

000
001
002
003
004
005
006
007
008
009
010
011

Abstract

Both consumer cameras and camera phones produce images that often exceed 10 megapixels. Yet computing and matching local features in images of this size can easily take more than twenty seconds using fast 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 swiftly match large images without compromising on matching precision or recall. Fast-Match derives its speed from only computing features in those parts of the image that can be confidently matched. We show that Fast-Match is an order of magnitude faster than Ratio-Match, yet often doubles the precision on difficult to match image pairs at equal recall. In addition we prove that when one image is known in advance, Fast-Match can achieve linear performance $O(n)$ in the number of feature points n .

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 unconstrained conditions such as perspective change with non-planar scenes [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 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: The *general* variation functions as a traditional matching algorithm and matches two unknown images albeit faster and more robustly than conventional methods. The *retrieval* variation 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. 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.

The problem that *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.

Fast-Match makes use of an angular assumption to efficiently find new matches in the geometric neighbourhood 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* 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.

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

2. Related work

As noted, the cost of computing SIFT keypoints and descriptors is addressed by other local image features such as SURF [5], BRIEF [6], or 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 distinctive-

108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
ness, but neither have been shown to consistently outperform SIFT [2, 9].

132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
Efforts to reduce the computational costs of finding nearest neighbours to feature points have largely focused on metric trees. Naively the set of nearest neighbours 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 (a typical three megapixel image may contain anywhere from 500 to 5000 feature points). 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 $O(n \log n)$, 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 [12] 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. 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.

154
155
156
157
158
159
160
161
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 match 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 [1–3, 16].

162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
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 match, but the average ratio of the best and the average of second best matches 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 et al. 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].

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 [20–22]. Similar global



Figure 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.

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 [25–27]. For cases that require more flexibility such as scenes with moving elements and non-planar perspective change Pairwise constraints provide an alternative to the more rigid global constraints. Introduced by Leordeanu and Herbert [28], pairwise constraints work by applying a geometric constraint across matches 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 Hough 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].

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 com-

216
217
218
219
220
221
puted. Finding and computing descriptors can easily account for 80% of the time spent matching for bigger images (cf. Figure 7). For this reason *Fast-Match* is designed to only compute features for the part of the image we hope to match.

222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
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 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. 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.

237
238
239
240
241
242
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 can do some off-line computations. For the *general* variation we make those computations on the fly instead and do not assume anything to be pre-computed.

243 244 **Algorithm 1** Fast-Match

245
Require: $I_{\text{query}}, I_{\text{target}}$: images, $\text{maxiter} \in \mathbb{N}, \tau \in [0, 1]$
246 $M_{\text{seed}} \leftarrow \text{seed_matches}(I_{\text{query}}, I_{\text{target}})$
247 $M_{\text{final}} \leftarrow \emptyset$
248 $C_{\text{final}} \leftarrow \emptyset$
249 $M_{\text{seen}} \leftarrow \emptyset$
250 **while** $M_{\text{seed}} \neq \emptyset \wedge i < \text{maxiter}$ **do**
251 $M_{\text{round}} \leftarrow \text{get_matches}(M_{\text{seed}})$
252 $C_{\text{round}} \leftarrow \text{get_confidence}(M_{\text{round}})$
253 $M_{\text{seed}} \leftarrow \text{get_seeds}(M_{\text{round}} \setminus M_{\text{seen}}, C_{\text{round}}, \tau)$
254 $M_{\text{seen}} \leftarrow M_{\text{seen}} \cup M_{\text{seed}}$
255 $M_{\text{final}} \leftarrow M_{\text{final}} \cup M_{\text{round}}$
256 $C_{\text{final}} \leftarrow C_{\text{final}} \cup C_{\text{round}}$
257 **end while**
258 **return** $M_{\text{final}}, C_{\text{final}}$

261 3.2. Initiating Seeds

262
263
Depending on the scenario, several strategies can be used
264 to obtain a set of initial seed matches. In practice we have
265 found it efficient to resize both images to thumbnails and
266 use *Ratio-Match* to obtain a set of matches and ratios that
267 we then threshold with the confidence threshold τ to obtain
268 the initial seed matches. The thumbnails created need to be
269 sized so they on one hand ensure that objects of interest are

270
271
still detectable by a keypoint detector while on the other re-
272 main small to ensure that we find matches efficiently. We
273 determine empirically that a thumbnail size of 300×300
274 pixels maintains a good balance between speed and accu-
275 racy. We found this size to be independent of the original
276 image size as long as we are matching images that are big-
277 ger than one megapixel.

278 3.3. Collecting Matches and Computing Confidence

279
280
If a seed match yields a connection between two points
281 p_q in the *query image* and p_t in the *target image*, then we are
282 interested in collecting all matches between the regions R_q
283 and R_t centered around p_q and p_t respectively. From each
284 region we can extract a set of feature points between which
285 we try to find a set of matches M_{qt} and a set of confidence
286 scores C_{qt} .

287
288
Lowe and others have shown that the distance between
289 two SIFT descriptors is much less indicative of a true corre-
290 spondence than the ratio between the best and second best
291 match [1–3, 16]. This ratio is more formally defined as fol-
292 lows: If we let f_q be a feature in the *query image* and f_t, f_b
293 be the two nearest neighbours of f_q in the *target image* then
294 the ratio r is defined as follows:

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

295
296
Here $d(f_i, f_j)$ is the distance between the features f_i and f_j .
297 For SIFT this is the Euclidean distance. The distance be-
298 tween two feature points is greater when the feature points
299 resemble each other less. For this reason we have higher
300 confidence in a match with a low r -value. Using r as a
301 measure of match confidence presumes that we expect each
302 feature in the *target image* to have at most one true cor-
303 respondence in the *target image*. Intuitively if we try to
304 find a match for a feature f_i in an image that does not have
305 any true correspondences, then we would expect the two
306 closest neighbours to be roughly equally well matched with
307 f_i . The effectiveness of *Ratio-Match* over a naïve nearest
308 neighbor match ([1, 2]) supports this notion. For this rea-
309 son we attribute high confidence to matches where the clos-
310 est neighbour is dramatically closer to f_i than the second
311 closest neighbour and discard the rest.

312
313
We are faced with a problem when applying this tech-
314 nique to obtain the set of confidence scores C_{qt} , since for
315 any match in M_{qt} we only know the nearest neighbours
316 amongst the features of R_q and R_t . To get around this
317 problem we either assume to know one of the images be-
318 forehand (the *retrieval* variation) or – in case we cannot
319 make that assumption – we compute the features of one of
320 the images (the *general* variation). For a given match be-
321 tween features f_q and f_t we can now find the second clos-
322 est neighbour f_b and calculate the confidence as
323 $r = d(f_q, f_t)/d(f_q, f_b)$.

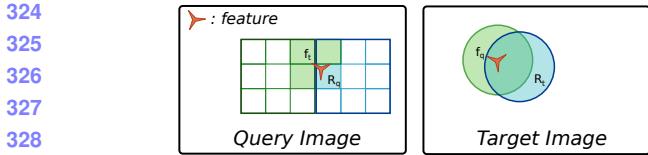


Figure 2: Exploration of features based on match. The areas shaded blue are the areas that were searched to obtain the match. The areas shaded green are candidate areas for obtaining more matches.

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} .

3.4. 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 confidence levels C_i that we can use to predict whether the neighbourhood of R_i is worth exploring.

There are many possible heuristics for predicting possible regions including 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, as illustrated in Figure 2. 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 Figure 1 we illustrate this process in the *Fast-Match* image pair. The *query image* to the right in the figure has a small blue square for each region that has been used while matching the two images.

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, quite a bit of care has to be expended making sure that results are properly cached. A matrix containing ‘bins’ of features can conveniently 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.

3.5. Computational Complexity

The difference between the *general* and *retrieval* variation 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 number of feature points, while finding $d(f_q, f_b)$ for all n features in the *target image* can be done in $O(n \log n)$ 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 number of possible seed matches is on average linear in the number of image features, and we also 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_1, f_2, \dots be the best, second best, etc 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 $\mathbb{P}(A_{ti}) \leq 0.5$ and A_{ti} is independent from A_{tj} when $i \neq j$.

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

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

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.d. stochastic 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 $\mathbb{E}(X_t) = \sum_{i=1}^{\infty} i 0.5^i = 2$, 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 $d(f_t, f_q) \leq d(f_t, f_b)$. 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 $S_n = X_1 + \dots + X_n$, in

324	378
325	379
326	380
327	381
328	382
329	383
330	384
331	385
332	386
333	387
334	388
335	389
336	390
337	391
338	392
339	393
340	394
341	395
342	396
343	397
344	398
345	399
346	400
347	401
348	402
349	403
350	404
351	405
352	406
353	407
354	408
355	409
356	410
357	411
358	412
359	413
360	414
361	415
362	416
363	417
364	418
365	419
366	420
367	421
368	422
369	423
370	424
371	425
372	426
373	427
374	428
375	429
376	430
377	431

432 which case $\mathbb{E}(S_n) = 2n$. This proves that the expected
 433 amount of seed matches is linear in the number of target
 434 features.
 435

436 For the above case it is theoretically possible that for ev-
 437 ery one of the n features we end up considering kn seed
 438 matches for some constant k , i.e. a quadratic behaviour. Us-
 439 ing Chebyshev’s inequality we can provide an upper bound
 440 on the probability of this event:

$$441 \quad \mathbb{P} \left(\left| \frac{S_n}{n} - \mathbb{E}(X_t) \right| \geq kn \right) \leq \frac{\text{Var} \left(\frac{S_n}{n} \right)}{k^2 n^2} = \frac{6}{k^2 n^3} \quad (2)$$

442 It is clear that for any constant k we can pick a number
 443 of features n which render the likelihood of a quadratic be-
 444 haviour infinitesimal. In essence the larger n becomes, the
 445 less likely *Fast-Match* is to exhibit super linear behaviour.
 446

447 The noteworthy part of this result is that for the *retrieval*
 448 variation we can match two images in $O(n)$. Since there are
 449 no large constants involved this makes it possible to rapidly
 450 match very large images. In particular *Fast-Match* is suited
 451 for cases where we are looking to match several large *query*
 452 images to one *target image*. In this case we can compute the
 453 features and distances of the *target image* and then match
 454 each *query image* in linear time. In contrast the general
 455 variation still has a complexity of $O(n \log n)$ due to the ne-
 456 cessity of calculating distances online.
 457

4. Experimental Setup

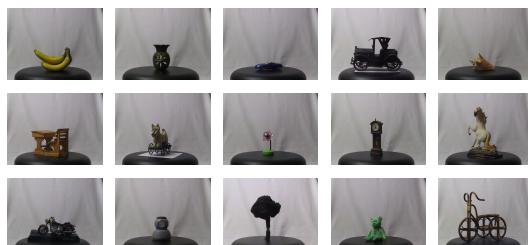
460 To evaluate *Fast-Match* on three sets of data. We use a
 461 dataset of 3024 image pairs featuring various 3D objects at
 462 different angles and lighting conditions to measure matching
 463 performance for real life objects when lighting and per-
 464 spective changes. To evaluate the performance of *Fast-
 465 Match* when only part of (or none of) the scene is shared
 466 between an image pair we conduct experiments on 400 ran-
 467 dom crops with different degree of image overlap taken
 468 from 4 image pairs, creating a challenging test set where
 469 large part of the images consists of background clutter sim-
 470 ilar in nature to the subject being matched. Finally we time
 471 the algorithm on two pairs of images with high pixel counts
 472 to measure performance in terms of speed.
 473

474 We compare the algorithm to the standard *Ratio-Match*
 475 [1] as well as the newer *Mirror-Match* [18]. We exclude the
 476 geometric algorithms discussed in related works from com-
 477 parison because even the fastest cases are two to three mag-
 478 nitudes slower than *Ratio-Match* and as such are too time
 479 consuming to use when matching features in large images.
 480

4.1. Evaluation of Fast-Match on 3D Objects

482 The 3D Objects dataset by Moreels and Perona [3] al-
 483 lows us to experimentally compare matching algorithms
 484 over a large range of object and surface types rotated on a
 485 turnstile and photographed from every angle in increments

486 of 5 degrees. 15 objects from the dataset are shown in Fig-
 487 ure 3. We use images of 84 different objects photographed
 488 under three different lighting conditions at 12 different an-
 489 gle intervals, conducting experiments with a total of 3024
 490 image pairs. All photos taken with a consumer camera in
 491 3.1 megapixel resolution.
 492



493 Figure 3: 15 objects from the 3D Objects dataset by Moreels
 494 and Pietro [3].
 495

496 To validate matches, Moreels and Perona propose a
 497 method using epipolar constraints [3, p.266]. According to
 498 their experiments, these constraints are able to identify true
 499 correspondences with an error rate of 2%. We use their pro-
 500 posed method to generate the ground truth for the evaluation
 501 of our framework.
 502

503 To compute the total number of possible true corre-
 504 spondences, we take each feature in a *query image* and
 505 count how many of them have a feature in the *target im-
 506 age* which would satisfy the epipolar constraints. To avoid
 507 matching non-moving background and foreground objects
 508 we excluded features with no correspondences from being
 509 matched. This was necessary because a relatively small
 510 number of actual true correspondences between folds in the
 511 background material were mistakenly counted as false pos-
 512 itives when evaluated which impacted the precision of test in
 513 particular on cases with few true correspondences. In prac-
 514 tice only few feature points were excluded for this reason.
 515

516 We evaluate all matching algorithms from our frame-
 517 work on the 3D Objects dataset by matching images at dif-
 518 ferent angular intervals. For each object we pick the *query*
 519 image as the image taken at 10 degrees rotation for calibra-
 520 tion stability. We then match this image with the same ob-
 521 ject turned an additional Δ degrees, $\Delta \in \{5, 10, \dots, 60\}$.
 522 For every angle interval we compare images taken under 3
 523 different lighting conditions as provided by the dataset. We
 524 include all objects in the database for which photos at 5 de-
 525 gree angle intervals are available except for the “Rooster”
 526 and “Sponge” objects due to image irregularities. For each
 527 angle this means that a total of 252 image pairs are matched.
 528 We calculate the result these pairs using a weighted aver-
 529 age based on the number of actual correspondences for each
 530 pairs. This ensures that image pairs that are more difficult
 531 to match are not contributing less to the end result because
 532 less actual matches would be found.
 533

540
541 **4.2. Evaluation of Fast-Match on partially overlap-**
542 **ping scenes**



551 Figure 4: Four image pairs from the Graffiti dataset by
552 Mikolajczyk et al. [2]. From left to right the images are
553 ‘Boat’, ‘Graf’, ‘Bark’ and ‘Wall’.

554 To evaluate how well *Fast-Match* handles partially overlapping scenes we make use of four images from the Graffiti dataset by Mikolajczyk et al. [2] as displayed in Figure 4 all featuring challenging perspective change. For each of the four image pairs we generate a hundred pairs of smaller crops each crop randomly positioned within the original image. This process ensures that image pairs that does not contain any overlap in scene are still visually similar, making it more challenging for a matching algorithm to avoid false positives, i.e. matching local image features that are not true correspondences.

555 In practice the amount of overlap between pairs in a test set will depend on the overlap and viewpoint change in the source image pair. To give a rough idea, Table 1 shows the overlap of patch pairs created from images 1 and 3 of the *Graf* image.

556 Table 1: Overlap in the set of 100 patch pairs created from
557 the ‘Graf’ image pair (Figure 4).

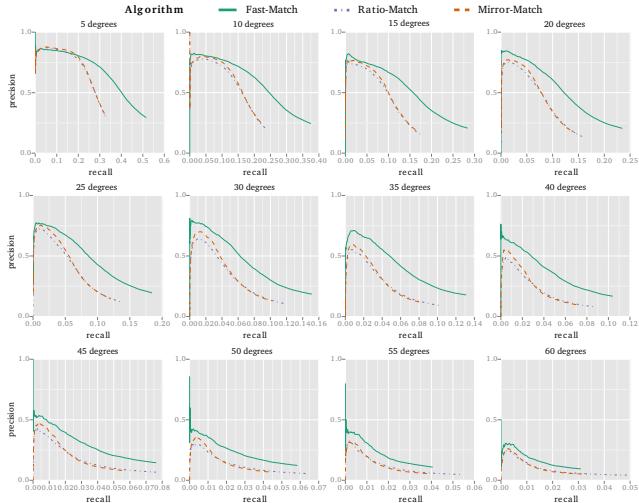
Amount of overlap:	0%	< 50%	> 50%
Number of patch pairs:	21	54	25

558 Given a potential match between two pixels p_1 and p_2 ,
559 $m = (p_1, p_2)$, and a homography H relating the two images
560 I_1 and I_2 , we can calculate if m is an inlier by checking if
561 the two points satisfy the following criteria:

$$|Hp_1 - p_2| + |H^{-1}p_2 - p_1| < d_{\max}$$

562 That is, the distance between p_1 translated to I_2 and p_2 plus
563 the distance between p_2 translated to I_1 should be less than
564 a certain threshold (we use $d_{\max} = 5$ pixels here).

565 As with the 3D Objects we accumulate results across the
566 hundred image crop pairs by using an average weighted by
567 the amount of true correspondences between each cropped
568 pair. This ensures that difficult to match and trivial image
569 pairs have an equal impact on the evaluation result despite
570 the difficult image pairs yielding far less actual matches.



571 Figure 5: Results for the 3D Objects dataset. Each plot con-
572 tains the average precision and recall over 84 objects photo-
573 graphed under 3 different lighting conditions weighted
574 over the number of true correspondences for each pair.

575 **4.3. Configuration of Fast-Match**

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

582 To achieve a good balance between speed and robustness
583 we let the region in which we extract features be a square
584 window with a side length of 90 pixels. From empirical
585 tests we found that this number works well with the SIFT
586 feature descriptor, but for other descriptors different num-
587 bers might lead to better performance. The window is split
588 in to nine smaller squares (see also Figure 2). When a seed
589 match falls in any of these squares we match only the fea-
590 tures within the smaller square. We let both the regions and
591 the smaller squares overlap each other at all edges with 25
592 pixels in order to capture feature points lying close to an
593 edge. For the image we have already cached we find all
594 features within a radius of 50 pixels of the seed match.

595 **5. Results**

596 Figure 5 shows the performance of the different match-
597 ing methods in our proposed framework for 12 increas-
598 ingly bigger angle differences. The results are shown in
599 a precision-recall plot to make it easy to compare perfor-
600 mance in terms of precision at similar levels of recall. For
601 each plot we show the accumulated results on all 3D ob-
602 jects, weighted by the number of possible true corre-
603 spondences for the individual objects. This ensures that each ob-

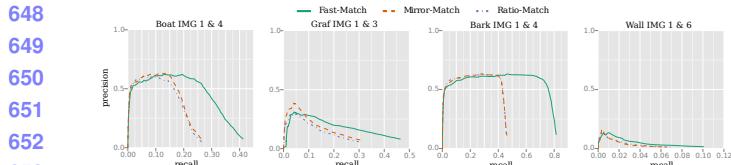


Figure 6: Results for the Graffiti dataset. Each plot contains the average result over 100 semi overlapping image patches based on the same two images weighted over the number of true correspondences for each pair.

ject contributes equally to the final result, despite some objects resulting in disproportionately more matches than others.

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

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

In Figure 6 we see the performance of the three different matching methods over 400 crop pairs based on four image pairs. As for the 3D Objects the results are shown in a precision-recall plot weighted by the number of possible true correspondences for the individual crop pairs.

At low recall we see a similar performance across all algorithms, but as the algorithms return more matches, *Fast-Match* shows drastically better precision at similar recall levels than *Mirror-Match* and *Ratio-Match*. While this difference is more noticeable in the case of the ‘Bark’ and ‘Boat’ images, the ‘Graf’ and ‘Wall’ also see a drastic improvement in recall but as precision is lower in general for these two images the effect is less visible.

For all four image pairs we see a precision dip at very low recall which is explained by the presence of image crop pairs with no or little overlap. When matching these pairs we might often find matches even though there are no or few possible true correspondences which in effect negatively impacts the results at very low recall rates.

Figure 7 demonstrate the speed of *Fast-Match* on image sizes as is typically produced by modern consumer cameras

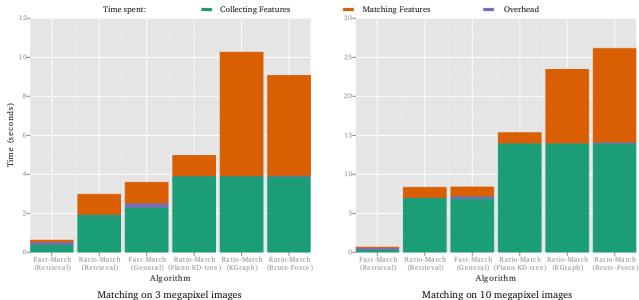


Figure 7: The duration in seconds of different variations of *Fast-Match* and *Ratio-Match*. The top plot shows results for two 10 MPixel images two 3 Mpixel images is used for the comparison at the bottom. From the left, the first two results presume a precached image while the rest do not require knowing any inputs beforehand. The image pairs used can be seen in Figure 1

and camera phones. In Figure 7 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 pre-computed 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 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.

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

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. We show that these results hold true for matching cases featuring partial or no overlap by comparing *Fast-Match* on 400 crop pairs of planar scenes. *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.

References

- [1] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision* **60** (2004) 91–110
- [2] Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence* **27** (2005) 1615–1630
- [3] Moreels, P., Perona, P.: Evaluation of features detectors and descriptors based on 3D objects. *International Journal on Computer Vision* **73** (2007) 263–284
- [4] Heinly, J., Dunn, E., Frahm, J.M.: Comparative evaluation of binary features. In: Proc. European Conference on Computer Vision (ECCV). (2012) 759–773
- [5] Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: Proc. European Conference on Computer Vision (ECCV). (2006) 404–417
- [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
- [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
- [8] Ke, Y., Sukthankar, R.: PCA-SIFT: A more distinctive representation for local image descriptors. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Volume 2., IEEE (2004) II–506
- [9] Li, J., Allinson, N.M.: A comprehensive review of current local features for computer vision. *Neurocomputing* **71** (2008) 1771–1787
- [10] Beis, J.S., Lowe, D.G.: 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) 1000–1006
- [11] Lowe, D.G.: Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Volume 2., Ieee (1999) 1150–1157
- [12] Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP (1). (2009) 331–340
- [13] Dong, W., Moses, C., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of the 20th international conference on World wide web, ACM (2011) 577–586
- [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
- [15] Baumberg, A.: Reliable feature matching across widely separated views. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1. (2000) 774–781
- [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
- [17] Brown, M., Szeliski, R., Winder, S.: Multi-image matching using multi-scale oriented patches. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1. (2005) 510–517
- [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
- [19] Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG* **28** (2009) 24

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863

- 864 [20] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to
865 image analysis and automated cartography. Communications of the ACM **24** (1981) 381–395 918
866
867
868
869 [21] Torr, P.H.S., Zisserman, A.: MLESAC: A new robust 919
870 estimator with application to estimating image geometry. Computer Vision and Image Understanding **78** 920
871 (2000) 138–156 921
872
873 [22] Szeliski, R.: Computer Vision: Algorithms and Applications. Springer (2010) 922
874
875
876 [23] Kim, J., Choi, O., Kweon, I.S.: Efficient feature 923
877 tracking for scene recognition using angular and scale 924
878 constraints. In: Proc. IEEE/RSJ International Conference 925
879 on Intelligent Robots and Systems (IROS), Nice, 926
880 France (2008) 4086–4091 927
881
882 [24] Schmid, C., Mohr, R.: Local grayvalue invariants for 928
883 image retrieval. IEEE Transactions on Pattern Analysis 929
884 and Machine Intelligence **19** (1997) 530–535 930
885
886 [25] Torresani, L., Kolmogorov, V., Rother, C.: Feature 931
887 correspondence via graph matching: Models and 932
888 global optimization. In: Proc. European Conference 933
889 on Computer Vision (ECCV). Springer (2008) 596– 934
609 935
890
891 [26] Yarkony, J., Fowlkes, C., Ihler, A.: Covering trees 936
892 and lower-bounds on quadratic assignment. In: Proc. 937
893 IEEE Conference on Computer Vision and Pattern 938
894 Recognition (CVPR), IEEE (2010) 887–894 939
895
896 [27] Cho, M., Lee, J., Lee, K.M.: Reweighted random 940
897 walks for graph matching. In: Proc. European Conference 941
898 on Computer Vision (ECCV). Springer (2010) 942
492–505 943
899
900 [28] Leordeanu, M., Hebert, M.: A spectral technique for 944
901 correspondence problems using pairwise constraints. 945
902 In: Proc. IEEE Conference on Computer Vision and 946
903 Pattern Recognition (CVPR). Volume 2., San Diego, 947
904 CA (2005) 1482–1489 948
905
906 [29] Yuan, Y., Pang, Y., Wang, K., Shang, M.: Efficient 949
907 image matching using weighted voting. Pattern Recognition Letters **33** (2012) 471–475 950
908
909 [30] Pang, Y., Shang, M., Yuan, Y., Pan, J.: Scale 951
910 invariant image matching using triplewise constraint and 952
911 weighted voting. Neurocomputing **83** (2012) 64–71 953
912
913 [31] Cho, M., Lee, J., Lee, K.M.: Feature correspondence 954
914 and deformable object matching via agglomerative 955
915 correspondence clustering. In: Proc. International 956
916 Conference on Computer Vision (ICCV), IEEE (2009) 957
1280–1287 958
917