Using Image Sequences for Long-Term Visual Localization

Erik Stenborg[1,3] Torsten Sattler[2] Lars Hammarstrand[3]
[1] Zenseact [2]Czech Technical University in Prague [3]Chalmers University of Technology
erik.stenborg@zenseact.com torsten.sattler@cvut.cz lars.hammarstrand@chalmers.se

Abstract

Estimating the pose of a camera in a known scene, i.e.*, visual localization, is a core task for applications such as self-driving cars. In many scenarios, image sequences are available and existing work on combining single-image localization with odometry offers to unlock their potential for improving localization performance. Still, the largest part of the literature focuses on single-image localization and ignores the availability of sequence data. The goal of this paper is to demonstrate the potential of image sequences in challenging scenarios,* e.g.*, under day-night or seasonal changes. Combining ideas from the literature, we describe a sequence-based localization pipeline that combines odometry with both a coarse and a fine localization module. Experiments on long-term localization datasets show that combining single-image global localization against a pre-built map with a visual odometry / SLAM pipeline improves performance to a level where the extended CMU Seasons dataset can be considered solved. We show that SIFT features can perform on par with modern state-of-the-art features in our framework, despite being much weaker and a magnitude faster to compute. Our code is publicly available at github.com/rulllars.*

1. Introduction

Visual localization is the problem of estimating the position and orientation, *i.e.*, the pose, of a camera in an a-priori known scene. It is a key part of applications such as self-driving cars and other autonomous robots as well as mixed and virtual reality systems.

State-of-the-art visual localization methods establish 2D-3D matches between pixels in a camera image and 3D point positions in the scene [6–8, 14–16, 22, 29, 30, 34, 46, 65, 68, 80, 82]. The 2D-3D matches are then used for camera pose estimation, *e.g.*, by applying a PnP solver [1,31,41, 43,45,46] inside a robust estimator such as RANSAC [4,9, 17,25,47,66]. These visual localization methods typically use either local image descriptors [19,22,34,52] to explicitly match 2D features to 3D scene points or use machine learning, *e.g.*, via a random forest [15,16] or a convolutional neural network (CNN) [6,7,14], to regress the corresponding 3D scene coordinate per pixel. They build a scene representation, *e.g.*, a 3D Structure-from-Motion (SfM) model for local features or a CNN for scene coordinate regression, from a set of reference images.

Following classical methods [21, 33, 49, 50, 68, 77, 89], most modern localization algorithms localize each image independently of all other query images. They work well as long as sufficiently many matches can be established for each individual query image. However, strong condition changes, *e.g.*, day-night changes, between the reference and query images drastically reduce the number of available matches, often leading to localization failure. To avoid these failures, modern localization approaches use advanced local features [22, 32, 34, 53, 66, 72] and/or robust scene representations [46,72,81,82] that work better under changing conditions for long-term localization.

Some applications, *e.g.*, localizing historical photos or providing an initial pose estimate for a robot, require single images to be localized, and here the single-image algorithms remain highly relevant. However, in many other applications, *e.g.*, autonomous driving, the cameras record image sequences rather than taking individual pictures. There is a significant body of work on using sequences for visual localization [23,28,35,54,55,58,61,71,84,91]. These works combine single-image localization against a pre-built scene representation with local camera tracking based on visual and/or inertial odometry. The main insight is that localization and odometry complement each other: infrequent localization helps to prevent drift in the odometry. In turn, odometry enables real-time pose estimates, even if the localization algorithm is not real-time capable, and provides temporal constraints between frames. These approaches should result in more accurate localization under challenging conditions as they can bridge longer periods of time where not enough matches are available for localization, with the slight drawback of requiring more computations and memory. Still, recent works on localization almost exclusively focus on individual images.

This paper is aimed at answering the question: to which

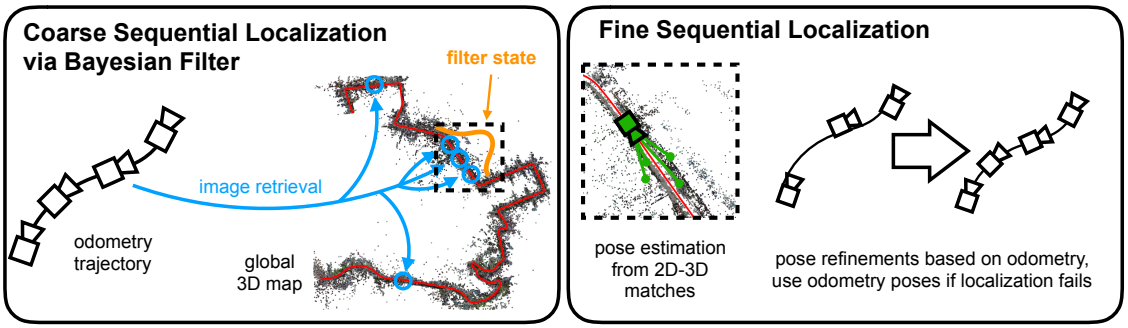

Figure 1. This paper implements a sequential localization system based on good practice from the literature: a coarse localization system (left) uses a Bayesian filter [3] that combines location hypotheses obtained via image retrieval against a global map with the trajectory estimated by the odometry. The coarse system also prunes retrieval results that are too far from the current location belief (orange). The remaining retrieved images are used in the fine localization stage (right) for pose estimation from 2D-3D matches [33, 65, 69]. Since individual pose estimates are noisy, the odometry trajectory is used for smoothing [54, 58, 84]. The odometry is further used to handle localization failures, *e.g.*, insufficiently many matches, by using odometry instead of localization poses.

degree does sequential localization boost performance under challenging conditions compared to single-image localization? In particular, we are interested in determining whether existing datasets, considered challenging for single-image methods, are already solved by using sequences and whether using sequences allows us to use weaker local features for localization. To this end, we use a modular pipeline that easily allows the integration of modern feature-based localization algorithms. Our pipeline, shown in Fig. 1, heavily inspired by existing works, consists of three stages: a coarse localization stage, based on a discrete Bayesian position filter [3], combines odometry information with global image-level descriptors [2, 83]. It provides a coarse estimate about the current position of an autonomous vehicle. Inspired by [28, 51], this coarse estimate is used as a prior for a single-image localization module that provides pose estimates with respect to a global map. Finally, a fine localization stage similar to [35, 54, 55, 58, 61, 84] refines the pose information provided by the single-image algorithm.

We do not intend to develop a novel sequence-based localization algorithm. Rather, we show that when image data is sequential, making use of odometry should be the first step taken to improve performance, before *e.g.*, looking into more advanced image features. The main contributions of this paper are: (**1**) Experimental results on established benchmark datasets [67] show that sequence-based methods significantly outperform single-image-based approaches and that modern techniques can make the localization problem seem harder than it actually is by focusing on individual images. (**2**) We analyze the impact of using sequence information in all stages through ablation studies and show that each stage benefits from them. (**3**) We show that sequence-based approaches achieve near optimal results on the popular extended CMU Seasons benchmark [3, 67]. (**4**) We show that, in the context of sequence-based localization, classical SIFT [52] features can perform on par with modern features such as D2-Net [22]. This is an important result, as SIFT features are computationally much more efficient but have also been shown to perform worse on the single-image localization task. Our results thus encourage a direction of research that uses as simple features as possible for efficiency while compensating their weaker discriminative power through sequential information. (**5**) We make our code publicly available at github.com/rulllars to foster research on sequence-based localization.

2. Related work

Single-image localization. One can trace the roots of visual localization way back to Kruppa [42], through Moravec's work [59] on rovers and into modern days. Most notable examples from history such as [50, 59, 73] are quite similar to modern methods in that they detect point features, match them to a map and then compute the pose from 3 or more correspondences. Modern visual localization methods often also do that, but have a few things in common that previous works do not. They are all based on learned features in some way or another, and they typically use multiple stages to perform localization. The multi-stage localization starts at a coarse level, and then proceeds to a finer localization using a locally sub-selected map of feature points. Examples from this family of visual localization include HF-Net [65], KAPTURE [32], and D2-Net [22]. HF-Net uses SuperPoint [19] features for the fine level localization and either NetVLAD [2] or a custom made aggregation of the last layer in SuperPoint as *whole image descriptor* for coarse level localization. KAPTURE uses R2D2 [34] features and the AP-GeM global image descriptor [64] for coarse localization. D2-Net does feature detection and description in one go using the same network, which leads to good features for fine level localization. It does not specify a *whole image descriptor* to be used for coarse localization, however, in experiments it is usually

paired with DenseVLAD [83] or NetVLAD [2]. The asymmetric matching of whole images to sparse keypoints that is used in S2DNet [29, 30], achieves something similar, with the added benefit of higher accuracy in the found correspondences compared to previous methods.

Another recent line of work on single-image localization has focused on machine learning [7, 8, 10, 11, 14, 15, 37, 38, 57, 75, 86, 88]. Scene coordinate regression approaches [7,8,14,15,57,75,88] train a random forest or convolutional neural network (CNN) to predict the corresponding 3D coordinate for each pixel. The resulting 2D-3D matches are then used for RANSAC-based pose estimation, reaching state-of-the-art accuracy on small-scale scenes, but currently not on larger and more complex scenes [8,72,79]. Absolute pose regression approaches [11,37,38,86] directly regress the camera pose from an input image, but are less accurate than 3D structure-based ones [6, 86] and are approximately at the level of accuracy of image retrieval [70]. While relative pose regression can achieve a level of accuracy comparable to structure-based methods [20], such approaches have not yet been evaluated in the context of localization in changing scenes. In this paper, we thus focus on traditional approaches based on local features, which have been shown to work in such a scenario [22,65,67].

SLAM/VO. Simultaneous Localization and Mapping (SLAM) and Visual Odometry (VO) are well known problems with a multitude of solutions that range back several decades [12]. Popular open source options for pure visual SLAM are PTAM [39], LSD-SLAM [24], LDSO [27], and ORB-SLAM [60]. Other solvers also make use of inertial constraints, *e.g.*, the extension to ORB-SLAM [61], Kasyanov *et al.* [36], Okvis [48], VINS-Mono [63] and ROVIO [5]. All of them perform quite well in benchmarks [18], and thus other factors are more important when selecting an algorithm to base further work on.

Sequential image localization. [35] describe Corvis, a full visual inertial SLAM system for both mapping and localization. [58, 84] propose localization methods where an agent runs a pure visual SLAM system locally and contacts a server for global localization against a pre-built map. [54, 55] use compressed maps to solve global localization both on the client side [54] and as a server process [55], and also integrates an IMU in their system. [23] present a consistent localization filter based on a Cholesky-Schmidt-Kalman filter that uses image correspondences and an IMU for localizing in a pre-defined map. [71] present Maplab, an open source framework for benchmarking visual inertial mapping and localization. Maplab includes ROVIO [5] augmented with a localization component and a number of other open source components, *e.g.*, [48]. [76] use semantically labeled point clouds and IMU in FGSN. Recently, there are also learning based methods, such as KFNet [91], for solving the sequential localization problem in a data driven way. Common to all these approaches is that they combine visual(-inertial) odometry with localization against a pre-built scene representation.

Our work builds heavily on these previous works. We use ORB-SLAM [60] as a SLAM system and add our localization extensions on top of it. Additionally, we make use of correspondences to a pre-built map. This map uses SIFT [52] or D2-Net [22] to detect and describe feature points, in order to get better invariance to seasonal changes compared to the ORB features built into ORB-SLAM. We also build a coarse localization system based on the discrete Bayesian position filter from Badino et al [3], but replace its custom descriptors with the more recent DenseVLAD descriptors [83]. The choice of these algorithms is based on a combination of performance, availability of source code and customizability. We do not claim that our sequential localization pipeline beats state-of-the-art results of prior work. However, we still show that sequence-based localization significantly improves upon single-image localization in challenging conditions, even when using simpler features and existing algorithms.

The approaches discussed above, including ours, integrate single-image localization into a VO/SLAM system. A complementary approach is to integrate VO/SLAM trajectories into a localization system [13, 26, 40, 78, 85]. Modeling a sequence of images as a generalized camera [62] enables these methods to localize multiple images simultaneously. Naturally, the localization results could again be integrated into VO/SLAM, *e.g.*, using our method.

3. Problem formulation

The problem is to localize query images relative to a global map built from reference images with known poses. Query images arrive in a sequence taken from a single camera or multiple cameras, mounted rigidly on a moving body. The images are captured asynchronously at times $[t_1, \ldots, t_n]$, and we assume that the intrinsics and the extrinsic calibration of the cameras with respect to the platform are known. The platform is accompanied by relative motion constraints in the form of an odometry signal and sometimes also rough absolute position constraints in the form of a GPS observation. Thus, given a sequence of query images and additional positional constraints, we seek to estimate the trajectory of the moving body w.r.t. the map.

Odometry constraints come from noisy observations of the average translational, $\mathbf{v}_k = [v_k^x, v_k^y, v_k^z]^T$, and angular velocities, $\boldsymbol{\omega}_k = [\omega_k^x, \omega_k^y, \omega_k^z]^T$, between two time instances. Similarly, a GPS signal, $\mathbf{P}_k^{\text{gps}}$, if available, is sampled at the same time instances, and gives noisy estimates of the absolute position.

From the query images and odometry constraints, we want to estimate the 4×4 pose matrices, $\mathbf{T}_{W,B_k} \in SE(3)$,

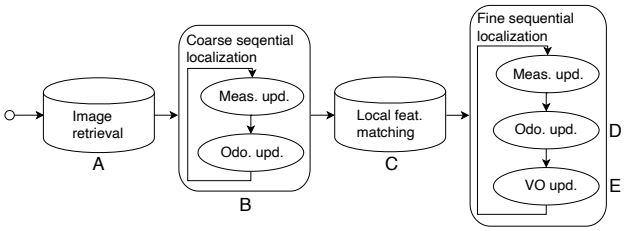

Figure 2. Our system is split into two separate loops with the local feature matching algorithm sandwiched in between, and an image retrieval algorithm in front. We label the individual components from A to E, and refer to those in our ablation study. A: DenseVLAD [83], B: Coarse filter [3], C: SIFT [52] or D2-Net [22], D: Odometry (6-DoF), E: Original ORB-SLAM [60].

of the body (B_k) at times t_k in the world coordinate system (W). Thus, the trajectory that we seek to estimate is described by the poses $\{\mathbf{T}_{W,B_1}, ..., \mathbf{T}_{W,B_n}\}$. We consider this trajectory estimation problem as a batch problem, *i.e.*, also measurements after time t_k are available when estimating the pose at time t_k. However, to be useful for certain applications, the presented methods could also be used in an online setting with minor modifications, potentially at the price of a lower accuracy.

4. System overview and models

We choose a system design that is divided in four separate components shown in Fig. 2. We follow the coarse to fine progression that is often used by single-image approaches: (A) image retrieval for each query image against the map provides a set of locations potentially visible in the image. (B) the coarse localization module combines this information with the odometry signal to obtain a coarse location estimate. In turn, the coarse estimate is used to filter out unlikely retrieval results, *i.e.*, retrieved reference images taken too far from the estimate. (C) as for single-image methods, the remaining retrieved images are used to guide feature matching: 2D-3D matches between features in the query image and 3D points from a global 3D map that are visible in the retrieved images are computed and used for pose estimation. The output is a camera pose as well as the set of 2D-3D matches that are inliers to the pose. (D, E) the fine localization module refines these poses using the odometry trajectory and the inlier matches.

A main reason for this design is to make integration with existing single-image localization algorithms easier. If the components are reorganized slightly such that the local feature matching takes place in the loop inside the last block instead of as a stand alone component, it is possible to run the system online in a causal way. The coarse localization is a rough localization, *e.g.*, in an autonomous driving scenario to identify the correct road segment and the robot's rough position along it. We use the sequential datasets from [67] where the trajectories mostly follow the same route. In this case, there is no need to model, *e.g.*, forks in the road. Thus, for our coarse model, the state is simply a scalar that indicates how far along the reference trajectory the robot is. Lateral offset and rotation offset from the reference trajectory is not estimated by the coarse localization. For fine localization, we use a 6-DoF state that describes the robot pose in the same coordinate frame as the reference trajectory. One could include additional state variables to track, *e.g.*, body velocity, acceleration, sensor bias, and various calibrations. For simplicity we skip such variables and instead use noise models that cover the inaccuracies caused by this.

Body motion model. The odometry data describe the body motion over time, and form constraints by integrating linear and angular speed over time as

$$\mathbf{T}_{W,B_{k+1}} = \mathbf{T}_{W,B_k} f(\mathbf{v}_k, \boldsymbol{\omega}_k) \tag{1}$$

$$f(\mathbf{v}_k, \boldsymbol{\omega}_k) = \begin{bmatrix} (e^{[\tau_k \boldsymbol{\omega}_k + \mathbf{q}_k^{\omega}]_\times}) & (\tau_k \mathbf{v}_k + \mathbf{q}_k^{v}) \\ \mathbf{0} & 1 \end{bmatrix} , \tag{2}$$

where the $[\cdot]_\times$-operator creates a skew symmetric matrix from a vector, $\tau_k = t_k - t_{k-1}$ is the time interval between images, $\mathbf{q}_k^{\omega} \sim \mathcal{N}(\mathbf{0}, \tau_k \mathbf{I} \sigma_\omega^2)$ is the noise in the angular velocity measurements, and $\mathbf{q}_k^{v} \sim \mathcal{N}(\mathbf{0}, \tau_k \mathbf{I} \sigma_v^2)$ is the noise in the velocity measurements. The noise is independent for each dimension and characterized by its standard deviations, σ_v and σ_ω, which are assumed to be constant parameters. The relative motion $f(\mathbf{v}_k, \boldsymbol{\omega}_k) = \mathbf{T}_{B_k,B_{k+1}}$ describes the motion from time t_k to t_{k+1}, in the coordinate frame of the body at time t_k. For coarse localization, we only use the forward component of the velocity, v_k^x.

Coarse and fine maps. For both coarse and fine localization, the reference trajectory and the images that belong to it are converted into a map, comprising a set of locations, a set of image feature descriptors, and some auxiliary data.

For coarse localization, the map is a set of tuples, comprising the distance along the route x_k, the pose $\mathcal{P}_k^R$, and an image-level descriptor $\mathcal{D}_k^R$. The image-level descriptors are high dimensional vectors of unit length, *e.g.*, DenseVLAD [83], that describe the content of the image in a way that is more robust to appearance changes than just using the pixel values. Using interpolation, we can now view both the image-level descriptor and the pose of the reference trajectory as functions of a continuous distance along the route.

For fine localization, we build a point cloud map, where each point is triangulated using the known poses of the reference trajectory and the matched local image descriptors in the reference images. The map $\mathcal{M}$ then consists of m 3D feature points, $\mathcal{U}_k$, with attached descriptor vectors, $\mathcal{D}_k$, and visibility information, $\mathcal{V}_k$, that indicate from which reference images the points were observed $\mathcal{M} = \{(\mathcal{U}_1, \mathcal{D}_1, \mathcal{V}_1), ..., (\mathcal{U}_m, \mathcal{D}_m, \mathcal{V}_m)\}$.

Coarse image measurement model. Beside the motion

model, we need measurement models for the filters used for coarse and fine localization. For coarse localization this can be realized using image-level descriptors and image retrieval against a database (map). We use a simple measurement model for the image-level descriptor, $\mathbf{D}^Q$, of the query image as a function of the distance traveled along the route

$$\mathbf{D}^Q(x) = \mathcal{D}^R(x) + \mathbf{d} \tag{3}$$

$$\mathbf{d} \sim \mathcal{N}(\mathbf{0}, \sigma_D^2 \mathbf{I}) \ , \tag{4}$$

where $\mathcal{D}^R(x)$ is the database descriptor at position x. This model ignores the normalization of the query image descriptor and also approximates the descriptor noise, $\mathbf{d}$, as independent and Gaussian, which is overly simplified, but makes things easier when implementing a filter.

Coarse GPS measurement model. Another, and arguably simpler, approach to coarse localization is to use GPS if available. The measurement we receive from a GPS is a 3D position, $\mathbf{P}^{\text{gps}}$ which is modeled as

$$\mathbf{P}^{\text{gps}}(x) = \mathcal{P}^R(x) + \mathbf{p} \tag{5}$$

$$\mathbf{p} \sim \mathcal{N}(\mathbf{0}, \sigma_P^2 \mathbf{I}) \ . \tag{6}$$

Fine measurement model. The fine measurement model is more complicated since the measurement now consists of both an image coordinate and a feature descriptor. A common approach is to first use the descriptor for selecting the best match between image point and map, requiring the match to be some factor closer than the second best [52]. After this, the image coordinate observation is modeled as

$$\begin{bmatrix} a_{jk} \\ b_{jk} \end{bmatrix} = g(\mathbf{x}_k, \mathbf{m}_j, \mathbf{C}_k) + \mathbf{r}_{jk} \ , \tag{7}$$

where (a_{jk}, b_{jk}) is the coordinate in the image plane corresponding to the observation of 3D map point $\mathbf{m}_j$ at time t_k, $\mathbf{C}_k$ are the known camera parameters of image k, and $\mathbf{r}_{jk}$ is observation noise. The observation noise $\mathbf{r}_{jk}$ can be considered normally distributed if the match is known to be correct, however, matches are quite often wrong, and thus, a more robust noise model that considers outliers is needed. The noise model we use is independent and identical for both a- and b-directions in the image. We assume that outliers are uniformly distributed over the image, and that the probability of correct matches is constant.

This measurement model is used both for observations of points from the pre-existing map (*global map*) and from the map that is built online as part of the odometry (from here on denoted *local map*). A notable difference between the two maps is that map point positions of the pre-existing map are considered constant, while the local map is unknown to start with, and thus the map points need to be estimated simultaneously with the poses, as done in SLAM.

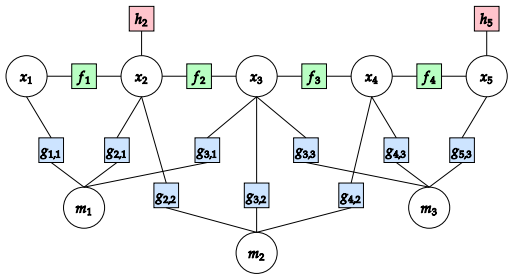

Figure 3. Factor graph representation of a small problem.

The models and variables that we solve for can be illustrated using a factor graph. Fig. 3 shows a small example with 5 poses and 3 points in the local map. The variables to be estimated are shown as white disks, while the models that form constraints on the variables are shown as colored squares. The motion model constraints are shown with green squares labeled f_k, the observations of local map points are shown as blue squares labeled g_{kj}, and the observations to the fixed, pre-existing map are shown as red squares labeled h_k. Note that the red squares are unitary constraints since the points in the pre-existing map are fixed. In terms of the components shown in Fig. 2, the components C, D, and E are represented by the red, green, and blue squares in the factor graph, respectively.

5. Implementation

Coarse sequential smoother. The coarse localization is based on the Topometric 1-D filter by Badino *et al*. [3]. It predicts an approximate position by first estimating a 1-D position along the mapping route and then mapping this to a 6-DoF pose, which is a simpler problem compared to directly predicting a 6-DoF pose. The estimator is a discrete Bayesian smoother, based on the discrete Bayesian filter used in [3], but with an added backward smoothing pass. The backward pass is particularly useful to resolve uncertain matches early in the trajectory or towards the end of long outages when no, or only few, images are available to the forward pass. These uncertainties get resolved in the backwards pass by leveraging information from images in the future. We discretize the mapping route into $N = 20,000$ equally long segments, which translates into about 0.25 m per segment for the Extended CMU Seasons dataset from [3,67]. The state is represented as a probability mass function, *i.e*., a vector, $\mathbf{P}^x = [p_1^x, ..., p_N^x]$, where the first element, p_1^x, represents the probability of being somewhere in the first segment, p_2^x represents the probability of being somewhere in the second segment, *etc*. This state can easily represent the highly multi-modal density that occurs when there are multiple locations with similar image-level descriptors.

The reference image-level descriptors are only available at some discrete locations along the route that do not co-

Component	time [s]
DenseVLAD (CPU)	0.35
SIFT (GPU)	0.06
D2-Net (GPU)	1.2
Coarse Smoother	0.030
Fine smoother (w/o VO)	0.015
Fine smoother (w/ VO)	0.15

Table 1. Average time per image for executing each component on a 2017 laptop with an Intel Core i7-7700HQ CPU @ 2.80GHz, 32 GB RAM, and a GeForce GTX 1050 Ti Mobile GPU.

incide with the equidistant subdivision chosen for the state representation. We thus choose to represent the likelihood at position x_k by the interpolation of the likelihood from the two nearest positions available from the reference route.

Fine sequential smoother. We base the fine localization smoother on the popular ORB-SLAM implementation [60]. ORB-SLAM provides a solution for the local map-factors (blue squares in Fig. 3). On top of that, we add support for odometry input (the green factors in Fig. 3), localization to a fixed map (the red factors in Fig. 3), and camera rigs, *i.e.*, multiple cameras that are rigidly mounted on a moving body. More details on the fine sequential smoother are given in the supplementary material.

6. Experimental Evaluation

For experimental evaluation, we use the Extended CMU Seasons [3, 67] and the RobotCar Seasons [56, 67] datasets from the popular visual localization benchmark in [67]. Both datasets are generated from cameras mounted on a car, and thus provide images of mainly road scenes. Still, both provide significant challenges in the form of seasonal changes of vegetated scenes (CMU Seasons) and day-night changes (RobotCar). To measure localization performance, we follow [67] and report the percentage of query images localized within Xm of position and a Y degrees of rotation error. As in [67], we use fine (0.25m, 2°), medium (0.5m, 5°), and coarse (5m, 10°) thresholds. Execution time performance for each component of the system is reported in Tab. 1. Adding components that process multiple image frames with odometry or visual odometry clearly has drawbacks in terms of execution time as well as memory requirements. However, switching from using simple SIFT features to D2-Net features clearly adds a larger penalty in terms of execution time.

The Extended CMU Seasons and RobotCar Seasons datasets from [67] are both subsets of originally larger datasets [3, 56]. In both cases, spatial subsampling was used, while the CMU dataset still is in sequence (albeit with larger gaps), the RobotCar dataset is split into a set of seperate locations. As larger gaps in the images sequences, caused by spatial subsampling, can present problems for the tracker in a SLAM system, we use the original image sequence where no images were removed. However, localization results are evaluated only on the images used by the Extended CMU Seasons and RobotCar Seasons datasets to ensure comparable results with single-image methods. For the CMU dataset the odometry signal was generated from the ground truth poses, by adding noise and bias roughly corresponding to a normal automotive spec IMU, while for RobotCar, we have used the linear and rotational speed signals provided by the original dataset.

We use both D2-Net [22] features, which have been shown to be reasonably robust to seasonal and illumination changes, and classical SIFT [52] features, which are much more efficient but less robust. For coarse localization, we use DenseVLAD [83] (DVLAD) as the image-level descriptor on CMU. On RobotCar, however, the coarse localization is based on GPS observations as the vehicle during query sometimes travel in the opposite direction compared to the images in the map base resulting in very large orientation errors using image retrieval.

We compare our sequence-based results with generalized cameras, as implemented by Wald *et al*. [87], and state-of-the-art, published single-image methods chosen based on the top-performing methods from visuallocalization.net. For the generalized cameras, the components D and E are replaced by a generalized PnP solver [44], while keeping components A, B and C as before.

Coarse localization. In the datasets used for evaluation, some locations are difficult because the environment exhibits repeated patterns or a general lack of features. As we can see from the top row in Tab. 2, image retrieval using image-level descriptors (DVLAD (A)) struggles to localize a considerable portion of images in all three parts of the CMU seasons dataset. The effect of using sequences for coarse localization can be seen on the second row (DVLAD (A, B)), where the low precision regime improves to nearly 100%, just by adding the longitudinal odometry information. Note that, while the effect of the motion model is very noticeable on the coarse localization, it does not reach the same precision as structure-based methods do due to the inherent coarseness of the method. However, combining the single-image D2-Net baseline (which uses image retrieval (A) and feature matching (C)) with the coarse localization model (D2-Net (A, B, C)) leads to a considerable improvement on the CMU dataset. This is especially true for the challenging park subset, which consists predominantly of vegetation that changes drastically with seasons. Similar observations can be made when using SIFT.

Fine localization. Adding sequence information through odometry to the fine level localization has an equally large effect on the performance as for coarse localization. Combining D2-Net with the fine localization module (D, E) further improves performance (*c.f*. Tab. 2). This approach

Method	Details	CMU / Urban	CMU / Suburban	CMU / Park	RobotCar / Day	RobotCar / Night
		.25 / .50 / 5.0 m 2 / 5 / 10 °	.25 / .50 / 5.0 m 2 / 5 / 10 °	.25 / .50 / 5.0 m 2 / 5 / 10 °	.25 / .50 / 5.0 m 2 / 5 / 10 °	.25 / .50 / 5.0 m 2 / 5 / 10 °
DVLAD [83]	A	22.2 / 48.6 / 92.8	9.8 / 26.6 / 85.2	10.3 / 27.1 / 77.0	8.1 / 31.8 / 90.7	1.3 / 5.7 / 25.5
Our (DVLAD)	A, B	41.9 / 72.9 / 99.8	17.1 / 46.9 / 99.5	17.7 / 55.6 / 99.9	-	-
Our (SIFT)	A, B, C *	85.4 / 88.9 / 91.9	67.8 / 72.6 / 79.4	51.5 / 55.2 / 60.2	-	-
Our (SIFT)	A, C *	85.5 / 88.9 / 91.9	67.8 / 72.6 / 79.4	51.6 / 55.2 / 60.2	-	-
Our (SIFT)	A, C	85.7 / 89.5 / 95.3	68.0 / 73.5 / 88.7	52.5 / 58.8 / 78.3	-	-
Our (SIFT)	A, B, C	87.3 / 92.3 / 98.4	69.3 / 77.8 / 96.8	56.1 / 71.9 / 96.8	-	-
D2-Net [22]	A, C	94.0 / 97.7 / 99.1	93.0 / 95.7 / 98.3	89.2 / 93.2 / 95.0	53.4 / 78.7 / 95.2	22.0 / 41.0 / 54.7
Our (D2-Net)	A, B, C	95.6 / 98.7 / 99.9	96.0 / 97.9 / **100.0**	95.7 / 99.1 / 99.8	53.8 / 79.4 / 96.0	39.7 / 74.8 / **95.6**
Our (SIFT)	A, B, C, D	98.4 / **100.0** / **100.0**	99.1 / 99.8 / **100.0**	96.9 / 99.7 / **100.0**	54.9 / 79.8 / 95.3	11.7 / 20.8 / 32.9
Our (D2-Net)	A, C, D	98.6 / **100.0** / **100.0**	98.9 / 99.9 / **100.0**	97.7 / 99.7 / **100.0**	54.1 / 79.6 / 96.0	36.5 / 71.7 / 91.9
Our (D2-Net)	A, B, C, D, E	**98.9** / **100.0** / **100.0**	98.3 / 99.1 / **100.0**	97.9 / **100.0** / **100.0**	-	-
Our (D2-Net)	A, C, D, E	**98.9** / 99.9 / **100.0**	98.5 / 99.5 / **100.0**	98.1 / **100.0** / **100.0**	-	-
Our (D2-Net)	A, B, C, D	98.2 / 99.9 / **100.0**	99.0 / 99.8 / **100.0**	**98.2** / **100.0** / **100.0**	51.8 / 80.1 / **100.0**	44.2 / 76.0 / **95.6**
D2-Net + Gen Cam	rig	98.2 / **100.0** / **100.0**	99.0 / 99.9 / **100.0**	98.0 / 99.9 / **100.0**	51.9 / 80.3 / 99.2	44.9 / 75.4 / 95.5
D2-Net + Gen Cam	10 cams	98.2 / 99.9 / 99.9	99.0 / 99.9 / 99.9	97.8 / 99.8 / 99.8	52.1 / 80.0 / 99.5	**45.0** / 75.4 / 95.5
D2-Net + Gen Cam	20 cams	97.3 / 99.4 / 99.6	98.7 / 99.6 / 99.7	97.2 / 99.5 / 99.6	51.6 / 79.9 / 99.5	**45.0** / 75.6 / 95.5
D2-Net + Gen Cam	50 cams	92.4 / 97.1 / 97.9	93.7 / 97.8 / 98.3	90.7 / 97.7 / 98.6	49.8 / 79.9 / 99.6	42.4 / **76.8** / 95.5
SIFT + Gen Cam	rig	92.5 / 96.0 / 97.3	86.0 / 90.1 / 93.3	66.3 / 71.2 / 75.8	51.9 / 80.0 / 98.9	14.2 / 25.0 / 42.2
SIFT + Gen Cam	10 cams	94.4 / 97.5 / 98.5	90.3 / 94.5 / 97.2	73.5 / 79.9 / 85.7	52.4 / 80.2 / 99.1	16.7 / 28.0 / 45.9
SIFT + Gen Cam	20 cams	94.1 / 97.4 / 98.6	91.4 / 95.2 / 98.1	75.9 / 83.1 / 89.2	52.0 / 80.2 / 99.2	18.2 / 31.9 / 51.0
SIFT + Gen Cam	50 cams	88.2 / 95.5 / 97.5	83.0 / 93.0 / 97.4	71.6 / 82.9 / 91.2	50.5 / 79.8 / 99.3	18.7 / 36.5 / 59.6
KAPTURE-R2D2-APGeM [32]		96.7 / 98.9 / 99.7	94.4 / 96.8 / 99.2	83.6 / 89.0 / 95.5	**55.1** / **82.1** / 97.3	28.8 / 58.8 / 89.4
Hierarchical-Localization [65]		91.6 / 96.4 / 99.1	84.7 / 91.5 / 98.6	69.3 / 77.8 / 90.5	-	-
Asym. Hypercol. Matching [29]		65.7 / 82.7 / 91.0	66.5 / 82.6 / 92.9	54.3 / 71.6 / 84.1	45.7 / 78.0 / 95.1	22.3 / 61.8 / 94.5
Vis. Loc. w/ Dense Semantic Map [74]		-	-	-	54.6 / 81.9 / 96.9	14.8 / 33.0 / 51.3
PFSL FGSN [46, 76]		94.1 / 99.3 / **100.0**	**100.0** / **100.0** / **100.0**	97.6 / 99.9 / 99.9	-	-

Table 2. **Evaluation on the Extended CMU Seasons and RobotCar Seasons datasets**. Results from our ablation study are ordered in ascending order of performance in the high precision regime on the park part. A to E indicate which components from Fig. 2 are used; A: Image retrieval, B: coarse-level odometry, C: local feature matching, D: fine-level odometry, E: fine-level visual odometry. Methods marked with (*) only use the poses directly from the local features algorithm and do not fall back to coarse localization results when feature-based pose estimation fails. This shows that coarse localization only makes a difference for the images that SIFT matching fails on. Three state-of-the-art single-image algorithms are included for reference below the double lines. The bottom line also includes a sequential method as reference. Note that for the RobotCar dataset, sequential data was only available for about 80% of the original dataset. Thus, we limited the evaluation to this part. Due to feature tracking failures during night, VO failed and was not included for RobotCar.

clearly outperforms state-of-the-art single-image methods and performs on par, or better, than the sequence-based approach from [46, 76]. [46, 76] use a particle filter approach based on a scene representation specifically trained for the CMU Seasons dataset. In contrast, our approach does not require such training.

Based on our results, we can consider the Extended CMU Seasons dataset to be practically solved in terms of pose accuracy. However, this does not imply that the dataset itself is solved. Rather, optimizing for accuracy *and* run-time or accuracy *and* memory consumption instead of *only* accuracy will become important. In this regard, our results obtained with SIFT suggest that sequence-based localization could play an important part for this research direction: Tab. 2 shows that when using odometry our algorithm using SIFT (A, B, C, D) essentially reaches the same level of accuracy as when using the more robust D2-Net features. In contrast, SIFT without odometry (A, C) performs clearly worse. This shows that sequence-based localization enables us to use weaker local features that are more efficient to compute (optimizing for accuracy *and* run-time). At the same time, it shows that sequence information can bridge periods of time where localization against the global map fails, which can be exploited for map compression (optimizing accuracy *and* memory) [54].

While the Extended CMU Season is practically solved in terms of pose accuracy, Tab. 2 shows that this is not the case for the RobotCar dataset. The reason that the smoother does not reach 100% in the coarse regime for RobotCar, is that there is a part of the trajectory where the car drives the other way during the mapping sequence. Even when using the GPS-based coarse localization, this area is difficult for the nighttime queries: the rear-facing camera is almost useless for localization (facing the wrong way) and the two side cameras exhibit quite significant noise and motion blur causing problems for the D2-Net matching. After manual inspection, we see that there is not a single correct match in this area for the night sequences. However, a human could probably work out enough correspondences in these images to get a reasonable localization. This, together with the fact that the fast ORB features used by ORB-SLAM did not manage to initialize tracking in many of the night sequences from RobotCar, is an indication that localization at night from a map built during daytime conditions is still an unsolved problem, even when using sequences instead of single-images. Yet, we observe that using sequences significantly improves localization performance in night conditions compared to only using individual images.

Interval \ Bias	"A, B, C, D, E" ×1	×2	×5	"A, B, C, D" ×1	×2	×5
0.1 s	97.7	97.8	98.0	97.4	97.3	95.9
0.5 s	97.1	97.4	97.7	97.8	97.6	95.8
2.0 s	97.1	97.4	97.2	96.5	95.4	71.7
5.0 s	93.7	94.2	93.5	93.9	73.1	33.5
10.0 s	85.9	85.2	80.6	63.4	33.0	6.0

Table 3. Evaluation of varying interval between localization and gyro bias on the Extended CMU Seasons dataset. We show the percentage of images localized within the threshold (0.25m / 2°).

In the higher precision regimes (0.25 m, 0.5 m) for the RobotCar daytime queries, sequence information provides no improvement and there is a significant gap to 100%. This gap is harder to explain, considering that quite similar conditions from the CMU dataset show performance very close to 100%. One possibility is that the ground truth poses are not precise enough for this regime. The facts that no methods listed on visuallocalization.net reaches 70% in the high precision regime for daytime queries, and that recently significantly improved poses for another dataset from [67] were released [90], support this idea.

Comparison with generalized cameras. Tab. 2 also reports results for sequence-based methods that localize trajectories of images. Here, the same odometry used by our method is used to obtain the poses that define the trajectories. We evaluate variants that use only the multi-camera rig and that use longer sequences consisting of 10, 20, and 50 images. Note that longer sequences can decrease localization accuracy as there is no mechanism to reduce drift in the odometry poses (which is more prominent for longer sequences). In contrast, this is not a problem for our approach, which uses localization against the map to correct odometry drift. Generalized camera-based approaches perform equally well as our method when using D2-Net. For SIFT features, we observe that our method outperforms gen. camera-based approaches for all but the RobotCar nighttime queries. For the latter, typically very few matches are found using SIFT, resulting in frequent localization failures and thus drift in the odometry. Here, using matches distributed over multiple images for pose estimation improves performance. Given the complementary nature of gen. camera-based methods and our approach, combining them into a single system is an interesting direction for future research.

When visual odometry makes a difference. Visual odometry and regular, non-visual, odometry both provide the means to relate the location of nearby images in a sequence. Naturally, one would use all information sources available for best results. However, good visual odometry is not quite as simple as regular odometry in terms of computational resources and development complexity. So when is it worth the effort to consider visual odometry?

In contrast to inertial measurements, images for visual odometry are captured at a lower frequency and do not exhibit the bias built into inertial sensors, which allow reducing the higher amount of drift that is typical for inertial odometry. To answer the question, we thus vary the amount of bias added to the angular velocity measurements (varying the quality of the odometry sensors) and the amount of time between successful matches against the pre-existing map (simulating a sparser map). Tab. 3 unsurprisingly shows that visual odometry is most useful when the odometry sensors are inaccurate and the map is sparse. The main insight from this experiment is that, with reasonably good odometry (the "×1" should be representative of a normal automotive spec gyroscope), there is no difference until rather long prediction times, *i.e.*, most of the benefit of using sequences can be gained by a rather small addition in sensor cost and computational complexity.

7. Conclusions

The main focus of this paper was to, in the context of long-term localization, investigate to what degree using sequence information improves localization performance over the predominantly single-image-based methods used in the literature. To this end, we implemented a sequence-based localization pipeline based on good practice from the literature. Through detailed experiments, the paper contributes the following insights to the literature: (1) sequential information can significantly improve performance under challenging conditions such as seasonal and day-night changes, to the point where the Extended CMU Seasons dataset can be considered solved in terms of pose accuracy. (2) sequential information has a relatively larger importance for successful localization in hard conditions than switching from traditional image features to more robust and complex one. For CMU Seasons, we observe that localization performance is practically identical for traditional SIFT features and state-of-the-art D2-Net features when using sequence information. This result opens up interesting avenues of research, such as using cheaper features for efficiency while compensating for their lack in descriptive power through sequential localization. To foster such research, we make our code publicly available. (3) we show that the observed gain in performance can be achieved with simple inertial odometry, *i.e.*, without visual odometry, if decent odometry sensors are available and localization to the pre-existing map is not too sparse. This opens up the possibility to using simpler, and thus easier to implement, odometry systems.

Acknowledgments. This work was supported by the Swedish Foundation for Strategic Research (Semantic Mapping and Visual Navigation for Smart Robots), the Chalmers AI Research Centre (VisLocLearn), the EU Horizon 2020 project RICAIP (grant agreeement No 857306), and the European Regional Development Fund under IMPACT No. CZ.02.1.01/0.0/0.0/15 003/0000468.

References

[1] C. Albl, Z. Kukelova, and T. Pajdla. Rolling Shutter Absolute Pose Problem With Known Vertical Direction. In *CVPR*, 2016.

[2] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[3] H. Badino, D. Huber, and T. Kanade. Visual topometric localization. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 794–799. IEEE, 2011.

[4] D. Barath, J. Noskova, M. Ivashechkin, and J. Matas. MAGSAC++, a Fast, Reliable and Accurate Robust Estimator. In *CVPR*, 2020.

[5] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 298–304. IEEE, 2015.

[6] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. DSAC - Differentiable RANSAC for Camera Localization. In *CVPR*, 2017.

[7] E. Brachmann and C. Rother. Learning Less is More - 6D Camera Localization via 3D Surface Regression. In *CVPR*, 2018.

[8] E. Brachmann and C. Rother. Expert sample consensus applied to camera re-localization. In *ICCV*, 2019.

[9] E. Brachmann and C. Rother. Neural-guided RANSAC: Learning where to sample model hypotheses. In *ICCV*, 2019.

[10] E. Brachmann and C. Rother. Visual camera relocalization from RGB and RGB-D images using DSAC. *arXiv:2002.12324*, 2020.

[11] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. Geometry-aware learning of maps for camera localization. In *CVPR*, 2018.

[12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[13] F. Camposeco, T. Sattler, and M. Pollefeys. Minimal Solvers for Generalized Pose and Scale Estimation from Two Rays and One Point. In *ECCV*, 2016.

[14] T. Cavallari, L. Bertinetto, J. Mukhoti, P. Torr, and S. Golodetz. Let's take this online: Adapting scene coordinate regression network predictions for online rgb-d camera relocalisation. In *3DV*, 2019.

[15] T. Cavallari, S. Golodetz, N. Lord, J. Valentin, V. Prisacariu, L. Di Stefano, and P. H. S. Torr. Real-Time RGB-D Camera Pose Estimation in Novel Scenes using a Relocalisation Cascade. *TPAMI*, 2019.

[16] T. Cavallari, S. Golodetz, N. A. Lord, J. Valentin, L. Di Stefano, and P. H. S. Torr. On-The-Fly Adaptation of Regression Forests for Online Camera Relocalisation. In *CVPR*, 2017.

[17] O. Chum and J. Matas. Optimal Randomized RANSAC. *PAMI*, 30(8):1472 –1482, 2008.

[18] J. Delmerico and D. Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509. IEEE, 2018.

[19] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[20] M. Ding, Z. Wang, J. Sun, J. Shi, and P. Luo. CamNet: Coarse-to-fine retrieval for camera re-localization. In *ICCV*, 2019.

[21] M. Donoser and D. Schmalstieg. Discriminative Feature-to-Point Matching in Image-Based Locallization. In *CVPR*, 2014.

[22] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-net: A trainable cnn for joint detection and description of local features. In *CVPR 2019-IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[23] R. C. DuToit, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis. Consistent map-based 3d localization on mobile devices. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6253–6260. IEEE, 2017.

[24] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[25] M. Fischler and R. Bolles. Random Sampling Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography. *CACM*, 24:381–395, 1981.

[26] V. Fragoso, J. DeGol, and G. Hua. gdls*: Generalized pose-and-scale estimation given scale and gravity priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2210–2219, 2020.

[27] X. Gao, R. Wang, N. Demmel, and D. Cremers. Ldso: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204. IEEE, 2018.

[28] M. Geppert, P. Liu, Z. Cui, M. Pollefeys, and T. Sattler. Efficient 2D-3D Matching for Multi-Camera Visual Localization. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[29] H. Germain, G. Bourmaud, and V. Lepetit. Sparse-to-dense hypercolumn matching for long-term visual localization. In *2019 International Conference on 3D Vision (3DV)*, pages 513–523. IEEE, 2019.

[30] H. Germain, G. Bourmaud, and V. Lepetit. S2DNet: Learning Accurate Correspondences for Sparse-to-Dense Feature Matching. In *ECCV*, 2020.

[31] R. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *IJCV*, 13(3):331–356, 1994.

[32] M. Humenberger, Y. Cabon, N. Guerin, J. Morat, J. Revaud, P. Rerole, N. Pion, C. de Souza, V. Leroy, and G. Csurka. Robust image retrieval-based visual localization using kapture, 2020.

[33] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From Structure-from-Motion Point Clouds to Fast Location Recognition. In *CVPR*, 2009.

[34] R. Jerome, P. Weinzaepfel, C. De Souza, and M. Humenberger. R2D2: Reliable and Repeatable Detectors and Descriptors. In *NeurIPS*, 2019.

[35] E. S. Jones and S. Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430, 2011.

[36] A. Kasyanov, F. Engelmann, J. Stückler, and B. Leibe. Keyframe-based visual-inertial online slam with relocalization. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6662–6669. IEEE, 2017.

[37] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017.

[38] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*, 2015.

[39] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[40] L. Kneip, H. Li, and Y. Seo. Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability. In *European Conference on Computer Vision*, pages 127–142. Springer, 2014.

[41] L. Kneip, D. Scaramuzza, and R. Siegwart. A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *CVPR*, 2011.

[42] E. Kruppa. *Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung*. Hölder, 1913.

[43] Z. Kukelova, M. Bujnak, and T. Pajdla. Real-Time Solution to the Absolute Pose Problem with Unknown Radial Distortion and Focal Length. In *ICCV*, 2013.

[44] Z. Kukelova, J. Heller, and A. Fitzgibbon. Efficient Intersection of Three Quadrics and Applications in Computer Vision. In *CVPR*, 2016.

[45] V. Larsson, Z. Kukelova, and Y. Zheng. Making Minimal Solvers for Absolute Pose Estimation Compact and Robust. In *ICCV*, 2017.

[46] V. Larsson, T. Sattler, Z. Kukelova, and M. Pollefeys. Revisiting Radial Distortion Absolute Pose. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[47] K. Lebeda, J. E. S. Matas, and O. Chum. Fixing the Locally Optimized RANSAC. In *British Machine Vision Conference (BMVC)*, 2012.

[48] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[49] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide Pose Estimation Using 3D Point Clouds. In *ECCV*, 2012.

[50] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *European conference on computer vision*, pages 791–804. Springer, 2010.

[51] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele. Real-Time Image-Based 6-DOF Localization in Large-Scale Environments. In *CVPR*, 2012.

[52] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[53] Z. Luo, L. Zhou, X. Bai, H. Chen, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan. Aslfeat: Learning local features of accurate shape and localization. In *CVPR*, 2020.

[54] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems*, 2015.

[55] S. Lynen, B. Zeisl, D. Aiger, M. Bosse, J. Hesch, M. Pollefeys, R. Siegwart, and T. Sattler. Large-scale, real-time visual-inertial localization revisited. *The International Journal of Robotics Research*, 0(0), 2020.

[56] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 36(1):3–15, 2017.

[57] D. Massiceti, A. Krull, E. Brachmann, C. Rother, and P. H. Torr. Random Forests versus Neural Networks - What's Best for Camera Relocalization? In *ICRA*, 2017.

[58] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-dof localization on mobile devices. In *European conference on computer vision*, pages 268–283. Springer, 2014.

[59] H. P. Moravec. The stanford cart and the cmu rover. *Proceedings of the IEEE*, 71(7):872–884, 1983.

[60] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[61] R. Mur-Artal and J. D. Tardós. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

[62] R. Pless. Using Many Cameras as One. In *CVPR*, 2003.

[63] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[64] J. Revaud, J. Almazan, R. S. de Rezende, and C. R. de Souza. Learning with Average Precision: Training Image Retrieval with a Listwise Loss. In *ICCV*, 2019.

[65] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019.

[66] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4938–4947, 2020.

[67] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8601–8610, 2018.

[68] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla. Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization? In *CVPR*, 2017.

[69] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt. Image Retrieval for Image-Based Localization Revisited. In *BMVC*, 2012.

[70] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *CVPR*, 2019.

[71] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425, 2018.

[72] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler. Semantic Visual Localization. In *CVPR*, 2018.

[73] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international Journal of robotics Research*, 21(8):735–758, 2002.

[74] T. Shi, H. Cui, Z. Song, and S. Shen. Dense semantic 3d map based long-term visual localization with hybrid features. *arXiv preprint arXiv:2005.10766*, 2020.

[75] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *CVPR*, 2013.

[76] E. Stenborg, C. Toft, and L. Hammarstrand. Long-Term Visual Localization Using Semantically Segmented Images. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[77] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson. City-Scale Localization for Cameras with Known Vertical Direction. *PAMI*, 39(7):1455–1461, 2017.

[78] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. gdls: A scalable solution to the generalized pose and scale problem. In *European Conference on Computer Vision*, pages 16–31. Springer, 2014.

[79] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. inloc: Indoor visual localization with dense matching and view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[80] H. Taira, I. Rocco, J. Sedlar, M. Okutomi, J. Sivic, T. Pajdla, T. Sattler, and A. Torii. Is This the Right Place? Geometric-Semantic Pose Verification for Indoor Visual Localization. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[81] C. Toft, C. Olsson, and F. Kahl. Long-term 3D Localization and Pose from Semantic Labellings. In *ICCV Workshops*, 2017.

[82] C. Toft, E. Stenborg, L. Hammarstrand, L. Brynte, M. Pollefeys, T. Sattler, and F. Kahl. Semantic Match Consistency for Long-Term Visual Localization. In *The European Conference on Computer Vision (ECCV)*, 2018.

[83] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):257–271, 2018.

[84] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. Global localization from monocular slam on a mobile phone. *IEEE transactions on visualization and computer graphics*, 20(4):531–539, 2014.

[85] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. A minimal solution to the generalized pose-and-scale problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 422–429, 2014.

[86] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-Based Localization Using LSTMs for Structured Feature Correlation. In *ICCV*, 2017.

[87] J. Wald, T. Sattler, S. Golodetz, T. Cavallari, and F. Tombari. Beyond controlled environments: 3d camera re-localization in changing indoor scenes. *arXiv preprint arXiv:2008.02004*, 2020.

[88] L. Yang, Z. Bai, C. Tang, H. Li, Y. Furukawa, and P. Tan. SANet: Scene Agnostic Network for Camera Localization. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

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

[90] Z. Zhang, T. Sattler, and D. Scaramuzza. Reference Pose Generation for Visual Localization via Learned Features and View Synthesis. *arXiv*, 2005.05179, 2020.

[91] L. Zhou, Z. Luo, T. Shen, J. Zhang, M. Zhen, Y. Yao, T. Fang, and L. Quan. Kfnet: Learning temporal camera relocalization using kalman filtering. In *Proc. CVPR*, 2020.