# VocMatch: Efficient Multiview Correspondence for Structure from Motion

Michal Havlena and Konrad Schindler

Institute of Geodesy and Photogrammetry, ETH Zürich, Switzerland
{michal.havlena,schindler}@geod.baug.ethz.ch

**Abstract.** Feature matching between pairs of images is a main bottleneck of structure-from-motion computation from large, unordered image sets. We propose an efficient way to establish point correspondences between *all* pairs of images in a dataset, without having to test each individual pair. The principal message of this paper is that, given a sufficiently large visual vocabulary, *feature matching can be cast as image indexing*, subject to the additional constraints that index words must be rare in the database and unique in each image. We demonstrate that the proposed matching method, in conjunction with a standard inverted file, is 2-3 orders of magnitude faster than conventional pairwise matching. The proposed vocabulary-based matching has been integrated into a standard SfM pipeline, and delivers results similar to those of the conventional method in much less time.

**Keywords:** Feature matching, Image clustering, Structure from motion.

## 1 Introduction

Recent developments in large-scale structure-from-motion (SfM) from unordered datasets make it possible to automatically construct 3D models from tens or even hundreds of thousands of photos downloaded from Internet photo-sharing sites [1,4]. The two most expensive subtasks in unordered SfM are *(i)* feature matching between images and *(ii)* bundle adjustment. The efficiency required to process such large image sets is achieved mainly by limiting those two tasks to cleverly selected subsets of images, and massively parallelizing them. Specifically, even with parallelization it would be prohibitive (quadratic in the size of the image set) to match all pairs of images, hence the set of candidate pairs to be matched is first pruned to only those pairs for which it is likely that enough correspondences can be found. There are two popular ways of reducing the number of candidate image pairs: either image indexing via shared visual words [20,14], or clustering with global image descriptors such as GIST [15] to select a subset of "iconic" images. Both methods bring a significant speed-up, but also tend to miss some image pairs that could in fact be matched, leading to unnecessary fragmentation of the resulting models.

Here, we propose an efficient way to match *all* pairs of images in a database *without* the need to exhaustively test each individual image pair. The method

D. Fleet et al. (Eds.): ECCV 2014, Part III, LNCS 8691, pp. 46–60, 2014.
© Springer International Publishing Switzerland 2014

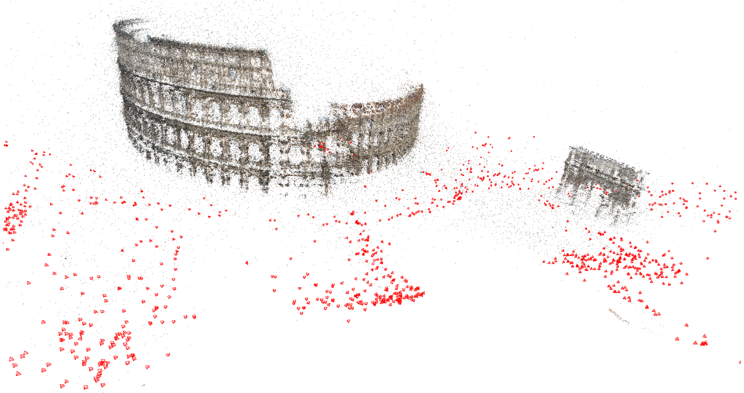


**Fig. 1.** Colosseum and Arch of Constantine, one of 13 models in the ROME dataset computed with Bundler from feature point matches of the proposed *VocMatch* method. Finding matches between all 13,049 images of Rome took $\approx 2$ hours on a single CPU core. Red pyramids denote camera poses, the 3D point cloud is visualized by small dots colored from the respective images.

builds upon a simple observation: if we employ a visual vocabulary to quantize the feature descriptors from an image, then similar features will be mapped to the same visual word. As we increase the size of the vocabulary, fewer features will map to each word, until eventually most visual words will only be found *once in a given image*. We show that with a huge vocabulary the quantization is fine enough to directly define the multi-view matching: many visual words appear in multiple images, but are unique in every one of them, hence no subsequent matching step is required to establish correspondence. By construction, the approach directly finds a list of matches for the same point across multiple images. Following [12] we call such a list a "track", borrowing the term from sequential SfM. The set of all discovered tracks can be fed into a standard SfM pipeline to reconstruct camera poses and a sparse 3D reconstruction. Although the clustering and export steps required for this integration are still quadratic in the size of the image set, they only require simple operations and are orders of magnitude faster than full matching.

We make the following contributions: *(i)* we show that, with a sufficiently large vocabulary, exhaustive feature matching in large image sets can be cast as image indexing, subject to the additional constraints that index words must be rare in the database and unique in each image; *(ii)* we demonstrate that the proposed matching method, in conjunction with a standard inverted file, yields a speed-up by three orders of magnitude, and along the way delivers the clustering of the image set into independent 3D models for free; *(iii)* we integrate the proposed matching scheme into a standard SfM pipeline, and obtain results comparable with the conventional approach in much less time, see Figure 1.

## 2 Related Work

An essential prerequisite for any feature-based SfM method is to find tracks, *i.e.* lists of corresponding feature points in different images that tentatively are projections of the same 3D point. A standard incremental SfM pipeline like Bundler [21] or VisualSFM [22], starts by detecting salient image points and encoding them with a descriptor, typically SIFT [11]. For unordered datasets no assumptions about the ordering of images can be made to constrain the matching problem—unlike SfM from video, where feature points are tracked over time [18], or SfM supported by external navigation sensors, where approximate camera poses are known before matching [8]. Hence, tracks are generated by finding two-view correspondences between all pairs of images and transitively chaining them to multiview correspondences.

In a set of $N$ images there are $\frac{1}{2}N(N-1)$ different pairings, thus as the size of the image set increases one quickly is faced with millions of image pairs. It thus becomes necessary to prefilter the possible pairs with some proxy that is much faster than feature matching. A popular method is to represent images as *tf-idf* vectors (weighted bags-of-words [20]) and compute the scalar product between those vectors [6,1]. That scalar product (*i.e.* the cosine between the two high-dimensional *tf-idf* signatures) is often a good proxy for the the amount of overlap between the images' fields of view, and thus an indication how likely it is to find corresponding scene points. Pairwise feature matching still needs to be done, but only for the smaller set of image pairs that have similar *tf-idf* signatures. Since the proxy is not perfect it can (and often does) happen that some matchable image pairs are discarded, too. In some cases this will disconnect image blocks that show the same scene and lead to unwanted fragmentation of the 3D model.

Another group of methods aims at reducing the size of the image set, and as a consequence also the set of possible pairs. Data from Internet photo-sharing sites such as Flickr [23] often contain many similar views of the same scene. For the purpose of reconstruction these are redundant (3D reconstruction can reliably be performed using only one of them), therefore it has been proposed to cluster images based on a cheap signature such as the GIST descriptor [15] and only retain one best representative per cluster [9,4]. Alternatively, one can construct a camera graph in which edges connect images with high *tf-idf* similarities, and search for the minimum dominating set of that graph, *i.e.* the smallest subset of images that adequately covers the entire camera network [7]. When pruning images rather than image pairs there is an even greater risk of fragmentation, because in most cases removing an image disconnects several possible pairs.

Our proposed method is orthogonal to the two approaches described above. We neither reduce the number of images nor restrict the set of possible pairings, but rather present a way to greatly speed up the multiview matching itself. In particular, our method naturally fits with most works described above, since it also starts with quantizing descriptors into visual words, which existing methods [6,1,7] do anyway. Given the quantization we then generate an inverted file, in time linear in the number $N$ of images. The trick in our method is to make the quantization so fine that the file entries directly correspond to feature tracks.

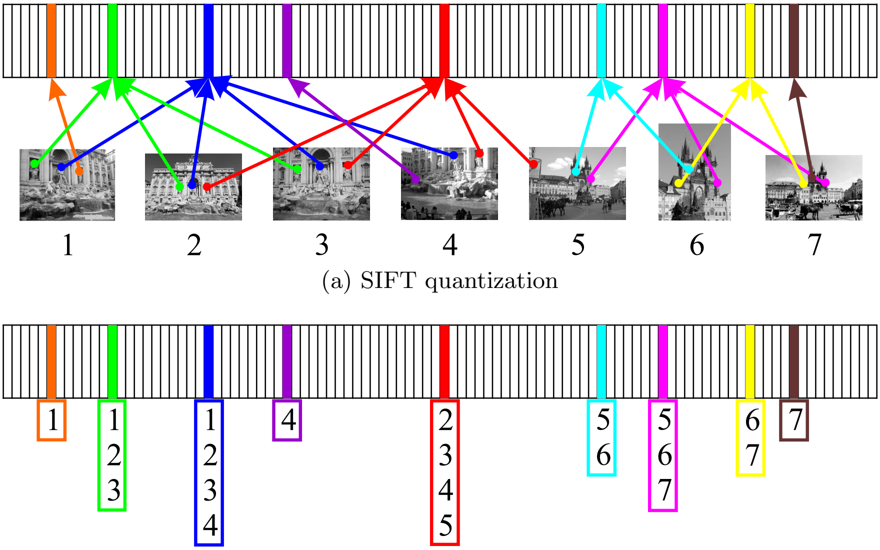
Our way of generating tracks can be seen as a limiting case of [19], where conventional matching is performed, but only between those features that map to the same visual word of a smaller vocabulary. Instead, we make sure that every image contains at most one instance of every relevant word, such that the matching becomes trivial.

Another view of the proposed method is as an extreme representative of a class of image retrieval methods which, given a large image database, aim to find images showing the same location [17,3]. These methods also index and query the database with *tf-idf* vectors w.r.t. a large visual vocabulary, verify the retrieved images using the spatial layout of feature points, and optionally expand the retrieved set by resubmitting the results as further query images. The verification step amounts to robustly fitting a geometric transformation which aligns corresponding feature points. Our method could be seen as a retrieval system that submits every image of the database as query so as to directly get maximally expanded sets of spatially related images, and then runs full SfM on these sets as geometric verification.

## 3 Method

The method builds on a huge, publicly available visual vocabulary that allows for a fine quantization into 16 million visual words [12]. The vocabulary has 2 layers (4,096×4,096 words) and was trained on a database of 11 billion SIFT descriptors computed at Hessian-Affine interest points [16]. To use the vocabulary we resample all images in the database to a common resolution, extract the same features, and quantize them with two rounds of approximate nearest-neighbor search (FLANN, [13]). Note that all these operations are linear in the number of images and can be massively parallelized.

Next, a simple loop through all processed images generates an inverted file, *i.e.* for each visual word a list of all images in which that word appears, see Figure 2. For our purposes two additional conditions must be fulfilled to include an image in the list: *(i)* the word must be *unique* (appear only once) in the respective image; and *(ii)* the word must be *rare* (appear in $\leq$1% of all images of the database). Condition *(i)* is used for technical reasons: it avoids having to resolve ambiguous correspondences, which would mean to again revert to some form of explicit matching. Due to the fine quantization the condition rejects only a negligible fraction of feature points—in our experiments on average 2.5% of the detected features in an image, which corresponds to what was observed in [2]. Condition *(ii)* on the other hand improves both the quality and the efficiency of the method. The idea behind this condition is similar to the one behind *tf-idf* weighting: words that appear rarely are more informative and/or more discriminative than those which are present in too many images. If a word appears in, say, 10% of all images then it is unlikely that the hundreds of image points are projections of the same 3D feature, even if the word is unique in each



(a) SIFT quantization

(b) inverted file generation

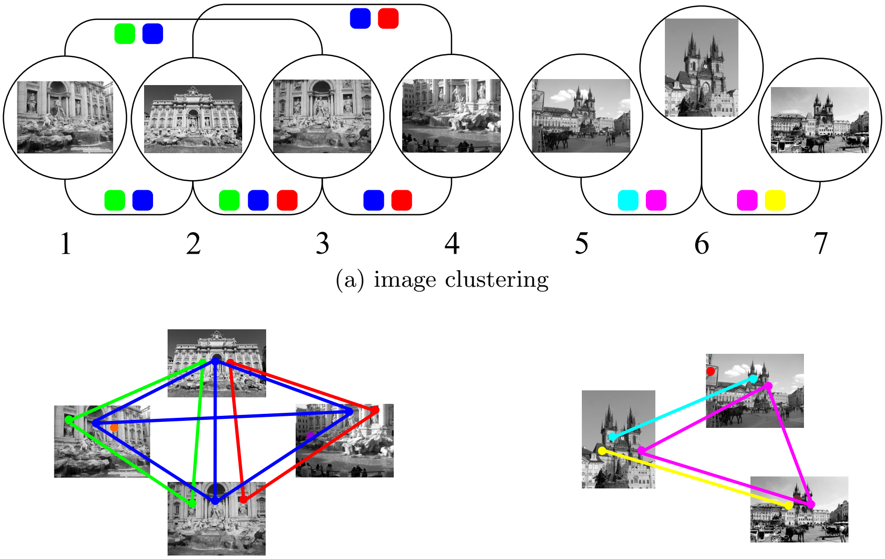
**Fig. 2.** Records of the inverted file correspond to feature tracks. Every record stores all instances of a visual word that were found in the database. If each instance is from a different image then this implies that their associated feature points are in correspondence (assuming the quantization were perfect).

image. Moreover, for later steps one needs to restore matches between image pairs.[1] Since the number of pairwise matches is combinatorial in the length of the inverted file records, limiting that length also improves efficiency. Again, the large vocabulary size ensures that only few words are affected—in our experiments $\approx$3,500 out of 16M words were eliminated because they appeared too frequently, whereas more than 600,000 words did not form a track because they were not detected in any of the database images. For simplicity, whenever we talk about "visual words" or the "vocabulary" in the following, we only mean those visual words which fulfill both conditions.

### 3.1 Clustering

Even though our method in principle generates matches between all images, most crowd-sourced image collections naturally decompose into several smaller clusters that are disconnected (*i.e.* there are no matches between different clusters, or so few matches that there is no point in reconstructing them together).

[1] The step back to two-view or three-view correspondences is necessary anyway, because incremental SfM pipelines work by chaining epipolar or trifocal geometries.



(a) image clustering

(b) export of matches

**Fig. 3.** Extracting clusters and pairwise matches from the inverted file. *(a)* Images belong to a cluster if they share a sufficient number of unique visual words (*i.e.* matches) with at least one of its members. *(b)* Finding two-view correspondences amounts to enumerating, in each record of the inverted file, all 2-element subsets that fall in the same cluster.

A sensible, and widely used strategy for SfM is to reconstruct each cluster separately. The clustering into independent models can be done using only information present in the inverted file, therefore the *clusters fall out naturally* in the proposed vocabulary-based matching framework. It is easy to count the number of putative matches for each pair of images (words detected in both images) and record them in a symmetric $N \times N$ matrix $\mathbf{Q}$, by iterating through all records in the inverted file. To find clusters that are only weakly connected to each other, all one has to do is threshold the matrix with a minimum desired number of matches $Q_{min}$. The resulting binary adjacency matrix $\mathbf{B}$ indicates which images have enough matches to be part of the same cluster, and connected component search on $\mathbf{B}$ yields the independent clusters. Note, if $Q_{min}$ is set to the minimum number of correspondences that the SfM pipeline demands for epipolar geometry estimation, then no two-view connections are lost. In practice we set the threshold slightly higher, $Q_{min} = 50$, since very weak connections in the camera network are a potential source of drift and gross pose errors. Clusters with $< 100$ images are discarded and not considered further. This weeds out small sets of isolated or unrelated images, which are invariably present in crowd-sourced datasets.

The threshold $Q_{min}$ would perhaps be sufficient for more controlled datasets. On the contrary, for images downloaded from photo-sharing sites the number of feature points can vary drastically, even if image sizes are normalized. To counter this issue, we additionally compute the fraction of matching features: given a pair of images $i$ and $j$ that have $F_i$, respectively $F_j$ visual words, we normalize the number of putative matches $Q_{ij}$ (shared visual words) with the maximum possible number of matches $min(F_i, F_j)$. For images $i$ and $j$ to be connected we then require not only the absolute count of putative matches $Q_{ij}$, but also the normalized count $Q_{ij}/min(F_i, F_j)$ to be above a threshold (set to 1.5% in our experiments).

### 3.2 Integration with SfM

The purpose of our matching procedure is to utilize the correspondences in an incremental SfM pipeline. Here we use Bundler [21], which provides a simple interface to plug in our matcher. To feed the incremental relative pose estimation, our tracks must be converted to pairwise matches, which we do by once more iterating through the records of the inverted file, and exporting all possible two-view combinations in a track to the list of matches for the corresponding image pairs. Obviously, two-view matches must only be exported for image pairs that both fall into the same cluster, see Figure 3. Note that some tracks may extend across more than one cluster. In that case (and assuming that the clusters are correct), clustering also helps to break down incorrectly merged tracks into correct ones, see Online Resource 1. Following the default setting of Bundler, we only export two-view matches if an image pair has at least 16 putative matches, otherwise no relative pose is computed. Having generated all two-view correspondences we run Bundler with the default settings, using the focal length specified in the EXIF tag as approximation whenever there is one available.

As an obvious baseline we also run Bundler with conventional matching. Corresponding SIFT descriptors from two images are found in the standard way with approximate nearest-neighbor (ANN) search in KD-trees. This two-view matching is run exhaustively over all pairs of images within a cluster. Note that pairwise matching only within clusters is already vastly more efficient than exhaustively checking all pairs of images in the dataset. On the contrary, in our method the complete set of tracks is inherently available, and independent clusters only afford small savings during the export of two-view matches. Nevertheless, the proposed vocabulary-based matching is three orders of magnitude faster than full ANN matching on our datasets, whereas the quality of the obtained 3D models is nearly on par. We could not test ANN matching without the preceding clustering step, because it turned out to be intractable for our datasets.

## 4 Experiments

The proposed method is validated using two datasets sourced from Flickr [23]. The first one, MERGED, was kindly provided by the authors of [5]. It consists

**Table 1.** Time spent for different steps of the method. Note, SIFT extraction would need to be performed by any feature-based SfM method, and feature quantization is needed by any methods that involves bag-of-words signatures [6,1]. The actual time needed to generate tracks is less than 1 hour for both datasets. Exporting two-view matches takes about as long as track generation.

| | MERGED [5] | ROME [10] |
|---|---|---|
| number of images | 9,469 | 13,049 |
| SIFT extraction [16] (6 threads) | 8h 40min | 11h 30min |
| feature quantization [13] | 5h 20min | 6h 50min |
| inverted file generation | **35min** | **41min** |
| counting number of shared VWs | **10min** | **17min** |
| number of discovered clusters | 3 | 13 |
| export of matches | **30min** | **1h 15min** |

of 9,469 images from three different landmarks: roughly one third of them depicts the Fontana di Trevi in Rome, another third the Duomo in Milan, and the last third the Old Town Square in Prague. All images are resampled to a size of $\approx 3$ Mpix for further processing, a standard practice for SfM with uncontrolled data. The second dataset, ROME, is very similar to the one of [10]. The small difference is caused by the fact that the authors only provide feature descriptors on the project website. Since their interest point detector and SIFT implementation are incompatible with the pretrained vocabulary we use, we had to download the original images again from Flickr. A small number of images were no longer available, which causes the difference. In total our version consists of 13,049 images depicting several famous landmarks of Rome (Colosseum, St. Peter's Basilica, Fontana di Trevi, Pantheon, *etc.*). Again all images are resized to $\approx 3$ Mpix.

We point out an important difference between the two datasets. MERGED is the raw result of searching Flickr with text tags, whereas ROME contains only images that were connected to a 3D model in [10], *i.e.* for these images SfM computation already succeeded once. The more realistic situation is the one without pre-filtering, where the matching and clustering must also cope with unusable and unrelated images, as in the MERGED set. Still we also test on the ROME data, which it is more widely known and used, in order to better put our method into context.

### 4.1 Feature Extraction and Clustering

As described in Section 3, we extract Hessian-affine interest points in all input images, convert them to SIFT descriptors, and quantize them against the 2-layer (4,096×4,096) visual vocabulary of [12] with ANN search using FLANN [13]. One can exploit the hierarchical nature of the vocabulary to gain some efficiency, by grouping the features from several images together in order to lower the number

**Table 2.** Clustering results for MERGED. The dataset contains a significant number of clutter images not directly related to the query. The recovered clusters cover the large majority of relevant images. The wrongly included images in $M_3$ are mostly caused by similarly decorated Christmas trees that were present at different sites.

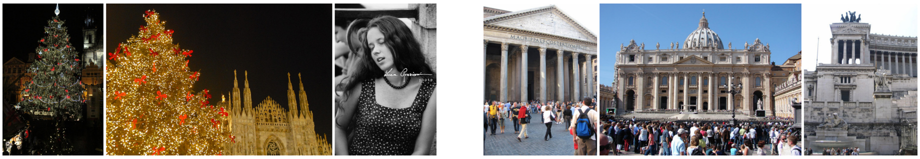
| | # img | $M_1$ | $M_2$ | $M_3$ |
|---|---|---|---|---|
| DITREVI | 3,110 | 1,973 | 0 | 1 |
| DUOMO | 3,104 | 0 | 580 | 11 |
| OLDTOWN | 3,255 | 0 | 0 | 1,402 |

**Table 3.** Clustering results for ROME. The dataset mainly contains related images because they were connected to a 3D model in [10]. With few exceptions (one of which is a mistake in the ground truth) the recovered clusters are clean. See text for details.

| | # img | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $R_6$ | $R_7$ | $R_8$ | $R_9$ | $R_{10}$ | $R_{11}$ | $R_{12}$ | $R_{13}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COLOSS. | 1,664 | 1,392 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| DITREVI | 1,518 | 0 | 1,381 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ST.PET. | 1,129 | 0 | 0 | 967 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| COL_IN | 1,089 | 0 | 0 | 0 | 774 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ST.P_IN | 952 | 0 | 0 | 0 | 0 | 728 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PANTH. | 775 | 0 | 0 | 721 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PAN_IN | 672 | 0 | 0 | 0 | 0 | 0 | 512 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ALTPAT | 604 | 0 | 0 | 498 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ANGELO | 304 | 0 | 0 | 0 | 0 | 0 | 0 | 251 | 0 | 0 | 0 | 0 | 0 | 0 |
| SPAN_DN | 220 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SPAN_UP | 212 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 136 | 0 | 0 | 0 | 0 | 0 |
| ARCH | 211 | 166 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| FORUM | 208 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 138 | 0 | 0 | 0 | 0 |
| SENAT | 206 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 156 | 0 | 0 | 0 |
| COLON. | 206 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 158 | 0 | 0 |
| SISTINE | 195 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 108 | 0 |
| ST.P_TV | 182 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 152 |
| *other* | 2,702 | 0 | 0 | 225 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

of FLANN calls in the second layer. Note that in the following only the visual word index is needed for our method, so in principle one need not store the original descriptors after they have been quantized.

Table 1 shows computation times measured on a standard desktop machine (Intel(R) Core(TM) i7-3930K with 24 GB of RAM) running 64bit Linux. Apart from SIFT extraction running in 6 threads, all steps are currently single-threaded and implemented in MATLAB. In particular, due to format and implementation details of the pretrained vocabulary, an older, single-threaded version of FLANN had to be used for the quantization, so our timings are a conservative upper bound for the potential of the method.



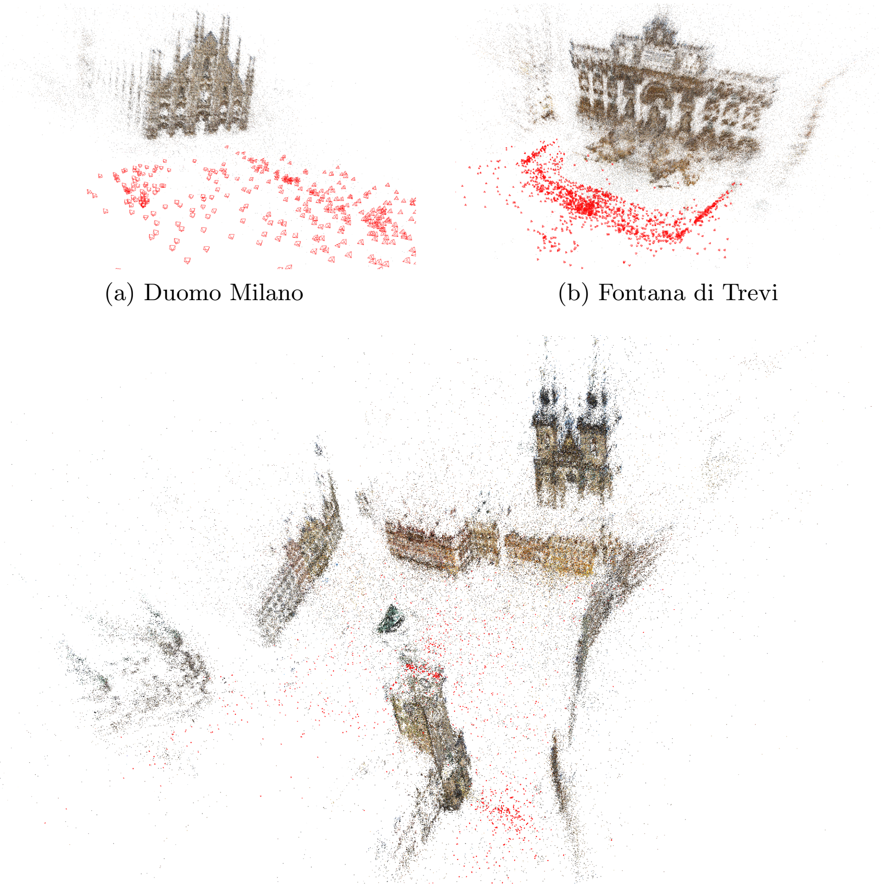
(a) MERGED

(b) ROME

**Fig. 4.** *(a)* The only inaccuracies in the clustering of MERGED. *(b)* The sole failure of vocabulary-based clustering for ROME. See text for details.

After generating the inverted file the absolute and relative numbers of shared visual words are counted for all pairs of images. Note that this step is only needed for incremental SfM—if approximate pose information is available from external sources, then one might be able to filter outliers based on the full tracks and directly proceed to bundle adjustment. Constructing the adjacency graph and searching connected components takes less than 10 seconds. The resulting clusters nicely match the structure of the data, see Tables 2 and 3. For MERGED the three ground truth clusters are given explicitly, because the images were found with three different queries. We point out that the "ground truth" clusters each contain a significant amount of unrelated images, so one should not expect the estimated clusters to include all images. The clusters are very clean: we verified by visual inspection that the overwhelming majority of unrelated images was correctly rejected and very few unrelated images are contained in any of the three retained clusters. Regarding confusions between clusters, only 12 images are assigned to the wrong cluster. None of these mistakes are specific to the proposed vocabulary-based matching: the 11 images from DUOMO are due to similarly decorated Christmas trees present at both locations. The 1 failure image from DITREVI is due to a dotted dress, which also matches the Christmas trees, see Figure 4a. All 12 images are later removed during SfM computation. The reason for the low number of images in $\mathrm{M}_2$ is a technical issue: the relevant DUOMO images are actually made up of three only weakly connected sub-clusters for the front facade, the inside and the roof. These are indeed recovered separately, but only the one for the front facade is large enough to be exported ($\geq 100$ images).

For ROME we consider the division into individual 3D models (which is provided together with the data) as the ground truth clustering. Also in this case the recovered clusters are clean w.r.t. the ground truth grouping, with two exceptions. In $\mathrm{R}_1$ our clustering corrects the ground truth: the Constantine ARCH and the outside of COLOSSEUM are located near each other and can in fact be reconstructed into a connected 3D model. $\mathrm{R}_3$ is an actual failure case, where multiple sites (the front of St. Peter's Basilica, the front of the Pantheon, and the Altare della Patria) are mixed, because the images are dominated by similar column structures, see Figure 4b. Nevertheless, all the sub-models could be reconstructed. The recovered cluster for SPANISH STEPS DOWN had less than



(a) Duomo Milano

(b) Fontana di Trevi

(c) Old Town Square Prague

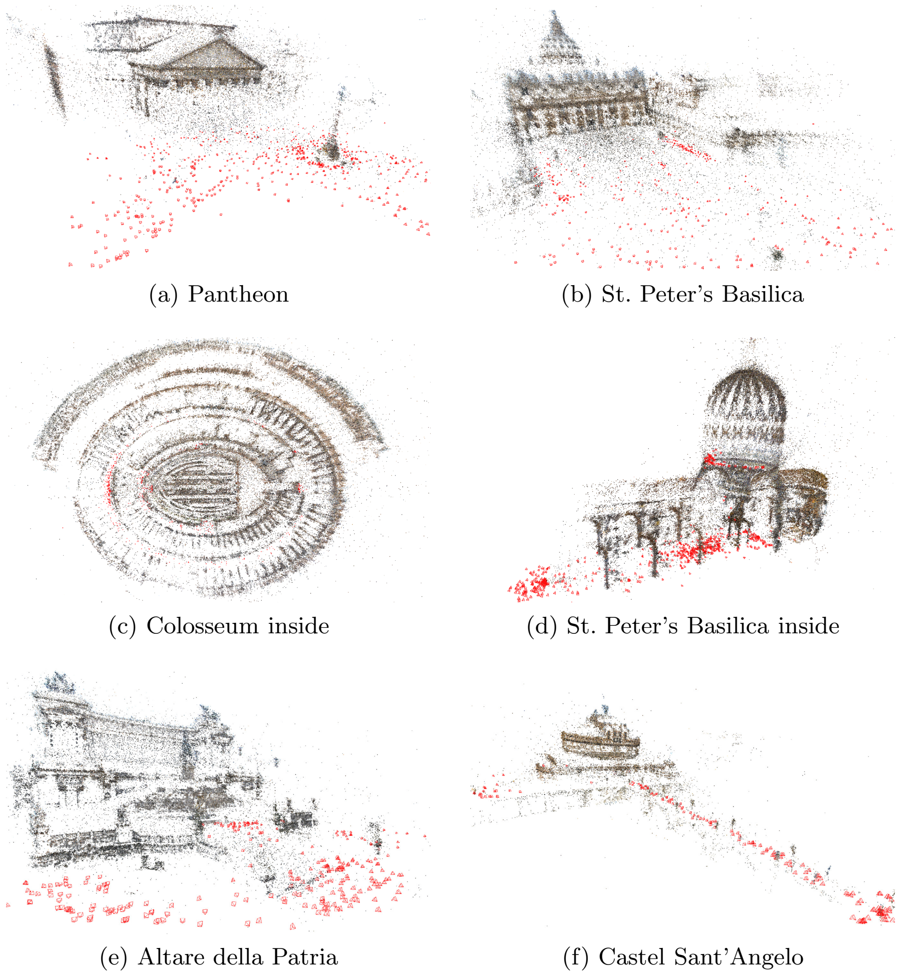
**Fig. 5.** SfM results for the MERGED dataset (matches created by the proposed method). Estimated camera poses are denoted by red pyramids.

100 images and thus was removed. Since for the ROME dataset clean ground truth clusters without unrelated clutter are given we could also check the completeness of our clustering. On average, $> 77\%$ of all images were found for a given landmark.

Overall, the quality of clustering is more than satisfactory, considering the fact that it was performed without any geometric verification.

### 4.2 Export of Matches and 3D Reconstruction

After exporting the matches for each cluster, Bundler was called with the default settings to perform sparse 3D reconstruction, see Figures 5, 1, and 6. In most cases, all or almost all relevant images from a cluster could be oriented.



(a) Pantheon

(b) St. Peter's Basilica

(c) Colosseum inside

(d) St. Peter's Basilica inside

(e) Altare della Patria

(f) Castel Sant'Angelo

**Fig. 6.** SfM results for the ROME dataset (matches created by the proposed method). Estimated camera poses are denoted by red pyramids.

This demonstrates that for practically all images the inlier fraction of the proposed vocabulary-based matching is high enough to successfully RANSAC correct epipolar geometries. The worst failure is that for PANTHEON 20% of the images, which show the inside of the building, could not be connected.

As baseline we also perform standard pairwise ANN matching (Bundler default) in each image cluster. As expected, the running time is orders of magnitude longer, even if the matching is parallelized in 24 threads, see Table 4. Note that state-of-the-art CUDA-enabled SiftGPU matching from VisualSFM for a slightly smaller number of SIFT features was not much faster than the multi-threaded CPU matching either, the proposed method—export of matches

**Table 4.** Time needed for exhaustive pairwise matching within the clusters (in hours). Three methods are compared: multi-threaded ANN CPU matching, SiftGPU (CUDA-enabled GPU matching), and the proposed method. For the first two methods matching all image pairs without prior clustering would take orders of magnitude longer.

| cluster | # img | # pair | 24× CPU | $\sum$ CPU time | 448× GPU | proposed |
|---|---|---|---|---|---|---|
| M$_1$ | 1,973 | 1,945,378 | 35:00 | 822:40 | 11:10 | |
| M$_2$ | 580 | 167,910 | 3:30 | 75:50 | 0:55 | |
| M$_3$ | 1,414 | 998,991 | 15:00 | 341:40 | 5:40 | |
| MERGED | total time | | 53h 30min | 1,240h 10min | 17h 45min | 30min |
| | speed-up | | 107× | 2,480× | 35× | |
| R$_1$ | 1,558 | 1,212,903 | 17:25 | 404:00 | 7:00 | |
| R$_2$ | 1,381 | 952,890 | 13:35 | 314:20 | 5:30 | |
| R$_3$ | 2,411 | 2,905,255 | 30:25 | 716:30 | 16:45 | |
| R$_4$ | 774 | 299,151 | 6:06 | 138:50 | 1:45 | |
| R$_5$ | 728 | 264,628 | 3:20 | 76:10 | 1:30 | |
| R$_6$ | 512 | 130,816 | 0:55 | 20:15 | 0:45 | |
| R$_7$ | 251 | 31,375 | 0:27 | 8:30 | 0:10 | |
| R$_8$ | 136 | 9,180 | 0:11 | 3:20 | 0:03 | |
| R$_9$ | 138 | 9,453 | 0:14 | 4:00 | 0:03 | |
| R$_{10}$ | 156 | 12,090 | 0:10 | 2:50 | 0:04 | |
| R$_{11}$ | 158 | 12,403 | 0:27 | 8:15 | 0:04 | |
| R$_{12}$ | 108 | 5,778 | 0:05 | 1:30 | 0:02 | |
| R$_{13}$ | 152 | 11,476 | 0:20 | 6:30 | 0:04 | |
| ROME | total time | | 73h 40min | 1,705h | 33h 45min | 1h 15min |
| | speed-up | | 59× | 1,364× | 27× | |

from the inverted file—is still roughly 30× faster. Comparing the quality of the reconstructed models we note the following:

*(i)* Practically the same number of correct camera poses are reconstructed with both methods. For most applications this is *the* most important quality criterion. In localization/navigation-type applications the camera pose is the immediate goal. If the goal is 3D object modeling then SfM also serves mainly to recover the camera poses, since the final model is constructed in a subsequent step with some form of dense matching or 3D surface reconstruction.

*(ii)* Full pairwise matching on average generates about twice as many matches as the vocabulary-based method, whereas both yield approximately the same number of 3D structure points. It seems that vocabulary-based matching is in fact stricter and cannot always find the complete track for an object point. This is in all likelihood also the reason why Bundler took significantly longer to construct the 3D models from conventional pairwise matches.

*(iii)* The vocabulary-based point clouds are a bit noisier, which means that the correspondences detected with the proposed method are either not quite as accurate, or that they contain more epipolar-consistent miss-matches. The lower accuracy is somewhat expected, since shorter tracks mean fewer rays per point

and thus higher uncertainty of the triangulation. While the issue still needs to be investigated in more detail, we believe that also miss-matches play a role: in Internet photo collections many images are typically taken from the same height with similar viewing direction, *e.g.* viewing the front side of a monument from street level. Under that viewing geometry repetitive or diffuse appearance along the (horizontal) epipolar lines can lead to miss-matches. Such matches will be rejected by the 2$^{\text{nd}}$-best ratio test in standard SIFT matching, but the vocabulary-based approach includes no such test.

## 5 Conclusions

We have shown that multiview correspondence can be efficiently established even for large datasets, if one replaces pairwise image feature matching with image indexing, using only visual words that appear at most once in every image. What makes the proposed method practical is the realization that today's visual vocabularies are large enough to ensure that, even in datasets $> 10{,}000$ images, the overwhelming majority of all words are in fact unique in all images they appear in.

Formally speaking, the steps required for *feature track generation* with the proposed *VocMatch* method, namely *(i)* interest point extraction, *(ii)* feature quantization against a fixed vocabulary, and *(iii)* inverted file generation are all linear in the size of the image set. The additional steps needed to *integrate* the method with incremental SfM, namely *(iv)* clustering based on match counts and *(v)* export of two-view matches are still quadratic. But they involve only very simple operations and in practice are orders of magnitude faster than full ANN matching.

Regarding scalability of the proposed approach, for 100,000 images with 10,000 features per image, the inverted file would take 6 GB of RAM (6 bytes per feature) and the clustering matrix **Q** would take 10 GB (2 bytes per image pair). Extrapolating the measured times, inverted file generation would take 6 hours (linear in the total number of features) and generation of **Q** 16 hours (quadratic in the track length) using a single thread, which are still very competitive times. For 1,000,000 images the method is currently not feasible, storing matrix **Q** in RAM would be impossible (1 TB).

An open question is when a vocabulary becomes *too large* to be useful for matching. It seems clear that a too fine quantization will cause tracks to become fragmented, or lost altogether. We did not experience problems in our experiments, but note that the critical size is also dependent on the dataset: image sets with few interest points or weak connectivity are more vulnerable to a loss of correspondences. To counter the problem it might be possible to exploit the similarity between different visual words, either by direct comparison or by analyzing the descriptors they were trained on, and augment the vocabulary with neighborhood information. This could potentially allow one to also match descriptors if they do not quantize to the exact same visual word.

## References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building Rome in a day. In: ICCV (2009)
2. Chum, O., Perdoch, M., Matas, J.: Geometric min-Hashing: Finding a (thick) needle in a haystack. In: CVPR (2009)
3. Chum, O., Philbin, J., Sivic, J., Isard, M., Zisserman, A.: Total recall: Automatic query expansion with a generative feature model for object retrieval. In: ICCV (2007)
4. Frahm, J.-M., et al.: Building Rome on a cloudless day. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 368–381. Springer, Heidelberg (2010)
5. Havlena, M., Hartmann, W., Schindler, K.: Optimal reduction of large image databases for location recognition. In: BD3DCV (2013)
6. Havlena, M., Torii, A., Knopp, J., Pajdla, T.: Randomized structure from motion based on atomic 3D models from camera triplets. In: CVPR (2009)
7. Havlena, M., Torii, A., Pajdla, T.: Efficient structure from motion by graph optimization. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 100–113. Springer, Heidelberg (2010)
8. Klingner, B., Martin, D., Roseborough, J.: Street view motion-from-structure-from-motion. In: ICCV (2013)
9. Li, X., Wu, C., Zach, C., Lazebnik, S., Frahm, J.-M.: Modeling and recognition of landmark image collections using iconic scene graphs. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 427–440. Springer, Heidelberg (2008)
10. Li, Y., Snavely, N., Huttenlocher, D.P.: Location recognition using prioritized feature matching. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 791–804. Springer, Heidelberg (2010)
11. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV 60(2), 91–110 (2004)
12. Mikulik, A., Perdoch, M., Chum, O., Matas, J.: Learning vocabularies over a fine quantization. IJCV 103(1), 163–175 (2013)
13. Muja, M., Lowe, D.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP (2009)
14. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2006)
15. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV 42(3), 145–175 (2001)
16. Perdoch, M., Chum, O., Matas, J.: Efficient representation of local geometry for large scale object retrieval. In: CVPR (2009)
17. Philbin, J., Chum, O., Isard, M., Šivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR (2007)
18. Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. IJCV 59(3), 207–232 (2004)
19. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2D-to-3D matching. In: ICCV (2011)
20. Šivic, J., Zisserman, A.: Video Google: Efficient visual search of videos. In: Toward Category-Level Object Recognition, CLOR (2006)
21. Snavely, N., Seitz, S., Szeliski, R.: Modeling the world from internet photo collections. IJCV 80(2), 189–210 (2008)
22. Wu, C.: VisualSFM: A visual structure from motion system (2013), `http://ccwu.me/vsfm`
23. Yahoo!: Flickr: Online photo management and photo sharing application (2005), `http://www.flickr.com`