

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

Anonymous ACCV 2014 submission

Paper ID \*\*\*

## Abstract.

## 1 Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 129 2 Matching Fast and Slow

130  
 131 In this chapter we will introduce the fundamentals of *Fast-Match* and motivate  
 132 the design choices as we go. The algorithm is made of 3 components; finding  
 133 seeds, finding matches, and exploring for other places where matches might be.  
 134 After discussing a few design choices in Section 2.1 we discuss each component

090

091

092

093

094

095

096

097

098

099

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

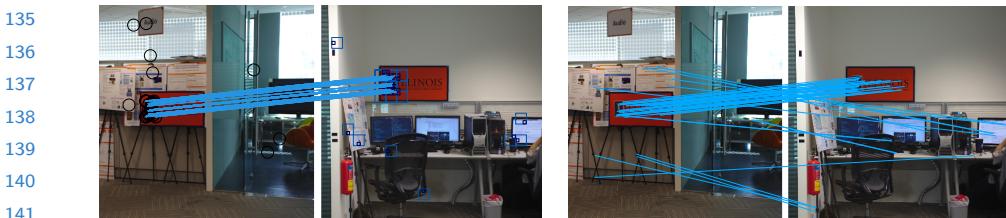
130

131

132

133

134



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

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

## 2.1 Considerations

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

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

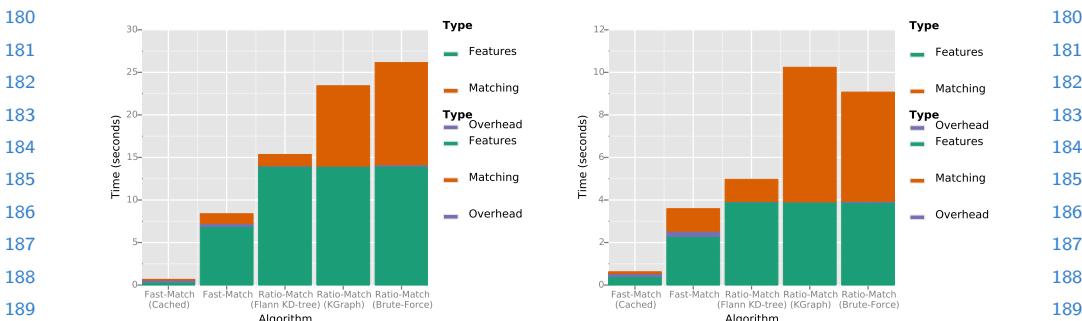


Fig. 2: The duration in seconds of different variations of *Fast-Match* and *Ratio-Match*. The top plot shows results for two 10 MPixel images two 3 Mpixel images is used for the comparison at the bottom. In both plots from left to right: 1. The proposed *Fast-Match* algorithm with one image pre-cached.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

## 2.2 Initiating Seeds

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

## 2.3 Collecting Matches and Computing Confidence

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

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

225 **Algorithm 1** Fast-Match

---

```

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

```

---

241 expect a one-to-one matching of the features between the two images. Intuitively  
 242 if we try to find a match for a feature  $f_i$  in an image that does not have any true  
 243 correspondences, then we would expect the two closest neighbours to be roughly  
 244 equally well matched with  $f_i$ . For this reason we attribute high confidence to  
 245 matches where the closest neighbour is dramatically closer to  $f_i$  than the second  
 246 closest neighbour and discard the rest.

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

255 In the case that we know the image beforehand, we can further optimize this  
 256 step. If we assume the *target image* is known in advance we can approximate the  
 257 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,  
 258 which means we can pre-calculate  $d(f_t, f_b)$  for all features in the cached *target*  
 259 *image* before we start matching.

261 **2.4 Exploring for Matches**

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

268 There are many possible heuristics for predicting possible matching regions  
 269 including using local and global epipolar assumptions and partial graph isomor-

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

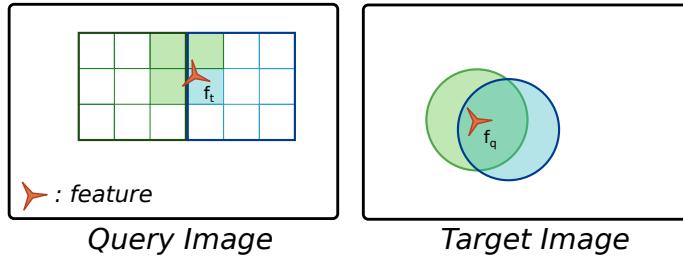


Fig. 3: Exploration of features based on match. The areas shaded blue are the areas that were searched to obtain the match. The areas shaded green are candidate areas for obtaining more matches.

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

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

## 2.5 Computational Complexity

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

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

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

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

325 Here  $F_{\text{query}}$  is the set of features in the *query image*. That is,  $X_t$  is the number  
 326 of features in the query image closer to  $f_q$  than  $f_b$ . For each feature  $f_t$ ,  $X_t$  is an  
 327 i.i.d stochastic variable.

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

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

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

$$339 \mathbb{P}\left(\left|\frac{S_n}{n} - \mathbb{E}(X_t)\right| \leq f(n)\right) \leq \frac{\text{Var}\left(\frac{S_n}{n}\right)}{(f(n))^2} \quad (\text{chebyshev}) \quad (2)$$

$$340 \leq \frac{6}{n(f(n))^2} \quad (3)$$

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

343 The noteworthy part of this result is that for the *retrieval* approach we can  
 344 match two images in  $O(n)$ . Since there are no large constants involved this makes  
 345 it possible to rapidly match very large images. In particular *Fast-Match* is suited  
 346 for cases where we are looking to match several large *query images* to one *target*  
 347 *image*. In this case we can compute the features and distances of the *target image*  
 348 and then match each *query image* in linear time.

### 3 Experimental Setup

355 We evaluate *Fast-Match* over 3024 image pairs featuring various 3D objects at  
 356 different angles. We compare the algorithm to the standard *Ratio-Match*[1] as  
 357 well as the newer *Mirror-Match*[18]. These two algorithms were selected because

360 they are magnitudes faster than the fastest geometric algorithms while at the  
 361 same time providing robust matches.  
 362

### 363 3.1 Evaluation of Fast-Match on 3D Objects

364 The 3D objects dataset by Moreels and Pietro [3] allows us to experimentally  
 365 compare matching algorithms over a large range of object and surface types ro-  
 366 tated on a turnstile and photographed from every 5 degree turn. 15 objects from  
 367 the dataset are shown in Figure 4. We use images of 84 different objects un-  
 368 der three different lighting conditions at 12 different angle intervals, conducting  
 369 experiments with a total of 3024 image pairs.  
 370



383 Fig. 4: 15 objects from the 3D Objects dataset by Moreels and Pietro [3].  
 384

385  
 386 To validate matches, Moreels and Pietro propose a method using epipolar  
 387 constraints [3, p.266], which is outlined in Figure 5. According to their experi-  
 388 ments, these constraints are able to identify true correspondences with an error  
 389 rate of 2%. We use their proposed method to generate the ground truth for the  
 390 evaluation of our framework.  
 391

392 To compute the total number of possible correspondences, we take each fea-  
 393 ture in a *query image* and count how many of them have a feature in the *target*  
 394 *image* which would satisfy the epipolar constraints outlined above. When using  
 395 this dataset, features with no correspondences were not included in the set of  
 396 features for testing, so as to avoid matching non-moving background and fore-  
 397 ground objects.  
 398

399 We evaluate all matching algorithms from our framework on the 3D Objects  
 400 dataset by matching images at different angular intervals. For each object we  
 401 pick the *query* image as the image taken at 10 degrees rotation for calibration  
 402 stability. We then match this image with the same object turned an additional  $\Delta$   
 403 degrees,  $\Delta \in \{5, 10, \dots, 60\}$ . For every angle interval we compare images taken  
 404 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.  
 405

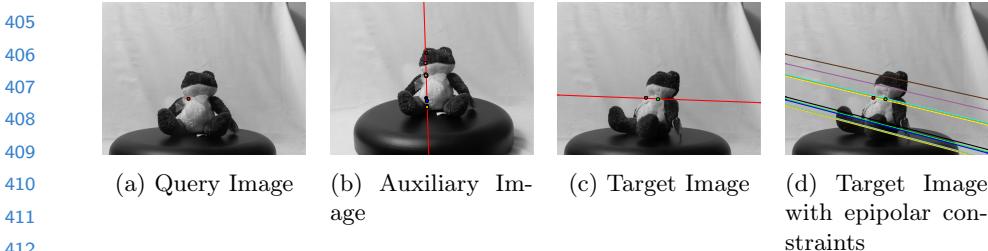


Fig. 5: Creating epipolar constraints based on three source images [3]. (a) *Query image* marked with the position of the feature we are attempting to match. (b) *Auxiliary image*, taken at the same rotation as the *query image* but from a higher elevation angle. The line going through the image is the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

### 3.2 Configuration of *Fast-Match*

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

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

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

450	<b>4 Results</b>	450
451		451
452	<b>5 Conclusion</b>	452
453		453
454	<b>References</b>	454
455		455
456	1. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," <i>International Journal on Computer Vision</i> , vol. 60, no. 2, pp. 91–110, 2004.	456
457	2. K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," <i>IEEE Trans. Pattern Analysis and Machine Intelligence</i> , vol. 27, no. 10, pp. 1615–1630, 2005.	457
458	3. P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3D objects," <i>International Journal on Computer Vision</i> , vol. 73, no. 3, pp. 263–284, 2007.	458
459	4. J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in <i>Proc. European Conference on Computer Vision (ECCV)</i> , 2012, pp. 759–773.	459
460	5. H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in <i>Proc. European Conference on Computer Vision (ECCV)</i> , 2006, pp. 404–417.	460
461	6. M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in <i>Proc. European Conference on Computer Vision (ECCV)</i> , 2010, pp. 778–792.	461
462	7. S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in <i>Proc. International Conference on Computer Vision (ICCV)</i> . IEEE, 2011, pp. 2548–2555.	462
463	8. Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in <i>Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)</i> , vol. 2. IEEE, 2004, pp. II–506.	463
464	9. J. Li and N. M. Allinson, "A comprehensive review of current local features for computer vision," <i>Neurocomputing</i> , vol. 71, no. 10, pp. 1771–1787, 2008.	464
465	10. J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in <i>Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on</i> . IEEE, 1997, pp. 1000–1006.	465
466	11. D. G. Lowe, "Object recognition from local scale-invariant features," in <i>Computer vision, 1999. The proceedings of the seventh IEEE international conference on</i> , vol. 2. Ieee, 1999, pp. 1150–1157.	466
467	12. M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in <i>VISAPP (1)</i> , 2009, pp. 331–340.	467
468	13. W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in <i>Proceedings of the 20th international conference on World wide web</i> . ACM, 2011, pp. 577–586.	468
469	14. R. Deriche, Z. Zhang, Q.-T. Luong, and O. Faugeras, "Robust recovery of the epipolar geometry for an uncalibrated stereo rig," in <i>Proc. European Conference on Computer Vision (ECCV)</i> , 1994, pp. 567–576.	469
470	15. A. Baumberg, "Reliable feature matching across widely separated views," in <i>Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)</i> , vol. 1, 2000, pp. 774–781.	470
471	16. J. Rabin, J. Delon, and Y. Gousseau, "A statistical approach to the matching of local features," <i>SIAM Journal on Imaging Sciences</i> , vol. 2, no. 3, pp. 931–958, 2009.	471
472		472
473		473
474		474
475		475
476		476
477		477
478		478
479		479
480		480
481		481
482		482
483		483
484		484
485		485
486		486
487		487
488		488
489		489
490		490
491		491
492		492
493		493
494		494

CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.

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

- 495 17. M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale  
496 oriented patches," in *Proc. IEEE Conference on Computer Vision and Pattern  
497 Recognition (CVPR)*, vol. 1, 2005, pp. 510–517.  
498 18. J. T. Arnofred, S. Winkler, and S. Süsstrunk, "Mirror Match: Reliable feature point  
499 matching without geometric constraints," in *Proc. IAPR Asian Conference on Pat-  
500 tern Recognition (ACPR)*, 2013, pp. 256–260.  
501 19. C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: a ran-  
502 domized correspondence algorithm for structural image editing," *ACM Transac-  
503 tions on Graphics-TOG*, vol. 28, no. 3, p. 24, 2009.  
504 20. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model  
505 fitting with applications to image analysis and automated cartography," *Commu-  
506 nications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.  
507 21. P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with applica-  
508 tion to estimating image geometry," *Computer Vision and Image Understanding*,  
509 vol. 78, no. 1, pp. 138–156, 2000.  
510 22. R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.  
511 23. J. Kim, O. Choi, and I. S. Kweon, "Efficient feature tracking for scene recognition  
512 using angular and scale constraints," in *Proc. IEEE/RSJ International Conference  
513 on Intelligent Robots and Systems (IROS)*, Nice, France, 2008, pp. 4086–4091.  
514 24. C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE  
515 Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp.  
516 530–535, 1997.  
517 25. L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph  
518 matching: Models and global optimization," in *Proc. European Conference on Com-  
519 puter Vision (ECCV)*. Springer, 2008, pp. 596–609.  
520 26. J. Yarkony, C. Fowlkes, and A. Ihler, "Covering trees and lower-bounds on quadratic  
521 assignment," in *Proc. IEEE Conference on Computer Vision and Pattern Recog-  
522 nition (CVPR)*. IEEE, 2010, pp. 887–894.  
523 27. M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching,"  
524 in *Computer Vision-ECCV 2010*. Springer, 2010, pp. 492–505.  
525 28. M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems  
526 using pairwise constraints," in *Proc. IEEE Conference on Computer Vision and  
527 Pattern Recognition (CVPR)*, vol. 2, San Diego, CA, 2005, pp. 1482–1489.  
528 29. Y. Yuan, Y. Pang, K. Wang, and M. Shang, "Efficient image matching using  
529 weighted voting," *Pattern Recognition Letters*, vol. 33, no. 4, pp. 471–475, 2012.  
530 30. Y. Pang, M. Shang, Y. Yuan, and J. Pan, "Scale invariant image matching using  
531 triplewise constraint and weighted voting," *Neurocomputing*, vol. 83, pp. 64–71,  
532 2012.  
533 31. M. Cho, J. Lee, and K. M. Lee, "Feature correspondence and deformable ob-  
534 ject matching via agglomerative correspondence clustering," in *Proc. International  
535 Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 1280–1287.  
536 32. L. Wu, Y. Niu, H. Zhang, H. Zhu, and L. Shen, "Robust feature point matching  
537 based on local feature groups (LFGs) and relative spatial configuration," *Journal  
538 of Computational Information Systems*, vol. 7, no. 9, pp. 3235–3244, 2011.  
539 33. H.-Y. Chen, Y.-Y. Lin, and B.-Y. Chen, "Robust feature matching with alternate  
hough and inverted hough transforms," in *Computer Vision and Pattern Recog-  
nition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2762–2769.