Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization

Simon Lynen*, Torsten Sattler†, Michael Bosse*, Joel Hesch‡, Marc Pollefeys† and Roland Siegwart*
*Autonomous Systems Lab, ETH Zurich
†Computer Vision and Geometry Group, Department of Computer Science, ETH Zurich
‡Google Inc., Mountain View, CA

***Abstract*—Accurately estimating a robot's pose relative to a global scene model and precisely tracking the pose in real-time is a fundamental problem for navigation and obstacle avoidance tasks. Due to the computational complexity of localization against a large map and the memory consumed by the model, state-of-the-art approaches are either limited to small workspaces or rely on a server-side system to query the global model while tracking the pose locally. The latter approaches face the problem of smoothly integrating the server's pose estimates into the trajectory computed locally to avoid temporal discontinuities.**

In this paper, we demonstrate that large-scale, real-time pose estimation and tracking can be performed on mobile platforms with limited resources without the use of an external server. This is achieved by employing map and descriptor compression schemes as well as efficient search algorithms from computer vision. We derive a formulation for integrating the global pose information into a local state estimator that produces much smoother trajectories than current approaches. Through detailed experiments, we evaluate each of our design choices individually and document its impact on the overall system performance, demonstrating that our approach outperforms state-of-the-art algorithms for localization at scale.

I. Introduction and Related Work

Being able to reliably estimate and track the pose of a robot with respect to a global map at a high frequency is fundamental to unlock path-planning, obstacle avoidance and manipulation. Designing systems that provide accurate localization at a large scale while running on platforms with limited computational resources is thus a key problem in robotics. Such systems are also highly relevant for Augmented Reality (AR) applications, *e.g.*, for mobile phones, which require high-quality pose estimates relative to a global map to correctly project virtual objects into the camera view. One crucial aspect for localization systems, independent whether they are used for agile flying robots, manipulation tasks, or AR, is that they provide a (nearly) perfect registration to a global model with subsequent pose estimates being free of discontinuities.

Using visual and inertial sensors, *i.e.*, a camera and an inertial measurement unit (IMU), enables localization under a wide range of conditions without relying on the availability of GPS or WiFi. State-of-the-art methods for online, low-latency visual (inertial) localization assume the existence of a 3D model of the scene. This model is often pre-built from a set of database images using Structure-from-Motion (SfM). Since SfM reconstructs each 3D scene point from multiple views, each such 3D landmark can be associated with a set of local

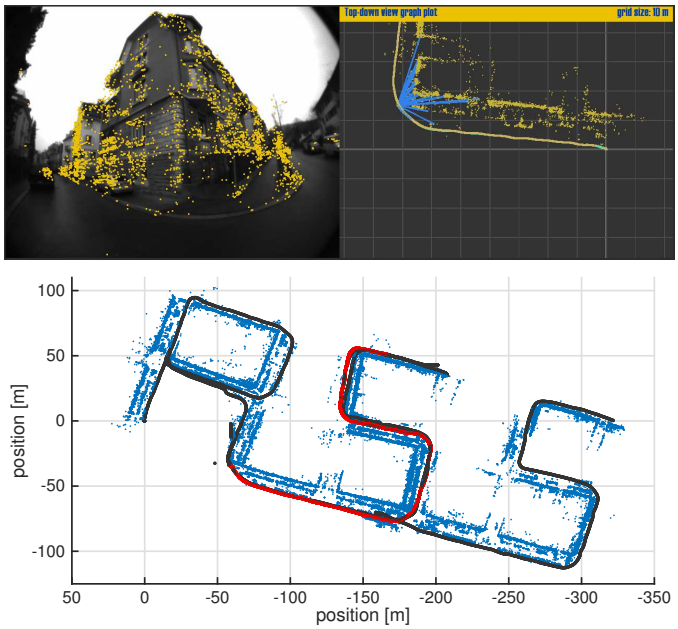

Fig. 1. The proposed system uses efficient nearest neighbor search techniques and projected binary feature descriptors to achieve highly accurate localization against a large "RSS" 3D model in real-time. (Top-left) Projecting the map points (yellow) into the camera view shows the quality of our 2D-3D alignment. (Top-right) Blue lines connect image features to their corresponding 3D model points. (Bottom) The 3D points of the model (blue), the trajectory taken during model building (black), and the current trajectory (red).

image descriptors [41]. Given a novel image together with features [6, 28, 23] extracted from it, the camera pose w.r.t. to the model can be computed from 2D-3D matches between image features and landmarks using relative [37] or absolute [18] pose solvers. In order to match the features against the 3D landmarks, approaches from robotics can be used to first retrieve relevant database images [8, 16, 48]. The image features are then matched against the landmarks observed in the retrieved keyframes [3, 11]. Alternatively, matching can also be done directly against the full model [20, 26, 32, 44, 53]. Once an initial pose has been computed, real-time markerless tracking [3, 27] or simultaneous localization and mapping (SLAM) [32, 44, 52] track the camera pose without having to match against the model in every frame.

Localization approaches that keep the full scene model on the device [3, 11, 20, 27, 53] are usually restricted to

small workspaces due to the high memory requirements of the landmark descriptors. More scalable approaches assume the availability of an external server, which sends the relevant model parts to the device [4] or even performs the actual localization [32, 51, 52]. The latter methods run SLAM on the device, enabling them to handle the latency of transmitting images to the server and to track the pose for periods where localization against the global model fails [32, 44, 52]. Besides removing restrictions on the workspace, tracking a local pose estimate enables the high frequency and smooth pose estimates that are essential for robot control. However, relying on a server connection implies that WiFi/GSM is available, which in turn already gives a very accurate prior for global localization. Instead of outsourcing computations, we compress the global model by both removing unnecessary scene geometry and quantizing the landmark descriptors. By drastically compressing global maps our approach is able to handle building scale models on clients having computational resources similar to micro aerial vehicles or mobile phones. Our approach is based on the observation that pose estimates relative to the global model are only required infrequently when combined with robust local pose tracking.

To align the local SLAM map with the global model from the server, state-of-the-art methods apply loosely coupled approaches that employ the 2D-3D matches used by server-side global localization: Se et al. [44] triangulate local landmarks and perform 3D-3D landmark alignment to the global map. Both Middelberg et al. [32] and Ventura et al. [52] first compute an initial alignment using either the camera poses returned by the server or the global landmark positions. For all following frames that are sent to the server, they then optimize the alignment by including the global 2D-3D matches into the bundle adjustment (BA) of the local map. To limit the computational complexity, they perform a windowed version of SLAM based on a limited number of keyframes. Keyframes which are furthest away from the current pose [32] or oldest [52] are discarded together with their constraints to the local and global model. Discarding (instead of properly marginalizing) measurements however has been shown to lead to suboptimal estimation performance [12, 45, 35, 24]. The removal of constraints from the optimization furthermore leads to discontinuities in the resulting pose estimate as the minimum of the cost function changes. In this paper, we therefore show how to properly handle these global constraints.

Overall, this paper makes the following contributions:

- We propose a large-scale system that entirely runs on devices with limited computational and memory resources while offering highly accurate, real-time localization.
- We provide an extensive experimental evaluation that demonstrates that large-scale localization on such devices is feasible. We evaluate each component individually and also demonstrate its influence on the overall performance.
- We propose a direct inclusion of 2D-3D matches from global localization into the local visual-inertial estimator, leading to smoother trajectories and orders of magnitude faster run-times compared to sliding window BA.
- We demonstrate the feasibility of state-of-the-art matching and compression schemes for mobile localization.

II. System Overview

In an offline stage, a 3D point cloud is reconstructed from a set of database images using SfM. The model is then compressed by removing less important landmarks and quantizing the 3D point descriptors before being stored on the mobile system. The device runs a keyframe-based visual inertial SLAM method to smoothly track the movement of the camera. For each keyframe, visual features are extracted and their descriptors are matched against the descriptors of the 3D model. The resulting 2D-3D matches are then used to robustly estimate the camera pose via RANSAC [14]. The pose computed for the first keyframe gives the initial position and orientation of the mobile system w.r.t. to the global model. For all subsequent keyframes, the inliers to the estimated pose are integrated into the state estimator to provide additional constraints besides the features tracked by SLAM.

A. Descriptor Extraction and Projection

In order to match interest points detected in an image against the 3D model, the same feature type has to be used for model construction and matching. To enable real-time performance on mobile devices efficient binary descriptors [2] are used and projected to a real-valued space following [7, 29]. The projection is designed such that the L_2 distance between descriptors in the projected space matches the Likelihood Ratio Test (LRT) statistic [7]. The LRT is the hypothesis test that best separates matching from non-matching descriptors for a given maximum false-positive matching rate [36]. After projection the descriptor is reduced to 10 dimensions by removing the dimensions with the lowest signal-to-noise ratio. While this does not significantly reduce the memory footprint (from 64 to 40 bytes), the projection vastly increases the efficiency of the k-nearest neighbor (kNN) search at minimal loss of precision [29]. More importantly, the projection enables the use of high-quality kNN algorithms for real-valued vectors [15, 38, 40, 5] and thus avoids the problems of kNN searches in binary spaces [50].

III. Global 3D Model Creation and Compression

The global localization model is built from a sequence of database images using standard SfM techniques. The metric scale of the resulting map, which is essential for any robotic application, is recovered by including IMU data. A loop-closure pipeline, combining the work of Galvez-Lopez and Tardos [16] with the FABMAP algorithm by Cummins and Newman [9], identifies places that have been visited multiple times. Following Leutenegger et al. [24], the model is refined through non-linear optimization [1] by minimizing a cost function $J(\mathbf{x})$ that depends on a state x which involves camera and structure parameters. The cost function contains both the (weighted) reprojection errors $\mathbf{e}_\mathrm{r}$ and the (weighted) temporal

error term from the IMU $\mathbf{e}_\mathrm{s}$:

$$J(\mathbf{x}) := \underbrace{\sum_{i=1}^{I} \sum_{j \in \mathcal{J}(i)} \mathbf{e}_\mathrm{r}^{i,j\,T} \mathbf{W}_\mathrm{r}^{i,j} \mathbf{e}_\mathrm{r}^{i,j}}_{\text{visual}} + \underbrace{\sum_{i=1}^{I-1} \mathbf{e}_\mathrm{s}^{i\,T} \mathbf{W}_\mathrm{s}^{i} \mathbf{e}_\mathrm{s}^{i}}_{\text{inertial}} \quad , \tag{1}$$

where i denotes the camera frame index and j denotes the landmark index. The set $\mathcal{J}(i)$ contains the indices of landmarks visible in the i^{th} frame. Furthermore, $\mathbf{W}_\mathrm{r}^{i,j}$ represents the inverse measurement covariance of the respective landmark, and $\mathbf{W}_\mathrm{s}^{i}$ the inverse covariance of the i^{th} IMU constraint.

A. Model Compression

To reduce the memory footprint of storing the model, the 3D point cloud is compressed by approximately solving a (NP-complete) set cover problem: Both Li et al. [26] and Park et al. [39] try to select a minimal subset of all landmarks such that each keyframes observes at least N_thres (here 15) selected landmarks in order to guarantee robust localization. We initially remove every second key-frame since consecutive frames have a highly similar visual appearance and descriptors contribute a large portion to the model size. Inspired by Li et al. [26], who iteratively *select* landmarks, our approach then uses a greedy algorithm that iteratively *removes* landmarks from the model which are not observed often while at the same time making sure no keyframe is left with less than N_thres observations. Finally, only the minimal amount of data required for localization is retained, which are the 3D landmark positions together with their corresponding descriptors as well as landmark covisibility information. Since the reduction process removes the landmarks which have been observed the least or have a large estimated covariance, initially the overall quality of the model improves, while only at high reduction rates an impact on the localization quality is apparent [26].

Product Quantization [21] is used to compress the descriptors of the remaining landmarks: The 10-dimensional descriptor space is split into M_PQ subspaces of equal dimensionality, *i.e.*, each descriptor is split into M_PQ parts of length $D_\text{PQ} = 10/M_\text{PQ}$. For each subspace, a visual vocabulary with k_PQ words is learned through k_PQ-means clustering. These vocabularies are then used to quantize each part of a landmark descriptor individually. A descriptor is thus represented by the indices of the closest cluster center for each of its parts. This quantization can significantly reduce the memory requirements. *E.g.*, when using two vocabularies ($M_\text{PQ} = 2$) with $k_\text{PQ} = 256$ centers each, storing a descriptor requires only 2 bytes instead of 40 bytes.

The squared Euclidean distance between a regular descriptor $\mathbf{d} = (\mathbf{d}_1 \ \cdots \ \mathbf{d}_{M_\text{PQ}})$, $\mathbf{d}_j^T \in \mathbb{R}^{D_\text{PQ}}$, and a quantized descriptor represented by a set of indices $(i_1, \ldots, i_{M_\text{PQ}})$ is computed as

$$\sum_{j=1}^{M_\text{PQ}} \left(\mathbf{d}_j - \mathbf{c}_j(i_j)\right)^2 \quad . \tag{2}$$

Here $\mathbf{c}_j(i_j)$ is the word corresponding to index i_j in the j^{th} vocabulary. Notice that the distance from $\mathbf{d}_j$ to all words in the j^{th} vocabulary can be pre-computed and stored in a look-up table [21]. Consequently, only table look-ups and additions are required to compute the descriptor distance.

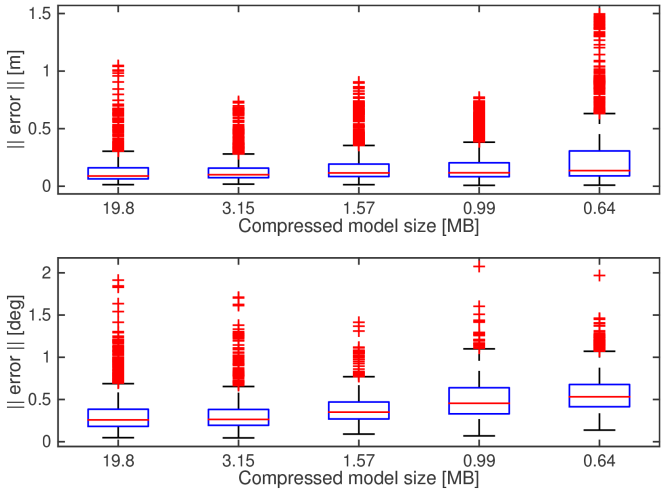

Fig. 2. Starting from an initial model of size 136.7 MB we apply a pruning technique to remove 3D points that have not been observed often. This reduces the model size to 19.8 MB. Combining pruning with a descriptor compression technique, we can fit the model in as little as 0.64 MB while not incuring substantial penalties on the localization quality. The plot shows the error statistics for (top) position and (bottom) orientation when localizing against models from different pruning levels.

Decomposing the descriptor space such that each component has a similar variance reduces the quantization error of product quantization [17]. As a result, Eq. (2) better approximates the true descriptor distance between the two original descriptors. This balancing can be achieved by permuting the rows of a rotation matrix that aligns the descriptor space with its principal directions [17]. Notice that this balancing step does not introduce any computational overhead as the rotation matrix is pre-computed and then pre-multiplied with the projection matrix from Section II-A.

IV. Localization Against the Global Model

Given the global model, 2D-3D matches between the features found in the current keyframe and the 3D landmarks are established via descriptor matching. These correspondences are then used to estimate the camera pose. A popular approach in related work is to use image retrieval techniques [38, 46] to identify a set of database images which are similar to the current keyframe [9, 20, 48, 16, 31, 30, 34]. 2D-3D matches between features and landmarks from the retrieved images can easily be established based on common visual words. Unfortunately, this approach often introduces so many wrong matches that RANSAC-based pose estimation becomes infeasible [43]. Thus, pairwise descriptor matching between the query keyframe and the top-ranked retrieved images is performed to reduce the fraction of outliers [20, 43].

The bag-of-words model becomes less distinctive as the database grows, *i.e.*, more and more unrelated views are found among the top ranked images [22]. Using a finer vocabulary can reduce the problem of voting for unrelated images

[22, 43, 47]. Yet, most large-scale localization approaches rely on directly matching the feature descriptors against the landmark descriptors without any intermediate retrieval step [25, 26, 41, 42, 49]. A popular approach to accelerate the matching process is to build a kd-tree for (approximate) nearest neighbor search on top of the landmark descriptors [25, 49]. While offering an excellent search performance, a kd-tree is rather slow due to backtracking and irregular memory access [41]. In addition, the kd-tree would need to be rebuilt when extending the global model, which is undesirable for SLAM scenarios. Current state-of-the-art methods for efficient large-scale localization [41, 42] instead use an inverted index: Given a fixed-size vocabulary, each feature descriptor from the current frame is assigned to its closest word. Exhaustive search through all landmark descriptors assigned to this word then yields the nearest neighboring landmark. This approach is faster than kd-tree search and can be accelerated even further through prioritization [41, 42], *i.e.*, stopping the search once a fixed number of matches has been found. Notice that search structures based on inverted indexes allow extentions to the 3D model by simply appending descriptors.

In this paper, an inverted index is used to accelerate nearest neighbor search. Obviously, larger vocabularies are desirable as fewer descriptors will be stored for every word. Yet, using a larger vocabulary implies higher assignment times and memory consumption. Both of these problems can largely be circumvented by using an *inverted multi-index* [5]: Similar to product quantization, the descriptor space is split into two parts and a visual vocabulary $\mathcal{V}_i$ containing N_{imi} words is trained for each part. The product of both vocabularies then defines a large vocabulary $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ containing N_{imi}^2 words. Finding the nearest word $\omega = (\omega_1, \omega_2) \in \mathcal{V}$ for a descriptor $\mathbf{d}$ consists of finding the nearest neighboring words ω_1, ω_2 from the two smaller vocabularies $\mathcal{V}_1$ and $\mathcal{V}_2$, which is accelerated using a kd-tree. Thus, using an inverted multi-index not only reduces the memory requirements but also accelerates the visual word assignments. Each landmark descriptor is assigned to its closest word from $\mathcal{V}$. For each feature that should be matched against the model, the X nearest words from each vocabulary are found. From the product of these two sets of words, the feature descriptor is matched against all landmark descriptors stored in the nearest Y words. When using product quantization, one product quantizer is learnt for each word from $\mathcal{V}_1$ and $\mathcal{V}_2$ to encode the residuals between the word and the assigned descriptors.

A. Covisibility Filtering and Pose Recovery

Often, fewer than 10% of all features found in the current frame have a corresponding landmark [41]. Most of the wrong matches can be eliminated by a threshold on the maximal descriptor distance or Lowe's ratio test [28]. However, some correspondences will still pass these tests, causing problems during camera pose estimation since RANSAC's run-time increases exponentially with the outlier ratio [14].

A popular strategy to filter out wrong matches is to employ the covisibility relation between the different landmarks [25, 48, 42]. For example, only matches whose landmarks form clusters in the covisibility graph [48, 42] are kept. Covisibility filtering works well on large models since wrong matches tend to be randomly distributed all over the scene. However, it is less effective for smaller scenes where a larger fraction of landmarks is covisible. Lynen et al. [29] proposed an algorithm that works well for both small and larger models, but is currently limited to offline processing of datasets. Due to the the simplicity and efficiency as well as the ability to run online we use the covisibility filter from [42]: A 3D model defines a undirected, bipartite *visibility graph* [26], where the two sets of nodes correspond to the database images and the 3D landmarks in the map. A landmark node and a image node are connected if the landmark is visible in the corresponding database image. The landmarks from a given set of 2D-3D matches and their corresponding database images then form a set of connected components in this visibility graph. The covisibility filter from [42] simply removes all matches whose 3D point does not belong to the largest connected component. These correspondences are then used to estimate the camera pose in a few milliseconds by applying a 3-point solver [14] inside a preemptive RANSAC [10] loop. Afterwards, the pose is refined by applying the direct least squares PnP solver [19] on all inlier correspondences.

V. Local Pose Tracking

The pose of the platform is tracked in real time using a visual-inertial sliding window estimator with on-the-fly feature marginalization similar to the work of Mourikis et al. [33]. The temporally evolving state in this estimator is given by

$$\mathbf{x}_E = ({}^L\boldsymbol{P}_I \ {}^L\boldsymbol{v}_I \ \mathbf{b}_g \ \mathbf{b}_a) \ , \tag{3}$$

where ${}^L\boldsymbol{P}_I$ denotes the pose of the platform as the coordinate transformation of the IMU frame of reference w.r.t. the local SLAM frame of reference. The translational velocity estimate of the IMU frame w.r.t. the local SLAM frame is denoted as ${}^L\boldsymbol{v}_I$. $\mathbf{b}_g$, $\mathbf{b}_a \in \mathbb{R}^3$ denote the estimate of the time varying gyroscope and accelerometer bias, modeled as random walk processes driven by the zero-mean, white, Gaussian noise vectors $\mathbf{n}_{bg}$ and $\mathbf{n}_{ba}$.

Besides the evolving state $\mathbf{x}_E$, the full estimated state $\hat{\mathbf{x}}_k$ at time k also includes the position and orientation of N cameras which form the sliding window of poses [33]:

$$\hat{\mathbf{x}}_k = (\hat{\mathbf{x}}_E \ {}^L\boldsymbol{P}_{C_1} \ \cdots \ {}^L\boldsymbol{P}_{C_N}) \ . \tag{4}$$

Here, ${}^L\boldsymbol{P}_{C_i}$, $i = 1 \dots N$, denote the estimates of the pose of the ith camera.

Using measurements from the IMU increases robustness and accuracy while providing a metric pose estimate. At the same time, the proper marginalization of past measurements [12, 45, 35, 24] is key to obtain a smooth pose estimate. This is particularly relevant when including measurements to the global model which are often not in perfect agreement with the locally observed structure, *e.g.*, due to moving objects during model creation or drift in the local pose estimates.

A. Global Updates to the Local State Estimation

In order to boot-strap the localization system, descriptors from the keyframes of the local SLAM system are matched against the model as described in Section IV. Once an estimate of the pose ${}^{G}\boldsymbol{P}_{C}$ of the camera w.r.t. the global model is available, the frame of reference of the local SLAM system is aligned with the global map using the relative transformation from global model to local SLAM frame of reference ${}^{G}\boldsymbol{P}_{L}$:

$$
\begin{aligned}
{}^{G}\boldsymbol{P}_{L} &= {}^{G}\boldsymbol{P}_{C} \otimes {}^{C}\boldsymbol{P}_{I} \otimes {}^{I}\boldsymbol{P}_{L} && (5)\\
{}^{I}\boldsymbol{P}_{L} &= {}^{L}\boldsymbol{P}_{I}^{-1}. && (6)
\end{aligned}
$$

Here, $\otimes$ denotes the transformation composition operator. Using the computed transformation ${}^{G}\boldsymbol{P}_{L}$, the estimator state and covariance are transformed to be with respect to the global frame of reference. We project this transformation along the direction of gravity, assuming that the global map is gravity aligned via including IMU measurements into the optimization.

Once the frame of reference of the local SLAM system is aligned with the global frame of reference, 2D-3D matches that are inliers to the global pose estimate can be directly used to update the estimator. There are existing approaches which include information from the global model into the local optimization such as the algorithm by Middelberg et al. [32] and Ventura et al. [52]. Ventura et al. [52] use a transformation calculated by a server to transform points from the global coordinate system to the local frame of reference. By applying this transformation, the non-linear refinement, which is subsequently carried out, cannot improve the alignment beyond the errors contained in the transformation estimate provided by the server since this transformation is not part of the error-term.

In the algorithm proposed in this paper, every associated 2D-3D match provides a measurement of the form

$$
\boldsymbol{z}_{i}^{(j)} = \frac{1}{{}^{C_i}z_j} \begin{bmatrix} {}^{C_i}x_j \\ {}^{C_i}y_j \end{bmatrix} + \boldsymbol{n}_{i}^{(j)} \ , \tag{7}
$$

where $[{}^{C_i}x_j \ {}^{C_i}y_j \ {}^{C_i}z_j]^T = {}^{C_i}\boldsymbol{p}_j$ denotes the position of the jth 3D point expressed in the frame of reference of camera i. To obtain the residual for updating the EKF, we express the expected measurement $\hat{\boldsymbol{z}}_{i}^{(j)}$ as a function h of the state estimate $\hat{\mathbf{x}}_k$ and the position of the landmark ${}^{G}\boldsymbol{p}_{\ell}$ expressed in the global frame of reference:

$$
\boldsymbol{r}_{i}^{(j)} = \boldsymbol{z}_{i}^{(j)} - \hat{\boldsymbol{z}}_{i}^{(j)} = \boldsymbol{z}_{i}^{(j)} - h({}^{G}\boldsymbol{p}_{\ell}, {}^{G}\boldsymbol{P}_{C_i}) \ . \tag{8}
$$

By linearizing this expression around the state estimate we obtain (See [33] Eq. (29) for details.):

$$
\boldsymbol{r}_{i}^{(j)} \simeq \boldsymbol{H}_{GL_i}^{(j)} (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i) + \boldsymbol{n}^{(j)} \ . \tag{9}
$$

Here, $\boldsymbol{H}_{GL_i}^{(j)}$ denotes the global landmark measurement Jacobian with non-zero blocks for the pose of camera i.

When querying the map, it is not unusual to retrieve hundreds of matches from the image to the global map. To reduce the computational complexity of updating the estimator, all the residuals and Jacobians $\boldsymbol{r} = \boldsymbol{H}(\boldsymbol{x} - \hat{\boldsymbol{x}}) + \boldsymbol{n}$ are stacked to apply measurement compression [33]. More specifically, we apply a QR-decomposition to $\boldsymbol{H}$:

$$
\boldsymbol{H} = \begin{bmatrix} \boldsymbol{V}_1 \ \boldsymbol{V}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{T}_H \\ \mathbf{0} \end{bmatrix} \ , \tag{10}
$$

where $\boldsymbol{T}_H$ is an upper triangular matrix and $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ are unitary matrices with columns forming the basis for the range and the nullspace of $\boldsymbol{H}$, respectively. This operation projects the residual on the basis vectors of the range of $\boldsymbol{H}$, which means that $\boldsymbol{V}_1$ extracts all the information contained in the measurements. The residual from Eq. (9) can then be rewritten as

$$
\begin{bmatrix} \boldsymbol{V}_1^T \boldsymbol{r} \\ \boldsymbol{V}_2^T \boldsymbol{r} \end{bmatrix} = \begin{bmatrix} \boldsymbol{T}_H \\ \mathbf{0} \end{bmatrix} (\boldsymbol{x} - \hat{\boldsymbol{x}}) + \begin{bmatrix} \boldsymbol{V}_1^T \boldsymbol{n} \\ \boldsymbol{V}_2^T \boldsymbol{n} \end{bmatrix} \ . \tag{11}
$$

After discarding $\boldsymbol{V}_2^T \boldsymbol{r}$ in Eq. (11) since it only contains noise, we obtain the compressed residual formulation

$$
\boldsymbol{r}_n = \boldsymbol{V}_1^T \boldsymbol{r} = \boldsymbol{T}_H (\boldsymbol{x} - \hat{\boldsymbol{x}}) + \boldsymbol{n}_n \text{ with } \boldsymbol{n}_n = \boldsymbol{V}_1^T \boldsymbol{n} \ . \tag{12}
$$

Using Givens rotations, the residual $\boldsymbol{r}_n$ and the upper triangular matrix $\boldsymbol{T}_H$ for L matches can be computed efficiently in $\mathcal{O}((6N)^2 L)$ [33].

The MSCKF algorithm [33] processes a feature-track as soon as the track is broken or the entire window of keyframes is spanned by the track. In our local SLAM system, these completed tracks are used to triangulate landmarks and the resulting 3D points are then used to update the estimator. Since we are using the same visual features for frame-to-frame tracking and global localization, we can use a match between the 3D model and a feature from a single frame to identify the corresponding feature track. This information can now be used to form a constraint to the 3D model that involves all keyframes that are spanned by the corresponding feature track. We found that forming a constraint which involves all key-frames which are part of the track gives a lower tracking error than performing single camera updates [33]. To avoid double counting information, care must be taken that feature measurements are used to either formulate a constraint in the local SLAM *or* to the 3D map, but not both.

For memory reasons the global model only stores the marginal covariance for each 3D landmark, which we use to formulate the measurement uncertainty of the update. We want to emphasize that by applying the update in this way, we are correlating the state estimate with the map in the same way that this was done by Mourikis et al. [33]. We see the improvements that can be achieved by avoiding this correlation as part of future work. During stationary phases, image features are repeatedly matched to the same 3D landmark. In order not to reuse information multiple times, we thus have to keep track which landmarks have already been used for updates.

VI. Experimental Evaluation

In the following, we evaluate the proposed system and its different components. After explaining how the datasets used for experimental evaluation were obtained, we first analyze each of our system's components (global localization and local pose tracking) before evaluating the full system.

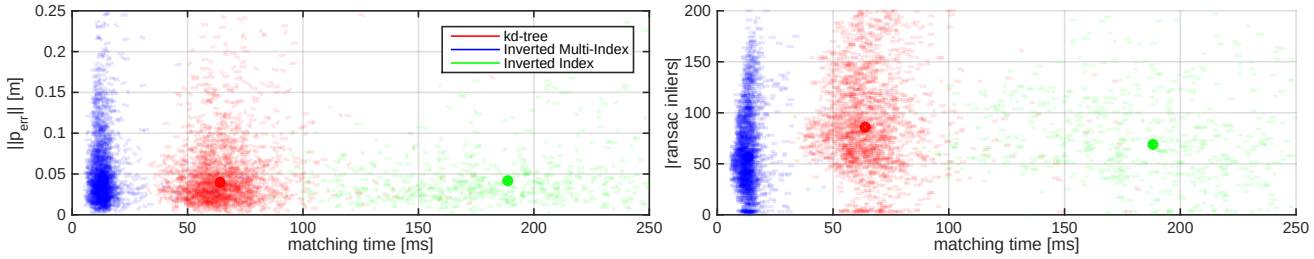

Fig. 3. Evaluation of different matching strategies: (Left) Position error of the global pose estimates computed from the 2D-3D matches established by each strategy. (Right) The number of inlier matches consistent with the estimated poses. The x-axis in both plots denotes the time required to compute the matches. Each point in the plots corresponds to a single pose estimate performed in the test sequence.

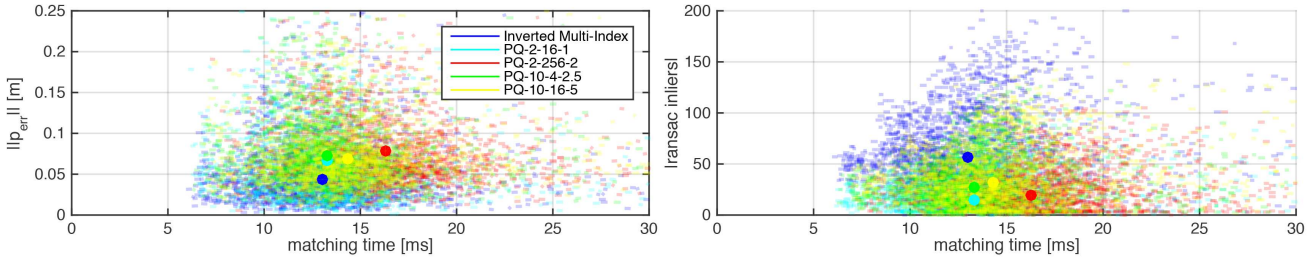

Fig. 4. Comparison of different parameters for product quantization with the inverted multi-index: (Left) Position error of the global pose estimates computed from the found 2D-3D matches. (Right) The number of inlier matches consistent with the estimated poses. The x-axis in both plots denotes the time required for finding the matches and each point in the plots corresponds to a single pose estimation attempt. PQ-10-16-5 denotes a product quantization variant that splits each descriptor into 10 parts, uses 16 cluster centers per part, and requires 5 bytes to store the quantized descriptor. The colored dots denote the median values for each variant

A. Data Acquisition

We build a model from a single run through a city center using the technique described in Section III. The resulting 3D model consists of 703,362 3D points associated with 3,213,618 descriptors, reconstructed from a trajectory of 1.44 km. We apply the model compression scheme discussed in Section III-A such that at least 15 of the selected landmarks are visible in each of the 4,212 keyframes. The compression scheme reduces the global map to 82,149 points with 471,296 descriptors, reducing the memory requirements from 136.7 MB to 19.8 MB.

We also record an additional dataset for evaluation of the algorithms where we perform moderate view-point changes. We obtain a ground truth trajectory by using the batch loop-closure algorithm of Lynen et al. [29] to find candidate 2D-3D matches for each frame in the evaluation sequence. These candidates are then filtered using spatial verification (*c.f.* Section IV-A) before we associate the observations from the evaluation dataset with the model landmarks. Finally we run a full-batch visual-inertial bundle adjustment (BA) (*c.f.* Section III) to align the evaluation trajectory with the model. After alignment, the poses of the evaluation trajectory are used as ground truth. The experiments where done on an Intel i7 (4980HQ) with 2.8 GHz and on a NVidia Tegra K1 powered mobile device. Table I compares timings for all parts of the pipeline for both architectures. For simplicity the remaining timings in the paper were obtained on the Intel i7. A video of the system localizing in real-time from a 1.4 km long model as well as from forward motion on a 500 m trajectory can be found in the supplementary material.

TABLE I
TIMING COMPARISON OF SEARCH AND OPTIMIZATION STEPS IN THE PROPOSED APPROACH FOR INTEL I7 AND NVIDIA TEGRA K1 CPUS FOR AN INDOOR NAVIGATION DATASET.

	Feat.det.	Kd-tree	Inv.Mult.Idx.	Opt. of [32]	EKF upd.
i7	7 ms	12 ms	3 ms	134 ms	2.5 ms
K1	34 ms	80 ms	26 ms	560 ms	9 ms

B. Global 2D-3D Matching

We first evaluate different approaches for establishing 2D-3D matches between the current keyframe and the global map. We compare using a kd-tree [13] (as in [25, 49]), a normal inverted index (as in [41, 42]), and an inverted multi-index. The inverted index is built using a visual vocabulary with 50,000 words while two vocabularies containing 1,000 words each are used for the inverted multi-index, *i.e.*, the inverted multi-index generates a product vocabulary with 1 million words. Both methods consider the landmark descriptors assigned to the 10 nearest words for nearest neighbor search.

Fig. 3 compares the localization accuracy, measured as the Euclidean distance between the ground truth and the calculated pose, the matching times, and the number of inlier matches after spatial verification for each method. Compared to the normal inverted index, the finer quantization induced by the inverted multi-index leads to faster search times since fewer landmark descriptor are stored per word. In contrast to the kd-tree, the inverted multi-index does not need to perform back-tracking (which often accounts for a large part of the search time), leading to faster search times. The larger variation in search times for the inverted index is caused by the fact that some words contain much more descriptors than others,

TABLE II

THE DIFFERENT PRODUCT QUANTIZATION VARIANTS USED IN SECTION VI-C. THE NUMBER OF BYTES REQUIRED TO STORE A DESCRIPTOR DEPENDS ON THE NUMBER OF PARTS M_{PQ} AND THE NUMBER k_{PQ} OF CLUSTER CENTERS USED FOR EACH PART. THE RESULTING MODEL SIZE INCLUDES 962 KB FOR THE LANDMARK POSITION AND COVISIBILITY INFORMATION. THE LAST COLUMN SPECIFIES THE MEMORY OVERHEAD FOR STORING THE QUANTIZERS.

M_{PQ}	k_{PQ}	Bytes per descriptor	Model Size [MB]	Overhead [MB]
2	256	2	2.72	9.8
2	16	1	2.28	0.6
10	16	5	4.08	0.6
10	4	2.5	2.96	0.15

a common effect caused by k-means clustering. As evident from Fig. 3(right), using the inverted multi-index results on average in fewer inlier matches than provided by the other two search structures. Yet, this does not translate to a worse localization accuracy (*c.f.* Fig. 3(left)). Based on the results, we recommend using an inverted multi-index instead of one of the two search structures that are commonly used for large-scale localization.

C. Global Matching with Product Quantization

The previous experiment used the original 10 dimensional descriptors. Thus, we next evaluate the impact of using quantized descriptors on the matching performance. We use the inverted multi-index and train a product quantizer for each word from each of the two individual vocabularies used by the index. We evaluate different combinations of M_{PQ}, the number of parts each descriptor is split into, and k_{PQ}, the number of cluster centers used for each part. Table II specifies the settings used as well as the resulting model sizes and the memory overhead required for storing the quantizers. Setting $M_{\text{PQ}} = 10$ denotes that the original descriptor is split into 10 parts. This means that we train 5 vocabularies for each word in the individual vocabularies of the multi-index, each containing only a single dimension of the original descriptor. For $M_{\text{PQ}} = 10$ and $k_{\text{PQ}} = 16$, we store 80 floating point values for each word from both individual vocabularies. Thus, 0.15 MB are needed to store the cluster centers of all 2000 words.

As can be seen from Table II, using product quantization enables us to drastically reduce the memory consumption of the global map from 136.7 MB to between 2.28 MB and 4.08 MB. Notice that for the setting $M_{\text{PQ}} = 2$ and $k_{\text{PQ}} = 256$, it is cheaper to store the map than storing the cluster centers of all product quantizers.

Fig. 4 compares the matching performance obtained when using product quantization vs. using the original descriptors. As can be seen, using product quantization instead of the original descriptors leads to finding fewer matches and thus fewer correspondences that pass RANSAC-based pose estimation. In addition, it happens more often that not enough matches for applying pose estimation are found. Nevertheless, we observe only a small difference in the resulting localization error especially since our robust local SLAM is able to bridge periods without updates from the global model (See Section VI-D and Fig. 5). Given that the search times are comparable, we thus recommend using product quantization due to the significant decrease in memory consumption.

TABLE III

COMPARING THE PROPOSED EKF-BASED ESTIMATOR UPDATE WITH SLIDING WINDOW BUNDLE ADJUSTMENT (BA): WE REPORT THE MEAN TIME *t-up* REQUIRED TO UPDATE THE ESTIMATOR BASED ON THE GLOBAL 2D-3D MATCHES AND THE MEAN POSITION $||\bar{p}_{err}||$ AND ORIENTATION ERROR $||\bar{\theta}_{err}||$ OF EACH METHOD (INCL. STD-DEV.). FOR SLIDING WINDOW BA, WE EXPERIMENT WITH USING DIFFERENT NUMBERS OF KEYFRAMES IN THE WINDOW (FIRST NUMBER) AND DIFFERENT NUMBERS OF BA ITERATIONS (SECOND NUMBER).

	t-up [ms]	$\|\|\bar{p}_{err}\|\|$ [m]	$\|\|\theta_{err}\|\|$ [deg]
EKF	2.9 ± 1.5	0.17 ± 0.12	0.32 ± 0.16
BA-10-10	163.0 ± 43.0	0.13 ± 0.15	0.41 ± 0.17
BA-10-5	138.8 ± 36.5	0.12 ± 0.14	0.58 ± 0.15
BA-5-10	100.3 ± 31.9	0.11 ± 0.13	0.53 ± 0.15
BA-5-5	77.6 ± 31.6	0.12 ± 0.14	0.57 ± 0.10
BA-5-2	38.6 ± 14.4	0.14 ± 0.16	0.54 ± 0.16

D. Local Pose Tracking

Next, we evaluate the pose tracking quality achieved with our state estimator that directly includes global 2D-3D matches as EKF updates. We are interested in the position and orientation accuracy of our system, as well as the smoothness of the computed trajectories, and the time required for the state updates. We compare our system to an implementation of [32] since it is the best performing algorithm currently available for mobile device localization. This system performs local SLAM using a sliding window BA which optimizes only local parameters and keeps the global model fixed. To further improve its performance and allow a fair comparison we included IMU constraints in the BA and use the Ceres [1] solver which exploits the structure of the problem. Furthermore we evaluate different parameter settings for the BA-based methods. Table III compares our method against the implementation of [32] in which both estimators are fed with the exact same data and the same constraints to the map. For every camera frame, we evaluate the error between the estimated pose and the ground truth as the Euclidean distance between the positions and the disparity angle in orientation. As evident from the table, our EKF-based approach achieves a positional accuracy similar to the best-performing BA approach while offering a more accurate estimate of the camera orientation. At the same time, our approach is more than one order of magnitude faster than the most efficient BA variant. While BA is typically run in a separate thread to avoid blocking pose tracking [32], the proposed EKF-based estimator is efficient enough to be directly run on each frame.

Fig. 6 shows that even though both approaches offer a similar mean pose accuracy, our estimator achieves a much better temporal consistency. We capture this measure by comparing the change in the error between ground truth and estimate for position and orientation. Avoiding discontinuities in pose tracking is a vital property for obstacle avoidance and robot control, where large pose jitter can cause problems when determining a path that prevents a collision.

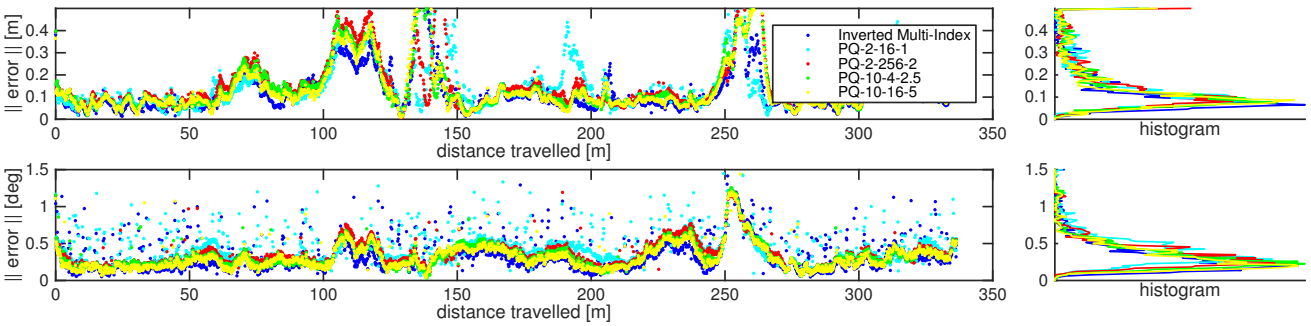

Fig. 5. Comparison of the position and orientation error for different parameters of the product quantization and the inverted multi-index: (Top) Position and (bottom) orientation error of the global pose estimates compared with the ground-truth over the distance travelled. It is worth noting that even though the product-quantized descriptors require significantly less memory than the original descriptors, the final error on the pose estimate is only marginally influenced. Again, PQ-10-16-5 denotes a product quantization variant that splits each descriptor into 10 parts, uses 16 cluster centers per part, and requires 5 bytes to store the quantized descriptor.

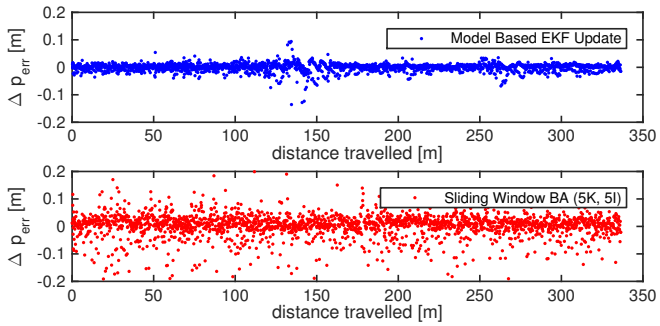

Fig. 6. Compared to a sliding window bundle adjustment approach similar to [32], the proposed direct inclusion of the global 2D-3D matches into EKF gives significantly smoother trajectories as evident from the difference in the pose error between subsequent frames.

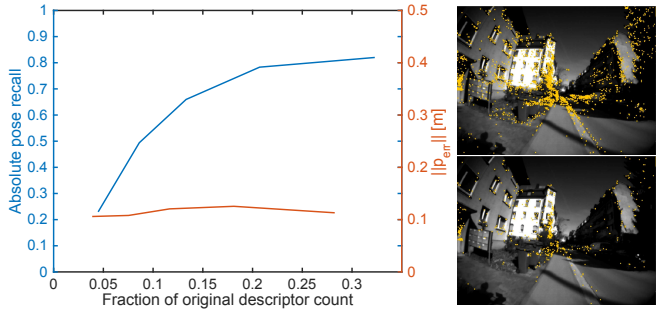

Fig. 7. Comparison of pose estimates using a forward facing camera and a model that underwent different levels of pruning (right). While the number of successfully recovered poses from p3p decreases for increased pruning (left), the overall tracking error stays low since sporadic map-updates are enough to keep the frame of reference of the local SLAM aligned with the model.

E. Failure due to Aggressive Compression

Increasing the compression ratio of a model has two different effects on the localization performance. If too little points are being matched to a given frame, the initial alignment of the local SLAM can be inaccurate or fail. On the other hand if the model is too compressed only few matches pass the RANSAC outlier rejection. Given our robust local SLAM however it is sufficient to localize against the global map only sporadically without much impact on the overall accuracy as shown in Fig. 7 for a forward moving camera. Similar to [41] we require a minimum number RANSAC inliers before we accept a localization result (here 10). We found that such a constraint effectively avoids outlier pose-results being passed to the estimators. Additional to this outlier rejection we additionally perform a Mahalanobis distance test on a per constraint basis. Despite a performance degradation with high compression rates, the results also show that it is possible to obtain an accurate trajectory with a compressed map, which validates our system design (See Fig. 2).

VII. Conclusion

In this paper, we have presented a framework for estimating and tracking the camera pose relative to a global 3D map. Model and descriptor compression enables our system to reduce the memory required to store the global map from 136 MB to about 3 MB without a significant impact on the accuracy of our pipeline. Using efficient search structures accelerates global localization against the map as well. In contrast to current state-of-the-art methods for large-scale localization, the proposed framework can thus run on a system with limited computational and memory resources without relying on an external server. We have proposed a novel formulation for incorporating the global pose estimates into a visual-inertial SLAM system. In contrast to state-of-the-art approaches, which rely on sliding window bundle adjustment, our method achieves a comparable pose tracking accuracy while being an order of magnitude or more faster and producing trajectories with a higher temporal consistency. We have evaluated each component of our system individually while also measuring the impact of each stage on the following parts of the pipeline. The results of our evaluation will be interesting for everyone working on (scalable) real-time pose estimation and tracking approaches.

For future work, we plan to accelerate global localization by integrating knowledge about the system's current position into the matching process.

Acknowledgments

The research leading to these results has received funding from Google's project Tango.

REFERENCES

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. https://code.google.com/p/ceres-solver/.

[2] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast Retina Keypoint. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[3] Clemens Arth, Daniel Wagner, Manfred Klopschitz, Arnold Irschara, and Dieter Schmalstieg. Wide Area Localization on Mobile Phones. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009.

[4] Clemens Arth, Manfred Klopschitz, Gerhard Reitmayr, and Dieter Schmalstieg. Real-Time Self-Localization from Panoramic Images on Mobile Devices. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[5] Artem Babenko and Victor Lempitsky. The Inverted Multi-Index. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision (ECCV), 2006 IEEE European Conference on*.

[7] Michael Bosse and Robert Zlot. Keypoint Design and Evaluation for Place Recognition in 2D Lidar Maps. *Robotics and Autonomous Systems*, 2009.

[8] Mark Cummins and Paul Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 2008.

[9] Mark Cummins and Paul Newman. Highly Scalable Appearance-Only SLAM — FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[10] David D. Nistér. Preemptive RANSAC for live structure and motion estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003.

[11] Zilong Dong, Guofeng Zhang, Jiaya Jia, and Hujun Bao. Keyframe-Based Real-Time Camera Tracking. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[12] Tue-Cuong Dong-Si and Anastasios I Mourikis. Motion Tracking with Fixed-lag Smoothing: Algorithm and Consistency Analysis. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[13] Jan Elseberg, Stéphane Magnenat, Roland Siegwart, and Andreas Nüchter. Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration. *Journal of Software Engineering for Robotics*, 2012.

[14] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 1981.

[15] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software (TOMS)*, 1977.

[16] Dorian Galvez-Lopez and J. D. Tardos. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics and Automation*, 2012.

[17] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized Product Quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[18] JA Grunert. Das pothenotische Problem in erweiterter Gestalt nebst Bemerkungen über seine Anwendungen in der Geodäisie. *Grunerts Archiv fiir Mathematik und Physik*, 1841.

[19] Joel A. Hesch and Stergios I. Roumeliotis. A Direct Least-squares (DLS) solution for PnP. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[20] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From Structure-from-Motion Point Clouds to Fast Location Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[21] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.

[22] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2008.

[23] Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[24] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization. *International Journal of Robotics Research (IJRR)*, 2014.

[25] Yunpeng Li, Noah Snavely, Daniel Huttenlocher, and Pascal Fua. Worldwide Pose Estimation Using 3D Point Clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2010.

[26] Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. Location Recognition using Prioritized Feature Matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2010.

[27] Hyon Lim, Sudipta N. Sinha, Michael F. Cohen, and Matthew Uyttendaele. Real-time Image-based 6-DOF Localization in Large-Scale Environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[28] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 2004.

[29] Simon Lynen, Michael Bosse, Paul Furgale, and Roland Siegwart. Placeless Place-Recognition. In *3D Vision*

(3DV), International Conference on, 2014.

[30] Will Maddern, Michael Milford, and Gordon Wyeth. CAT-SLAM: Probabilistic Localisation and Mapping using a Continuous Appearance-based Trajectory . *International Journal of Robotics Research (IJRR)*, 2012.

[31] Christopher Mei, Gabe Sibley, and Paul Newman. Closing loops without places. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2010.

[32] Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-DOF Localization on Mobile Devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014.

[33] A.I. Mourikis, N. Trawny, S.I. Roumeliotis, A.E. Johnson, A. Ansar, and L. Matthies. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics (T-RO)*, 2009.

[34] Liz Murphy and Gabe Sibley. Incremental Unsupervised Topological Place Discovery. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[35] Esha D. Nerurkar, Kejian J. Wu, and Stergios I. Roumeliotis. C-KLAM: Constrained Keyframe-Based Localization and Mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[36] Jerzy Neyman and Egon S. Pearson. *On the problem of the most efficient tests of statistical hypotheses*. Transactions of the Royal Society of London Series A, 1992.

[37] David Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2004.

[38] David Nister and Henrik Stewenius. Scalable Recognition with a Vocabulary Tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[39] Hyun Soo Park, Yu Wang, Eriko Nurvitadhi, James C Hoe, Yaser Sheikh, and Mei Chen. 3D Point Cloud Reduction Using Mixed-Integer Quadratic Programming. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013.

[40] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[41] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast Image-Based Localization using Direct 2D-to-3D Matching. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[42] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving Image-Based Localization by Active Correspondence Search. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2012.

[43] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image Retrieval for Image-Based Localization Revisited. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.

[44] Stephen Se, David G. Lowe, and James J. Little. Vision-Based Global Localization and Mapping for Mobile Robots. *IEEE Transactions on Robotics (T-RO)*, 2005.

[45] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 2010.

[46] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision (ICCV), IEEE International Conference on*, 2003.

[47] Henrik Stewénius, Steinar H Gunderson, and Julien Pilet. Size Matters: Exhaustive Geometric Verification for Image Retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2012.

[48] Elena Stumm, Christopher Mei, and Simon Lacroix. Probabilistic Place Recognition with Covisibility Maps. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[49] Linus Svarm, Olof Enqvist, Magnus Oskarsson, and Fredrik Kahl. Accurate Localization and Pose Estimation for Large 3D Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[50] Tomasz Trzcinski, Vincent Lepetit, and Pascal Fua. Thick Boundaries in Binary Space and their Influence on Nearest-Neighbor Search. *Pattern Recognition Letters*, 2012.

[51] Jonathan Ventura and Tobias Hollerer. Wide-Area Scene Mapping for Mobile Visual Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2012.

[52] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. Global Localization from Monocular SLAM on a Mobile Phone. *IEEE Transactions on Visualization and Computer Graphics*, 2014.

[53] Andreas Wendel, Arnold Irschara, and Horst Bischof. Natural Landmarkbased Monocular Localization for MAVs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.