Scalable 6-DOF Localization on Mobile Devices

Sven Middelberg[1], Torsten Sattler[2], Ole Untzelmann[1], and Leif Kobbelt[1]

[1] Computer Graphics Group, RWTH Aachen University, Aachen, Germany
[2] Department of Computer Science, ETH Zürich, Zürich, Switzerland

Abstract. Recent improvements in image-based localization have produced powerful methods that scale up to the massive 3D models emerging from modern Structure-from-Motion techniques. However, these approaches are too resource intensive to run in real-time, let alone to be implemented on mobile devices. In this paper, we propose to combine the scalability of such a global localization system running on a server with the speed and precision of a local pose tracker on a mobile device. Our approach is both scalable and drift-free by design and eliminates the need for loop closure. We propose two strategies to combine the information provided by local tracking and global localization. We evaluate our system on a large-scale dataset of the historic inner city of Aachen where it achieves interactive framerates at a localization error of less than 50cm while using less than 5MB of memory on the mobile device.

1 Introduction

Determining both position and orientation of a camera relative to a 3D model of a scene, also referred to as the image-based localization problem [26], is an important step in many applications, such as location recognition [21,26], navigation [10,16,30,22,24], and Augmented Reality (AR) [17,8,18,25].

We distinguish between two fundamental approaches to solve the image-based localization problem. *Local* methods operate in unknown environments, using simultaneous localization and mapping (SLAM) techniques to concurrently construct a 3D model and estimate the camera pose relative to it [8,9,11,17]. Due to the limited computational capabilities of mobile devices like smartphones and tablet PCs, local approaches for mobile localization, *e.g.*, PTAM [8,17], are usually confined to small workspaces. In contrast, *global* localization approaches assume that a 3D model of the scene is already available. Recent progress in Structure-from-Motion (SfM), allowing the rapid reconstruction of large scenes consisting of millions of 3D points [1], has lead to a focus on global image-based localization methods that are able to scale to such large datasets [21,26]. These approaches achieve impressive localization performance, but rely on powerful and memory intensive local descriptors such as SIFT [23] to establish 2D-3D matches between image features and scene points. As a result, they require computation and memory capacities that significantly exceed what current mobile devices can offer. Especially localization scenarios for micro aerial vehicles [30,24] place

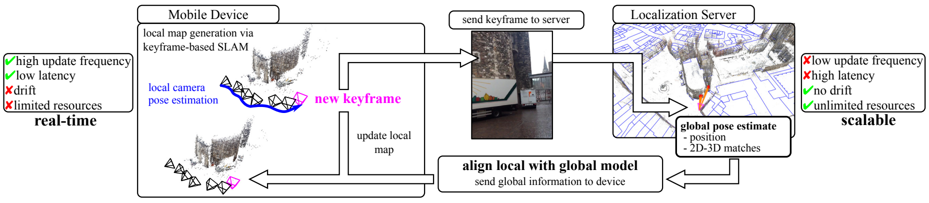

Fig. 1. We use a small local map, constructed on the mobile device, to perform real-time pose estimation. The keyframes used for the reconstruction are also send to a localization server to align the local map to the server reconstruction.

very hard restrictions on the available hardware due to weight and power constraints. Nevertheless, the flexibility of mobile devices creates a strong need for large-scale mobile localization, *e.g.*, for city-scale augmented reality. Recently, Lim *et al.* [22] and Ventura and Höllerer [29] proposed approaches for real-time, large-scale mobile pose tracking. However, both approaches require to keep a 3D model of the environment on the device, which can already consume more than 100MB for a scene of size 8m $\times$ 5m [22], limiting their applicability for mobile devices with their hard memory restrictions.

In this paper, we propose a truly scalable approach that is both real-time capable and memory efficient and thus ideally suited for mobile devices. The key idea, illustrated in Fig. 1, is to combine the information provided by local pose estimation running on the device and a global localization method running on an external server. Our system tracks the camera relative to a local map, built and maintained on the device using keyframe-based SLAM [17], in real-time. The localization server then provides global position estimates for the keyframes, allowing us to align the local map with the global one. Since the device only needs a small map of the relevant part of the scene, the run-time of the resulting algorithm is *independent of the size of the global model.* Since the global pose estimates are drift-free, there is no need to explicitly handle loop-closures.

2 Overall Approach

While very intuitive, our combination of local and global localization introduces some interesting challenges, as it requires to align two different coordinate systems. For this, we need to take into account that the accuracy of global pose estimates is unknown and incorporate means to detect and handle inaccurate global poses. In addition, the local mapping and tracking method has to be robust enough to handle a delay of up to multiple seconds before the global pose estimates are available, which is caused by the latency of querying the localization server. As the main contribution of our paper, we therefore propose two distinct alignment strategies, compare them experimentally in terms of localization accuracy and demonstrate that both strategies result in an image-based localization system that runs in real-time on mobile devices.

The paper is structured as follows. The remainder of this section reviews related work. Sec. 2 gives an overview of our approach, while Sec. 3 explains the local and global localization approaches employed by our method. Sec. 4 then details the two alignment strategies. Sec. 5 provides an experimental evaluation of the resulting localization systems.

Related Work. Classical SLAM approaches simultaneously estimate in every frame both the current position of the camera and a 3D representation of the scene [9,11]. As the 3D maps grows larger, the mapping part of SLAM becomes more expensive. To handle larger maps, *parallel tracking and mapping* (PTAM) approaches decouple the tracking and mapping part into two separate threads to prevent that mapping slows down tracking [17]. The mapping thread uses SfM techniques such as bundle adjustment [28] to generate a high quality map from certain *keyframes*. Advanced PTAM approaches are able to handle multiple 3D maps [8] or to run on mobile phones [18]. In contrast to SLAM and PTAM, which construct sparse maps from features found in images, *dense tracking and mapping* (DTAM) builds a dense 3D model by estimating depth maps [25]. This allows to track every pixel using the photometric error, which is much more stable under rapid movement than feature-based tracking. However, a powerful GPU is required to construct the map in real-time.

The localization approach of Arth *et al.* [5] uses SURF-like features [6] to determine the pose of a camera relative to a small 3D model. The model is kept out-of-core and is manually divided into multiple segments that fit into the memory of a mobile phone. Lim *et al.* propose to avoid complex, scale-invariant features [22]. Starting from an SfM point cloud, they extract more efficient descriptors for every 3D point across multiple scales to simulate scale-invariance. Once a pose is estimated from 2D-3D matches, they try to track the matching points over time using binary BRIEF descriptors [7]. Direct 2D-3D matching is only triggered if the number of currently tracked points is less than a threshold. The matching is distributed over several frames to keep processing times low. Ventura and Höllerer [29] propose to query an image-based localization server to estimate the camera pose of the first image, allowing subsequent real-time pose tracking relative to the server-side 3D model, which is also kept locally on the mobile device. To efficiently match features in the following images to this model, the pose of the previous image is used to cull 3D points behind the camera and outside the image. Image features are then matched to the remaining points via a patch-based approach similar to that of Klein and Murray [18].

In order to obtain an image-based localization method that is able to run in real-time on mobile devices with restricted computational and memory capabilities and that scales to very large datasets, *e.g.*, to a city-scale, we exploit the distinct advantages of local and global pose estimation approaches. Currently, global localization approaches do not achieve the interactive response times required for continuous camera pose tracking and are computationally too complex to run on mobile devices [21,26]. Thus, we employ a keyframe-based PTAM technique [17] to construct a local map of the environment on the mobile device itself and use it to track the camera pose in real-time [17,8]. While the keyframes are used to reconstruct the scene locally, we also send the corresponding images to a localization server, which computes and returns global localization results, *e.g.*, the position of the keyframe with respect to the server's 3D model. This global information is utilized to align the local map to the global server-side reconstruction. Hence, having properly aligned the local map to the global reconstruction, local tracking produces globally registered camera poses. Since the camera pose is tracked solely with respect to the globally aligned local map, the processing time and memory footprint on the mobile device only depend on its size, which, in turn, scales with the number of keyframes. The number of keyframes, however, increases with the length of the sequence. Thus, we keep only the currently relevant parts of the scene in the local map, by limiting the number of keyframes.

The crucial aspect of our approach is the alignment of the local map to the global reconstruction. If done right, this alignment implicitly avoids that our local model is affected by drift since the global localization is drift free. However, a simple affine transformation between the two coordinate frames is clearly not enough as avoiding drift requires us to non-affinely adapt the local map. To avoid drift, we need to carefully integrate the global information directly into the construction and refinement of the local map. There are two major aspects that need to be considered when performing such an integration. The first is that both the poses estimated locally on the device itself and the global information are not perfect but have certain, unrelated errors, which have to be handled accordingly to obtain a stable alignment, which preserves the consistency of the local map. The second challenge lies in the latency induced by sending an image to the server over 3G, 4G, or WiFi. The latter problem is overcome by decoupling pose tracking with respect to the local map from mapping and global alignment using two separate threads as in PTAM [17]. This way, local pose tracking can be continued, while the mapping thread handles the global alignment.

As a main contribution of this paper, we propose two strategies for incorporating global pose information into the local-to-global alignment process. The first strategy adapts the local map in a way that it aligns the keyframe positions estimated locally on the device with the corresponding global positions provided by the server, similar to the approach for fusion of SfM and GPS proposed by Lhuillier [20]. While this is a quite intuitive strategy, it is susceptible to inaccuracies in the global pose estimates. We therefore propose a second strategy that directly relies on the global 2D-3D matches used by the server to estimate the global poses instead of the keyframe positions themselves. We will discuss these two strategies in more detail in Sec. 4 before we compare them experimentally in Sec. 5. In the following, we offer a brief description of the local and global pose estimation methods currently employed by our method.

3 Local and Global Pose Estimation

Our localization approach consists of two separate modules handling the local and global camera pose estimation, respectively. The local pose estimation part, performed on the mobile device itself, has to run in real-time. Thus, we employ the inexpensive BRISK detector and its binary descriptor [19]. We consider only one scale-space octave and use a 256 bit, scale- but non-rotation-invariant version of the descriptor. The task of the global localization process, running on an external server, is to provide the information required to align the local map to the globally, geo-registered server reconstruction.

3.1 Local Pose Tracking Using SLAM

Following a PTAM approach [17], we use separate CPU threads for camera pose estimation and local map construction and alignment, since the latter does not need to run in real-time. We use some of the optimizations proposed by Lim *et al.* [22], such as simpler descriptors for tracking, to accelerate localization.[1]

Initializing the Local Model. Given two images, taken from viewpoints different enough to guarantee a certain baseline and similar enough to share enough feature matches, we initialize the local 3D model of the scene using SfM techniques [17]. Since IMUs belong to the standard equipment of modern smartphones and tablets, we exploit knowledge about the gravity direction by using a 3-point-relative-pose solver [13] to estimate the essential matrix inside a RANSAC loop [12]. We assume that the camera of the device is calibrated, but the internal calibration required by the solver could also be provided from the localization server. The initial local map is then obtained using triangulation followed by bundle adjustment [28].

Local Camera Pose Estimation. Following Lim *et al.*, we try to avoid BRISK descriptor matching between image features and scene points. Instead, we prefer to track already matched 3D points using simple 8×8 intensity patches as local descriptors. We extract the intensity patches only for BRISK keypoints in a $24{\times}24$ window surrounding the positions of the 3D points projected into the previous frame. The camera pose is then estimated using a 3-point-absolute-pose solver inside a RANSAC loop [12]. We consider a pose as valid if it has at least 10 inliers. To account for changes in viewpoint and scene structure that may occur during long sequences, we adapt the BRISK detection threshold dynamically following a simple strategy. With each new keyframe, we set the detection threshold to the maximal value, with which at least a number of η keypoints are found in the image. However, we enforce a lower limit of 30 for the detection threshold, to prevent the detection of features due to image noise. The value η is a framework parameter, which will be evaluated in Sec. 5.

Local Map Update. Once we have determined the pose of the current frame, we check whether it fulfills the keyframe properties and should thus be used to extend the local map. As in PTAM [17], the current frame becomes a keyframe if it could be successfully localized and it has a certain minimal distance to all previous keyframes. Additionally, to account for fast camera motion and viewpoint changes, we select a frame as a keyframe if the number of pose inliers is low and the majority of inliers is located in either the lower, upper, left or right half of the image. We then triangulate new map points from 2D-2D matches to the previous keyframes and apply bundle adjustment [28]. In order to limit the

[1] An iOS demo is available at `http://www.graphics.rwth-aachen.de/localization`.

computational load of maintaining the local map and its size in memory, we limit the number of keyframes to κ, which is an additional framework parameter. If more than κ keyframes are present, we delete the one that has the largest distance to the new keyframe. All associations between the deleted keyframe and points in the local map are removed. Finally, 3D points that are not observed by at least two of the remaining keyframes are deleted entirely.

3.2 Server-Based Global Localization

Besides using the keyframes to extend and refine the local map, we also send each of them to the localization server to obtain localization information relative to a large, global model, which covers the entire scene. We assume that the global model was reconstructed using SfM and we associate each 3D point with the RootSIFT descriptors [3,23] found in the images used for feature matching. Once we receive a keyframe, we employ the publicly available implementation by Sattler *et al.* [26] to establish 2D-3D matches between SIFT features found in the received image and the 3D points in the scene. Since we assume a calibrated camera, we also send its focal length to the server and apply a 3-point-absolute-pose solver to estimate the camera pose inside a RANSAC loop [12]. The resulting pose is accepted as correct if it has at least 12 inliers [26] and we subsequently refine the pose iteratively. Besides reporting the estimated position to the mobile device, we also send the 2D feature and 3D point positions of the inlier matches used to estimate the poses. Since [26] stops matching after finding 100 matches, the overhead of additionally transmitting the inlier data is negligible.

4 Aligning the Local Map Globally

The two crucial aspects of our localization approach are the alignment of the local map to a global, well-defined coordinate frame, as well as the prevention of drift. The latter cannot be accomplished by loop-closure methods, since only a constant number of keyframes is kept. Instead, the key idea of our framework is to manage both issues by exploiting the global, drift-free information provided by the localization server. The most simple approach to obtain such an alignment is to compute the affine transformation that best aligns the local and global keyframe positions. However, such a transformation does not correct inherent inaccuracies of the local map and thus cannot prevent drift. Instead, we need to incorporate the global localization information directly into the construction of the local map to enable a non-affine alignment. In the following, we first introduce our notation. We then detail our two alignment strategies outlined in Sec. 2. Both strategies are based on the idea of using global information in the bundle adjustment to "anchor" the local map. To prevent drift, we incorporate these constraints not only into the initial bundle adjustment after the map has been initialized, but also into the bundle adjustments that go along with the keyframe updates. The first method thereby uses the global keyframe positions while the second exploits the global 2D-3D matches between the images and the server reconstruction that were originally used to estimate the global poses. One might argue that it would also be reasonable to use 3D-3D correspondences between the local and global map points in the alignment. However, the local points are reconstructed from BRISK features [19] and the global points from SIFT features [23]. Thus, we do not expect a significant overlap between both point sets. Since we aim to achieve an as loose coupling as possible between the local tracker and the global localization approach to maintain a high degree of modularity, we consider the requirement that both map points are reconstructed from the same features too restrictive. For this reason, we do not explore the use of 3D-3D correspondences, although it might help in the alignment.

Let $\mathbf{P}_j^L$ and $\mathbf{P}_j^G$ denote the position of the j-th 3D point in the local and global map, respectively. Consequently, let $\mathbf{p}_j^i$ be the observed image position of the j-th point in the i-th keyframe. The local and global poses of a keyframe are defined by rotation matrices $\mathtt{R}_i^L$, $\mathtt{R}_i^G$ and positions $\mathbf{t}_i^L$, $\mathbf{t}_i^G$ in the local and global coordinate system, respectively. Furthermore, let $\mathtt{K}_i$ denote the intrinsic matrix of the i-th keyframe. Since we assume a calibrated setting, $\mathtt{K}_i$ is known. Bundle adjustment adapts the camera parameters $\mathtt{R}_i^L$, $\mathbf{t}_i^L$ and points $\mathbf{P}_j^L$ in order to minimize the sum of squared reprojection errors

$$\chi_1 = \sum_i \sum_j \delta_{i,j} \cdot d(\mathbf{p}_j^i, \mathtt{K}_i \mathtt{R}_i^L (\mathbf{P}_j^L - \mathbf{t}_i^L))^2 \ . \tag{1}$$

Here, $\delta_{i,j} \in \{0, 1\}$ indicates whether the j-th point is visible in the i-th keyframe and $d(\mathbf{x}, \mathbf{y})$ is the 2D Euclidean distance between the 2D positions $\mathbf{x}$ and $\mathbf{y}$. The term $\mathtt{K}_i \mathtt{R}_i^L (\mathbf{P}_j^L - \mathbf{t}_i^L)$ is the projection of the j-th point onto the i-th keyframe.

Alignment Using the Global Keyframe Positions. Assuming that the global keyframe positions $\mathbf{t}_i^G$ returned by the server are sufficiently accurate, our first strategy to align the local and global 3D models tries to enforce that the local keyframe positions $\mathbf{t}_i^L$ closely match the estimates provided by the server. Therefore, we want to minimize the sum of squared Euclidean distances

$$\chi_2 = \sum_i \left|\left|\mathbf{t}_i^G - \mathbf{t}_i^L\right|\right|^2 \tag{2}$$

between the local and global keyframe positions. In order to be robust to errors in the global localization, we add a keyframe constraint $\left|\left|\mathbf{t}_i^G - \mathbf{t}_i^L\right|\right|^2$ to χ_2 only if the Euclidean distance between $\mathbf{t}_i^L$ and $\mathbf{t}_i^G$ is less than 2 meters. When refining the local map, we thus need to minimize both the sum of reprojection errors and the distances between the keyframe positions, resulting in the energy functional

$$e^1 = \chi_1 + \chi_2 \ . \tag{3}$$

Notice that the first term χ_1 minimizes a geometric error while the second term χ_2 minimizes an algebraic error. To combine both error types, we follow the idea of [20] and normalize our objective function using the initial values χ_1^0, χ_2^0 obtained by evaluating the terms using the original, unoptimized camera poses, to normalize both error terms. Thus the objective function becomes

$$e_\alpha^1 = (1 - \alpha) \cdot \chi_1 / \chi_1^0 + \alpha \cdot \chi_2 / \chi_2^0 \ , \tag{4}$$

where the parameter $\alpha \in [0, 1]$ is used to weight the energy functionals χ_1 and χ_2 differently. A high value for α enforces a stronger alignment of the local and global keyframe positions while potentially lowering the quality of the local map. On the other hand, a low value for α preserves the consistency of the local map but may result in a poor alignment. Thus, the choice for α strongly influences the quality of global pose estimation and is evaluated experimentally in Sec. 5.

The minimization of e^1_α has to be carried out iteratively by a non-linear least squares solver, which requires an adequate initialization. For this purpose, we compute an affine transformation $\mathtt{T}_1(\mathbf{t}^L_i) = \mathtt{s}\mathtt{R}\mathbf{t}^L_i + \mathtt{t}$, with a scaling factor $\mathtt{s}$, a rotation $\mathtt{R}$ and a translation $\mathtt{t}$, that minimizes $\sum_i \left|\left|\mathbf{t}^G_i - \mathtt{T}_1\left(\mathbf{t}^L_i\right)\right|\right|^2$. Since this transformation adapts the local map only affinely, this gives us a sufficient initial alignment while it preserves the consistency of the local map. Following the approach by Horn [15], we need at least three pairs $\left(\mathbf{t}^L_i, \mathbf{t}^G_i\right)$ to estimate $\mathtt{T}_1$. However, if the server reconstruction is aligned with gravity, we can again utilize the gravity vectors provided by the device IMU to reduce the degrees of freedom of $\mathtt{T}_1$ from 7 to 5, which enables us to compute $\mathtt{T}_1$ from only two pairs $\left(\mathbf{t}^L_i, \mathbf{t}^G_i\right)$. Having computed $\mathtt{T}_1$, we replace every local point $\mathbf{P}^L_j$ by $\mathtt{T}_1(\mathbf{P}^L_j)$ and every camera pose $(\mathtt{R}^L_i, \mathbf{t}^L_i)$ by $\mathtt{T}_1(\mathtt{R}^L_i, \mathbf{t}^L_i) = (\mathtt{T}_1(\mathtt{R}^L_i), \mathtt{T}_1(\mathbf{t}^L_i))$, where $\mathtt{T}_1(\mathtt{R}^L_i) = \mathtt{R}^L_i\mathtt{R}^\mathsf{T}$. Note that this initial alignment has to be performed only once, when the local map is not yet aligned to the global coordinate frame. For later map updates, we assume that the local map is already adequately aligned to be used as initialization for the minimization of e^1_α directly.

Alignment Using the Global 2D-3D Matches. The strategy described above implicitly assumes that the positions computed by the localization server are accurate enough to improve the quality of the local map. However, this assumption might be too strict. For example, the server localization approach might find 2D-3D matches only in a small region of the keyframe. This results in an unstable configuration for pose estimation and thus a larger error in the global position, which negatively impacts the alignment. However, a central observation is that even if the pose is inaccurate, most of the inlier 2D-3D matches from which it was estimated are still correct. Instead of relying on the accuracy of the global poses, our second strategy thus directly incorporates these global inlier matches into the bundle adjustment using the sum of squared global reprojection errors

$$\chi_3 = \sum_i \sum_k \delta_{i,k} \cdot d(\mathbf{p}^i_k, \mathtt{K}_i\mathtt{R}^L_i(\mathbf{P}^G_k - \mathbf{t}^L_i))^2 \ , \tag{5}$$

resulting in the unweighted and weighted objective functions

$$e^2 = \chi_1 + \chi_3 \ , \tag{6}$$

$$e^2_\beta = (1-\beta) \cdot \chi_1/n_L + \beta \cdot \chi_3/n_G \ , \tag{7}$$

where n_L is the number of local matches and n_G is the number of global matches. In contrast to the first strategy, our second approach does not rely on accurate global pose estimation and integrates much more naturally into the map refinement process as only a single type of error is minimized. Again, a framework parameter $\beta \in [0, 1]$ is used to weight χ_1 and χ_3 differently. Although both χ_1 and χ_3 represent the same type of error, a normalization is required since there are usually many more local than global matches. However, compared to the first strategy, the second strategy enables a fairer weighting of both χ_1 and χ_3 using the number of local and global matches. Notice that we do not refine the global 3D point positions $\mathbf{P}_k^G$. Since the global SfM point cloud was reconstructed offline using state-of-the-art SfM techniques, we can assume that they are highly accurate and thus do not need to be refined. Incorporating the globally matching points thus prevents drift in the local map without adding additional degrees of freedom to the optimization process.

Unlike χ_2, the sum of squared global reprojection errors χ_3 is highly susceptible to errors in the poses $\{\mathtt{R}_i^L, \mathbf{t}_i^L\}$. Without proper initialization, we observed convergence to bad local minima of e_β^2. Thus, we have to take further precautions to ensure a good initialization for the initial and all further map alignments as newly inserted keyframes may have poor local pose estimates. Thus, for the first alignment, the initialization with $\mathtt{T}_1$ is followed by a second affine mapping

$$\mathtt{T}_2 = \arg\min_{\mathtt{T}} \sum_i \sum_k \delta_{i,k} \cdot d(\mathbf{p}_k^i, \mathtt{K}_i \mathtt{T}(\mathtt{R}_i^L)(\mathbf{P}_k^G - \mathtt{T}(\mathbf{t}_i^L)))^2 \ , \tag{8}$$

which is computed with a non-linear least squares solver. Since $\mathtt{T}_2$ does not affect the local reprojection errors, but minimizes the global reprojection errors it results in a better initialization for the minimization of e_β^2.

For subsequent alignments, we have to make sure that the pose estimate of a newly inserted keyframe is not only optimal with respect to the local, but also with respect to the global matches. Thus, in contrast to the first strategy, the local pose of a new keyframe is computed in a RANSAC-PnP solver [12] from both the local and the global matches. We expect a significantly higher number of local than global matches, since the global localization approach by Sattler *et al.* [26] returns at most 100 global matches as mentioned in Section 3.2. Thus, since we jointly process both types of matches in a RANSAC loop, this additionally allows to detect and remove outliers in the global matches, making the strategy robust to erroneous global localizations.

5 Experimental Evaluation

We experimentally evaluated our framework and the proposed alignment strategies on a second generation iPad Mini, which provides a dual-core 1.3 GHz Apple A7 CPU and 1GB of memory. In our implementation, we make intensive use of the Grand Central Dispatch multithreading API and the ARM Neon SIMD instruction set, *e.g.*, for detection, description, and matching of BRISK features. For bundle adjustment and other non-affine refinements, we use the Google Ceres optimization library [2]. The server reconstruction covers the historic inner city of Aachen (about 40k m^2), was computed from 3k images and consists of 1.5M 3D points and 7.28M SIFT descriptors [27]. In the following, we present evaluation results for 3 different sequences, which were captured with

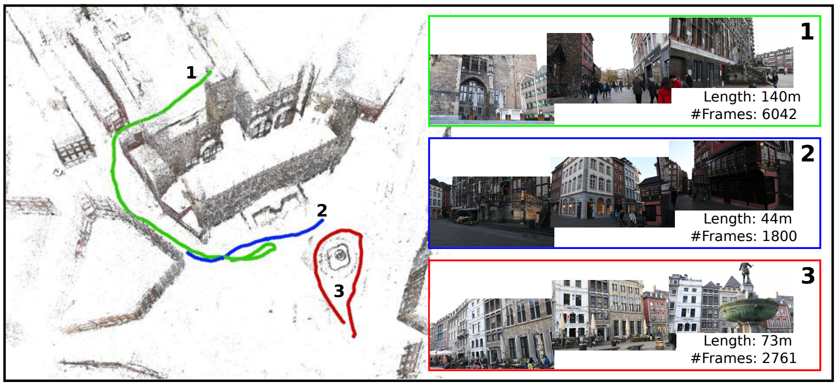

Fig. 2. Server reconstruction and ground truths of the evaluated sequences (*left*). Sample images from the sequences (*right*).

a Canon EOS 5D Mark II camera at a resolution of 1920×1080 and processed on the iPad at a resolution of 1024×576. The field of view has been adjusted to resemble that of mobile devices. We obtained ground truth pose estimates for each sequence by SfM, where we incrementally added the sequence images to the server reconstruction. For the initialization of the local map, we selected the first keyframe manually and set the second keyframe to the first sequence frame, which allowed us to evaluate the entire sequence. The gravity vectors were extracted from the ground truths. The low quality of current mobile device cameras accompanied by, *e.g.*, strong rolling shutter distortions prevented us from capturing trackable evaluation sequences with the iPad camera itself. However, with increasing quality of built-in cameras and ongoing research with the aim to overcome these issues [4,14], we expect our approach to be applicable to actual mobile device cameras in the near future. The server reconstruction and the evaluation sequences are depicted in Fig. 2.

5.1 Comparison of the Proposed Alignment Strategies

We compare the mean position and orientation error of both proposed alignment strategies e^1_α and e^2_β for Seq. 1, 2 and 3 and several choices for the weighting parameters α and β. Additionally, we also evaluated the pose accuracy for the respective unweighted alignments. To compute the position and orientation errors we compared each pose estimate against ground truth.

Table 1 reports the results. The orientation error for the objective functions e^1 and e^1_α is always large, since this strategy does not incorporate any global orientation information. With respect to position error, the first strategy delivers the best results for $\alpha \geq 0.4$. Due to a significantly higher number of local than global matches, the unweighted functional of the second strategy e^2 performs poorly compared to the weighted versions. The best localization results are obtained for $\beta \geq 0.3$. As predicted, these results show that the second strategy outperforms

Table 1. Impact of α, β on mean position and orientation error. The first entry in each cell is the position error [m], the second the orientation error [Deg].

Seq.	e^1	$e^1_{0.1}$	$e^1_{0.2}$	$e^1_{0.3}$	$e^1_{0.4}$	$e^1_{0.5}$
1	0.52/4.96	0.27/4.03	0.78/12.7	0.40/6.55	0.30/4.33	0.27/4.10
2	0.52/2.39	0.54/13.7	0.40/10.9	0.43/9.42	0.31/5.83	0.34/7.99
3	0.24/1.81	0.20/2.42	0.19/1.79	0.17/2.41	0.21/4.40	0.17/2.50
Seq.	e^2	$e^2_{0.1}$	$e^2_{0.2}$	$e^2_{0.3}$	$e^2_{0.4}$	$e^2_{0.5}$
1	1.42/0.54	0.15/0.38	0.16/0.40	0.18/0.39	0.17/0.40	0.17/0.39
2	7.12/2.21	0.80/1.23	0.67/1.07	0.30/0.78	0.33/0.94	0.29/0.93
3	1.89/0.95	0.15/0.47	0.12/0.42	0.12/0.42	0.12/0.42	0.11/0.41

Table 2. Localization accuracy and timings for several choices of η (*left*) and κ (*right*). The first entry in each cell is the position error [m], the second entry the orientation error [Deg.], the third entry is the frame processing time [ms].

η	Seq. 1	Seq. 2	Seq. 3
500	0.19/0.38/42	0.24/0.76/39	0.11/0.40/41
1000	0.16/0.37/46	0.30/0.89/47	0.10/0.39/48
1500	0.17/0.39/53	0.29/0.93/61	0.11/0.41/53
2000	0.17/0.42/65	0.28/0.77/63	0.13/0.44/62

κ	Seq. 1	Seq. 2	Seq. 3
5	0.21/0.41/48	0.31/0.85/47	0.13/0.44/45
10	0.18/0.37/49	0.27/0.79/51	0.13/0.43/50
15	0.17/0.39/53	0.29/0.93/61	0.11/0.41/53
20	0.16/0.40/59	0.28/0.76/59	0.13/0.43/56

the first. Thus, if not explicitly stated otherwise, the following experiments were performed with alignment strategy $e^2_{0.5}$. One could argue that the first strategy could be improved by incorporating global keyframe orientations. While this is correct, we highly doubt that this would achieve the accuracy of the second strategy, since it would still rely on a single global pose estimate per keyframe. Furthermore, while the incorporation of the global matches results in an intuitive and simple objective function, the incorporation of orientations would require complex functions involving errors in the tangent space of SE(3).

Our framework depends on two additional parameters: the desired number of image features η and the maximum number of keyframes κ. Table 2 details the impact of different choices for η and κ. While these parameters do not affect the pose accuracy significantly, we observed that the smoothness of the pose estimates decreases with low values for η and κ. We decided for values of $\eta = 1500$ and $\kappa = 15$, which is a compromise in smoothness and processing time.

5.2 Accuracy, Efficiency and Scalability of Pose Estimation

Table 3 details localization results for the alignment strategies $e^1_{0.5}$ and $e^2_{0.5}$. Both strategies deliver accurate positions with mean errors between 0.11 and 0.34 m. However, only strategy $e^2_{0.5}$ is able to accurately estimate camera orientations. Furthermore, the standard deviations of the position and orientation errors are smaller for strategy $e^2_{0.5}$, which results in smoother trajectories. The mean size of the map is below 4MB for each sequence and for both strategies, which shows that our approach is suitable for large-scale mobile localization.

Fig. 4 depicts per frame statistics for Seq. 1 and strategy $e^2_{0.5}$. The threshold adaption is working properly as we constantly detect about 1500 features. The pose inaccuracies between frame 4000 and 4500 are caused by a rapid viewpoint

Table 3. Mean position and rotation error, processing time and map size for the alignment strategies $e^1_{0.5}$ and $e^2_{0.5}$

Seq.	$e^1_{0.5}$				$e^2_{0.5}$			
	P.Err.[m]	R.Err.[Deg.]	Time[ms]	Map[MB]	P.Err.[m]	R.Err.[Deg.]	Time[ms]	Map[MB]
1	0.27 ± 0.24	4.10 ± 4.48	53.8	3.99	0.17 ± 0.11	0.39 ± 0.23	52.8	3.20
2	0.34 ± 0.18	7.99 ± 6.27	56.4	2.79	0.29 ± 0.16	0.93 ± 0.68	60.1	3.41
3	0.17 ± 0.11	2.50 ± 2.73	54.0	3.17	0.11 ± 0.10	0.41 ± 0.33	52.6	3.98

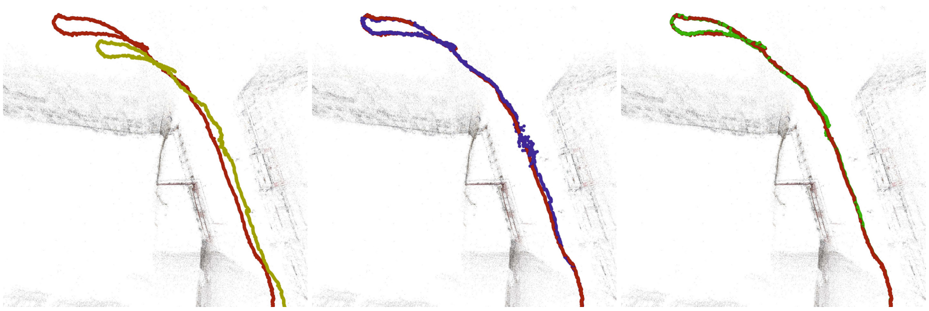

Fig. 3. Part of Seq. 1 with a one-off alignment from the first two keyframes (*yellow*), strategies $e^1_{0.5}$ (*blue*), $e^2_{0.5}$ (*green*) and ground truth (*red*). Strategies $e^1_{0.5}$ and $e^2_{0.5}$ are well aligned to ground truth, while the one-off alignment is affected by drift. Strategy $e^2_{0.5}$ produces a smoother trajectory than $e^1_{0.5}$.

change. The peaks in the timing plot are caused by BRISK descriptor extraction and direct 2D-3D matching, which is triggered only if feature tracking fails. Besides feature detection, these tasks are most time consuming, as depicted in Fig. 5, which gives mean timings for the main components of pose estimation. As proposed by Lim *et al.* [22], descriptor extraction and direct matching could be distributed among multiple consecutive frames to achieve smoother timings, leaving feature detection as the main bottleneck. The total number of direct matching frames for Seq. 1 was 1304. The bottom curve in Fig. 4 demonstrates that the map size is nearly constant as soon as κ keyframes are in the map. Fig. 3 compares the trajectories for Seq. 1 and strategies $e^1_{0.5}$ and $e^2_{0.5}$, as well as a third trajectory that was aligned only once, from the first two keyframes, with ground truth. A single, initial alignment is not sufficient for robust large-scale pose estimation. However, by careful incorporation of additional global information with every new keyframe, the proposed strategies are able to prevent drift. Furthermore, strategy $e^2_{0.5}$ results in a much smoother trajectory than $e^1_{0.5}$.

Comparison to State-of-the-Art. We additionally evaluated our framework on the FLIGHT1 sequence of the state-of-the-art approach by Lim *et al.* [22]. Fig. 6 shows that the alignment strategy $e^2_{0.5}$ produces a qualitatively similar trajectory as the ground truth. Since we do not know the actual scale of the ground truth, we cannot compare the position error to the results of Lim *et al.*. The mean orientation error was 0.28 degree, compared to 1.7 degree with the approach of Lim *et al.*. Lim *et al.* state that the map is an 8m × 5m room that

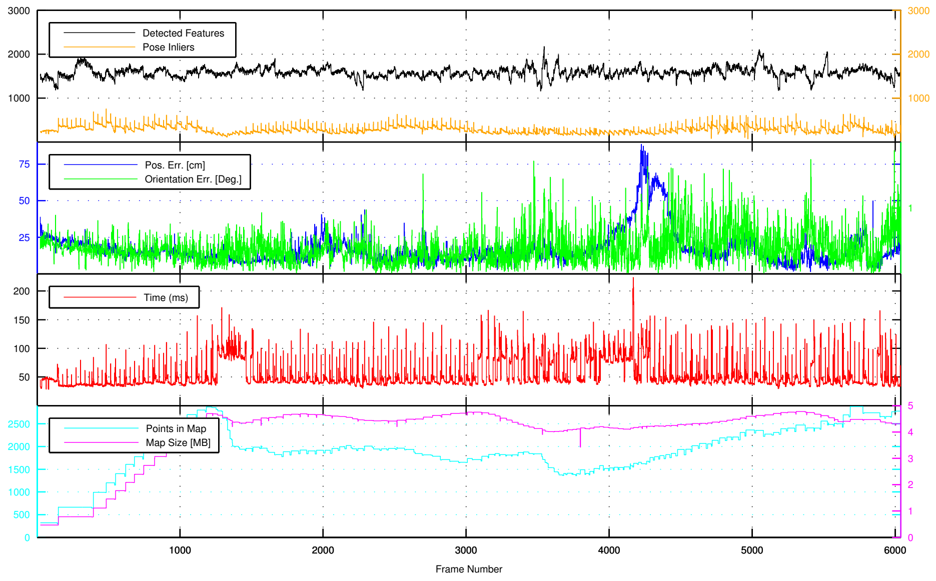

Fig. 4. Evaluation of Seq. 1. From top to bottom: detected features and inliers; position and orientation error; frame processing time; map points and size.

consumes 124MB of memory on the device. The mean memory footprint with our approach is only 1.03MB. The mean pose estimation time was 24.3ms.

The pose estimation approach by Ventura and Höllerer [29] relies on a single global pose estimate and is not capable to track the relative motion during the latency period. Thus, it is prone to both, high server latency and failed global localization. On the other hand, our approach is robust to these problems, since it is able to track the camera pose locally until sufficient global information is available for the alignment. Afterwards, if the server fails to localize a keyframe or the server response is pending, the global constraints belonging to this keyframe, but not the local reprojection errors, are omitted in the alignment.

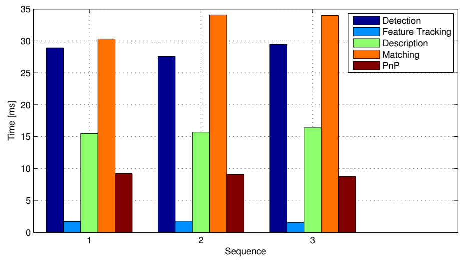

Fig. 5. Mean processing times of the main components of pose estimation

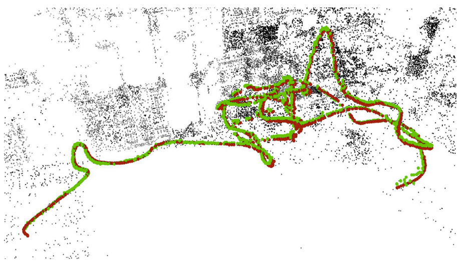

Fig. 6. FLIGHT1 sequence [22]: The $e^2_{0.5}$-trajectory (*green*) is qualitatively similar to the ground truth (*red*)

Table 4. Impact of false positive localizations on mean position and rotation error for Sequence 1 and strategies $e_0.5^1$ and $e_0.5^2$

False Positives	$e^1_{0.5}$		$e^2_{0.5}$	
	P.Err.[m]	R.Err.[Deg.]	P.Err.[m]	R.Err.[Deg.]
0%	0.27 ± 0.24	4.10 ± 4.48	0.17 ± 0.11	0.39 ± 0.23
10%	0.31 ± 0.35	6.63 ± 9.22	0.18 ± 0.15	0.40 ± 0.23
20%	0.32 ± 0.39	6.43 ± 9.39	0.18 ± 0.13	0.40 ± 0.23
30%	0.38 ± 0.51	7.48 ± 11.7	0.19 ± 0.15	0.40 ± 0.23
40%	0.44 ± 0.67	6.80 ± 8.20	0.19 ± 0.16	0.39 ± 0.23

Robustness to False Positive Global Localizations. As discussed in Section 4, both proposed alignment approaches take precautions to be robust to errors in the global localization. Li *et al.* [21] report false positive rates of less than 5.3%. Thus, to evaluate the robustness of the proposed strategies, we artificially set the number of false positive localizations to up to 40%. For every global localization, we randomly decided if it is a false positive localization. If so, we randomly selected a global keyframe pose and geometrically consistent global 2D-3D matches. Table 4 reports the impact on mean position and rotation errors for Sequence 1 and strategies $e^1_{0.5}$ and $e^2_{0.5}$. While the number of false positive global localizations has a notable impact on the localization accuracy and standard deviation for strategy $e^1_{0.5}$, it has almost no effect for strategy $e^2_{0.5}$.

6 Conclusion and Future Work

In this paper, we presented a truly scalable image-based localization approach that runs in real-time on a mobile device. The idea of our approach is to combine real-time pose tracking relative to a small 3D map constructed on the device itself with global pose information provided by a remote localization server. We proposed two strategies to align the local to the global model and to prevent drift in the camera pose. In the future, we would like to additionally utilize the localization server to overcome the major challenges of SLAM-based pose tracking, which are map initialization and recovery from tracking loss. While the proposed alignment approaches are intuitive and satisfying with regard to localization performance, we want to explore how additional global information, *e.g.*, 3D-3D correspondences between local and global map points or known scene geometry, can help to even further improve the alignment.

Acknowledgments. We gratefully acknowledge support by PREServ (EFRE 300268402). We also thank Sudipta N. Sinha for kindly providing the Flight1 dataset and Tobias Weyand and Bastian Leibe for their valuable support.

References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building Rome in a Day. In: ICCV (2009)
2. Agarwal, S., Mierle, K.: Ceres Solver: Tutorial & Reference. Google Inc.
3. Arandjelović, R., Zisserman, A.: Three Things Everyone Should Know to Improve Object Retrieval. In: CVPR (2012)
4. Arth, C., Klopschitz, M., Reitmayr, G., Schmalstieg, D.: Real-Time Self-Localization from Panoramic Images on Mobile Devices. In: ISMAR (2011)
5. Arth, C., Wagner, D., Klopschitz, M., Irschara, A., Schmalstieg, D.: Wide Area Localization on Mobile Phones. In: ISMAR (2009)
6. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded-Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
7. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: Binary Robust Independent Elementary Features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 778–792. Springer, Heidelberg (2010)
8. Castle, R.O., Klein, G., Murray, D.W.: Video-Rate Localization in Multiple Maps for Wearable Augmented Reality. In: ISWC (2008)
9. Davison, A.J., Reid, I.D., Molton, N., Stasse, O.: MonoSLAM: Real-Time Single Camera SLAM. PAMI 29(6), 1052–1067 (2007)
10. Dong, Z., Zhang, G., Jia, J., Bao, H.: Keyframe-Based Real-Time Camera Tracking. In: ICCV (2009)
11. Eade, E., Drummond, T.: Scalable Monocular SLAM. In: CVPR (2006)
12. Fischler, M., Bolles, R.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. ACM 24(6), 381–395 (1981)
13. Fraundorfer, F., Tanskanen, P., Pollefeys, M.: A Minimal Case Solution to the Calibrated Relative Pose Problem for the Case of Two Known Orientation Angles. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 269–282. Springer, Heidelberg (2010)
14. Hedborg, J., Forssén, P.E., Felsberg, M., Ringaby, E.: Rolling Shutter Bundle Adjustment. In: CVPR (2012)
15. Horn, B.K.P.: Closed-Form Solution of Absolute Orientation Using Unit Quaternions. JOSA A 4(4), 629–642 (1987)
16. Irschara, A., Zach, C., Frahm, J.M., Bischof, H.: From Structure-from-Motion Point Clouds to Fast Location Recognition. In: CVPR (2009)
17. Klein, G., Murray, D.: Parallel Tracking and Mapping for Small AR Workspaces. In: ISMAR (2007)
18. Klein, G., Murray, D.: Parallel Tracking and Mapping on a Camera Phone. In: ISMAR (2009)
19. Leutenegger, S., Chli, M., Siegwart, R.: BRISK: Binary Robust Invariant Scalable Keypoints. In: ICCV (2011)
20. Lhuillier, M.: Fusion of GPS and Structure-from-Motion Using Constrained Bundle Adjustments. In: CVPR (2011)
21. Li, Y., Snavely, N., Huttenlocher, D., Fua, P.: Worldwide Pose Estimation Using 3D Point Clouds. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part I. LNCS, vol. 7572, pp. 15–29. Springer, Heidelberg (2012)

22. Lim, H., Sinha, S.N., Cohen, M.F., Uyttendaele, M.: Real-Time Image-Based 6-DOF Localization in Large-Scale Environments. In: CVPR (2012)
23. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. IJCV 60(2), 91–110 (2004)
24. Meier, L., Tanskanen, P., Heng, L., Lee, G.H., Fraundorfer, F., Pollefeys, M.: PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight Using Onboard Computer Vision. Autonomous Robots 33(1-2), 21–39 (2012)
25. Newcombe, R.A., Lovegrove, S., Davison, A.J.: DTAM: Dense Tracking and Mapping in Real-Time. In: ICCV (2011)
26. Sattler, T., Leibe, B., Kobbelt, L.: Improving Image-Based Localization by Active Correspondence Search. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part I. LNCS, vol. 7572, pp. 752–765. Springer, Heidelberg (2012)
27. Sattler, T., Weyand, T., Leibe, B., Kobbelt, L.: Image Retrieval for Image-Based Localization Revisited. In: BMVC (2012)
28. Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle Adjustment – A Modern Synthesis. In: Triggs, B., Zisserman, A., Szeliski, R. (eds.) Vision Algorithms: Theory and Practice. LNCS, vol. 1883, pp. 298–372. Springer, Heidelberg (2000)
29. Ventura, J., Höllerer, T.: Wide-Area Scene Mapping for Mobile Visual Tracking. In: ISMAR (2012)
30. Wendel, A., Irschara, A., Bischof, H.: Natural Landmark-based Monocular Localization for MAVs. In: ICRA (2011)