

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

004 Anonymous ACCV 2014 submission

006 Paper ID ***

009 **Abstract.** Both consumer cameras and camera phones produce images
 010 that often exceed 10 mega pixels. Yet computing and matching local im-
 011 age features in images of this size can easily take more than twenty sec-
 012 onds using fast matching algorithms. This is much too slow for interactive
 013 applications 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 the 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* and demonstrate that *Fast-Match*
 019 often doubles the precision on difficult to match image pairs at equal
 020 recall. In addition we prove that when one image is known in advanced,
 021 *Fast-Match* can achieve linear performance matching in the amount of
 022 feature points.

023 **1 Introduction**

025 Whenever we match local image features we are faced with a choice between
 026 performance and precision. On one hand SIFT features proposed by Lowe [1]
 027 have shown again and again to compare favorably to other local image descrip-
 028 tors especially under unrestrained conditions such as perspective change with
 029 non-planar scenes [2,3,4]. On the other, SIFT keypoints and descriptors are slow
 030 to compute, the main *raison d'être* for other local image features. In many applica-
 031 tions of computer vision we would like the increase the computational perfor-
 032 mance in order to work on larger images, bigger image sets, at faster frame rates
 033 or with more limited hardware but we cannot give up the additional precision
 034 that SIFT affords us over other local features without negative effects for our
 035 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 two
 040 variations of the algorithm: a *general* and a *retrieval* variation. The *general* vari-
 041 ation functions as a traditional matching algorithm and matches two unknown
 042 images albeit faster and more robustly than conventional methods. The *retrieval*
 043 variation on the other hand assumes that we know one of the images we intend
 044 to match beforehand but given this trade-off this variation offers performance

045 a magnitude faster than existing matching methods without compromising on
 046 accuracy. We will unimaginatively refer to the proposed algorithm as *Fast-Match*
 047 in this paper and make clear from the context whether this refers to the *retrieval*
 048 or *general* variation. As an aside we adopt an artificial distinction between the
 049 words *match* and *correspondence* to make it clear when a match between two
 050 images is correct. The word *match* will be used to denote a result from a matching
 051 algorithm that may or may not be correct while a *correspondence* refers to
 052 an actual visual correspondence between two points in two images.

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

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

066 Efforts to reduce the computational costs of finding nearest neighbours to
 067 feature points has largely been focused on metric trees. Naïvely the set of near-
 068 est neighbours between features in two images can computed by brute force in
 069 $O(n^2)$ where n is the number of features in the two images. Typically a three
 070 mega pixel image contains anywhere from 500 to 5000 feature points. When
 071 SIFT was originally published Lowe proposed using the Best-Bin-First method
 072 to approximately search for nearest neighbours [10,11]. This reduces the com-
 073 putational complexity to $O(n \log n)$ but even approximate metric trees are hard
 074 pressed to compete with brute force due to the high dimensionality of SIFT and
 075 the constant costs incurred with constructing and searching in a metric tree.
 076 Later work by Muja and Lowe has focused on improving approximate nearest
 077 neighbour searches by using several KD-Trees simultaneously while optimizing
 078 the tree structure using K-Means to cluster similar features [12]. Recent work
 079 on knn-graphs shows a lot of promise for high dimensional cases [13]. We later
 080 review these improvements and their effect on efficiently matching large scale
 081 images.

082 A wealth of matching methods have focused instead on increasing the ef-
 083 ficiency of local feature matching. *Fast-Match* builds upon the foundation of
 084 *Ratio-Match* introduced originally by Deriche et al. [14] and Baumberg [15] even
 085 though Lowe is usually credited for introducing it [1]. They both propose using
 086 the ratio of the similarity of the best to second best correspondence of a given
 087 point to evaluate how unique it is. Their finding has later been tested by sev-
 088 eral independent teams, all concluding that thresholding based on this ratio is

090 generally superior to thresholding based on similarity or returning all nearest
 091 neighbours [1,2,3,16].

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

100 Another inspiration for the design of *Fast-Match* is *Patch-Match* as intro-
 101 duced by Barnes et al. [19]. Like *Fast-Match*, *Patch-Match* is an iterative algo-
 102 rithm for fast image matching, but unlike *Fast-Match* which is focused on sparse
 103 local image features, *Patch-Match* is designed for dense image features. It works
 104 by randomly creating a set of matches from both images and then iteratively
 105 improving it.

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

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

132 *Fast-Match* makes use of a angular assumption to efficiently find new matches
 133 in the geometric neighbourhood of already confirmed matches. However this
 134 constraint is only locally applied to increase the number of matches, making



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

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

Fast-Match robust to outliers. In addition *Fast-Match* does not rely on an initial set of matches and derives much of its speed from the fact that not even an initial set of local image features is required, making it faster than *Ratio-Match* on most occasions.

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

2 Matching Fast and Slow

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

2.1 Considerations

If we set out to design a truly fast matching algorithm, we cannot rely just on optimizing the matching step once the descriptors from both images have been found and computed. As illustrated in Figure 2 finding and computing descriptors can easily account for 80% of the time spent matching for bigger images. For this reason *Fast-Match* is designed to only compute features for the part of the image we hope to match.

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

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

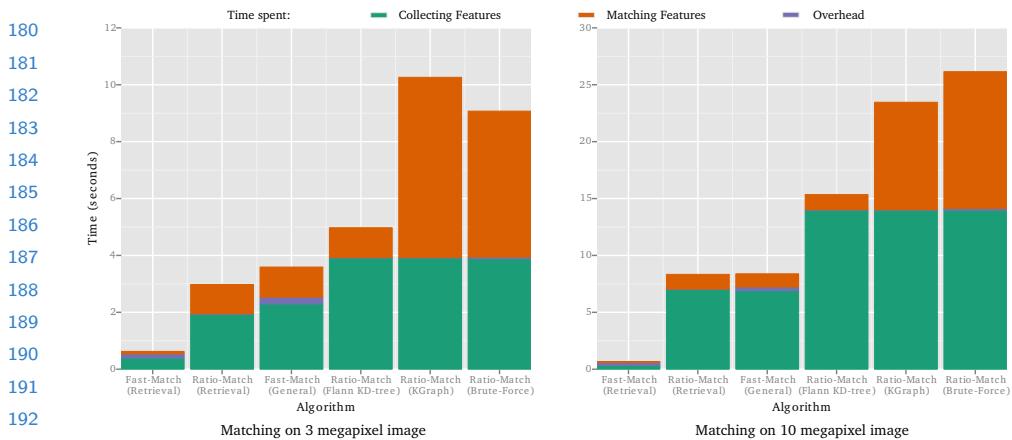


Fig. 2: The duration in seconds of different variations of *Fast-Match* and *Ratio-Match*. The top plot shows results for two 10 MPixel images two 3 Mpixel images is used for the comparison at the bottom. In both plots from left to right: 1. The proposed *Fast-Match* algorithm with one image pre-cached.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

the final precision and recall can be adjusted by thresholding the matches on their confidence scores at the end.

We introduce two variations of the algorithm, the a *general* and the *retrieval* variation. The *retrieval* variation 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 that anything can be pre-computed. In Figure 1 we illustrate the result of matching *Fast-Match* on two images compared with the result of *Ratio-Match* for the same images.

2.2 Initiating Seeds

To obtain a set of initial seed matches several strategies can be used depending on the use case. For the case of matching a pair of two images we can get a rough group of seed matches by matching the thumbnails of the two images with for example *Ratio-Match*. However if we were matching images in a series, such as frames from a movie we could instead make use of a subset of the matches from the last frame to seed *Fast-Match* on the next. In practice we have found it efficient to resize both images to thumbnails of 300 by 300 pixels and use ratio match to obtain a set of matches and ratios that we then threshold with the confidence threshold τ to obtain the initial seed matches.

225	Algorithm 1 Fast-Match	225
226	Require: I_{query}, I_{target} : images, $\text{maxiter} \in \mathbb{N}$, $\tau \in [0, 1]$	226
227	$M_{seed} \leftarrow \text{seed_matches}(I_{query}, I_{target})$	227
228	$M_{final} \leftarrow \emptyset$	228
229	$C_{final} \leftarrow \emptyset$	229
230	$M_{seen} \leftarrow \emptyset$	230
231	while $M_{seed} \neq \emptyset \wedge i < \text{maxiter}$ do	231
232	$M_{round} \leftarrow \text{get_matches}(M_{seed})$	232
233	$C_{round} \leftarrow \text{get_confidence}(M_{round})$	233
234	$M_{seed} \leftarrow \text{get_seeds}(M_{round} \setminus M_{seen}, C_{round}, \tau)$	234
235	$M_{seen} \leftarrow M_{seen} \cup M_{seed}$	235
236	$M_{final} \leftarrow M_{final} \cup M_{round}$	236
237	$C_{final} \leftarrow C_{final} \cup C_{round}$	237
238	end while	238
239	return M_{final}, C_{final}	239
240		240

2.3 Collecting Matches and Computing Confidence

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

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

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

Here $d(f_i, f_j)$ is the distance between the features f_i and f_j . For SIFT this is the euclidean distance. 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 measure for 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

253	$r = r(f_q, (f_t, f_b))$	253
254		254
255	$= \frac{d(f_q, f_t)}{d(f_q, f_b)}.$	255
256		256
257		257
258		258
259		259
260		260
261		261
262		262
263		263
264		264
265		265
266		266
267		267
268		268
269		269

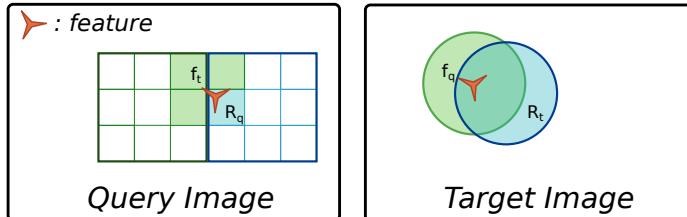


Fig. 3: Exploration of features based on match. The areas shaded blue are the areas that were searched to obtain the match. The areas shaded green are candidate areas for obtaining more matches.

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

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

2.4 Exploring for Matches

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

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

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

323

324 2.5 Computational Complexity

325

326 The difference between the *general* and *retrieval* variation of *Fast-Match* consists
 327 in whether we compute the features in the *target image* (general) or if we presume
 328 the features are computed offline (retrieval). Computing the features is linear in
 329 the amount of features, while finding $d(f_q, f_b)$ for all n features in the *target*
 330 *image* can be done in $O(n \log n)$ using metric trees.

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

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

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

$$346 X_t = |\{f_q \mid f_q \in F_{\text{query}}, d(f_q, f_t) < d(f_q, f_b)\}| \quad (1) \\ 347$$

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

351 In the worst case, each feature in the *target image* has a true correspondence
 352 in the *query image* and $\mathbb{P}(A_{ti} = 0.5)$, in which case we find can find $\mathbb{P}(X_t =$
 353 $n) = \mathbb{P}(A_{ti})^n = 0.5^n$, as well as the expected value $\mathbb{E}(X_t) = \sum_{i=1}^{\infty} 0.5^i i = 2$, and
 354 the variance $\text{Var}(X_t) = \sum_{i=1}^{\infty} 0.5^i (i - 2)^2 = 6$.

355 For $\tau = 1$ we accept a seed match when $\tau < r$, that is, when $d(f_t, f_q) \leq$
 356 $d(f_t, f_b)$. That means that for any feature in the *query image*, we accept at
 357 most X_t matches. To find the total amount of seed matches given n , we let
 358 $S_n = X_1 + \dots + X_n$, in which case $\mathbb{E}(S_n) = 2n$ which shows that the expected
 359 amount of seed matches is linear in the number of target features.

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360 For the above case it is theoretically possible that we for every one of the n
 361 features end up considering kn seed matches for some constant k , i.e. a quadratic
 362 behaviour. We can provide an upper bound on the probability of this event:

$$364 \quad \mathbb{P} \left(\left| \frac{S_n}{n} - \mathbb{E}(X_t) \right| \geq kn \right) \leq \frac{\text{Var} \left(\frac{S_n}{n} \right)}{k^2 n^2} \quad (\text{chebyshev}) \quad (2)$$

$$366 \quad \leq \frac{6}{k^2 n^3} \quad (3)$$

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

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

3 Experimental Setup

383 We evaluate *Fast-Match* over 3024 image pairs featuring various 3D objects at
 384 different angles to measure precision and recall, as well as two pairs of images
 385 with high pixel counts to measure performance in terms of speed. We compare the
 386 algorithm to the standard *Ratio-Match*[1] as well as the newer *Mirror-Match*[18].
 387 These two algorithms were selected because they are magnitudes faster than the
 388 fastest geometric algorithms while at the same time providing robust matches.

3.1 Evaluation of Fast-Match on 3D Objects

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

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

403 To compute the total number of possible correspondences, we take each fea-
 404 ture in a *query image* and count how many of them have a feature in the *target*

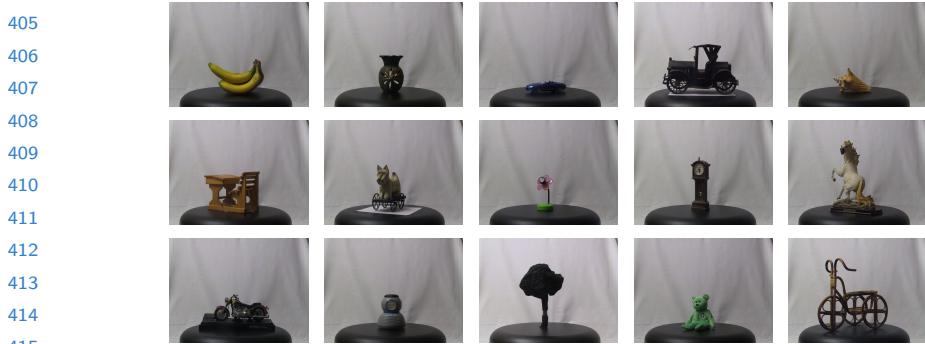


Fig. 4: 15 objects from the 3D Objects dataset by Moreels and Pietro [3].

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.

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

3.2 Configuration of *Fast-Match*

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

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

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

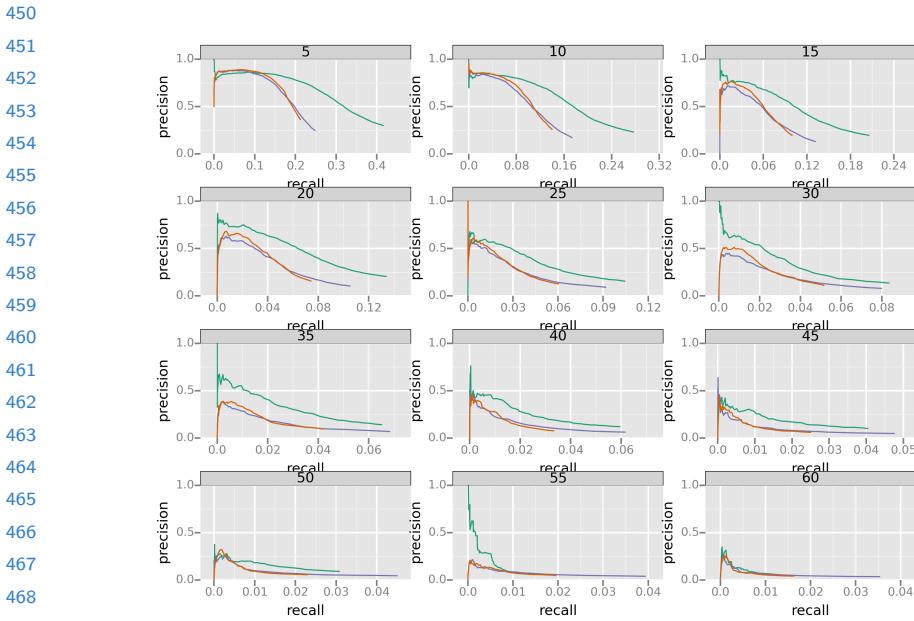


Fig. 5: Results for the 3d objects dataset. Each plot contains data accumulated from 84 objects photographed under 3 different lighting conditions

4 Results

Figure 5 shows the performance of the different matching methods in our proposed framework for 12 increasingly bigger angle differences. The results are shown in a precision / recall plot to make it easy to compare performance in terms of precision at similar levels of recall. For each plot we show the accumulated results on all 3D objects, weighted by the number of possible true correspondences for the particular object. This ensures that each object 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 *Ratio-Match* and the other algorithms at low recall while more than doubling precision at higher recalls. Finally for image pairs with angular differences over 50^0 the performance of all algorithms decline with *Fast-Match* still showing better results for the most part.

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

495 that we are likely to find correct correspondences in the regions that we are
 496 matching. However lenient thresholds can easily impact the matching speed, so
 497 in practical situations we will usually have to choose between matching fast and
 498 precise or slower with high recall.

499 Figure 1 and 2 demonstrate the speed of *Fast-Match* on image sizes as is typi-
 500 cally produced by modern consumer cameras and camera phones. In Figure 2
 501 we compare the speed of the two variations of *Fast-Match* to different variations
 502 of *Ratio-Match* on two image pairs, one with two images of 10 mega pixel and
 503 another where the same image pair has been scaled to 3 mega pixels. For the
 504 larger images we see the clearest difference in performance. The retrieval variant
 505 of *Fast-Match* matches the image pairs in around one second while a retrieval
 506 variant of *Ratio-Match* with precomputed features spends eight seconds on the
 507 same task. This result is roughly equal to the time the general variation of *Fast-*
 508 *Match* spends matching the two images without any pre-computations. For the
 509 nearest neighbor search we note that for large images there are very substantial
 510 speed gains to be had by choosing the right algorithm for finding nearest neigh-
 511 bors, with Muja et al.’s Flann matcher being vastly superior to brute force and
 512 KGraph.

513 For the 3 mega pixel picture the pattern repeats itself, although with smaller
 514 margins separating *Fast-Match* and *Ratio-Match*. When images are smaller, com-
 515 putations like obtaining the initial seed matches are comparatively more costly.

516 Figure 1 shows the result from matching the pair of three mega pixel images
 517 using *Fast-Match* and *Ratio-Match*. An almost identical result can be obtained
 518 using the pair of 10 mega pixel pictures instead. In the figure we see how *Fast-*
 519 *Match* only ends up matching the part of the images that are likely to have
 520 correspondences. This highlights both a strength and a weakness of *Fast-Match*.
 521 It is this myopic matching approach that yields *Fast-Match* its speed and preci-
 522 sion, but *Fast-Match* will not be the best candidate for almost identical images
 523 where *Fast-Match*’s step by step approach incurs too much overhead to compete
 524 in speed with *Ratio-Match*.

525

5 Conclusion

526 We have introduced *Fast-Match* in two forms, a general and a retrieval varia-
 527 tion. The retrieval variation assumes that we know one of the images that we are
 528 matching ahead of time in order to pre-cache feature points. The general vari-
 529 ation has no assumptions and performs the pre-caching of feature points online
 530 instead. We have proven that while the general variation has a computational
 531 complexity of $O(n \log n)$ with n being the number of feature points, the retrieval
 532 variation will on average run in $O(n)$. From experiments on large images we
 533 have shown that *Fast-Match* can be a magnitude faster than *Ratio-Match* while
 534 providing comparable results. We went on to evaluate *Fast-Match* to *Ratio-*
 535 *Match* and *Mirror-Match* using images of rotated 3D objects and demonstrated
 536 that *Fast-Match* outperforms the other algorithms significantly over 3024 image
 537 pairs and in most cases double the matching precision at similar recall rates.
 538

539

540 *Fast-Match* is particularly applicable in cases where we are interested in matching
 541 one image to multiple large images, in which case we can quickly pre-cache
 542 the image and use this data for each of the other images.

543

544 References

545

- 546 1. D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal on Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- 547 2. K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.
- 548 3. P. Moreels and P. Perona, “Evaluation of features detectors and descriptors based
 549 on 3D objects,” *International Journal on Computer Vision*, vol. 73, no. 3, pp.
 550 263–284, 2007.
- 551 4. J. Heinly, E. Dunn, and J.-M. Frahm, “Comparative evaluation of binary features,”
 552 in *Proc. European Conference on Computer Vision (ECCV)*, 2012, pp. 759–773.
- 553 5. H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in
 554 *Proc. European Conference on Computer Vision (ECCV)*, 2006, pp. 404–417.
- 555 6. M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: binary robust inde-
 556 pendent elementary features,” in *Proc. European Conference on Computer Vision
 557 (ECCV)*, 2010, pp. 778–792.
- 558 7. S. Leutenegger, M. Chli, and R. Y. Siegwart, “BRISK: Binary robust invariant scal-
 559 able keypoints,” in *Proc. International Conference on Computer Vision (ICCV)*.
 560 IEEE, 2011, pp. 2548–2555.
- 561 8. Y. Ke and R. Sukthankar, “Pca-sift: A more distinctive representation for local
 562 image descriptors,” in *Proc. IEEE Conference on Computer Vision and Pattern
 563 Recognition (CVPR)*, vol. 2. IEEE, 2004, pp. II–506.
- 564 9. J. Li and N. M. Allinson, “A comprehensive review of current local features for
 565 computer vision,” *Neurocomputing*, vol. 71, no. 10, pp. 1771–1787, 2008.
- 566 10. J. S. Beis and D. G. Lowe, “Shape indexing using approximate nearest-neighbour
 567 search in high-dimensional spaces,” in *Computer Vision and Pattern Recognition,
 568 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE, 1997, pp.
 569 1000–1006.
- 570 11. D. G. Lowe, “Object recognition from local scale-invariant features,” in *Computer
 571 vision, 1999. The proceedings of the seventh IEEE international conference on*,
 572 vol. 2. Ieee, 1999, pp. 1150–1157.
- 573 12. M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic
 574 algorithm configuration.” in *VISAPP (1)*, 2009, pp. 331–340.
- 575 13. W. Dong, C. Moses, and K. Li, “Efficient k-nearest neighbor graph construction for
 576 generic similarity measures,” in *Proceedings of the 20th international conference on
 577 World wide web*. ACM, 2011, pp. 577–586.
- 578 14. R. Deriche, Z. Zhang, Q.-T. Luong, and O. Faugeras, “Robust recovery of the
 579 epipolar geometry for an uncalibrated stereo rig,” in *Proc. European Conference
 580 on Computer Vision (ECCV)*, 1994, pp. 567–576.
- 581 15. A. Baumberg, “Reliable feature matching across widely separated views,” in *Proc.
 582 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1,
 583 2000, pp. 774–781.
- 584 16. J. Rabin, J. Delon, and Y. Gousseau, “A statistical approach to the matching of
 585 local features,” *SIAM Journal on Imaging Sciences*, vol. 2, no. 3, pp. 931–958,
 586 2009.

CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.

14 ACCV-14 submission ID ***

- 585 17. M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale
586 oriented patches," in *Proc. IEEE Conference on Computer Vision and Pattern
587 Recognition (CVPR)*, vol. 1, 2005, pp. 510–517.
- 588 18. J. T. Arnofred, S. Winkler, and S. Süsstrunk, "Mirror Match: Reliable feature point
589 matching without geometric constraints," in *Proc. IAPR Asian Conference on Pat-
590 tern Recognition (ACPR)*, 2013, pp. 256–260.
- 591 19. C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: a ran-
592 domized correspondence algorithm for structural image editing," *ACM Transac-
593 tions on Graphics-TOG*, vol. 28, no. 3, p. 24, 2009.
- 594 20. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model
595 fitting with applications to image analysis and automated cartography," *Commu-
596 nications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- 597 21. P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with applica-
598 tion to estimating image geometry," *Computer Vision and Image Understanding*,
599 vol. 78, no. 1, pp. 138–156, 2000.
- 600 22. R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.
- 601 23. J. Kim, O. Choi, and I. S. Kweon, "Efficient feature tracking for scene recognition
602 using angular and scale constraints," in *Proc. IEEE/RSJ International Conference
603 on Intelligent Robots and Systems (IROS)*, Nice, France, 2008, pp. 4086–4091.
- 604 24. C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE
605 Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp.
606 530–535, 1997.
- 607 25. L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph
608 matching: Models and global optimization," in *Proc. European Conference on Com-
609 puter Vision (ECCV)*. Springer, 2008, pp. 596–609.
- 610 26. J. Yarkony, C. Fowlkes, and A. Ihler, "Covering trees and lower-bounds on quadratic
611 assignment," in *Proc. IEEE Conference on Computer Vision and Pattern Recog-
612 nition (CVPR)*. IEEE, 2010, pp. 887–894.
- 613 27. M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching,"
614 in *Computer Vision-ECCV 2010*. Springer, 2010, pp. 492–505.
- 615 28. M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems
616 using pairwise constraints," in *Proc. IEEE Conference on Computer Vision and
617 Pattern Recognition (CVPR)*, vol. 2, San Diego, CA, 2005, pp. 1482–1489.
- 618 29. Y. Yuan, Y. Pang, K. Wang, and M. Shang, "Efficient image matching using
619 weighted voting," *Pattern Recognition Letters*, vol. 33, no. 4, pp. 471–475, 2012.
- 620 30. Y. Pang, M. Shang, Y. Yuan, and J. Pan, "Scale invariant image matching using
621 triplewise constraint and weighted voting," *Neurocomputing*, vol. 83, pp. 64–71,
622 2012.
- 623 31. M. Cho, J. Lee, and K. M. Lee, "Feature correspondence and deformable ob-
624 ject matching via agglomerative correspondence clustering," in *Proc. Interna-
625 tional Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 1280–1287.
- 626 32. L. Wu, Y. Niu, H. Zhang, H. Zhu, and L. Shen, "Robust feature point matching
627 based on local feature groups (LFGs) and relative spatial configuration," *Journal
628 of Computational Information Systems*, vol. 7, no. 9, pp. 3235–3244, 2011.
- 629 33. H.-Y. Chen, Y.-Y. Lin, and B.-Y. Chen, "Robust feature matching with alternate
630 hough and inverted hough transforms," in *Computer Vision and Pattern Recog-
631 nition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2762–2769.

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629