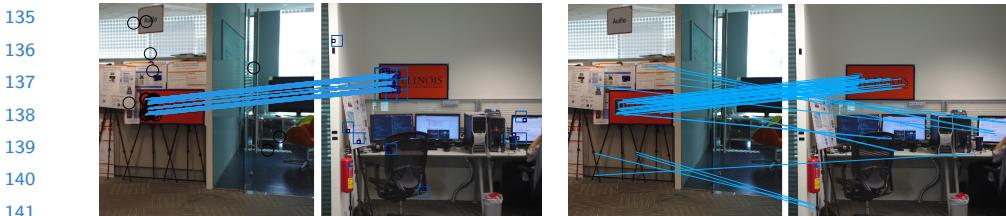


Fig. 1: The 50 best matches by the proposed *Fast-Match* algorithm (left) and the *Ratio-Match* algorithm [1] (right). The lines between the image pairs denote a match as proposed by the algorithm. For *Fast-Match* the zones of the right image where local features were computed has been marked with a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

*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].

### 3 Matching Fast and Slow

In this section we will introduce the fundamentals of *Fast-Match* and motivate the design choices as we go. The algorithm consists of 3 components; finding seeds, finding matches, and exploring for other places where matches might be.

#### 3.1 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. Finding and computing descriptors can easily account for 80% of the time spent matching for bigger images (cf. Figure 5). 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  $\tau$ , 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  $\tau$ , 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  $\tau$  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 a *general* and a *retrieval* variation of the algorithm. The latter assumes that we know one of the images that we intend to match ahead of time and

180 can do some off-line computations. For the *general* variation we make those  
 181 computations on the fly instead and do not assume anything to be pre-computed.  
 182

---

**184 Algorithm 1** Fast-Match
 

---

185 **Require:**  $I_{query}, I_{target}$  : images, maxiter  $\in \mathbb{N}$ ,  $\tau \in [0, 1]$   
 186  $M_{seed} \leftarrow \text{seed\_matches}(I_{query}, I_{target})$   
 187  $M_{final} \leftarrow \emptyset$   
 188  $C_{final} \leftarrow \emptyset$   
 189  $M_{seen} \leftarrow \emptyset$   
 190 **while**  $M_{seed} \neq \emptyset \wedge i < \text{maxiter}$  **do**  
 191      $M_{round} \leftarrow \text{get\_matches}(M_{seed})$   
 192      $C_{round} \leftarrow \text{get\_confidence}(M_{round})$   
 193      $M_{seed} \leftarrow \text{get\_seeds}(M_{round} \setminus M_{seen}, C_{round}, \tau)$   
 194      $M_{seen} \leftarrow M_{seen} \cup M_{seed}$   
 195      $M_{final} \leftarrow M_{final} \cup M_{round}$   
 196      $C_{final} \leftarrow C_{final} \cup C_{round}$   
 197 **end while**  
 198 **return**  $M_{final}, C_{final}$

---

200  
 201 **3.2 Initiating Seeds**

203  
 204 Depending on the scenario, several strategies can be used to obtain a set of  
 205 initial seed matches. For the case of matching a pair of two images we can get a  
 206 rough group of seed matches by matching the thumbnails of the two images with  
 207 for example *Ratio-Match*. However if we were matching images in a series, such  
 208 as frames from a movie we could instead make use of a subset of the matches  
 209 from the last frame to seed *Fast-Match* on the next. In practice we have found  
 210 it efficient to resize both images to thumbnails of  $300 \times 300$  pixels and use ratio  
 211 match to obtain a set of matches and ratios that we then threshold with the  
 212 confidence threshold  $\tau$  to obtain the initial seed matches.  
 213

214 **3.3 Collecting Matches and Computing Confidence**

215  
 216 If a seed match yields a connection between two points  $p_q$  in the *query image* and  
 217  $p_t$  in the *target image*, then we are interested in collecting all matches between  
 218 the regions  $R_q$  and  $R_t$  centered around  $p_q$  and  $p_t$  respectively. From each region  
 219 we can extract a set of feature points between which we try to find a set of  
 220 matches  $M_{qt}$  and a set of confidence scores  $C_{qt}$ .

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

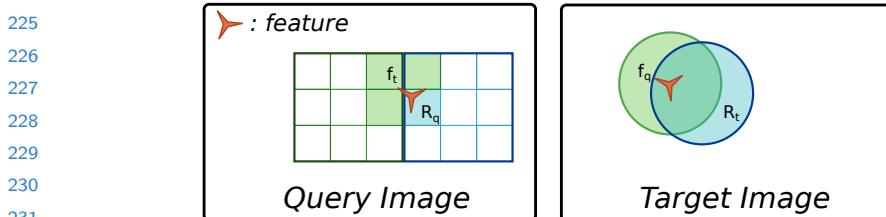


Fig. 2: Exploration of features based on match. The areas shaded blue are the areas that were searched to obtain the match. The areas shaded green are candidate areas for obtaining more matches.

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

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

Here  $d(f_i, f_j)$  is the distance between the features  $f_i$  and  $f_j$ . For SIFT this is the Euclidean distance. The distance between two feature points is greater when the feature points resemble each other less. For this reason we have higher confidence in a match with a low  $r$ -value. Using  $r$  as a measure of match confidence 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 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* variation) or – in case we cannot make that assumption – we compute the features of one of the images (the *general* variation). 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  $r = d(f_q, f_t)/d(f_q, f_b)$ .

In the retrieval scenario where we know an 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  $\hat{r} = d(f_t, f_q)/d(f_t, f_b)$ . 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. When these distances are tied to their corresponding feature in the target image, it becomes trivial to calculate  $\hat{r}$ .

270    **3.4 Exploring for Matches**

270

271  
 272    In each iteration we compute a new set of seed matches, i.e. positions that  
 273    might yield more matches in the image. For each region  $R_i$  evaluated during the  
 274    collection step we now have a set of matches  $M_i$  and a set of confidence levels  $C_i$   
 275    that we can use to predict whether the neighbourhood of  $R_i$  is worth exploring.

276    There are many possible heuristics for predicting possible regions including  
 277    local and global epipolar assumptions and partial graph isomorphisms. However,  
 278    for the sake of simplicity and speed we have chosen a straightforward approach  
 279    based on weak angular assumptions, as illustrated in Figure 2. Assuming that the  
 280    *target image* has been cached either in advance or before the matching step, the  
 281    blue shaded areas show  $R_q$  and  $R_t$  for a given seed match. In the *target image* we  
 282    collect all features in a given radius while we compute features in the rectangular  
 283     $R_q$  in the *query image*. For performance reasons we compute all features in the  
 284    blue square but match only the features inside the shaded area. Next a match  
 285    is found in the collection step between  $f_q$  and  $f_t$ . Based on the position of  $f_q$  in  
 286     $R_q$  we select three areas with potential for more matches. The center of each is  
 287    matched with the center position  $f_t$  to produce three seed matches for the next  
 288    iteration. In Figure 1 we illustrate this process in the *Fast-Match* image pair.  
 289    The *query image* to the right in the figure has a small blue square for each region  
 290    that has been used while matching the two images.

291    In practice it is necessary that these squares overlap in order to detect features  
 292    lying close to the edges. This incurs a bit of overhead which is minimized by  
 293    only collecting features for groups of 9 squares. In order to avoid computing the  
 294    same matches or features twice, quite a bit of care has to be expended making  
 295    sure that results are properly cached. A matrix containing ‘bins’ of features can  
 296    conveniently be used to store features from different image regions. Similarly a  
 297    hash-set is suitable for keeping track of which regions have already been matched  
 298    and which matches have already been found.

300    **3.5 Computational Complexity**

300

301  
 302    The difference between the *general* and *retrieval* variation of *Fast-Match* consists  
 303    in whether we compute the features in the *target image* (*general*) or if we presume  
 304    the features are computed offline (*retrieval*). Computing the features is linear in  
 305    the number of feature points, while finding  $d(f_q, f_b)$  for all  $n$  features in the  
 306    *target image* can be done in  $O(n \log n)$  using metric trees.

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

313    We will show that the number of possible seed matches is on average linear  
 314    in the number of image features, and we also provide an upper bound for the

315 probability that it is not. We assume that outside of true correspondences, fea-  
 316 tures have better or at least equally good matches in terms of confidence in the  
 317 image they come from. More rigorously put: For any feature in the target image  
 318  $f_t$  let  $f_1, f_2, \dots$  be the best, second best, etc matching feature from either image  
 319 which is not a true correspondence. Let  $A_{ti}$  be the event that  $f_i$  is found in the  
 320 *query image*, then  $\mathbb{P}(A_{ti}) \leq 0.5$  and  $A_{ti}$  is independent from  $A_{tj}$  when  $i \neq j$ .

321 For a given feature in the *target image*  $f_t$ , its nearest neighbour in the *target*  
 322 *image*  $f_b$ , and the set of features in the *query image*  $F_{query}$ , we can define the  
 323 stochastic variable  $X_t$  as follows:

$$324 X_t = |\{f_q \mid f_q \in F_{query}, d(f_q, f_t) < d(f_q, f_b)\}| \quad (1)$$

325 That is,  $X_t$  is the number of features in the query image closer to  $f_q$  than  $f_b$ .  
 326 For each feature  $f_t$ ,  $X_t$  is an i.i.d. stochastic variable.

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

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

336 For the above case it is theoretically possible that for every one of the  $n$   
 337 features we end up considering  $kn$  seed matches for some constant  $k$ , i.e. a  
 338 quadratic behaviour. Using Chebyshev's inequality we can provide an upper  
 339 bound on the probability of this event:

$$341 \mathbb{P} \left( \left| \frac{S_n}{n} - \mathbb{E}(X_t) \right| \geq kn \right) \leq \frac{Var \left( \frac{S_n}{n} \right)}{kn^2} = \frac{6}{k^2 n^3} \quad (2)$$

342 It is clear that for any constant  $k$  we can pick a number of features  $n$  which  
 343 render the likelihood of a quadratic behaviour infinitesimal. In essence the larger  
 344  $n$  becomes, the less likely *Fast-Match* is to exhibit super linear behaviour.

345 The noteworthy part of this result is that for the *retrieval* variation we can  
 346 match two images in  $O(n)$ . Since there are no large constants involved this makes  
 347 it possible to rapidly match very large images. In particular *Fast-Match* is suited  
 348 for cases where we are looking to match several large *query images* to one *target*  
 349 *image*. In this case we can compute the features and distances of the *target*  
 350 *image* and then match each *query image* in linear time. In contrast the general  
 351 variation still has a complexity of  $O(n \log n)$  due to the necessity of calculating  
 352 distances online.

## 353 4 Experimental Setup

354 We evaluate *Fast-Match* over 3024 image pairs featuring various 3D objects at  
 355 different angles to measure precision and recall, as well as two pairs of images

360 with high pixel counts to measure performance in terms of speed. We compare  
 361 the algorithm to the standard *Ratio-Match* [1] as well as the newer *Mirror-*  
 362 *Match* [18]. These two algorithms were selected because they are magnitudes  
 363 faster than the fastest geometric algorithms while at the same time providing  
 364 robust matches.

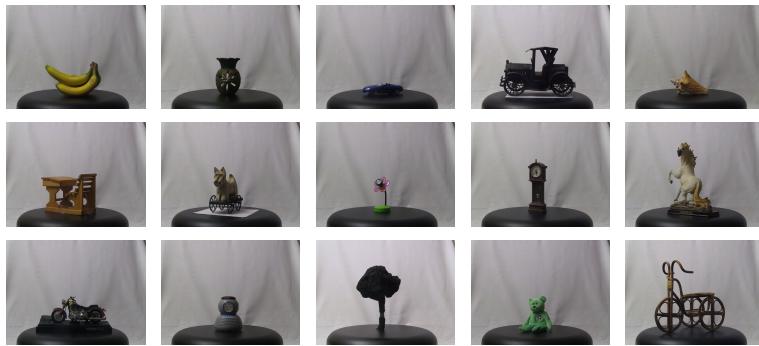
365

#### 366 4.1 Evaluation of Fast-Match on 3D Objects

367

368 The 3D Objects dataset by Moreels and Perona [3] allows us to experimentally  
 369 compare matching algorithms over a large range of object and surface types  
 370 rotated on a turnstile and photographed from every angle in increments of 5  
 371 degrees. 15 objects from the dataset are shown in Figure 3. We use images of  
 372 84 different objects photographed under three different lighting conditions at  
 373 12 different angle intervals, conducting experiments with a total of 3024 image  
 374 pairs. All photos a taken with a consumer camera in 3.1 mega pixel resolution.

375



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

390

391 To validate matches, Moreels and Perona propose a method using epipolar  
 392 constraints [3, p.266]. According to their experiments, these constraints are able  
 393 to identify true correspondences with an error rate of 2%. We use their proposed  
 394 method to generate the ground truth for the evaluation of our framework.

395

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. When using this dataset,  
 features with no correspondences were not included in the set of features for  
 testing, so as to avoid matching non-moving background and foreground objects.

400

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  $\Delta$   
 degrees,  $\Delta \in \{5, 10, \dots, 60\}$ . For every angle interval we compare images taken

401

402

403

404

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405 under 3 different lighting conditions as provided by the dataset. We include all  
 406 objects in the database for which photos at 5 degree angle intervals are available  
 407 except for the “Rooster” and “Sponge” objects due to image irregularities.  
 408

#### 409 4.2 Configuration of *Fast-Match*

410  
 411 The central parameters of *Fast-Match* are the confidence threshold  $\tau$  for selecting  
 412 seed matches and the final confidence threshold applied to the total set of  
 413 matches. For the experiments on the 3D Objects we let  $\tau = 0.9$  and created  
 414 precision/recall plots by varying the confidence threshold over the final set of  
 415 matches.

416 To achieve a good balance between speed and robustness we let the region in  
 417 which we extract features be a square with a side length of 90 pixels. On empirical  
 418 tests we found that anything smaller would result in decreased performance while  
 419 much bigger regions would decrease speed. The 90 pixel window is split in to  
 420 nine smaller squares (see also Figure 2). When a seed match falls in any of these  
 421 squares we match only the features within the smaller square. We let both the  
 422 regions and the smaller squares overlap each other at all edges with 25 pixels in  
 423 order to capture feature points lying close to an edge. For the image we have  
 424 already cached we find all features within a radius of 50 pixels of the seed match.  
 425

## 426 5 Results

427  
 428 Figure 4 shows the performance of the different matching methods in our pro-  
 429 posed framework for 12 increasingly bigger angle differences. The results are  
 430 shown in a precision-recall plot to make it easy to compare performance in terms  
 431 of precision at similar levels of recall. For each plot we show the accumulated  
 432 results on all 3D objects, weighted by the number of possible true correspondences  
 433 for the individual objects. This ensures that each object contributes equally to  
 434 the final result, despite some objects resulting in disproportionately more matches  
 435 than others.

436 At low angular differences all algorithms show similar performance at low re-  
 437 call, however *Fast-Match* is superior to *Ratio-Match* and *Mirror-Match* at higher  
 438 recall, more than doubling the precision at similar recall levels. This picture re-  
 439 mains the same at higher angular differences, but with a higher performance  
 440 gap between *Fast-Match* and the other algorithms at low recall while more than  
 441 doubling precision at higher recalls. For image pairs with angular differences of  
 442 more than 40° the performance of all algorithms declines, with *Fast-Match* still  
 443 coming out on top.

444 The high recall rates of *Fast-Match* are partially explained by the fact that  
 445 we can allow our confidence thresholds to be more lenient when we already  
 446 know that we are likely to find true correspondences in the regions that we are  
 447 matching. However, lenient thresholds can easily impact the matching speed,  
 448 so in practical situations we usually have to choose between matching fast and  
 449 precisely, or more slowly with higher recall.

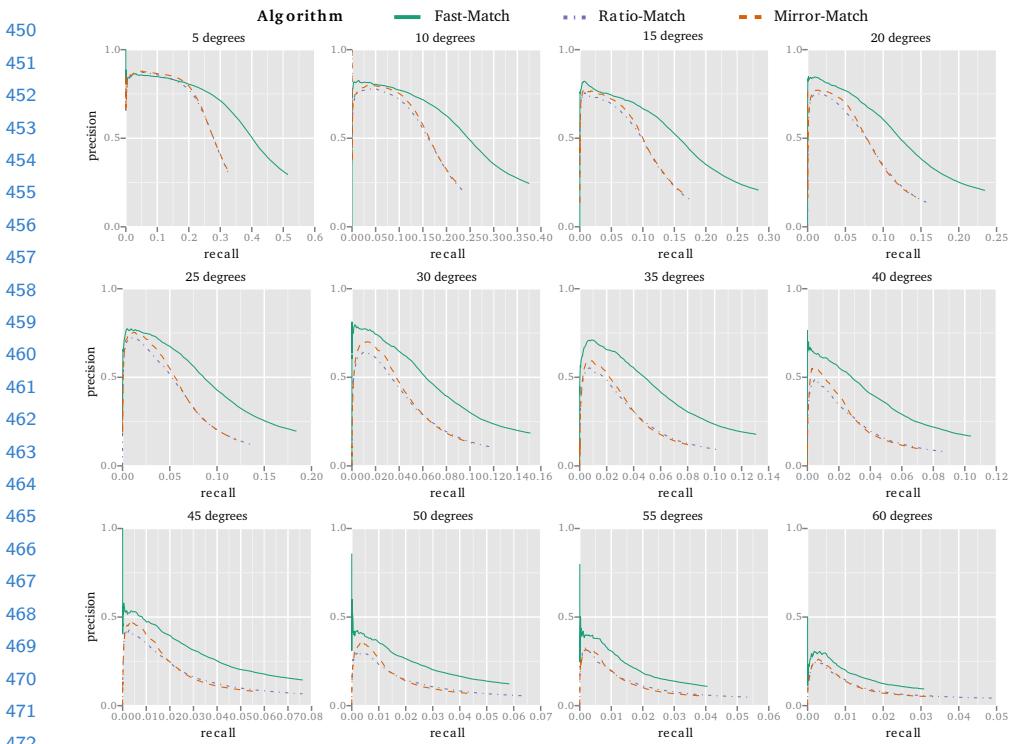


Fig. 4: Results for the 3D Objects dataset. Each plot contains data accumulated from 84 objects photographed under 3 different lighting conditions.

Figure 5 demonstrate the speed of *Fast-Match* on image sizes as is typically produced by modern consumer cameras and camera phones. In Figure 5 we compare the speed of the two variations of *Fast-Match* to different variations of *Ratio-Match* on two image pairs, one with two images of 10 megapixel and another where the same image pair has been scaled to 3 megapixels. For the larger images we see the clearest difference in performance. The retrieval variant of *Fast-Match* matches the image pairs in around one second while a retrieval variant of *Ratio-Match* with precomputed features spends eight seconds on the same task. This result is roughly equal to the time the general variation of *Fast-Match* spends matching the two images without any pre-computations. For the nearest neighbor search we note that for large images there are substantial speed gains to be had by choosing the right algorithm for finding nearest neighbors, with Muja et al.'s Flann matcher being vastly superior to brute force and KGraph.

For the 3 megapixel picture the pattern repeats itself, although with smaller margins separating *Fast-Match* and *Ratio-Match*. When images are smaller, computations like obtaining the initial seed matches are comparatively more costly.

Figure 1 shows the result from matching the pair of three megapixel images using *Fast-Match* and *Ratio-Match*. *Fast-Match* ends up only matching the part

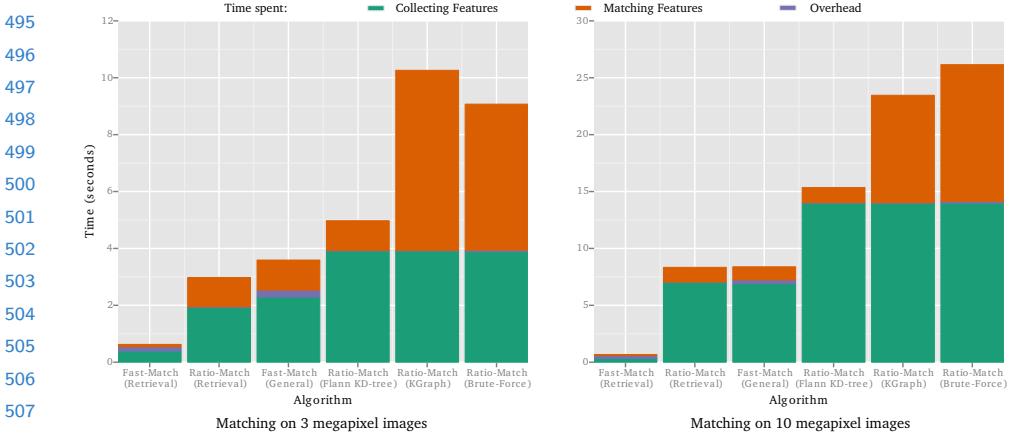


Fig. 5: The duration in seconds of different variations of *Fast-Match* and *Ratio-Match*. The top plot shows results for two 10 MPixel images two 3 Mpixel images is used for the comparison at the bottom. From the left, the first two results presume a precached image while the rest do not require knowing any inputs beforehand. The image pairs used be seen in Figure 1

of the images that are likely to have correspondences. This highlights both a strength and a weakness of *Fast-Match*. It is this myopic matching approach that is responsible for *Fast-Match*' speed and precision; on the other hand, *Fast-Match* is not the best candidate for images that are almost identical, because its step by step approach incurs too much overhead to compete in speed with *Ratio-Match* in those cases.

## 6 Conclusion

We have introduced *Fast-Match* in two forms, a general and a retrieval variation. The retrieval variation assumes that we know one of the images that we are matching ahead of time in order to pre-cache feature points. The general variation has no assumptions and performs the pre-caching of feature points online instead. We have proven that while the general variation has a computational complexity of  $O(n \log n)$  with  $n$  being the number of feature points, the retrieval variation will on average run in  $O(n)$ . From experiments on large images we have shown that *Fast-Match* can be a magnitude faster than *Ratio-Match* while providing comparable results. We further compared *Fast-Match* to *Ratio-Match* and *Mirror-Match* using images of rotated 3D objects to demonstrate that *Fast-Match* outperforms the other algorithms significantly over 3024 image pairs and in most cases doubles the matching precision at similar recall rates. *Fast-Match* is particularly applicable in cases where we are interested in matching one image to multiple large images, in which case we can quickly pre-cache the image and use this data for each of the other images.

## 540 References

- 541
- 542 1. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal on Computer Vision **60** (2004) 91–110
  - 543 2. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Analysis and Machine Intelligence **27** (2005) 1615–1630
  - 544 3. Moreels, P., Perona, P.: Evaluation of features detectors and descriptors based on 3D objects. International Journal on Computer Vision **73** (2007) 263–284
  - 545 4. Heinly, J., Dunn, E., Frahm, J.M.: Comparative evaluation of binary features. In: Proc. European Conference on Computer Vision (ECCV). (2012) 759–773
  - 546 5. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: Proc. European Conference on Computer Vision (ECCV). (2006) 404–417
  - 547 6. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: binary robust independent elementary features. In: Proc. European Conference on Computer Vision (ECCV). (2010) 778–792
  - 548 7. Leutenegger, S., Chli, M., Siegwart, R.Y.: BRISK: Binary robust invariant scalable keypoints. In: Proc. International Conference on Computer Vision (ICCV), IEEE (2011) 2548–2555
  - 549 8. Ke, Y., Sukthankar, R.: PCA-SIFT: A more distinctive representation for local image descriptors. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). Volume 2., IEEE (2004) II–506
  - 550 9. Li, J., Allinson, N.M.: A comprehensive review of current local features for computer vision. Neurocomputing **71** (2008) 1771–1787
  - 551 10. Beis, J.S., Lowe, D.G.: Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (1997) 1000–1006
  - 552 11. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proc. International Conference on Computer Vision (ICCV). Volume 2., Ieee (1999) 1150–1157
  - 553 12. Muja, M., Lowe, D.G.: 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) 331–340
  - 554 13. Dong, W., Moses, C., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proc. International Conference on World Wide Web (IW3C2), ACM (2011) 577–586
  - 555 14. Deriche, R., Zhang, Z., Luong, Q.T., Faugeras, O.: Robust recovery of the epipolar geometry for an uncalibrated stereo rig. In: Proc. European Conference on Computer Vision (ECCV). (1994) 567–576
  - 556 15. Baumberg, A.: Reliable feature matching across widely separated views. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1. (2000) 774–781
  - 557 16. Rabin, J., Delon, J., Gousseau, Y.: A statistical approach to the matching of local features. SIAM Journal on Imaging Sciences **2** (2009) 931–958
  - 558 17. Brown, M., Szeliski, R., Winder, S.: Multi-image matching using multi-scale oriented patches. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1. (2005) 510–517
  - 559 18. Arnfred, J.T., Winkler, S., Süsstrunk, S.: Mirror Match: Reliable feature point matching without geometric constraints. In: Proc. IAPR Asian Conference on Pattern Recognition (ACPR). (2013) 256–260

CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.

14 ACCV-14 submission ID \*\*\*

- 585 19. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: PatchMatch: a random-  
586 ized correspondence algorithm for structural image editing. ACM Transactions on  
587 Graphics-TOG **28** (2009) 24
- 588 20. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model  
589 fitting with applications to image analysis and automated cartography. Communications  
590 of the ACM **24** (1981) 381–395
- 591 21. Torr, P.H.S., Zisserman, A.: MLESAC: A new robust estimator with application  
592 to estimating image geometry. Computer Vision and Image Understanding **78**  
(2000) 138–156
- 593 22. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer (2010)
- 594 23. Kim, J., Choi, O., Kweon, I.S.: Efficient feature tracking for scene recognition  
595 using angular and scale constraints. In: Proc. IEEE/RSJ International Conference  
596 on Intelligent Robots and Systems (IROS), Nice, France (2008) 4086–4091
- 597 24. Schmid, C., Mohr, R.: Local grayvalue invariants for image retrieval. IEEE Trans-  
598 actions on Pattern Analysis and Machine Intelligence **19** (1997) 530–535
- 599 25. Torresani, L., Kolmogorov, V., Rother, C.: Feature correspondence via graph  
600 matching: Models and global optimization. In: Proc. European Conference on  
Computer Vision (ECCV). Springer (2008) 596–609
- 601 26. Yarkony, J., Fowlkes, C., Ihler, A.: Covering trees and lower-bounds on quadratic  
602 assignment. In: Proc. Conference on Computer Vision and Pattern Recognition  
603 (CVPR), IEEE (2010) 887–894
- 604 27. Cho, M., Lee, J., Lee, K.M.: Reweighted random walks for graph matching. In:  
605 Proc. European Conference on Computer Vision (ECCV). Springer (2010) 492–505
- 606 28. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems  
607 using pairwise constraints. In: Proc. Conference on Computer Vision and Pattern  
Recognition (CVPR). Volume 2., San Diego, CA (2005) 1482–1489
- 608 29. Yuan, Y., Pang, Y., Wang, K., Shang, M.: Efficient image matching using weighted  
609 voting. Pattern Recognition Letters **33** (2012) 471–475
- 610 30. Pang, Y., Shang, M., Yuan, Y., Pan, J.: Scale invariant image matching using  
611 triplewise constraint and weighted voting. Neurocomputing **83** (2012) 64–71
- 612 31. Cho, M., Lee, J., Lee, K.M.: Feature correspondence and deformable object match-  
613 ing via agglomerative correspondence clustering. In: Proc. International Conference  
614 on Computer Vision (ICCV), IEEE (2009) 1280–1287
- 615 32. Wu, L., Niu, Y., Zhang, H., Zhu, H., Shen, L.: Robust feature point matching  
616 based on local feature groups (LFGs) and relative spatial configuration. Journal  
617 of Computational Information Systems **7** (2011) 3235–3244
- 618 33. Chen, H.Y., Lin, Y.Y., Chen, B.Y.: Robust feature matching with alternate hough  
619 and inverted hough transforms. In: Proc. Conference on Computer Vision and  
Pattern Recognition (CVPR), IEEE (2013) 2762–2769
- 620  
621  
622  
623  
624  
625  
626  
627  
628  
629