Cluster-Wise Ratio Tests for Fast Camera Localization

Raúl Díaz, Charless C. Fowlkes
Computer Science Department, University of California, Irvine
{rdiazgar,fowlkes}@uci.edu

Abstract

Feature point matching for camera localization suffers from scalability problems. Even when feature descriptors associated with 3D scene points are locally unique, as coverage grows, similar or repeated features become increasingly common. As a result, the standard distance ratio-test used to identify reliable image feature points is overly restrictive and rejects many good candidate matches. We propose a simple coarse-to-fine strategy that uses conservative approximations to robust local ratio-tests that can be computed efficiently using global approximate k-nearest neighbor search. We treat these forward matches as votes in camera pose space and use them to prioritize back-matching within candidate camera pose clusters, exploiting feature co-visibility captured by the 3D model camera pose graph. This approach achieves state-of-the-art camera pose estimation results on a variety of benchmarks, outperforming several methods that use more complicated data structures and that make more restrictive assumptions on camera pose. We carry out diagnostic analyses on a difficult test dataset containing globally repetitive structure which suggest our approach successfully adapts to the challenges of large-scale pose estimation.

1. Introduction

In this paper we consider the problem of estimating the full 6DOF camera pose of a query image with respect to a large-scale 3D model such as those obtained from a Structure-from-Motion (SfM) pipeline [27, 34, 16, 25]. A typical approach is to detect distinctive 2D feature points in a query image and perform correspondence search against feature descriptors associated with 3D points obtained from the SfM reconstruction. This initial matching is performed in descriptor space (e.g., SIFT [14] or SURF [3]) using an approximate k-nearest neighbor search implementation [17, 18]. Candidate 2D-3D correspondences are then further filtered using robust fitting techniques (e.g., RANSAC variants [10, 32, 15]) to identify inliers and the final camera pose estimated using an algebraic PnP solver and non-linear refinement. Camera pose estimation is a fundamental building block in many computer vision algorithms (e.g., incremental bundle adjustment), can provide strong constraints on object recognition (see e.g., [33, 8]), and is useful in robotics applications such as autonomous driving and navigation.

This work is supported by NSF grants IIS-1618806 and IIS-1253538.

Unfortunately, the performance of standard camera localization pipelines degrades as the size of the 3D model grows. Finding good correspondences becomes difficult in the large-scale setting due to two factors. First, standard 2D-to-3D *forward* matching is likely to accept bad correspondences of a query feature with the model since the feature space becomes cluttered with similar descriptors from completely different locations. Standard heuristics for identifying distinctive matches, such as the distance ratio-test of Lowe [14], which compares the distance to the nearest-neighbor point descriptor with that of the second-nearest neighbor, fail due to proximity of other model feature descriptors. Second, increasingly noisy correspondences obtained from the matching stage drives up the runtime of the robust pose estimation step, whose complexity typically grows exponentially with the number of outliers. These difficulties are particularly evident in large urban environments, where repeated structure is common and local features become less distinctive [31, 1].

Related Work: These problems are well known and have been approached in several ways in the literature. Works such as [13, 12] focus on generating a simplified 3D model that contains only a representative subset of distinctive model points. With a smaller model and prioritized search, it becomes possible to replace the traditional approach of 2D-to-3D *forward* matching, with 3D-to-2D *back* matching, allowing the ratio test to be performed in the sparser feature space of the query image.

An alternative to removing points from the model is to cluster and quantize model point feature descriptors. [21] use vocabulary trees to speed up forward matching by assigning each model point and each query feature to a vocabulary word, yielding faster runtimes since the vocabulary size is generally smaller than the model point cloud.

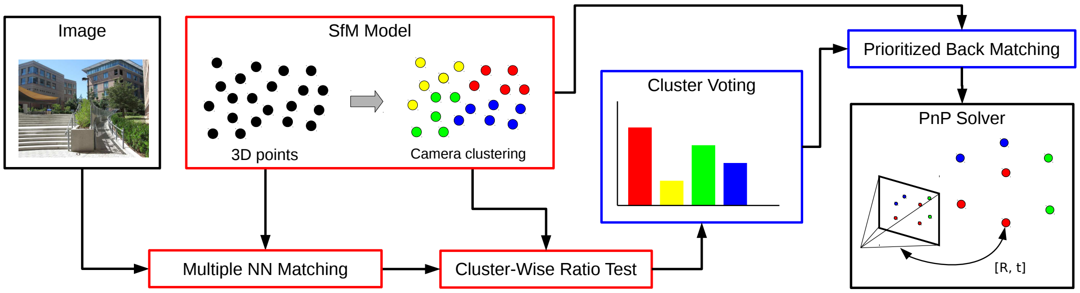

Figure 1. Overview of our camera pose-estimation pipeline. We first exploit multiple nearest neighbor search and camera pose clustering to identify candidate feature correspondences (red boxes, described in Section 2). We then utilize co-visibility to expand this set and prioritize back-matching of model features (blue boxes, described in Section 3).

A linear search for the first and second nearest neighbors is performed within each word bin, and a ratio test filters out non-distinct correspondences. [22] use active search in the vocabulary tree to prioritize back matching of 3D points close to those that have already been found and terminate early as soon as a sufficient number of matches have been identified.

A very different approach is taken in the works of [36, 29]. Camera localization is framed as a Hough voting procedure, where the geometric properties of SIFT (scale and orientation) provide approximate information about likely camera pose from individual point correspondences. By using focal length and camera orientation priors, each 2D-to-3D match casts a vote into the intersection of a visibility cone and a hypothesized ground-plane. Orientation and model co-visibility are further used to filter out unlikely matches, rapidly identifying the potential camera locations.

Our Contribution: Inspired by this prior work, we propose a fast, simple method for camera localization that scales well to large models with globally repeated structure. Our approach avoids complicated data structures and makes no hard *a priori* assumptions on camera pose (e.g., gravity direction of the camera). Our basic insight is to utilize a coarse-to-fine approach that rapidly narrows down the region of camera pose space associated with the query image. Specifically, we formulate a linear time voting process over camera pose space by assigning each single model view to an individual camera pose bin. This voting allows us to identify model views likely to overlap the query image and to prioritize back matching of those views against it while exploiting co-visibility constraints and local ratio testing.

Figure 1 gives on overview of our pipeline. Our first contribution (Section 2) is to introduce and analyze two ratio-tests that can be used to find distinctive matches in a pool of candidates produced by global k-nearest neighbor search (kNN). Our second contribution (Section 3) uses these forward matches as votes to prioritize back matching of model images against the query image. Extensive experimental evaluation (Section 4) suggests this approach scales well and outperforms existing methods on several pose-estimation benchmarks.

2. Ratio Tests for Global Matching

Forward-matching of query image points against a model is effective when the model is small. In such models, approximate nearest-neighbors are often true correspondences and ratio-testing is effective at discarding bad matches. In this section we first establish that clustering the model into smaller sub-models and performing forward-matching within each cluster is sufficient to achieve good performance for large models (Section 2.1). We then describe how to efficiently approximate exhaustive cluster-wise matching by global forward-matching using approximations to the local ratio test (Section 2.2) followed by back-matching.

2.1. Clustering and Exhaustive Local Matching

A naive approach to solving camera localization at large scale is to simply divide the 3D model into small pieces (clusters) and perform matching and robust PnP pose estimation for each cluster. This avoids the problems of global feature repetition and difficulties of high density in the feature space. However, this is infeasible from a computational point of view as it requires building a nearest-neighbor data structure for each cluster and matching to each cluster separately at test time. Consider a kd-tree, where searching for a match in a set with N descriptors is logarithmic in the set size: $O(log(N))$. If we divide the model into $|\mathcal{C}| = N/S$ clusters of constant size S, execution time is dominated by the number of clusters which grows linearly in the model size, $O(|\mathcal{C}|log(\frac{N}{|\mathcal{C}|})) = O(\frac{N}{S}log(S)) > O(log(N))$. While not practical at scale, we take this *exhaustive local matching* approach as a gold-standard baseline for evaluating our coarse-to-fine approach.

#clusters	#images	#inliers	ratio	error [m]	fwd [s]	RNSC [s]	total [s]
1 (global)	463	94	0.57	0.64	0.833	0.129	0.962
50	512	66	0.54	0.45	13.10	43.62	56.822
500	517	51	0.49	0.29	80.23	523.69	603.52

Table 1. Results on Eng-Quad using a standard localization framework applied to model clusters. Performing localization separately in each cluster improves the number of localized cameras and the median error accuracy, at the expense of longer runtimes due to exhaustive matching. We also report the number of inliers and inlier ratio, as well as forward matching, RANSAC, and total times.

Exhaustive Local Matching is effective but slow: To evaluate clustering and local matching, we use the Eng-Quad dataset from [9], and build two SfM models using COLMAP [25]. The first model contains only the training image set, while a second model bundles both the training and test images and is used for evaluating localization accuracy. We geo-register the resulting reconstructions with a GIS model so that the scale of the SfM model is approximately metric. 5129 training images of the 6402 were bundled, and 520 out of 570 test images were additionally bundled in the test model. The resulting point cloud has 579,859 3D points and 2,901,885 feature descriptors. We refer to these descriptors as *views* of the point.

To generate clusterings of the model, we construct a scene matrix S whose (i, j) entry contains the number of points that image pair I_i, I_j share in the SfM model. We performed spectral clustering [26] on the scene matrix using the 50 largest eigenvectors and produce three different granularities: no clustering at all (purely global), 50 clusters, and 500 clusters. To evaluate exhaustive local matching, we matched a query image against every cluster and select the one that produces the smallest localization error. For matching to a cluster, we use FLANN [18] to find the first and second NN of each query point and apply a standard ratio test with a $\tau = 0.7$ threshold. We ran RANSAC on each set of candidate cluster correspondences using a P3P solver [11] and a focal length prior based on the image EXIF metadata. Similar to [13], an image is considered to be successfully matched if it has at least 12 inlier correspondences with a reprojection error less than $\epsilon = 6px$.

Table 1 shows that exhaustive local matching within each cluster performs much better than global matching, with lower median error and fewer failures. However, the execution time grows roughly linearly with respect to the number of clusters, motivating our coarse-to-fine strategy.

2.2. Local Ratio Tests for Global Matches

How can we get the benefits of local cluster-wise matching while maintaining the computational cost associated with a single global nearest-neighbor search? Cluster-wise matching considers a nearest-neighbor per-cluster for each query point. To try and recover this larger pool of candidate correspondences using global search, we propose to retrieve the global top k nearest-neighbors for each query point. Fortunately, approximate kNN searches are not substantially more costly since those points typically live in adjacent leaves of the kd-tree (which must be explored even for a 1-NN retrieval). A larger set of candidate matches can address the problem of repeated structure by retrieving the set of multiple scene points that might correspond to a query point. However, it also results in a k-fold increase in outliers which we now address.

Algorithm 1 Global Forward Matching

INPUT: Query features Q, Model features $\mathcal{V}$, NN search depth k, ratio test threshold τ, match count threshold N_F
$\mathcal{M} = \emptyset$
while $|\mathcal{M}| < N_F$ **do**
  $q =$ random sample from Q
  $\{v_1, ..., v_{k+1}\} =$ kNN$(q, \mathcal{V}, k+1)$
  $\alpha_q = \frac{\|q - v_1\|}{\|q - v_{k+1}\|}$
  if $\alpha_q \leq \tau$ **then**
    $\mathcal{M} = \mathcal{M} \cup \{(q, v_1), ..., (q, v_k)\}$
  end if
end while
return $\mathcal{M}$

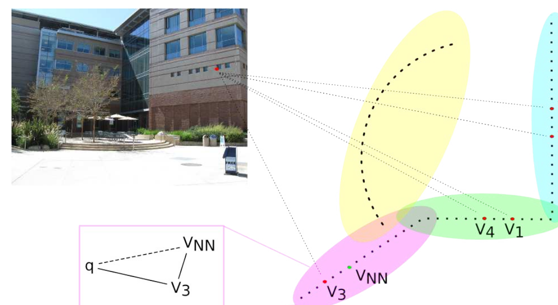

Figure 2. Cluster-wise ratio testing. Model views are divided into clusters. For a query feature q in the image (red dot), we search up to 5 nearest neighbors. Within clusters containing two or more matches, we can perform a standard local 1-ratio test within the cluster (e.g. v_1 and v_4 are the first and second NN in the green cluster). For the singleton v_3 in the pink cluster, we use an alternate *t-ratio* test (Eq. 2) based on the matched view's nearest neighbor v_{NN} (rather than the query point's true second nearest neighbor within the cluster).

We define a *view* $v \in \mathcal{V}$ as the 2D point observation of a 3D point $p \in \mathcal{P}$ in a particular model image $I \in \mathcal{I}$. Given a camera pose clustering $\mathcal{C}$ of the SfM model images, we assign the view descriptors of each image to their corresponding cluster $c \in \mathcal{C}$. Note that these clusters divide images in disjoint groups, but they do share common points, as a 3D point can have multiple views belonging to images assigned to different clusters. For a query image I with query features Q, we search for k approximate nearest neighbors using a global kd-tree structure built from all views $\mathcal{V}$.

Global k-ratio tests: We start with a conservative global ratio-test (Algorithm 1) to prune candidate matches by comparing the distance ratio of the first and $k+1$ nearest neighbor retrieved, as proposed by [35]. If the ratio is greater than threshold τ, we drop the query point. Otherwise, all k nearest neighbors pairs $\{(q, v_1), ..., (q, v_k)\}$ are included in the set of putative correspondences $\mathcal{M}$. This *global* ratio test is much more conservative than the standard first vs second NN test. In the remainder of this paper, we will refer to this global test as *k-ratio*, defined formally as $\frac{\|q-v_1\|}{\|q-v_{k+1}\|} \leq \tau$. The standard first versus second NN test will be referred as *1-ratio*.

Proposition 1. *If a candidate match fails the global k-ratio test, it also fails the local 1-ratio test.*

Proof. Let $\{v_{c_1}, v_{c_2}\} \subset \{v_1, ..., v_k\}$ be the first and second *local* nearest neighbors of a particular query feature q. Since the global set $\{v_1, ..., v_k\}$ is sorted by ascending distance, this implies that $\|q - v_{c_2}\| \leq \|q - v_{k+1}\|$, and $\|q - v_{c_1}\| \geq \|q - v_1\|$. Formally,

$$\frac{\|q - v_{c_1}\|}{\|q - v_{c_2}\|} \geq \frac{\|q - v_{c_1}\|}{\|q - v_{k+1}\|} \geq \frac{\|q - v_1\|}{\|q - v_{k+1}\|} \tag{1}$$

Hence, the local 1-ratio will always be equal or greater than the global k-ratio. This guarantees that any correspondence rejected by the k-ratio test would also have failed the local 1-ratio test. A correspondence passing the k-ratio test might not pass the local 1-ratio test, so the local 1-ratio test is a more stringent criteria. □

Cluster-wise ratio tests: After the initial global filtering, we would like to perform local ratio testing within each cluster. When more than two candidate matches for a query point belong to the same cluster, we can simply *re-rank* them and apply a standard 1-ratio test. For example, suppose two global matches (q, v_2) and (q, v_4) which are the second and fourth global NN of the query feature q fall into the same cluster. If v_2 and v_4 are views of distinct 3D points, then they are necessarily the first (q, v_{c_1}) and second (q, v_{c_2}) local nearest-neighbors of q in that cluster (see Figure 2). Any lower-ranked matches within the cluster can be ignored and the 1-ratio test applied to this pair.

When only a single global match falls within a cluster we can no longer perform an exact local 1-ratio test since we do not have immediate access to the 2nd nearest neighbor within that cluster. Instead we develop a bound based on the triangle inequality to define an alternate test for such cases which we refer to as the *t-ratio* test.

Given a local correspondence $v_{c_1} \in c$, we define $v_{NN} = kNN(v_{c_1}, c, 1)$ as the nearest neighbor of view v_{c_1} in the feature space defined by cluster c. Since v_{NN} is obtained purely from training data, we can pre-compute it offline and access it at test time. We define the *t-ratio* test as:

$$\frac{\|q - v_{c_1}\|}{\|q - v_{c_1}\| + \|v_{c_1} - v_{NN}\|} \leq \tau \tag{2}$$

Although we missed the local 2nd nearest neighbor in the global search, the distance $\|v_{c_1} - v_{NN}\|$ provides useful information on how far away the 2nd nearest neighbor might be.

Proposition 2. *If a candidate match fails the t-ratio test, it also fails the local 1-ratio test.*

Proof. Let q be a query feature, v_{c_1} and v_{c_2} the fist and second local nearest neighbors in a cluster c, and $v_{NN} = kNN(v_{c_1}, c, 1)$. We can bound the distance to the second nearest neighbor by the inequalities:

$$\|q - v_{c_2}\| \leq \|q - v_{NN}\| \leq \|q - v_{c_1}\| + \|v_{c_1} - v_{NN}\| \tag{3}$$

where the first inequality holds since $\|q - v_{c_2}\| \leq \|q - v\| \quad \forall\, v \in c \setminus v_{c_1}$, and the second holds by the triangle inequality. Thus,

$$\frac{\|q - v_{c_1}\|}{\|q - v_{c_2}\|} \geq \frac{\|q - v_{c_1}\|}{\|q - v_{c_1}\| + \|v_{c_1} - v_{NN}\|} \tag{4}$$

Consequently, a singleton match that fails the t-ratio test will always fail the local 1-ratio test. The t-ratio test thus only filters correspondences that would have failed the local ratio test if v_{c_2} was available. □

Back-matching and fitting: To provide additional robustness to outliers, we can *back match* views (model feature point descriptors) which were indicated as candidate correspondences from the forward matching. For any such candidate matching view, we search for the first and second nearest neighbor matches using a kd-tree built over the query image features and apply the 1-ratio test. We then select as the final set of correspondences the intersection of pairs (q, v) that passed the forward and back matching process. These pairs are cluster-wise *best buddies* [7], since each q and v of a pair are both discriminative features in the query and model feature space.

2.3. Cluster-wise ratio-tests are effective and fast

The cluster-wise ratio test, defined in Algorithm 2, prunes a large number of non-discriminative correspondences while still maintaining the locally unique matches. The complexity of this algorithm is linear in the number of forward correspondences N_F. For every local NN v_{c_1}, we simply look for its second NN pair v_{c_2} within the list of k nearest neighbors. The list of intra-cluster nearest neighbors is simply a view-to-v_{NN} vector that can be pre-computed offline and accessed at constant time, similar to vocabulary-based methods that store view-to-word assignments. Hence, at most N_F ratio tests will be performed.

Algorithm 2 Cluster-Wise Ratio Test

INPUT: matches $\mathcal{M}$, clusters $\mathcal{C}$, threshold τ
$\mathcal{M}_\mathcal{F} = \emptyset$
for $c \in \mathcal{C}$ **do**
  for $(q, v_{c_1}) \in \mathcal{M}$ with $v_{c_1} \in c$ **do**
    if $(q, v_{c_2}) \in \mathcal{M}$ with $v_{c_2} \in c$ **then**
      $\alpha_{(q,c)} = \frac{\|q-v_{c_1}\|}{\|q-v_{c_2}\|}$ ▷ local 1-ratio test
    else
      $v_{NN} = kNN(v_{c_1}, c, 1)$
      $\alpha_{(q,c)} = \frac{\|q-v_{c_1}\|}{\|q-v_{c_1}\|+\|v_{c_1}-v_{NN}\|}$ ▷ t-ratio test
    end if
    if $\alpha_{(q,c)} \leq \tau$ **then**
      $\mathcal{M}_\mathcal{F} = \mathcal{M}_\mathcal{F} \cup (q, v_{c_1})$
    end if
  end for
end for
return $\mathcal{M}_\mathcal{F}$

#clusters	#imgs	#inl	ratio	err.	fwd [s]	RT [s]	bck [s]	RNSC [s]	total [s]
1	481	115	0.74	0.69	0.821	0.008	0.021	0.046	0.895
50	477	127	0.59	0.66	0.818	0.008	0.028	0.061	0.915
500	480	133	0.56	0.61	0.821	0.009	0.038	0.066	0.934
5129	482	136	0.55	0.62	0.833	0.009	0.048	0.070	0.961

Table 2. Quantitative results on the 520 test image set using the proposed localization framework of algorithm 2 and best-buddy filtering. We used 5 nearest neighbors in the k-NN search. We evaluated four spatial subdivisions, including a finest clustering in which each camera in the model is considered a single cluster. Localization accuracy is competitive with exhaustive local matching with achieving runtimes comparable to global matching.

We evaluated this cluster-wise approach using the same settings as our gold standard baseline experiment. We added a finer division of the model, consisting of atomic clusters with a single image each. Table 2 shows the localization performance on these different granularities. A single global cluster gives surprisingly good results in the number of localized cameras, although it provides worse camera position results. This is due to the restrictiveness of the ratio test in denser search spaces, yielding fewer inliers and missing some discriminative correspondences that would improve results. As we increase the number of clusters, the localization errors are reduced (8 cm on average) thanks to the cluster-wise ratio test which provides more high confidence matches (at the expensive of longer RANSAC runtimes). We obtain best results using the finest clustering (a single model camera per cluster), successfully localizing 482 images. Compared to the gold-standard of Table 1, our strategy is competitive, by only dropping 5% in localization performance while being three orders of magnitude faster. Moreover, the finest single-image clusters provide the best result we can avoid running any complex clustering method (e.g., spectral clustering). We use single-image clusters in the remainder of the paper.

Dubrovnik - 800 test images

Method	top-1	top-2	top-5	top-10	Time [s]
$N_F = 50$	99.00%	99.38%	99.62%	99.88%	0.048
$N_F = 100$	99.62%	99.75%	99.88%	99.88%	0.085
$N_F = 200$	100%	100%	100%	100%	0.157

Eng-Quad - 520 test images

Method	top-1	top-2	top-5	top-10	Time [s]
$N_F = 50$	83.85%	85.96%	88.46%	88.65%	0.064
$N_F = 100$	84.62%	86.54%	89.62%	90.96%	0.125
$N_F = 200$	85.77%	87.69%	90.38%	91.35%	0.242
$N_F = 500$	86.15%	88.27%	90.96%	91.35%	0.502
All features	86.92%	89.23%	91.15%	91.92%	0.833

Table 3. We achieve perfect location recognition results on the Dubrovnik dataset using a random subset of 200 query features that pass the k-ratio and cluster-wise ratio tests, suggesting that our approach successfully finds local discriminative correspondences for all 800 test images. We also obtain good results in the more challenging Eng-Quad dataset, recognizing 478 (91.92%) images. This agrees with the baseline results obtained in Table 2.

3. Accelerating Matching by Pose Voting

As Table 1 suggests, with appropriate cluster-wise testing, forward matching now constitutes the primary computational bottleneck. Short of simplifying the model (e.g., as pursued by [12, 13]), how might we further accelerate the matching process? A natural strategy is to carry out forward matching incrementally and stop as soon as we have a sufficient number of matches to guarantee a good result. From this perspective, we can view forward matching as "voting" for the location of the query camera. Unlike [36, 29] where votes were cast into a uniformly binned camera translation space, we use each model camera pose as a putative bin to cast our votes (also used in [19]). We avoid additional data structures like vocabulary trees in favor of storing a simple but effective view-to-v_{NN} vector that enforces local uniqueness. Once we have accumulated enough votes to narrow down the camera pose to a few candidate clusters, we can terminate forward matching and carry out back matching with little loss in accuracy.

3.1. Coarse localization using cluster matching

To analyze how many votes are needed to determine a good localization, we frame the problem as that of *location recognition* [20, 30, 2, 24], namely producing a short ranked list of model images that depict the same general location as the query image. We follow the evaluation procedure of [5], reporting if there exists at least one image among the *top-k* images that shares 12 or more fundamental matrix inliers. We benchmark performance on two datasets: Eng-Quad and Dubrovnik [13].

The results in Table 3 are inspiring. Algorithm 2 is able to recognize the location of all 800 test images in the Dubrovnik dataset using 200 random features passing the k-ratio test. Results on the more challenging Eng-Quad

Algorithm 3 Prioritized Back Matching

INPUT: forward matches $\mathcal{M}_{\mathcal{F}}$, clustering $\mathcal{C}$, query features Q, threshold τ, minimum number of matches N_B, scene graph G
$H_c = |(q, v) \in \mathcal{M}_{\mathcal{F}} : v \in c| \quad \forall c$ ▷ Cast votes
$\mathcal{M}_{\mathcal{B}} = \emptyset, VC = \emptyset$
while $(|\mathcal{M}_{\mathcal{B}}| < N_B) \wedge (|VC| \leq 20)$ **do**
  $c^* = \text{argmax}_{c \notin VC} H$
  $\mathcal{M}_{c^*} = \emptyset$
  for $v \in c^*$ **do**
    $q_1, q_2 = kNN(v, Q, 2)$ ▷ Back match v to query
    $\alpha_v = \frac{\|v - q_1\|}{\|v - q_2\|}$
    if $\alpha_v \leq \tau$ **then**
      $\mathcal{M}_{c^*} = \mathcal{M}_{c^*} \cup (q_1, v)$
    end if
  end for
  $\mathcal{M}_{\mathcal{B}} = \mathcal{M}_{\mathcal{B}} \cup \mathcal{M}_{c^*}$
  if $|\mathcal{M}_{c^*}| \geq 12$ **then**
    for $(q, v) \in \mathcal{M}_{c^*}$ **do**
      for $c' \in \mathcal{C}$ with $v \in c'$ **do**
        $H_{c'} = H_{c'} + 1$ ▷ Update votes
      end for
    end for
  end if
  $VC = VC \cup c^*$ ▷ Update visited clusters (images)
end while
return $\mathcal{M}_{\mathcal{B}}$

dataset provide almost 92% accuracy on recognizing the landmarks of the 520 query images for which we have a ground truth pose. Importantly, a random subset of a few hundred query features achieves nearly as good recognition results as using all image features (a query image usually has 5,000 to 10,000 features). This suggests that the forward matching can be terminated early while still maintaining good localization performance.

3.2. Prioritized Back Matching

Determining the correct model image only provides rough camera location and additional work is needed to estimate the precise camera pose. To reap the computational benefits of subsampling, we thus modify our framework slightly. We use forward matching with a subset of N_F query features in order to identify likely model images. We then perform back matching within candidate images in order to expand the set of matches used for fine camera pose estimation. This back matching is carried out using a greedy prioritized search over images ranked by votes and further exploits co-visibility information encoded in the SfM model to find additional distinctive matches that were not identified during the forward (sub-sampled) matching.

Algorithm 3 describes our back-matching approach. Given the forward matches found using Algorithm 2, we select the most voted model image c^* and back-match all of its views against the query image using the standard 1-ratio test with threshold τ. The correspondences M_{c^*} found are added to the pool of back matched pairs $\mathcal{M}_{\mathcal{B}}$ used for the fine pose estimation. These back matches are also treated as votes. We use the SfM model's camera-point visibility graph G, to cast votes for other images that observe the same views as in M_{c^*}. These new votes increase the likelihood that neighboring images are selected for subsequent rounds of back-matching. To avoid introducing noise into the voting process, we only allow a back-matched image to cast votes if it depicts the same location (i.e., returns 12 or more matches). The algorithm terminates when $\mathcal{M}_{\mathcal{B}}$ is large enough to guarantee a good camera localization, or a certain total number of images have been back-matched.

Algorithm 4 Camera Localization

INPUT: Query features Q, Model features $\mathcal{V}$, co-visibility graph G, camera clusters $\mathcal{C}$, NN search depth k, ratio test threshold τ, match count thresholds N_F, N_B, projection error threshold ϵ
$\mathcal{M}$ = GLOBAL-FORWARD-MATCH($Q, \mathcal{V}, k, N_F, \tau$)
$\mathcal{M}_{\mathcal{F}}$ = CLUSTER-WISE-RATIO-TEST($\mathcal{M}, \mathcal{C}, \tau$)
$\mathcal{M}_{\mathcal{B}}$ = PRIORITY-BACK-MATCH($\mathcal{M}_{\mathcal{F}}, N_B, G, \tau$)
I_P, δ = ROBUSTFITTING($\mathcal{M}_{\mathcal{B}}, \epsilon$)
if $|\delta \leq \epsilon| \geq 12$ **then**
  return Camera Pose I_P
else
  return Error - Pose not found
end if

4. Benchmark Evaluation

We evaluated our approach (Algorithm 4) on three different datasets: Eng-Quad, Dubrovnik, and Rome. Rome is a large dataset of 15,179 training and 1,000 test images. Dubrovnik is a popular 6,044 training and 800 test image dataset whose SfM model is roughly aligned to geographic coordinates, allowing for quantitative metric evaluation. While Eng-Quad has fewer images, it is perhaps the most challenging due to the presence of strongly repeated structures in the modern architectural designs it depicts. When using P3P, we used EXIF metadata for Eng-Quad test images and ground-truth focal lengths from the SfM models for Dubrovnik and Rome. We also briefly analyze results on the city-wide SF-0 dataset [12].

Dubrovnik correctness: After carefully analyzing the original models provided for Dubrovnik, we found that *the test set ground truth was often wrong*, with extremely large focal lengths and misaligned 2D-3D correspondences. This in turn resulted in large errors in camera location and poor alignment between projection of 3D points and the corresponding 2D features. These problems are evident in results published elsewhere. For example, [23] report better results using P4Pf [4] than using P3P with the given "true" focal lengths. This is contrary to what should be expected: know-

Dubrovnik (Original) - 800 test images

Method	#images	#inliers	ratio	error [m] Q1	median	Q3	<18.3m	>400m	time [s]
Sattler [21]	783.9	≤100	-	0.4	1.4	5.9	685	16	0.31
Sattler [22]	795.5	≤200	-	0.4	1.4	5.3	704	9	0.25
Zeisl [36]	796	-	-	0.19	0.56	2.09	744	7	3.78
Svarm [28]	798	-	-	-	0.56	-	771	3	5.06
Ours (P3P)	800	358	0.65	1.09	7.92	27.76	550	10	0.62
Ours (P4Pf)	800	468	0.79	0.55	1.64	6.02	694	15	0.62

Eng-Quad - 520 test images

Method	#images	#inliers	ratio	error [m] Q1	median	Q3	time [s]
Sattler [21]	402	43	0.49	0.61	2.01	7.51	1.52
Sattler [22]	457	43	0.58	0.46	1.93	7.62	0.32
Ours (P3P)	509	112	0.66	0.33	0.67	1.47	0.69
Ours (P4Pf)	504	115	0.68	0.65	1.88	5.76	0.85

Dubrovnik (Corrected) - 777 test images

Method	#images	#inliers	ratio	error [m] Q1	median	Q3	<18.3m	>400m	time [s]
Sattler [21]	771	70	0.72	0.57	1.44	4.61	707	1	2.58
Sattler [22]	775	69	0.74	0.59	1.58	4.91	705	4	0.75
Ours (P3P)	777	591	0.88	0.33	0.66	1.60	759	1	0.48
Ours (P4Pf)	776	589	0.88	0.47	1.11	3.32	720	3	0.48

Rome - 1000 test images

Method	#images	#inliers	ratio	time [s]
P2F [13]	924	-	-	0.87
Sattler [21]	976.90	≤ 100	-	0.29
Sattler [22]	991	≤ 200	-	0.28
Ours (P3P)	999	281	0.54	0.75
Ours (P4Pf)	1000	458	0.83	0.74

Table 4. Quantitative results of our method compared to related methods for camera pose estimation.

ing the ground-truth focal length (P3P) should outperform joint estimation of pose and focal length (P4Pf). Examples are shown in the supplementary material.

For this reason, we rebuilt a new version of the Dubrovnik "ground-truth" model using the same set of keypoints provided for the original dataset and the excellent SfM package COLMAP [25]. We aligned the new model with the original one using a RANSAC-based Procrustes analysis so that the scale is approximately metric. After alignment, only 3853 of the recovered 6844 images were located within 3 meters from their original position in the model, further validating our concerns. Our reconstruction provided ground-truth for 777 of the 800 query images.

Anytime performance: The runtime of our algorithm for camera localization depends on two parameters: N_F and N_B. Setting these parameters trades off localization accuracy with faster execution times. Figure 3 shows the influence of these variables using the Eng-Quad dataset. We benchmarked forward matching times by randomly sampling query features until a desired number N_F pass the global ratio under fixed values for N_B. Similarly, we fixed N_F and evaluated different values for N_B. In both cases, the range of values tested vary from 50 up to 500 matched features. Figure 3 shows the number of registered images

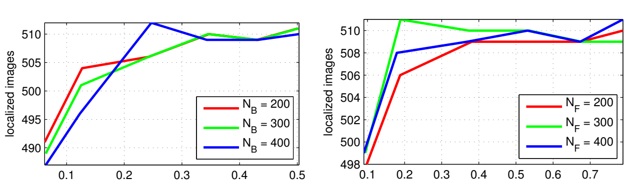

Figure 3. Anytime performance: querying a small number of features dramatically reduces runtime without a major loss in localization performance. The forward subsampling does not affect rough localization significantly and stabilizes after 0.3 seconds (left) regardless of the N_B value. Similarly, localization quickly plateaus after 0.2 seconds at back matching time for different values of N_F (right).

under these different configurations, and the time spent to achieve such a level of performance.

Experimental details: We tested our localization pipeline using the following settings: for each dataset, we built a global kd-tree index using all model view descriptors. We request $k = 5$ nearest neighbors and check 128 leaves. We set $\tau = 0.7$ across all of our ratio tests. We set $N_F = 200$ and $N_B = 200$ to provide a good balance between camera localization and execution time. Algorithm 3 stops after 20 back-matched images, which is a generous setting in these datasets (in most cases N_B is achieved in less than 5 loops). Experiments were performed using a single thread on an Intel i7-5930 CPU at 3.50GHz. We used the implementation of [21, 22] in Eng-Quad and the re-bundled Dubrovnik comparisons, running a single thread on an Intel i7-3770 CPU at 3.40GHz. We used a generic vocabulary tree and default parameters: $N_t = 100$ for [21] and $N_{3D} = 200$ for [22]. Unfortunately, implementations of [36, 28] were not available.

Camera Localization: We successfully localized all images in Dubrovnik, except one image in the corrected version using P4Pf. We achieved the smallest localization errors for all quartiles, and reported more images within the $18.3m$ threshold and fewer beyond the $400m$ mark. Despite finding a substantial higher number of inliers, our method yielded larger average errors with respect to the original Dubrovnik model due to its underlying defects in the ground-truth. [28] and [36] (after RANSAC), who use a shape-voting approximation to the rough image location rather than the traditional *match-and-RANSAC* pipeline, report smaller localization errors but at the cost of longer runtimes. Finally, we successfully localized all query images in the Rome dataset using P4Pf. Rome also suffers the similar inaccuracies as Dubrovnik, which resulted in the loss of one test image using P3P.

The benefits of our approach are more pronounced on Eng-Quad, due to its difficult characteristics. We localized

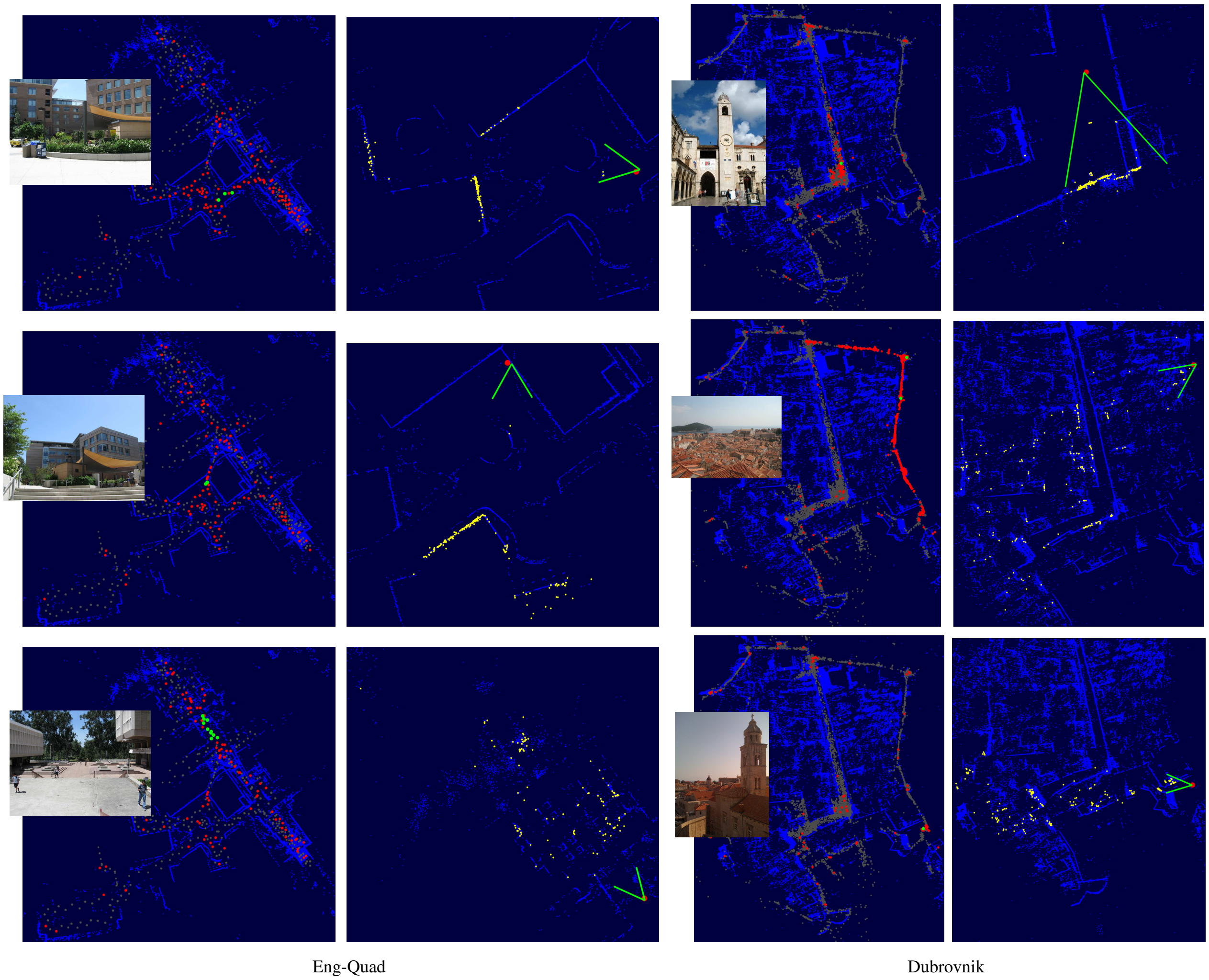

Figure 4. Qualitative localization results. Left column: model images (gray) are highlighted (red) if they received one vote from Algorithm 2. Algorithm 3 quickly recognizes model images from the same general area of camera pose space (green). Right column: correspondences used by the PnP solver (yellow), along the localized camera (green). Ground truth camera position is indicated with a red circle. Best viewed zoomed in color.

more than 100 and 50 additional cameras w.r.t. [21, 22] respectively, improving all localization errors except the first quartile using P4Pf. We obtain faster runtimes than [13, 36] while being competitive with those of [21, 22]. Notably, our approach adapts better to the more difficult Eng-Quad dataset, spending more time retrieving images with sufficient correspondences. On the other hand, we quickly recognize landmarks in Dubrovnik with the first or second top ranked images, quickly retrieving sufficient putative correspondence and yielding faster localization times.

Location Retrieval: We obtained an asymptotic recall of 66.63% on the SF-0 dataset using the protocol of [12]. At 95% precision, the recall drops to 52.30% using the *effective inlier count* of [19], falling below performance of other methods [6, 19, 1, 36] for location recognition. For this test we used less stringent parameters: $k = 7$, $N_F = 500$, $N_B = 300$, and back matched up to 50 images. We expect tuning these parameters and utilizing re-ranking heuristics exploited by other methods to provide a better approach for such location retrieval problems.

5. Conclusion

Alternatives to large-scale image localization have focused on reducing the density of the search space to quickly find discriminative correspondences. Here we have shown that retrieving multiple global nearest neighbors and filtering them using approximations to the ratio test can quickly identify candidate regions of pose space. Such regions can be further refined by back matching to yield state-of-the-art results in camera localization, even for datasets with challenging global repeated structure.

References

[1] R. Arandjelović and A. Zisserman. Dislocation: Scalable descriptor distinctiveness for location recognition. In *Asian Conference on Computer Vision*, pages 188–204. Springer, 2014. 1, 8

[2] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *Computer Vision–ECCV 2012*, pages 517–530. Springer, 2012. 5

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. 1

[4] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 6

[5] S. Cao and N. Snavely. Graph-based discriminative learning for location recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 700–707, 2013. 5

[6] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, et al. City-scale landmark identification on mobile devices. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 737–744. IEEE, 2011. 8

[7] T. Dekel, S. Oron, M. Rubinstein, S. Avidan, and W. T. Freeman. Best-buddies similarity for robust template matching. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2021–2029. IEEE, 2015. 4

[8] R. Díaz, S. Hallman, and C. C. Fowlkes. Detecting dynamic objects with multi-view background subtraction. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 273–280. IEEE, 2013. 1

[9] R. Díaz, M. Lee, J. Schubert, and C. C. Fowlkes. Lifting gis maps into strong geometric context for scene understanding. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9, March 2016. 3

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1

[11] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *CVPR*, 2011. 3

[12] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide Pose Estimation using 3D Point Clouds. In *ECCV*, 2012. 1, 5, 6, 8

[13] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *European Conference on Computer Vision*, pages 791–804. Springer, 2010. 1, 3, 5, 7, 8

[14] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, Nov. 2004. 1

[15] L. Moisan, P. Moulon, and P. Monasse. Automatic homographic registration of a pair of images, with a contrario elimination of outliers. *Image Processing On Line*, 2:56–73, 2012. 1

[16] P. Moulon, P. Monasse, and R. Marlet. Adaptive structure from motion with a contrario model estimation. In *ACCV*. 2012. 1

[17] D. M. Mount and S. Arya. Ann: library for approximate nearest neighbour searching. 1998. 1

[18] M. Muja and D. G. Lowe. Flann, fast library for approximate nearest neighbors. In *International Conference on Computer Vision Theory and Applications (VISAPP09)*. INSTICC Press, 2009. 1, 3

[19] T. Sattler, M. Havlena, F. Radenovic, K. Schindler, and M. Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition. In *ICCV*, 2015. 5, 8

[20] T. Sattler, M. Havlena, K. Schindler, and M. Pollefeys. Large-scale location recognition and the geometric burstiness problem. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 5

[21] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2D-to-3D matching. *ICCV*, 2011. 1, 7, 8

[22] T. Sattler, B. Leibe, and L. Kobbelt. Improving image-based localization by active correspondence search. In *European Conference on Computer Vision*, pages 752–765. Springer, 2012. 2, 7, 8

[23] T. Sattler, C. Sweeney, and M. Pollefeys. On sampling focal length values to solve the absolute pose problem. In *European Conference on Computer Vision*, pages 828–843. Springer, 2014. 6

[24] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007. 5

[25] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 3, 7

[26] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. 3

[27] N. Snavely, I. Simon, M. Goesele, R. Szeliski, and S. M. Seitz. Scene Reconstruction and Visualization From Community Photo Collections. *Proceedings of the IEEE*, 98(8):1370–1390, Aug. 2010. 1

[28] L. Svarm, O. Enqvist, F. Kahl, and M. Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016. 7

[29] L. Svarm, O. Enqvist, M. Oskarsson, and F. Kahl. Accurate localization and pose estimation for large 3d models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 532–539, 2014. 2, 5

[30] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 5

[31] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi. Visual place recognition with repetitive structures. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 1

[32] P. H. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. 1

[33] S. Wang, S. Fidler, and R. Urtasun. Holistic 3d scene understanding from a single geo-tagged image. 2015. 1

[34] C. Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 127–134. IEEE, 2013. 1

[35] A. R. Zamir and M. Shah. Image geo-localization based on multiple nearest neighbor feature matching using generalized graphs. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1546–1558, 2014. 4

[36] B. Zeisl, T. Sattler, and M. Pollefeys. Camera pose voting for large-scale image-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2704–2712, 2015. 2, 5, 7, 8