Efficient and Effective Cascade Correspondence Search for Large-scale Image-based Localization

Anonymous submission

Abstract

We propose a new method of 2D-3D correspondence search, namely Cascade Correspondence Search (CCS), for image-based localization problem. Our method integrates an efficient and compact indexing via cascade manner and an effective pose estimation method. Our indexing leverages efficient Hamming distance computation, and needs only compressed descriptors of SIFT. By multi-steps search (coarse search, refined search, and precise search), our method can quickly find out top nearest neighbors. To detect better inliers/outliers, we decide to shift the task of outliers rejection from matching step to pose estimation step. We first accept more correspondences from cascade search with relaxed test ratio, and we then propose a RANSAC-based pose estimation that can handle effectively these correspondences to expose more inliers. Our method achieves competitive accuracy, yet requires lower requirement of memory and computation than most of earlier works. Specifically, we achieve state-of-the-art performance on two of four considered benchmark datasets of image-based localization.

Introduction

Inferring camera pose from an image in large scale urban scene is a fundamental problem and important in many applications: autonomous navigation, location recognition, augmented reality, and high-secure application (GPS agnostic). As mobile applications become popular nowadays, developing an on-device localization system is on the trend (Arth et al. 2009; Lynen et al. 2015). Thus, it is advantageous to research a large-scale localization method, which is fast, memory efficient, and accurate.

Given a 3D model of the scene, the camera pose (e.g. 3D position and orientation) can be inferred from 2D-3D matches between 2D features in the image and 3D points in the model by applying n-point-pose solver inside a RANSAC loop (Fischler and Bolles 1981). The scene model was reconstructed from a dataset of images of previously seen locations, e.g. by recent advances in Structure-from-Motion (SfM) technique (Snavely, Seitz, and Szeliski 2006). The 2D-3D correspondences are established through approximate nearest neighbor search between descriptors of 2D local features and 3D points. To distinguish between inliers or outliers correspondences is an ill-posed problem. Although Lowe's ratio test (Lowe 2004) is widely used, it is too restrictive, and fails in many cases; especially, true in urban scenes which often exist many repetitive features (Sattler, Leibe, and Kobbelt 2012; Li et al. 2012). Some state-of-the-art methods therefore use rather complicated bi-directional matching procedure that combines 2D-3D and 3D-2D search. This approach first aims to reject as much as possible outliers in the matching step, and then performs geometric verification, e.g. by RANSAC, on a small set of correspondences. The reason is that the runtime of RANSAC increase exponentially with the percentage of false matches. In contrast, the other approach is to use lower ratio test threshold to accept larger amount of matches (both inliers and outliers), and filter incorrect correspondences based on geometric consistency (Svarm et al. 2014; Zeisl, Sattler, and Pollefeys 2015) rather than matching step. Clearly, the use of more matches in geometric solver can potentially obtain higher number of inliers. However, handling such a number of matches is computationally expensive. Therefore, current geometric solvers take seconds on powerful workstations for only a single query. Moreover, if considering also its processing time of 2D-3D correspondence search on, e.g. SIFT descriptors, its total latency is much longer. In addition, a SIFT descriptor requires at least 128 bytes, which is inefficient as needing to store a large scale 3D model on mobile devices.

In this work, we address the localization problem by relaxing the test ratio to have more tentative matches (multiple matches per query descriptor), and we propose a fast solution to handle it for pose estimation. Before of that, in order to seek multiple matches efficiently, we propose a new 2D-3D matching method. Our work tries to improve localization performance on two both aspects: 2D-3D matching and pose estimation. Our goal is to develop a practical localization method that achieves better trade-off among complexity, memory and accuracy. We make three main contributions: Firstly, we propose an efficient 2D-3D correspondence search method based on coarse-to-fine search scheme. Secondly, we propose a RANSAC based solution detect inliers from a large number of matches in low-computational complexity. Furthermore, we apply a simple prioritization scheme, but significantly accelerate the 2D-3D matching and pose estimation. Finally, we demonstrate our method on benchmark datasets of millions 3D points. The results show our method is faster than most of previous works, and achieves the competitive accuracy. To the best of our knowl-

edge, we are the first work confirms that using compressed descriptors can still potentially achieves state-of-the-art performance on benchmark datasets of large scale image-based localization.

Related Works

Early works of image-based localization can be divided into two categories: retrieval based approach and 3D model based approach (or direct search approach). Retrieval based methods (Zhang and Kosecka 2006; Zamir and Shah 2010; Chen et al. 2011; Agarwal, Burgard, and Spinello 2015) are closely related to image retrieval by matching query features against geo-tagged database images to find a set of similar ones. The query pose can be inferred from those references. This approach highly depends on the performance of image retrieval and does not leverage the benefit of 3D scene structure.

On the other hand, the model based approach directly performs 2D-3D matching between 2D features of query image and 3D points of 3D model. The 3D model is a point cloud reconstructed from given image dataset by SfM methods. This approach achieves better results than the first one as it imposes stronger geometric constraints. Preferably, it tells more information about 3D structure of the scene. The camera pose can then be computed from 2D-3D correspondences by RANSAC within n-pose solver inside. (Irschara et al. 2009) first perform image retrieval and then computes 2D-3D matches between 2D query features and 3D points visible in top retrieved images. Synthetic views of 3D points are generated to improve image registration. (Arth et al. 2009) divide the scene into multiple cells, use potentially visible set on cells to reduce the number of searched 3D points. However, their exhaustive feature matching is computationally expensive. As a result, the work is confined to small workspaces and requires user's input or external sensors (GPS, WiFi,...) to determine the cell candidates. (Li, Snavely, and Huttenlocher 2010) compress the 3D model and prioritize 3D points (given the prior knowledge from visibility graph) in 3D-2D correspondence search that allows the "common" views to be localized quickly. (Sattler, Leibe, and Kobbelt 2011) propose the efficient prioritization scheme which is able to early stop the 2D-3D direct search. (Ventura et al. 2014) and (Middelberg et al. 2014) employ the client-server architectures. Pose tracking is performed on devices to reduce latency, but sometimes the pose needs to be corrected by matching to global model to avoid the drift. While (Ventura et al. 2014) keep a part of global model locally to obtain fast tracking, (Middelberg et al. 2014) construct their own local map of the environment and using global results received from external server to align this map. (Lim et al. 2012) use Harris corner detectors and extract two binary features for tracking and 2D-3D matching. It avoids wasting computation by performing matching over only a small batch of tracked keypoints. (Lynen et al. 2015) implemented a pose estimation and tracking on device. It uses Inverted Multi-Index (IMI) (Babenko and Lempitsky 2012) for indexing. To reduce the computation and make it processable on device, they compress their 3D models by a similar method of (Li, Snavely, and Huttenlocher 2010).

To achieve better accuracy, some state-of-the-art methods propose the combination searches from 2D image features to 3D points and vice versa, which can recover some rejected positive matches due to Lowe's ratio test (Sattler, Leibe, and Kobbelt 2012) or removing outliers more efficiently because a large number created from the relaxed threshold (Li et al. 2012). Co-visibility constraints are also applied to filter out outliers in these works. All aforementioned methods make efforts to find reliable matches and avoid to generate false correspondences. Using bi-directional method often takes more time. A recent trend in localization shifts the task of finding correct correspondences from matching step to pose estimation step, by leverage geometric cues. (Svarm et al. 2014) propose an outliers filter with assumption of a known direction of gravitational vector and the rough estimate of the ground plane in 3D model. Consequently, the pose estimation problem can be casted into a 2D registration problem. Follow the same setup as (Svarm et al. 2014), (Zeisl, Sattler, and Pollefeys 2015) propose a filtering strategy based on Hough voting in linear complexity. In order to accelerate the method, they exploit the local feature geometry based verification, e.g. the viewing direction constraints, or the scale and orientation of 2D local features to early reject false matches before voting.

Proposed Method

Our proposed method, namely Cascade Correspondence Search (CCS), consists of two parts: (i) an efficient 2D-3D matching is to seek top ranked list of nearest neighbors in cascade manner and (ii) a fast and effective RANSAC enables to exploit inliers from a large number of correspondences that helps to boost the accuracy.

Cascade search for 2D-3D matching

Our method leverages the efficient computation of Hamming distance. We follow the Pigeonhole Principle on binary code (Norouzi, Punjani, and Fleet 2012) to further accelerate the search. The key idea is on following: A binary code $\mathbf{h}$, comprising d bits, is partitioned into m disjoint sub-binary vectors, $\mathbf{h}^{(1)}, ..., \mathbf{h}^{(m)}$, each has $\lfloor \frac{d}{m} \rfloor$ bits. For convenience, we assume that d is divisible by m. When two binary codes $\mathbf{h}$ and $\mathbf{g}$ differ at most r bits, then, at least, one of m sub-binary vectors, for example $\{\mathbf{h}^{(k)}, \mathbf{g}^{(k)}\}, 1 \leq k \leq m$, must differ at most $\lfloor \frac{r}{m} \rfloor$ bits. Formally, it can be written:

$$\|\mathbf{h} - \mathbf{g}\|_H \leq r \Rightarrow \exists k \in [1, m] : \left\|\mathbf{h}^{(k)} - \mathbf{g}^{(k)}\right\|_H \leq \left\lfloor \frac{r}{m} \right\rfloor \tag{1}$$

where $\|.\|_H$ is the Hamming distance. We apply it into our context of 2D-3D matching and propose additions and improvements:

- In our work, since the 3D models are built off-line and SIFT descriptors for 3D points are available during off-line processing, we propose to train a data-dependent hash function to improve matching accuracy.
- We use a single hash function to mapping from 128 bytes SIFT to d bits binary vector. Splitting the long d bits code

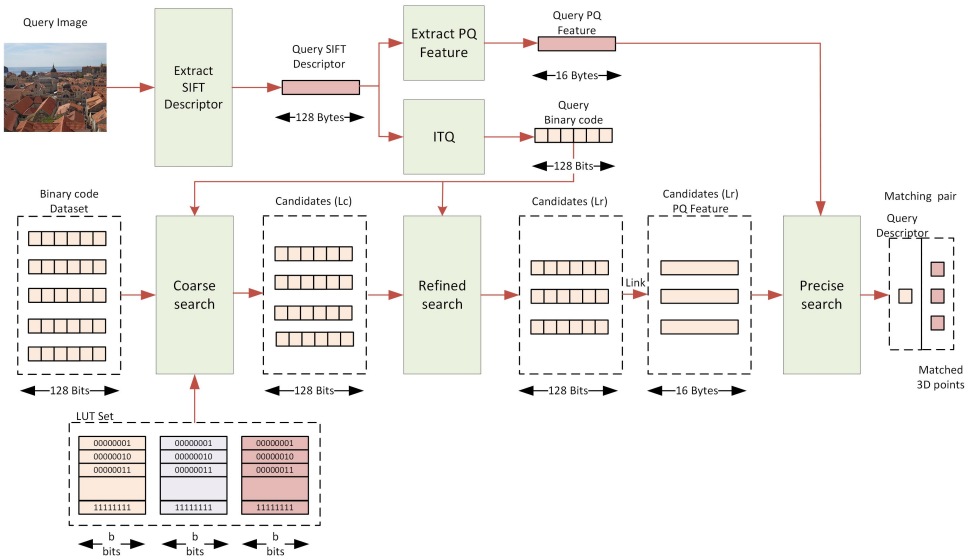

Figure 1: The pipeline of our cascade search per one query descriptor. It consists of three main steps: coarse search (16-bit LUT), refined search (128-bit) and precise search (16-byte).

into m short codes of b bits to construct m lookup tables (LUT) for coarse search and use full d bit vector for refined search.

- We add the precise search layer to the hashing scheme, and propose to use Product Quantization (PQ) (Jégou, Douze, and Schmid 2011), a fast and memory efficient method for precise search. Consequently, our work combines hashing and PQ in a single pipeline to leverage their strengths: Binary hash code enables fast indexing via Hamming distance-based comparison, while PQ achieves better matching accuracy. They are both compressed descriptors.
- We propose to use prioritizing scheme based on the number of candidates resulted from coarse search layer. Results show that this scheme significantly accelerates about $\sim 10\times$ with only slightly less accuracy. The valuation is demonstrated on Dubrovnik dataset in Table 2.

The pipeline of our proposed 2D-3D matching method is shown in Fig. 1. The method includes three main steps: coarse search, refined search and precise search. Two first steps are to quickly filter out a shorter list of candidates from N_p 3D points' descriptors, the last step to precisely determine the top ranked list. Let $d = 128$ be the feature dimension of SIFT descriptors. Given a 3D model and its points' descriptors, each descriptor $\mathbf{d} \in \mathbb{R}^{d\times 1}$ are pre-mapped into binary vector $\mathbf{h}$ in Hamming space $\mathbb{B}^{d\times 1}$: $\mathbf{h} = \text{sign}(\mathbf{Wd})$, where $\mathbf{W}$ is the transformation matrix, which can be learned via objective minimization:

$$\arg\min_{\mathbf{W},\mathbf{H}} \|\mathbf{H} - \mathbf{WD}\|_F^2 \tag{2}$$

where $\|.\|_F$ is Frobenius norm. $\mathbf{D}, \mathbf{H}$ are matrices of all point descriptors of 3D model (one descriptor per matrix's column) and its binarized code after transformation respectively. We solve the optimization problem by ITQ (Gong and Lazebnik 2011). Given learned hash function, all descriptors of the model are mapped into binary vectors and we store those vectors instead of original SIFT descriptors.

Method(m,b)	$\mathbf{L}_C$	**LUT size**
ITQ(4,32)	-	$4 \times 2^{32} \times 4$ = 64GB
ITQ(8,16)	$\sim$ **5K**	$8 \times 2^{16} \times 2$ **= 1048KB**
ITQ(16,8)	$\sim 60K$	16×2^{8} = 4096B
LSH(8,16)	$\sim 20K$	$8 \times 2^{16} \times 2$ = 1048KB

Table 1: The trade-off between the number of candidates $\mathbf{L}_C$ and the size of LUT experimented on Dubrovnik dataset.

Coarse search: We follow the principle (1) to create a LUT based data structure for fast search. We split binary vector $\mathbf{h}$ into m sub-vectors $\{\mathbf{h}^{(k)}\}, k \in [1:m]$ of b bits ($m * b = d$). In our work, we only select candidates differ at most $r = m - 1$ bits from the query ($\lfloor \frac{r}{m} \rfloor = 0$). In other words, a candidate's binary vector is potentially matched to the query's iff at least one of their sub-vectors are exactly the same. For training, we create m LUTs, where $\mathbf{LUT}^{(k)}$ for the sub-vector $\mathbf{h}^{(k)}$, and each LUT comprises of $K_b = 2^b$ buckets. One bucket links to a point-id list of 3D point that are assigned to buckets according to their binary sub-vectors. For searching, a query descriptor is first mapped into Hamming space, and was divided into m sub-binary vectors as above. And then looking up into the $\mathbf{LUT}^{(k)}$ to find a certain bucket that matches with binary code of $\mathbf{h}^{(k)}$. This results the point-id list $\mathbf{L}^{(k)}$:

$$\mathbf{L}^{(k)} = \mathbf{LUT}^{(k)}(\mathbf{h}^{(k)}), k = 1, .., m \tag{3}$$

Then, merging m point-id list to have the final list of

coarse search $\mathbf{L}_C = [\mathbf{L}^{(1)}, ..., \mathbf{L}^{(m)}]$. By using LUT, the search complexity of $\mathbf{h}^{(k)}$ is constant $O(1)$. This step results a short list $\mathbf{L}_C$ contains $|\mathbf{L}_C|$ candidates for the next search. It is important to choose appropriate values of m and b to trade-off between the memory requirement of LUT and computation time (which depends on the length of $\mathbf{L}_C$ that requires Hamming distance refining). As shown in Table 1, we map descriptors to binary codes by using ITQ with different settings, and also replace it with LSH (Charikar 2002). ITQ($m = 4$,$b = 32$) is impractical due to over-large size requirement of LUTs. ITQ($m = 16$,$b = 8$) results in too many candidates, which slows down the refined search. ITQ($m = 8$,$b = 16$) is the best option, results the short list, and requires a small amount of LUT memory (excluding the overhead memory of descriptors indexing). Using LSH results a longer list ($\sim 4\times$ of ITQ) of candidates, which means that learning hash mapping from data points by ITQ is more efficient than a random method LSH in our case. This is consistent with our experiments conducted later in Table 4.

Refined search: In this step, we use full d-bit code $\mathbf{h}$ to refine $\mathbf{L}_C$ list to pick out a shorter list $\mathbf{L}_R$ ($\mathbf{L}_R \leq 50$). First, we compute exhaustively the Hamming distance between the d-bit code of query to that of $\mathbf{L}_C$ candidates. Then, candidates are re-ranked according to these distances. Computing Hamming distance is efficient because we can leverage low-level machine instructions (XOR, POPCNT). Computing Hamming distance of two 128-bit vectors is significantly faster ($\geq 30\times$) than Euclidean distance of SIFT vectors, and accelerate ($\geq 4\times$) ADC (Jégou, Douze, and Schmid 2011) on our machine. Furthermore, Hamming distance of d-bit code has limit range of [0, 128], which allows us to build the online LUT during the refined search. By this way, selecting top candidates $\mathbf{L}_R$ search is accelerated. However, the limited range prevents us to precisely rank candidates. That leads to the last step of our pipeline.

Precise search: The purpose of precise search is to get $\mathbf{L}_R$ ranked better, so that we can choose the best candidate or remove outliers of matches before applying geometric verification. Furthermore, we can consider their order as an useful prior information. It plays an important role to reduce the complexity of pose estimation (discussed in Section of Geometric Verification). The approximated Euclidean distance by ADC of PQ (Jégou, Douze, and Schmid 2011) is used. The match between a query feature and 3D point is established if the distance ratio from the query to the first and second candidates passes the ratio test ν_h (Lowe 2004); otherwise, rejected as outliers. The sub-quantizers of PQ are trained once from an independent dataset, SIFT1M (Lowe 2004), and used in all experiments. In this step, we need to store PQ codes in addition to hashing code of two previous steps.

Prioritization: Finding all matches between 2D features and 3D points to infer camera pose is expensive because the query image can contain thousands of features. In practice, the method can early stop once found a sufficient number of matches (Sattler, Leibe, and Kobbelt 2011). Therefore, we perform prioritized search on descriptors of 2D image as follows: given a query descriptor, the coarse search returned the point-id list $\mathbf{L}_C$. We first continue refined and precise search with those query features having shorter list $|\mathbf{L}_C|$. A correspondence is established if the nearest candidate passes the ratio test with threshold ν_h on precise search. We stop the search once $N_{early} = 100$ correspondences have been found.

procedure CCS-RANSAC($\mathbf{Q}$, $\mathbf{P}$, $\{i_k\}_{k=1}^{N_h}$)
  $\epsilon \leftarrow p(1|H_g) = \frac{s}{N_h}$
  $\delta \leftarrow p(1|H_b) = 0.01$
  $A \leftarrow A(\delta, \epsilon)$
  $\mu \leftarrow \frac{1}{\epsilon^m * (1-\frac{1}{A})}$
  $nr \leftarrow 0$ ▷ the number of rejected times
  $iter \leftarrow 0$ ▷ the number of iterations
  while $iter \leq \mu$ **do**
    $iter \leftarrow iter + 1$
    I. Hypothesis
    Select a random sample of minimum size s from hypothesis set $\{\mathbf{q}_{i_k}, \mathbf{p}_{i_k 1}\}, i_k \in [1 : N_q], k = 1 : N_h$.
    Estimate model parameters θ fitting the sample.
    II. Pre-verification
    $k = 1$
    $\lambda_0 = 1$
    while $k <= N_h$ **do**
      Let $\rho_1 \leftarrow \rho_1(\theta, \{\mathbf{q}_i, \mathbf{p}_{i1}\})$ ▷ 0 or 1
      $\lambda_k \leftarrow \lambda_{k-1} * (\rho_1 * \frac{\delta}{\epsilon} + (1 - \rho_1) * \frac{1-\delta}{1-\epsilon})$
      if $\lambda_k > A$ **then**
        `bad_model = true` ▷ Reject sample
        `break`
      else
        $k \leftarrow k + 1$
      end if
    end while
    if `bad_model` **then**
      $nr \leftarrow nr + 1$
      $\hat{\delta} \leftarrow \delta * \frac{nr-1}{nr} + \epsilon * nr$ ▷ Re-estimate δ
      if $|\delta - \hat{\delta}| > 0.05$ **then**
        $\delta \leftarrow \hat{\delta}$ ▷ Update δ
        $A \leftarrow A(\delta, \epsilon)$ ▷ Update A
      end if
      `continue`
    end if
    III. Verification
    Compute cost $C \leftarrow \sum_i \rho_M(\theta, \{\mathbf{q}_i, \mathbf{P}_i\})$
    if $C^* \leq C$ **then**
      $C^* \leftarrow C, \theta^* \leftarrow \theta$ ▷ Update good model
      $\epsilon \leftarrow C^* * \frac{l_R}{N_q}$ ▷ Update ϵ
      $A \leftarrow A(\delta, \epsilon)$ ▷ Update A
      $\mu \leftarrow \frac{1}{\epsilon^m * (1-\frac{1}{A})}$ ▷ Update μ
    end if
  end while
end procedure

Figure 2: The algorithm of our proposed RANSAC.

Geometric verification

After 2D-3D matching, one query descriptor has one 3D point correspondence. Those correspondences (one-one matches) is then verified by RANSAC within 6-DLT algorithm inside. Empirically, we found two things: (i) ratio test ν_h tends to reject many good matches (ii) good can-

didates are not always highest-ranked in the list $\mathbf{L}_R$. It is probably due to repetitive features in buildings, that is a common issue of localization in urban environment (Sattler, Leibe, and Kobbelt 2016; Zeisl, Sattler, and Pollefeys 2015). Therefore, relaxing the threshold to accept more matches, and filtering wrong matches by using geometric verification seems a potential solution. Recent works (Svarm et al. 2014; Zeisl, Sattler, and Pollefeys 2015) aim to approach this way, but their geometric solvers are too slow for practical applications.

To address this issue, we propose a fast and effective RANSAC as follows: First, we relax the threshold $\nu > \nu_h$ to accept more matches and keep one-many candidates per query descriptor. We compute one-many matchings: given one query feature, we accept M top candidates in $\mathbf{L}_R$ list when $d_0/d_1 < \nu$. d_0 and d_1 are the first and second smallest distances of the query to $\mathbf{L}_R$ candidates. However, processing all these matchings leads to an exponential increase in computational time in RANSAC due to very low rate of inliers. We avoid this issue by considering its subset to generate hypothesis. As consequence, we propose two different sets of matchings in the hypotheses and verification stages of RANSAC. The first set contains the one-one (1-1) matchings that pass a ratio test with threshold ν_h. The second set with candidates found by relaxed threshold, and contains one-many (1-M) matchings as mentioned above. We propose to use the first set to generate hypotheses and the second set for verification. We found that using relaxed threshold and 1-M matchings in verification can increase the number of inliers, leading to the accuracy improvement. We further speed up our method by using the pre-verification step like (Chum and Matas 2008) in the middle, to quickly reject "bad" samples before performing verification.

For detail, let $\mathbf{Q} = \{\mathbf{q}_i\}$, and $\mathbf{P} = \{\mathbf{p}_{ij}\}$, $i = 1 : N_q$, where N_q is the number of query descriptors found their matchings. $\mathbf{q}_i$ is the 2D coordinate of i-th query and $\mathbf{p}_{ij}$ is the 3D coordinate of its corresponding j-th ranked 3D point. Those matchings found by 2D-3D matching with the threshold ν. One query $\mathbf{q}_i$ has many candidates (1-M) $\mathbf{P}_i = \{\mathbf{p}_{ij}\}, j = 1 : |\mathbf{P}_i|$, where $|\mathbf{P}_i|$ is the number of point candidates matched with each query $\mathbf{q}_i$: $|\mathbf{P}_i| \leq M$ (verification set). Without the loss of generality, $\mathbf{p}_{ij}$ were sorted in ascendant of ADC distance to the query $\mathbf{q}_i$: $d(\mathbf{q}_i, \mathbf{p}_{ij})$. And among N_q 1-M matchings, assuming that there are N_h 1-1 matchings $\{\mathbf{q}_{i_k}, \mathbf{p}_{i_k 1}\}$ (hypothesis set), $i_k \in [1 : N_q]$, $k = 1 : N_h$ found by the threshold ν_h. The detail of proposed algorithm is presented in Fig. 2. ϵ indicates the probability $p(1|H_g)$ that any randomly match is consistent with a "good" model. $s = 6$ indicates the minimum size of a sample can estimate the model parameters θ by using 6-DLT algorithm. δ indicates the probability $p(1|H_b)$ of a match being consistent with a "bad" model. The probability of rejecting a "good" sample ($\alpha = 1/A$). Here, H_g: the hypothesis that the model is "good", and H_b: the alternative hypothesis that the model is "bad". $\rho_1(\theta, .)$ and $\rho_M(\theta, .) \in \{0, 1\}$ are functions to evaluate whether 1-1 matches or 1-M matches are consistent with the model or not. $\rho_M(\theta, \{\mathbf{q}_i, \mathbf{p}_{ij}\}) = 1$ if $\exists k$, $\rho_1(\theta, \{\mathbf{q}_i, \mathbf{p}_{ik}\}) = 1$. C and θ are the cost (or the number of inliers) and model parameters respectively. The adaptive decision threshold $A = A(\delta, \epsilon)$ is computed similarly as (Chum and Matas 2008).

Figure 3: The registration rate and inliers ratio according to the number of candidates of $\mathbf{L}_R$.

Experimental Results

We evaluate our method on four datasets: Dubrovnik (Li, Snavely, and Huttenlocher 2010), Rome (Li, Snavely, and Huttenlocher 2010), Vienna (Irschara et al. 2009), and Aachen (Sattler et al. 2012). We use "mean descriptors" of each 3D point for all experiments. CCS, CCS_R indicate 1-1 RANSAC and new 1-M RANSAC with fixed number of iterations. CCS_RP indicates our method adopts prioritizing scheme, and Faster CCS_RP further includes accelerated RANSAC. Experiments are conducted on workstation: Intel Xeon Octa-core CPU E5-1260 3.70GHz, 64GB RAM.

Hashing based 2D-3D matching

First experiment is to determine a good test ratio threshold for precise search. It is conducted on Dubrovnik and Vienna datasets. We use ADC with Inverted File (Jégou, Douze, and Schmid 2011) with the number of coarse quantizer $K_c = 256$, 16 sub-vectors of SIFT, the number of sub-quantizers $K_{pq} = 2^8$, and the number of neighboring cells visited $w = 8$. We use small K_c and large w to ensure that quantization does not significantly affect the overall performance. In this experiment, we fix 5000 iterations to remove effect of randomness in RANSAC. A query image is "registered" if at least twelve inliers found, same as (Li, Snavely, and Huttenlocher 2010). This experiment suggests the threshold $\nu_h = 0.8$ is a good option (Fig. 1 in supplementary material).

Second experiment to choose the good size of $|\mathbf{L}_R|$ output from refined search. Conditions like the first experiment, except we choose the best threshold $\nu_h = 0.8$ for precise search. We validate our method with various number of candidates in $\mathbf{L}_R$. This experiment suggests that $|\mathbf{L}_R| = 40$ is a good option because increasing it does not significantly affect to the registration rate and inliers ratio (Fig. 3).

In the third experiment on Dubrovnik dataset, we study the influence of different indexing procedures on accuracy and computation, by comparing our method to two well-known PQ-based indexing schemes. Conditions like the second experiment, we compare our CCS to Inverted File (IV-

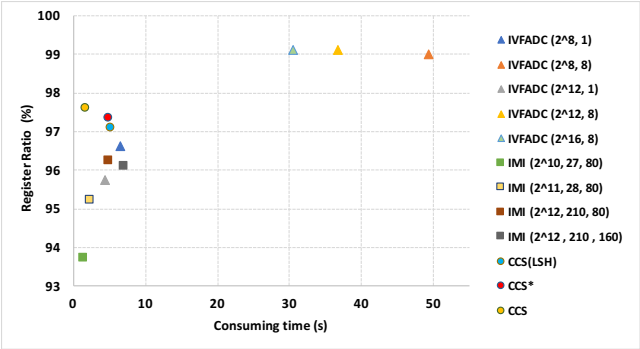

Figure 4: Comparison on indexing methods for PQ. Parameters for IVFADC (K_c, w), IMI ($K_c, w, |\mathbf{L}_R|$), CCS ($K_b = 2^{16}$, $|\mathbf{L}_R| = 40$). '*' is CCS ignoring the refined search.

Method	#reg	Median	Time (s)
Kd-tree	795	-	34.1*
(Li, Snavely, and Huttenlocher 2010)	753	9.3	
(Sattler, Leibe, and Kobbelt 2011)	782.0	1.3	2.32*
(Feng, Fan, and Wu 2016)	784.1		
(Sattler et al. 2012)	786	-	
(Sattler, Leibe, and Kobbelt 2012)	795.9	1.4	0.71*
(Sattler, Leibe, and Kobbelt 2016)	797	-	
(Cao and Snavely 2013)	796	-	
(Zeisl, Sattler, and Pollefeys 2015)	798	1.69	3.78+
(Zeisl, Sattler, and Pollefeys 2015)**	794	**0.47**	-
(Svarm et al. 2014)	798	0.56	5.06+
(Li et al. 2012)	**800**	-	
CCS	781	0.93	1.71
$\mathbf{CCS_R}$	796	0.89	1.71
$\mathbf{CCS_{RP}}$	794	1.06	**0.19**

Table 2: Results on Dubrovnik dataset. '*' are repeated on our machine, '+' reported only for outliers rejection/voting scheme taken from original papers. '**' after bundle adjustment. #reg indicates the number of registered images. (See detail in Table 2 of supplementary material)

FADC) (Jégou, Douze, and Schmid 2011), Inverted Multi-Index (IMI) (Babenko and Lempitsky 2012). We also compare to ourselves without refined search. We tune parameters of IVFADC and IMI for a fair comparison. Results in Fig. 4 demonstrate the efficiency of refined search, because removing this step slowing down CCS about $\sim 3\times$, though obtaining the similar registration rate. IVFADC with $w = 8$ visited cells achieves highest performances with different sizes of sub-quantizers, but it is too slow. Our method outperforms IVFADC (with $w = 1$) at both execution time and registration rate. IMI registers more queries when increasing the number of nearest neighbors w or the length of its re-ranking list $\mathbf{L}_R$ (same meaning as ours). Yet it also increases processing time. Our registration rate is higher than IMI, while our running time is competitive. We try to replace ITQ by LSH (Charikar 2002) in our same hashing scheme. Results show that using ITQ is $\sim 3\times$ faster than LSH. This is consistent with the parameter of the number of candidates reported in Fig. 4. Note that all three experiments above, we use 1-1 matchings and classical RANSAC.

Method	Rome	Vienna	Aachen
Kd-tree	983	221	317
(Li, Snavely, and Huttenlocher 2010)	924	204	-
(Sattler, Leibe, and Kobbelt 2016)	990.5	221	318
(Cao and Snavely 2013)	997	-	329
(Sattler et al. 2012)	984	227	327
(Feng, Fan, and Wu 2016)	979	-	298.5
(Li et al. 2012)	997	-	-
$\mathbf{CCS_R}$	991	**241**	**340**
$\mathbf{CCS_{RP}}$	991	236	338

Table 3: The number of registered images on Rome, Vienna, and Aachen datasets.

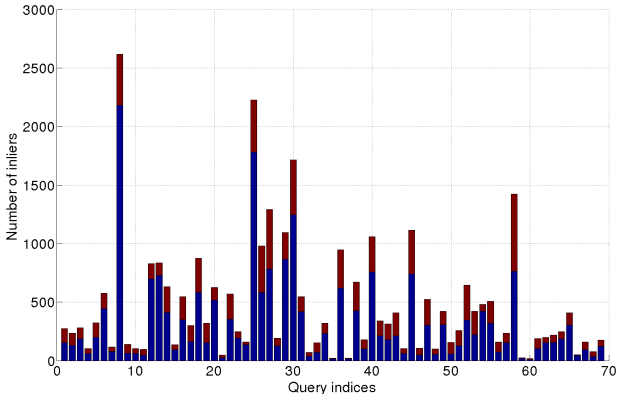

Figure 5: The contribution of inliers (on first 70 queries of Dubrovnik) found by threshold $\nu_h = 0.8$ (blue), and additional inliers found by relaxed threshold $\nu = 0.9$ (red).

Pose estimation and prioritization

In this section, we investigate the influence of our geometric verification, $\mathrm{CCS_R}$, that combines cascade search and proposed RANSAC with fixed number of 5000 iterations. We visualize the inliers found by our $\mathrm{CCS_R}$ on Dubrovnik dataset to understand the effects of ratio test. We adopt all candidates of $\mathbf{L}_R$, $M = |\mathbf{L}_R|$, in this experiment. Fig. 5 shows results of first 70 queries of Dubrovnik dataset (See similar experiment on Vienna dataset in Fig. 2 of supplementary material). For each query, the blue part is the number of inliers found by the strict ratio ν_h, and the red part is the additional ones found by the relaxed threshold ν. In average, relaxed threshold can increases about 65.4% of inliers from strict threshold, and contributes about 37.2% to the total number of inliers found by our $\mathrm{CCS_R}$. Another visualization is (Fig. 2 in supplementary material - on the right hand-side of first row) the average number of inliers contributed by 1-M matchings (from the second rank). The 1-M matches increase about average 15% of number of inliers from the strict threshold of 1-1, and about 7% of the total. It means if we use ν threshold and 1-M matchings, the method increase a significant number of inliers ($\geq 80\%$). We see on the right figure (Fig. 2 of supplementary material) that lower ranked candidates $<$ 5-th does not impact to total number of inliers; therefore to save the computation, we keep only $M = 5$ matchings after precise search.

Table 2 demonstrates the performance of $\mathrm{CCS_R}$ ($M = 5$).

Method	#reg. images	RANSAC (s)	Registration time (s)
Kd-tree	795	0.001	34.1
(Sattler, Leibe, and Kobbelt 2011)	782.0	0.01	2.33
(Sattler, Leibe, and Kobbelt 2012)	795.9	0.01	0.72
$\mathbf{CCS_{RP}}$	794	0.60	0.79
Faster $\mathbf{CCS_{RP}}$	793	0.03	**0.22**

Table 4: The processing times of RANSAC and the registration times.

Method	Vienna		Aachen	
	#reg. images	**Registration time (s)**	**#reg. images**	**Registration time (s)**
(Sattler, Leibe, and Kobbelt 2011)	206.9	2.17	-	1.93
(Sattler, Leibe, and Kobbelt 2012; 2016)	220	0.72	318	0.56
$\mathbf{CCS_{RP}}$	236	0.91	338	0.87
Faster $\mathbf{CCS_{RP}}$	228	**0.24**	335	**0.35**

Table 5: The running times (including RANSAC) on Vienna and Aachen datasets.

First, we see that $\mathrm{CCS_R}$ really outperforms CCS at both the number of registered images and errors. It confirms that using relaxed 1-M candidates per query improves the performance. The registration rate and running time of $\mathrm{CCS_R}$ is competitive to state of the art (faster than most of works), however its processing time can be further reduced by leveraging prioritizing scheme. We improve the cascade search speed with prioritized scheme ($\mathrm{CCS_{RP}}$). In the same Table 2, $\mathrm{CCS_{RP}}$ obtains similar performance as the full search but is about $\sim 10\times$ faster. By prioritizing scheme, we achieve the similar accuracy, but accelerate matching much faster than previous works. We also perform comparison using other standard datasets (Table 3). Our $\mathrm{CCS_{RP}}$ outperforms the state-of-the-art methods in registration rate on Vienna and Aachen datasets. In addition to that, our proposed method is more efficient with regards to memory because of the use of compressed descriptors. Note that when possible, we run the 2D-3D matching methods on our machine and measure their running time (excluding RANSAC time). This shows potential of using relaxed and 1-M matches for better accuracy. However, our version of $\mathrm{CCS_{RP}}$ (fixed 5000 iterations) used in above experiments can be further improved in term of execution time.

We accelerate it by using pre-verification step (Faster $\mathrm{CCS_{RP}}$). It remains competitive accuracy, but $\sim 20\times$ faster from RANSAC (5000 iterations) of $\mathrm{CCS_{RP}}$, as shown in Table 4. As a result, the total time of Faster $\mathrm{CCS_{RP}}$ is faster than $\mathrm{CCS_{RP}}$, and it needs totally only 0.2 - 0.3(s) to successfully register one query. As comparing to others, we outperform them both registration rate and execution time on Vienna and Aachen datasets (Table 5). Our proposed RANSAC (Faster $\mathrm{CCS_{RP}}$) executes as fast as classical RANSAC on small set of correspondences, e.g. 0.03(s) vs. 0.01(s) per Dubrovnik query in Table 4.

All settings of CCS requires low memory. Specifically, it requires 32 bytes (128-bit hash code and 16 bytes PQ code) to encode a SIFT descriptor. Using $m = 8$ LUTs have total 8×2^{16} buckets. Each bucket needs a 4-byte pointer referring to one point-id list. Let N_p is the point number of the 3D model, if N_p is large enough, the overhead memory is small can be ignored. m LUT refer equally m point-id of total N_p points. One point-id can be represented by a 4-byte integer number. N_p 3d point coordinates consume $N_p \times 12$ bytes. Our model needs the total of $N_p \times 76$ bytes, which is $\sim 2\times$ compressed of the original model of $N_p \times 144$ bytes, see Table 6 (ignoring the indexing structures of other methods that may require more memory). For example, Dubrovnik dataset has $N_p = 1,886,884$ 3D points (one mean descriptor per 3D point), which needs $\sim$140MB of total memory with our method. Similarly, $\sim$ 300MB, 117MB and 85MB for Rome, Aachen and Vienna respectively. These model sizes can be easily processed on the RAM of modern mobile phones or tablets.

	Our model	Original model
LUTs	$8 \times 2^{16} \times 4$	-
Point id	$8 \times N_p \times 4$	$N_p \times 4$
Point coordinates	$N_p \times 12$	$N_p \times 12$
Descriptors	$N_p \times (16 + 16)$	$N_p \times 128$
Total memory	$\approx N_p \times 76$	$N_p \times 144$

Table 6: Memory requirements (#bytes) for our model vs. original model.

Conclusion

We propose an efficient and effective 2D-3D correspondence search for image based localization. Our method improves both two aspects: 2D-3D matching and pose estimation. Firstly, we propose the cascade scheme with three steps to quickly seek out top nearest neighbors with respect to the query descriptor. Secondly, we propose a new RANSAC scheme to handle a large number of 1-M matches. Our new RANSAC significantly improve the accuracy, while remaining the similar execution time as classical RANSAC on small set of correspondences. Our method obtains competitive accuracy as compared to state-of-the-art methods on four benchmark datasets. Especially, we achieve the best performance of both processing time and registration rate on Aachen and Vienna datasets; and importantly, our method requires $\sim 2\times$ less memory footprint than most of works.

References

Agarwal, P.; Burgard, W.; and Spinello, L. 2015. Metric localization using google street view. In *IROS*.

Arth, C.; Wagner, D.; Klopschitz, M.; Irschara, A.; and Schmalstieg, D. 2009. Wide Area Localization on Mobile Phones. In *ISMAR*.

Babenko, A., and Lempitsky, V. S. 2012. The inverted multi-index. In *CVPR*.

Cao, S., and Snavely, N. 2013. Graph-based discriminative learning for location recognition. In *CVPR*.

Charikar, M. S. 2002. Similarity estimation techniques from rounding algorithms. In *ACM symposium on Theory of computing*, STOC '02.

Chen, D. M.; Baatz, G.; Koser, K.; Tsai, S. S.; Vedantham, R.; Pylvanainen, T.; Roimela, K.; Chen, X.; Bach, J.; Pollefeys, M.; Girod, B.; and Grzeszczuk, R. 2011. City-scale landmark identification on mobile devices. In *CVPR*.

Chum, O., and Matas, J. 2008. Optimal randomized ransac. *IEEE TPAMI* 30(8):1472–1482.

Feng, Y.; Fan, L.; and Wu, Y. 2016. Fast localization in large-scale environments using supervised indexing of binary features. *IEEE Trans. Image Processing* 25(1):343–358.

Fischler, M. A., and Bolles, R. C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24(6):381–395.

Gong, Y., and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*.

Irschara, A.; Zach, C.; Frahm, J.-M.; and Bischof, H. 2009. From structure-from-motion point clouds to fast location recognition. In *CVPR*.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *IEEE TPAMI* 33(1):117–128.

Li, Y.; Snavely, N.; Huttenlocher, D.; and Fua, P. 2012. Worldwide pose estimation using 3d point clouds. In *ECCV*.

Li, Y.; Snavely, N.; and Huttenlocher, D. P. 2010. Location recognition using prioritized feature matching. In *ECCV*.

Lim, H.; Sinha, S. N.; Cohen, M. F.; and Uyttendaele, M. 2012. Real-time image-based 6-dof localization in large-scale environments. In *CVPR*.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*.

Lynen, S.; Sattler, T.; Bosse, M.; Hesch, J. A.; Pollefeys, M.; and Siegwart, R. 2015. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*.

Middelberg, S.; Sattler, T.; Untzelmann, O.; and Kobbelt, L. 2014. Scalable 6-dof localization on mobile devices. In *ECCV*.

Norouzi, M.; Punjani, A.; and Fleet, D. J. 2012. Fast search in hamming space with multi-index hashing. In *CVPR*, 3108–3115.

Sattler, T.; Weyand, T.; Leibe, B.; and Kobbelt, L. 2012. Image retrieval for image-based localization revisited. In *BMVC*.

Sattler, T.; Leibe, B.; and Kobbelt, L. 2011. Fast image-based localization using direct 2d-to-3d matching. In *ICCV*.

Sattler, T.; Leibe, B.; and Kobbelt, L. 2012. Improving image-based localization by active correspondence search. In *ECCV*.

Sattler, T.; Leibe, B.; and Kobbelt, L. 2016. Efficient effective prioritized matching for large-scale image-based localization. *TPAMI* (99):1–1.

Snavely, N.; Seitz, S. M.; and Szeliski, R. 2006. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*.

Svarm, L.; Enqvist, O.; Oskarsson, M.; and Kahl, F. 2014. Accurate localization and pose estimation for large 3d models. In *CVPR*.

Ventura, J.; Arth, C.; Reitmayr, G.; and Schmalstieg, D. 2014. Global localization from monocular SLAM on a mobile phone. *IEEE Trans. Vis. Comput. Graph.* 20(4):531–539.

Zamir, A. R., and Shah, M. 2010. Accurate image localization based on google maps street view. In *ECCV*.

Zeisl, B.; Sattler, T.; and Pollefeys, M. 2015. Camera pose voting for large-scale image-based localization. In *ICCV*.

Zhang, W., and Kosecka, J. 2006. Image based localization in urban environments. In *International Symposium on 3D Data Processing, Visualization and Transmission*.