Real-time Image-based 6-DOF Localization in Large-Scale Environments

Hyon Lim
Seoul National University, Korea
hyonlim@snu.ac.kr

Sudipta N. Sinha Michael F. Cohen Matthew Uyttendaele
Microsoft Research, Redmond, USA
{sudipsin, mcohen, mattu}@microsoft.com

Abstract

We present a real-time approach for image-based localization within large scenes that have been reconstructed offline using structure from motion (Sfm). From monocular video, our method continuously computes a precise 6-DOF camera pose, by efficiently tracking natural features and matching them to 3D points in the Sfm point cloud. Our main contribution lies in efficiently interleaving a fast keypoint tracker that uses inexpensive binary feature descriptors with a new approach for direct 2D-to-3D matching. The 2D-to-3D matching avoids the need for online extraction of scale-invariant features. Instead, offline we construct an indexed database containing multiple DAISY descriptors per 3D point extracted at multiple scales. The key to the efficiency of our method lies in invoking DAISY descriptor extraction and matching sparingly during localization, and in distributing this computation over a window of successive frames. This enables the algorithm to run in real-time, without fluctuations in the latency over long durations. We evaluate the method in large indoor and outdoor scenes. Our algorithm runs at over 30 Hz on a laptop and at 12 Hz on a low-power, mobile computer suitable for onboard computation on a quadrotor micro aerial vehicle.

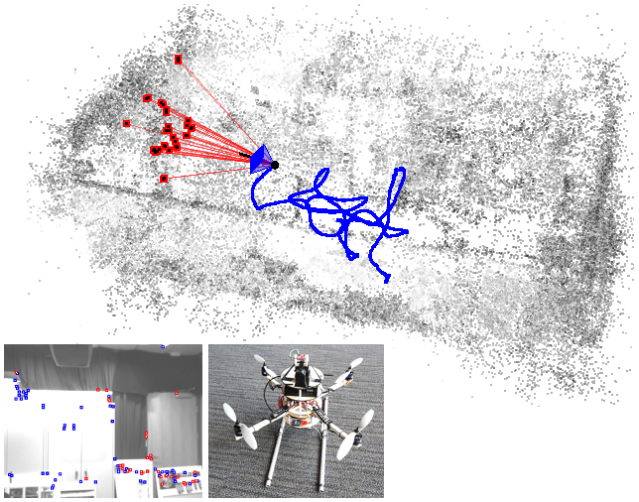

Figure 1: Our method can precisely localize a camera in real-time within a scene reconstructed offline using Sfm. The flight path of a quadrotor micro aerial vehicle (MAV) within a 8m $\times$ 5m room (reconstruction has 76K points), and 3D points that were matched to 2D features in the current frame (LOWER-LEFT), are shown.

1. Introduction

The problem of computing the position and orientation of a camera with respect to a geometric representation of the scene, which is referred to as *image-based localization*, has received a lot of attention in the computer vision community. It has important applications in location recognition [23, 26, 13, 18], autonomous robot navigation [24, 20, 1] and augmented reality [15, 10, 33]. Broadly speaking, there are two approaches to image-based localization. The first addresses the problem of simultaneous localization and mapping (SLAM), where the camera is localized within an unknown scene. In contrast, approaches in the second category use the knowledge of a prior map or 3D scene model. Several recent methods fall in the second category [13, 10, 18, 25], and this renewed interest has been sparked by progress in structure from motion (Sfm) [29, 14], which makes it possible to easily reconstruct large scenes in great detail.

Despite the scalability of recent approaches [13, 18, 25], real-time image-based localization in large environments remains a challenging problem. As the scene gets larger, recognizing unique identifiable landmarks becomes more challenging. In [13, 18, 25], this difficulty is overcome by using sophisticated image features such as SIFT [19], but these are too expensive to compute in real-time. On the other hand, some visual SLAM [15, 7, 36] systems are real-time, but their performance degrades in larger scenes, where map maintenance becomes progressively expensive. These techniques are also fragile if the camera moves too quickly, which makes them less attractive for persistently computing a precise camera pose over longer durations.

Recently, image-based localization has gained importance for autonomous aerial navigation, especially in GPS-

denied areas [1]. It is particularly attractive for micro-aerial vehicles (MAV), such as the PIXHAWK quadrotors [20], which have limited payload but are capable of full-fledged onboard vision processing. However, previous approaches [13, 18, 10, 25] are not fast enough for such platforms.

In this paper, we propose a new approach for continuously localizing a camera within large environments, which have already been reconstructed using Sfm. Our algorithm is real-time and runs over long periods with low fluctuations in the frame-rate. At its core lies a fast keypoint tracker. Keypoints (Harris corners) from one frame are tracked in the following frame by matching to candidate keypoints within a local search neighborhood [30] in the next frame. Inexpensive to compute, binary feature descriptors (BRIEF) [6] are used to find the best frame-to-frame match. This fast tracker is interleaved with a new, efficient approach to find corresponding 3D points in the Sfm reconstruction to the tracked keypoints. These 2D-3D correspondences then robustly determine the camera pose for each frame. For determining these correspondences, we match features using more expensive DAISY descriptors [31, 37], and a kd-tree index over the descriptors. This approach is related to recent work on *direct 2D-to-3D matching* [25]. However, in contrast to their work which focuses on localizing single images, we address the problem of continuous localization from video over long durations and propose several modifications to exploit spatio-temporal coherence.

We are able to achieve real-time performance by avoiding the need for scale-invariant keypoints at runtime. This is a key distinction from prior approaches [13, 10, 18, 25], which rely on the scale-invariance of SIFT [19]. However, matching features across different scales is important for reliable 2D-to-3D matching and we address this requirement by computing redundant descriptors during offline processing, *i.e*. multiple descriptors for each 3D point in our reconstruction, extracted at different scales from multiple images. By storing corresponding camera indices along with the descriptors, we can efficiently perform *place recognition*, which we use to prune false 2D-3D matches prior to geometric verification during camera pose estimation. Other indirect methods first perform keyframe recognition [10, 13] using global image descriptors [7] or with local image features and a hierarchical bag-of-visual-words model [21]. However, they incur the overhead of storing images or features in memory and having to geometrically verify the pairwise feature matches during online processing.

Our feature matcher can also be used for localizing a single image from scratch. This is vital for localizing the camera in the first frame or for quick relocalization when the camera is lost. However, at other times when tracking is successful, we adopt a much more efficient *guided matching* approach for 2D-to-3D matching, similar to strategies used in SLAM [9, 15]. Unlike traditional robust feature matching [19, 28], where ambiguous matches are usually pruned using a ratio-test [19, 13, 25], we recover multiple (one-to-many) 2D-3D match hypotheses for each tracked keypoint, and prune outliers later, during robust pose estimation. We optimize guided matching further, by distributing the computation in batches over a window of successive frames. By avoiding too many descriptor computations and kd-tree queries all at once, large fluctuations in the per-frame processing time are prevented. With lower per-frame latency, keypoints with known 3D point correspondences are typically tracked over longer sequences. This higher efficiency in tracking amortizes the cost of the relatively more expensive feature matching step, by requiring the matcher to be invoked less frequently over longer periods of time.

1.1. Related Work

A number of existing works in image-based localization have adopted an image-based retrieval approach to the problem, and used it for urban scene navigation [23] and city-scale location recognition [26]. However, these approaches often recover an approximate location estimate or may not even compute a complete 6-DOF pose. Existing work on markerless augmented reality [8, 16] addresses real-time 3D camera tracking but typically only with respect to specific objects, that often requires a CAD model of the object.

Approaches for 3D camera tracking using visual landmark recognition [28] based on SIFT features [19], was proposed for *global localization* and used for robot navigation [27]. However, the need for repeated pairwise image matching in these approaches makes them too slow for real-time systems. Efficient keypoint recognition with randomized trees [17] and random ferns [22] have enabled real-time camera tracking, but their significant memory requirements have limited their use beyond relatively small scenes.

SIFT features [19] are also used in recent work [13, 18, 25] on location recognition, where Sfm is used to estimate 3D coordinates for the landmarks. The approaches scale well and some variants use the GPU [13]. However, these methods address the single-image localization problem, and are not fast enough for real-time localization from video.

Recently, a method for continuous localization was proposed for scenes reconstructed using Sfm [10]. It uses keyframe recognition repeatedly on video frames to indirectly recover 2D-3D matches. SIFT feature extraction is also the bottleneck in their method, which runs at 6 fps on a single thread and at 20 fps using parallel threads on four cores. Although our approach is related, it differs in the following ways – we explicitly track keypoints using binary descriptors [6], to amortize the cost of feature matching over time, which is performed only as needed. Instead of keyframe-based matching, we use 2D-to-3D matching interleaved with tracking. This allows us to exploit spatio-temporal coherence and lowers the per-frame latency.

Real-time localization approaches for augmented reality on mobile devices have also been recently proposed [2, 33, 7]. However, these approaches derive their speedup from tracking relatively fewer features, making them less suitable for continuous 6-DOF localization in larger scenes or over longer durations. An efficient approach for tracking scale-invariant features in video was proposed in [30], but it was used for object recognition, not real-time localization.

Visual SLAM systems have recently been used for real-time augmented reality [9], by utilizing parallel threads for tracking and mapping (PTAM) [15], using multiple local maps [7] and performing fast relocalization using random ferns [36]. However, these approaches are susceptible to the problem of drift and error accumulation in the pose estimate, and existing solutions for fast relocalization do not scale to larger scenes. Visual SLAM (PTAM) [15] was recently used onboard a micro aerial vehicle for vision-based *position control i.e*. hovering at a pre-specified location [1], and autonomous navigation was demonstrated for scenes with salient visual features [20, 4]. However, these approaches focus more on the challenges of autonomous flight control, and so far have demonstrated vision-based localization either in small areas or within controlled scenes.

2. Key elements of the proposed approach

We now discuss our scene representation and the building blocks of the new 2D-to-3D matching approach. The offline and online stages of our algorithm are described in Sections 2.3 and 3 respectively. Real-time localization requires efficient 2D-to-3D matching in two specific scenarios. First, for initializing localization or relocalization, the camera pose must be efficiently computed from a single image from scratch [25, 18, 13, 10]. We call this *global matching*, which is challenging because the complete map must be searched. However, for intermediate video frames, a pose estimate computed from tracked features with known 3D point matches is used to significantly speed up the search for the keypoints with unknown correspondences; we call this *guided matching*.

Scene representation. Our representation consists of a 3D scene reconstruction in a global coordinate system, which is computed using Sfm [29] on an input sequence. This consists of the calibrated images, a 3D point cloud and a set of 2D-3D matches, that encode the views from which a particular 3D point was triangulated from during Sfm. The calibrated images are used to build a database of feature descriptors for the 3D points, and a kd-tree index is constructed over the descriptors to support efficient approximate nearest neighbor (ANN) queries during feature matching. We extract keypoints using the Harris corner detector at multiple scales and compute DAISY descriptors [31], in particular T2-8a-2r6s descriptors [37] for each keypoint[1]. Descriptors in the database are labeled with their image indices, and the mapping between descriptors and corresponding 3D points is saved in a lookup table. This makes retrieving 3D points corresponding to the descriptors in the database very efficient. We further optimize the point retrieval by grouping cameras into overlapping clusters and use them for *place recognition*, as described later in this section.

Multi-scale features. To avoid extracting scale invariant keypoints (*e.g*. DoG [19]) during online computation, offline we store in a database multiple descriptors for each 3D point from keypoints detected across a range of scales. First, multi-scale Gaussian image pyramids are computed and Harris corners are extracted in all levels of the pyramids. At each keypoint, its orientation is computed by finding a peak in the gradient orientation histogram [19], and a rotation invariant T2-8a-2r6s-32d DAISY descriptor [37] is computed from a resampled patch.

The 3D points in the map are then projected into the images they were triangulated from during Sfm. For each keypoint in a particular pyramid level of an image, the closest point amongst all 2D projections of the 3D points corresponding to that image is computed. If the closest point is within a threshold of τ pixels [2] (we set τ=2.0), that keypoint and its descriptor is assigned to the corresponding 3D point. This computation is performed for each image pyramid to generate all the descriptors for the 3D points in the map.

Having multiple descriptors, as described above, has an associated overhead in storage. However, the redundancy in this representation allows keypoints extracted at a fixed scale during online localization to be matched to a set of descriptors in the database, as long as one of the descriptors in this set was extracted at a similar scale. Using multiple descriptors per 3D point is advantageous for ANN queries during feature matching for reasons pointed out in [5], where using multiple descriptors boosted the accuracy of a simple nearest neighbor classifier for image classification.

Place recognition. In large scenes, global matching becomes more difficult due to greater ambiguity in feature descriptors. A query descriptor in an image could match descriptors for several different 3D points, which are similar in appearance. To address this, we perform coarse location recognition to filter as many incorrect 2D-3D matches as possible before the geometric verification step. As a result, fewer RANSAC [11] hypotheses will be required during robust pose estimation, making that step more efficient.

For place recognition, we cluster nearby cameras during the offline stage into *location classes*, which are identified by solving an *overlapping view clustering* problem [12],

[1] using PCA to reduce the dimension to 32.

[2] the distance is computed in the appropriate pyramid level.

where cameras with many Sfm points in common are grouped into the same cluster. We use an approach similar to the one proposed in [12], for clustering Internet image collections, which iterates between finding a disjoint partition of the cameras by analyzing the match graph from Sfm, and growing the clusters locally by including cameras from neighboring clusters to improve the coverage of 3D points [12, 18]. When localizing an image, the most likely location class is selected using a simple voting scheme over the set of matching descriptors returned by the ANN query on the image descriptors. Matched descriptors that correspond to 3D points that do not belong to the selected location cluster are removed. This approach is non-parametric in comparison to some approaches that cluster features in pose space [19], to achieve a similar goal. The global and guided matching are next described in more detail.

2.1. Global matching

Given 2D keypoints in an image and their corresponding DAISY descriptors denoted as $Q = \{q_i\}$, we seek to retrieve a set of 3D point correspondences for them. For each descriptor q_i, we perform a k-ANN query based on priority search using a kd-tree [3, 19], which retrieves approximate nearest neighbors $D_i = \{d_{ij}\}$ sorted by increasing distance $\{s_{ij}\}, j = 1 \dots k$, from q_i. For each neighbor d_{ij}, where $s_{ij} < \sigma s_{i0}$, the corresponding 3D point X_{ij} is retrieved, and every cluster that X_{ij} belongs to, receives a vote equal to its strength s_{i0}/s_{ij} [3]. We find the highest score $\tilde{s}$ amongst the clusters, and select the clusters that have a score of at least $\beta\tilde{s}$. The set of images in the selected clusters is denoted as S. We set parameters k = 50, σ = 2.0 and β = 0.8.

The set of retrieved descriptors D_i is filtered by retaining descriptors corresponding to the selected database images in S. Next, for each query q_i, we compute a set of retrieved 3D points, where a matching strength for each 3D point is obtained by summing the strengths of its corresponding descriptors d_{ij}, which were computed earlier. Finally, two set of matches are constructed. The first set contains the best 3D point match for each keypoint, where the best two matches based on matching strength, passed a ratio test with a threshold of 0.75 [19]. The second set contains one-to-many 3D point matches; for each keypoint all the matches with ratios greater than 0.75 were included. The two set of matches are used for pose estimation. The first set is used for generating RANSAC hypotheses whereas the second set is used in the verification step.

2.2. Guided matching

During guided matching, other than the usual set of keypoints and query descriptors, we are also given an additional set of keypoints with known 3D point correspondences. This knowledge will be exploited to efficiently retrieve 2D-3D matches for the query set. Unlike global matching, where a voting scheme was used to narrow down the search to a few images, here, the scope is computed by inspecting the known 2D-3D correspondences. Concretely, we count the number of 3D points (from the known matches) visible in each image and then select the top 30 database images where some 3D points were visible. The k-ANN search for the query descriptors is now constrained to retrieve descriptors that belong to one of the selected images.

Although this check could have been enforced after the nearest neighbor search step, significant speedup is obtained by avoiding unnecessary distance computations during the backtracking stage of the kd-tree search. Thus, by checking the descriptor's image label, the ones that are out-of-scope can be rejected early. We take the descriptors returned by the nearest neighbor query and obtain 3D point matches from them using the steps described in Section 2.1. The final matches are obtained after geometric verification is performed on the one-to-many 2D-3D matches using the camera pose estimate computed from the known matches.

2.3. Offline Preprocessing

The offline steps of our algorithm are now summarized.

- The input images are processed using Sfm.
- The cameras are grouped into overlapping clusters.
- Keypoints are extracted in Gaussian image pyramids and multiple DAISY descriptors are computed [4].
- A kd-tree is built for all the descriptors with image labels, and appropriate lookup tables are constructed.

During online localization, we currently assume that the feauture database and map will fit into main memory. However, an out-of-core approach should be possible for larger scenes, where the map is partitioned into overlapping sub-maps, kd-trees are built for each of them and only a few relevant sub-maps need to be memory at any time.

3. Real-time Localization

In this section, we introduce our approach for 2D keypoint tracking in video. We then discuss how guided matching is interleaved with tracking and finally describe pose estimation and filtering in brief. Algorithm 1 summarizes the online algorithm for localizing frame f, given the map M, and a track table T, which is updated every frame. The table T stores the features tracks, feature descriptors, multiple 3D point match hypotheses and other attributes.

3.1. Keypoint Tracking

To track features in video, we extract Harris corners in the original frame as described earlier. Next, for a $\mu \times \mu$

[3] Each 3D point could belong to multiple overlapping clusters.

[4] Our pyramid has two octaves and four sub-octaves.

pixel, square patch around each keypoint, a 256-bit BRIEF descriptor [6] is computed. The keypoints tracked in the prior frame are compared to all the keypoint candidates in the current frame, within a $\rho \times \rho$ search window around its respective positions in the prior frame. BRIEF descriptors are compared using Hamming distance [5], and the best candidate is accepted as a match, when the ratio between the best and second-best match is less than ψ. We set parameters μ=32, ρ=48 and ψ=0.8. In Algorithm 1, TRACK-2D performs keypoint tracking. When the feature count drops below κ_1(= 25), new candidates are added (ADD-GOOD-FEATURES) in regions where there are no tracked keypoints.

BRIEF descriptors lack rotational and scale invariance, but can be computed very fast. Using BRIEF, our method can track many more features than KLT [32], for a given computational budget. Keypoint extraction is the main bottleneck in our tracker. We have tried using FAST corners, but found Harris corners to be more repeatable. We do not prune the detected Harris corners using a non maximal suppression step, but instead, select all keypoint candidates that have a cornerness value greater than an adaptive threshold. The contrast-sensitive threshold is set to $\gamma\tilde{r}$ where $\tilde{r}$ is the maximum cornerness of keypoint candidates in the previous frame and $\gamma = 0.001$. We do not perform any geometric verification during tracking, but let the subsequent RANSAC-based pose estimation step handle outliers.

3.2. Distributing matching computation

When many new keypoints are added in ADD-GOOD-FEATURES, computing their DAISY descriptors and querying the kd-tree immediately will increase the latency in those frames. However, these matches are not needed right away. Therefore, we distribute this computation over several successive frames, performing guided matching only on a small batch of keypoints at a time (usually 100–150), until all pending keypoints have been processed (*i.e.* MATCHES-PENDING returns false). Our lazy evaluation strategy also reduces the overall number of descriptors/queries computed. This is because the tracker usually drops many features right after new keypoints are inserted into the track table and by delaying the matching computation, we avoid wasting computation on keypoints that do not get tracked. A verified 2D-3D match is saved in the track table and reused as long as the keypoint is accurately tracked. When fewer than κ_2 (= 10) 2D-3D matches are available to the tracker, it relocalizes by calling GLOBAL-MATCHING.

3.3. Pose Estimation and Filtering

Given 2D-3D matches, the 6-DOF pose is robustly computed. First, RANSAC is used with three-point pose estimation [11] to find the set of inliers, after which the pose parameters are refined using non-linear least squares optimization. If less than 10 inliers are found, the estimate is rejected. Finally, the pose is filtered with a discrete Kalman filter that take position and orientation as input and estimates velocity. It assumes a constant velocity, constant angular velocity motion model similar to [9]. Currently, this is implemented as two independent filters for position and orientation[6], similar to the implementation in [20], which explains why this choice is suitable for a quadrotor MAV.

[5] computed with bitwise XOR followed bit-counting using the parallel bit-count algorithm [http://graphics.stanford.edu/ seander/bithacks.html].

Algorithm 1 $P \leftarrow$ LOCALIZE-FRAME (f, T, M)

```
KALMAN-FILTER-PREDICT ( )
P ← ∅
K ← EXTRACT-KEYPOINTS (f)
η ← TRACK-2D (f, K, T)
if η < κ1 then
   ADD-GOOD-FEATURES (f, K, T)
end if
C1 ← FETCH-2D-3D-MATCHES-FROM-TABLE ( T )
if |C1| > κ2 then
   P ← ESTIMATE-POSE ( C1 )
   if MATCHES-PENDING ( T ) then
      GUIDED-MATCHING (f, T, P, M)
   end if
else
   GLOBAL-MATCHING (f, T, M)
end if
C2 ← FETCH-2D-3D-MATCHES-FROM-TABLE ( T )
if |C2| > κ2 ∧ C1 ≠ C2 then
   P ← ESTIMATE-POSE ( C2 )
end if
KALMAN-FILTER-UPDATE ( P )
return P
```

Name	size (m.)	Cams.	Pts.	D	L	Mem.
LAB	8×5	2111	76,560	1,019,253	450	124 MB
HALL	30×12	2749	88,248	1,377,785	253	111 MB
OUTDOOR1	from [10]	1448	120,313	1,241,045	188	117 MB
OUTDOOR2	from [10]	1011	26,484	1,282,227	126	107 MB

Table 1: DATASETS: The scene size, the #cameras (Cams.) and the #3D points (Pts.) in the offline 3D reconstruction, the #DAISY descriptors (D) in the database, the #camera clusters (L) and the in-memory footprint (Mem.) of each dataset is listed.

4. Experiments

We have tested our method on ten sequences from four different scenes. This is summarized in Tables 1 and 2. The eight LAB and HALL sequences were captured from a PtGrey Firefly-MV camera at 30 fps. Two of them were acquired with the camera mounted on our quadrotor MAV, while it was flown manually. OUTDOOR 1–2 are webcam

[6] We provide details in the supplementary material [35].

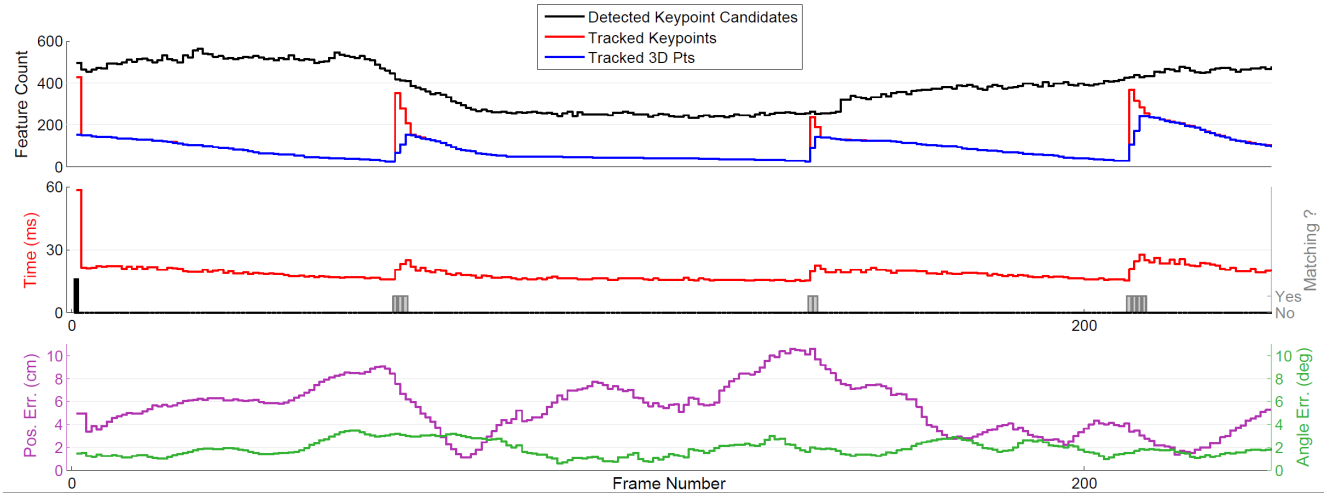

Figure 2: LAB-WALK1 SEQUENCE (237 FRAMES): [TOP] The #detected keypoints, #tracked keypoints and #keypoints with 2D-3D matches are shown. [MIDDLE] The red curves shows the per-frame processing time. Frames where guided matching and global matching is computed is shown with grey and tall black bars (first frame) respectively. [BOTTOM] The per-frame error in position (in cm.) and orientation (in degrees). The evaluation methodology is explained in the text. The maximum position error (within a 8 $\times$ 5 m. room) occurs when relatively fewer 3D points are being tracked, but this error becomes smaller as the system starts tracking more 3D points.

Map	Sequence	F	F_{loc}	T (ms.)	M	T_M (ms.)
LAB	WALK1	237	237 (100%)	19 ± 4	10 (4%)	27 ± 11
LAB	WALK2	3793	3790 (99.9%)	18 ± 3	210 (6%)	23 ± 6
LAB	FLIGHT1	1000	1000 (100%)	17 ± 2	34 (3%)	22 ± 5
LAB	FLIGHT2	1210	1204 (99.5%)	17 ± 3	47 (4%)	23 ± 7
HALL	WALK1	475	475 (100%)	17 ± 3	27 (6%)	20 ± 7
HALL	WALK2	713	712 (100%)	17 ± 2	30 (4%)	20 ± 4
HALL	WALK3	540	540 (100%)	16 ± 1	33 (6%)	18 ± 3
HALL	WALK4	201	201 (100%)	16 ± 8	4 (2%)	24 ± 7
OUTDOOR1		1033	1033 (100%)	21 ± 4	169 (16%)	25 ± 6
OUTDOOR2		605	603 (99.6%)	27 ± 10	115 (19%)	40 ± 15

Table 2: TIMINGS: The #frames (F), #frames localized (F_{loc}) and the timing on a laptop (T) (mean $\pm$ std. dev.) listed for all test sequences, as well as the number of frames in which matching was performed (M) and the average timing for those frames (T_M).

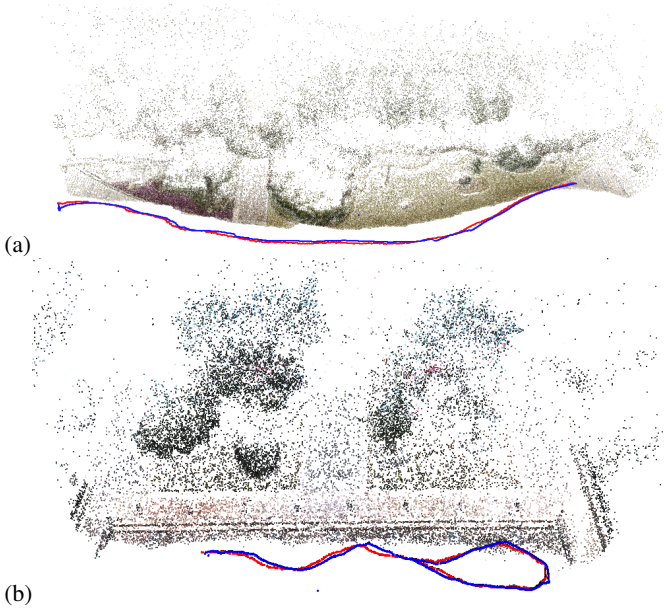

Figure 3: OUTDOOR SEQUENCES from [10]: The trajectories estimated by our method (in blue) are qualitatively similar to those computed using offline Sfm (in red).

sequences from [10], which were made publicly available online. The videos were processed at 640 $\times$ 480 resolution.

Timings. A single-threaded C++ implementation of our algorithm runs at an average frame-rate exceeding 30 Hz on all our datasets, on a laptop with an Intel Core 2 Duo 2.66GHz processor running Windows 7 (see Table 2). It is about fives times faster than the single-threaded implementation of the approach proposed in [10] with a reported frame-rate of 6Hz with a single core and 20Hz using four cores. The frame-rate of our method on the OUTDOOR datasets from [10] varies between 20 to 50Hz (see Table 2). Figure 4 shows how our efficient 2D-to-3D matching approach minimizes fluctuations in the frame-rate and lowers the processing overhead for extracting and matching DAISY features. For the two minute long LAB-WALK2 sequence, our system took 75 ms. in the first frame, 40–55 ms. for relocalization, and on average only 18 ms. per frame. The supplementary video shows our system in action [35].

Evaluation. To evaluate the accuracy of our estimates, we performed offline Sfm on the LAB-WALK1 sequence [7] and robustly registered this reconstruction to the original map.

[7] we processed all the 237 frames in addition to all the original images.

The true scale of the map was computed using known distances between scene landmarks. The camera position and orientation estimates from our method were compared to these pose estimates [8]. Figure 2 shows the error in pose alignment for the LAB-WALK1 sequence (along with other relevant statistics). The average position and orientation error for this sequence was 5.1 cm. and 1.7 degrees respectively. The size of the LAB scene was 8m $\times$ 5m.

The accuracy of our method can also be qualitatively judged from Figure 3, which shows the camera trajectories from our method and from offline Sfm on the OUTDOOR sequences. The two trajectories are well aligned. This confirms the accuracy of the position estimates, even though a quantitative evaluation was not possible due to the unknown map scale. The mean camera orientation error for the two sequences was 1.4 and 1.6 degrees respectively. According to [10], the visual SLAM system (PTAM) [15], failed on both these sequences due to fast camera motion, even when it was configured to run at slower than real-time (5 fps).

We also tested our method on the larger HALL scene which contains a narrow corridor, doorways and some textureless walls. The relatively fewer salient visual features makes 2D-to-3D matching more challenging in this scene. All frames from the four sequences were accurately localized using our approach at frame-rates exceeding 30 Hz (see Table 2). Figure 5 shows the recovered trajectories.

MAV Experiments. To evaluate the feasibility of our approach running onboard a MAV, we designed our own quadrotor vehicle mounted with the Firefly camera and a FitPC2i computer [34] running Windows 7. The FitPC, which has an Intel Atom Z550 2GHz CPU, 2GB RAM and a 64GB SSD drive, weighs less than 500gms. (incl. battery) and consumes only 10W at full load [9]. Our algorithm runs at about 12Hz on the FitPC. It was tested on the two flight sequences from the LAB dataset. All the frames processed from the FLIGHT1 sequence and 98.5% of those processed from the FLIGHT2 sequence were successfully localized [10]. The frame-rate in our case is comparable to the 5-10 Hz onboard visual SLAM system [15] used for vision-based position control [1]. These experiments show that if combined with an inertial measurement unit (IMU), our method could be used for autonomous aerial navigation in areas larger and more complex than what has been tackled before [1, 4, 20].

5. Conclusions and Future Work

In this paper, we have proposed a new approach for real-time video-based localization in scenes reconstructed offline using Sfm. Our algorithm efficiently combines keypoint tracking in video with direct 2D feature to 3D point matching, without requiring scale-invariant image features. It exploits spatio-temporal coherence, invoking expensive feature matching computations sparingly and distributes the computation over time. Our implementation can process 640 $\times$ 480 video faster than video-rate on a laptop. Preliminary tests demonstrate that it is practical for onboard computation on a micro aerial vehicle and could be practical for autonomous navigation. Our approach can fail when the camera faces a part of the scene poorly represented in the map. Using visual SLAM to dynamically extend the map in real-time to address this limitation is an interesting avenue for future work.

[8] orientation error was measured with the *angle metric*, where error between rotations R_1 and R_2 is the angle of the rotation $R_1 {R_2}^T$.

[9] Some advanced MAVs have faster onboard computers [20].

[10] When the processing exceeded 33 ms., we skipped the next frame.

References

[1] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart. Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments. In *ICRA*, 2011.

[2] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *ISMAR*, pages 73–82, 2009.

[3] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *SODA*, pages 271–280, 1993.

[4] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart. Vision based MAV navigation in unknown and unstructured environments. In *ICRA*, 2010.

[5] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *ECCV (4)'10*, pages 778–792, 2010.

[7] R. O. Castle, G. Klein, and D. W. Murray. Wide-area augmented reality using camera tracking and mapping in multiple regions. *CVIU*, 115(6):854 – 867, 2011.

[8] A. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real time markerless tracking for augmented reality: The virtual visual servoing framework. *TVCG*, 12, 2006.

[9] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *PAMI*, 26(6):1052–1067, 2007.

[10] Z. Dong, G. F. Zhang, J. Y. Jia, and H. J. Bao. Keyframe-based Real-time Camera Tracking. In *ICCV*, 2009.

[11] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981.

[12] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010.

[13] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, pages 2599–2606, 2009.

[14] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I.-S. Kweon. Pushing the envelope of modern methods for bundle adjustment. In *CVPR*, pages 1474–1481, 2010.

[15] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR*, November 2007.

[16] R. Koch, K. Koeser, B. Streckel, and J. F. Evers-Senne. Markerless image-based 3d tracking for real-time augmented reality applications. In *WIAMIS*, 2005.

[17] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *PAMI*, 28:2006, 2006.

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

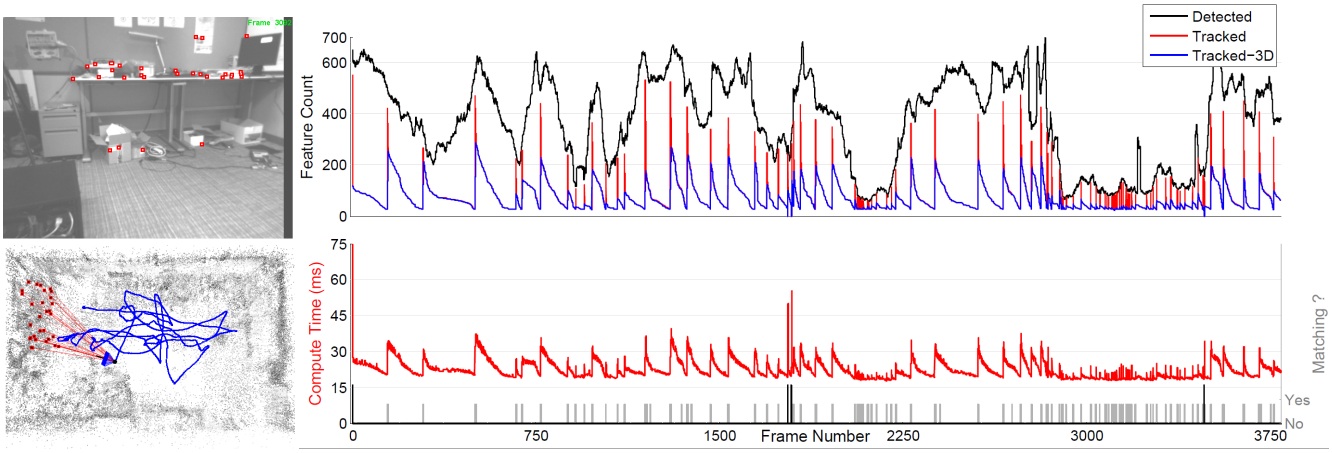

Figure 4: LAB-WALK2 SEQUENCE (3793 FRAMES): [LEFT] Frame f_{3092} and the camera trajectory for frames 1 – 3092 shown overlaid on the 3D reconstruction (top-view). [TOP-RIGHT] The #candidate keypoints (Detected), #tracked keypoints (Tracked) and #tracked keypoints with 2D-3D matches (Tracked-3D) for all the frames. [BOTTOM-RIGHT] The left axis shows the per-frame processing time (in ms.) and the short-grey and tall-black bars indicate the frames in which guided matching and global matching was invoked. Tracking was lost twice around f_{1800} and once around f_{3500} requiring relocalization. In spite of the lower tracking efficiency between frames $f_{2000} - f_{2200}$ and $f_{2800} - f_{3500}$, when relatively fewer keypoints were detected (in a dark corner of the room), the camera was consistently localized.

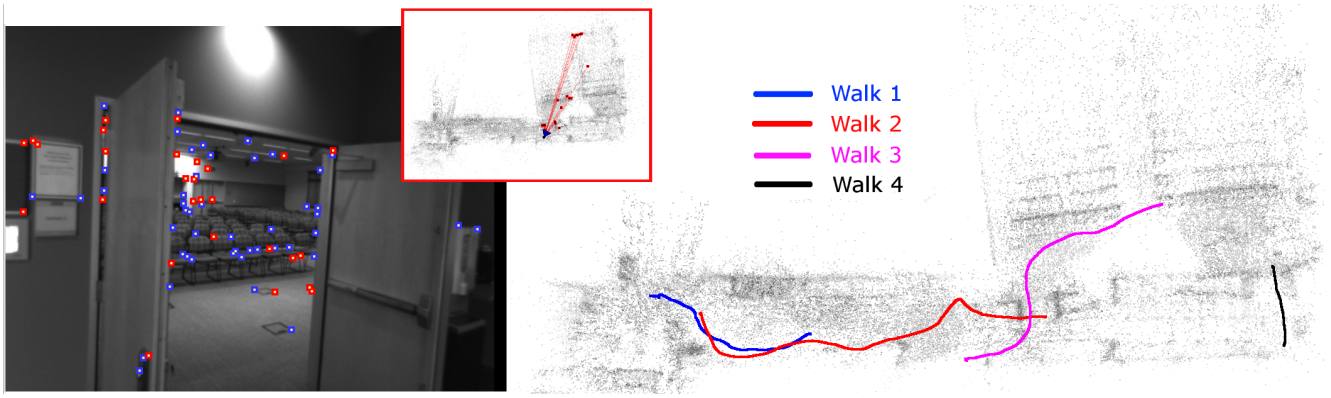

Figure 5: HALL-WALK SEQUENCES: [LEFT] A frame from the WALK3 sequence and the corresponding camera pose estimate shown in the top-view of the map (inset). [RIGHT] Computed trajectories of the camera in the corridor (WALK 1–2), and entering the room through different doors (WALK 3–4). Our approach worked well in this scene despite the presence of many ambiguous and confusing features.

[19] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[20] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. PIXHAWK: A system for autonomous flight using onboard computer vision. In *ICRA*, pages 2992–2997, 2011.

[21] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168, 2006.

[22] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *PAMI*, 32, 2010.

[23] D. Robertson and R. Cipolla. An image-based system for urban navigation. In *BMVC*, pages 819–828, 2004.

[24] E. Royer, M. Lhuillier, D. Michel, and J.-M. Lavest. Monocular vision for mobile robot localization and autonomous navigation. *IJCV*, 74:237–260, Sep 2007.

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

[26] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR (0)*, pages 1–7, 2007.

[27] S. Se, D. G. Lowe, and J. J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21:364–375, 2005.

[28] I. Skrypnyk and D. G. Lowe. Scene modelling, recognition and tracking with invariant image features. *ISMAR*, 0, 2004.

[29] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the World from Internet Photo Collections. In *IJCV*, 2008.

[30] D. Ta, W. chao Chen, N. Gelfand, and K. Pulli. SURFTrac: Efficient Tracking and Continuous Object Recognition using Local Feature Descriptors. In *CVPR*, 2009.

[31] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *CVPR*, 2008.

[32] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, CMU, 1991.

[33] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *TVCG*, 16:355–368, 2010.

[34] FitPC2i. `http://www.fit-pc.com/web/`.

[35] Supplementary Material. `http://research.microsoft.com/en-us/um/redmond/groups/ivm/loc6dof`.

[36] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In *ICCV*, 2007.

[37] S. A. J. Winder, G. Hua, and M. Brown. Picking the best DAISY. In *CVPR*, pages 178–185. CVPR, 2009.