Robust Image Retrieval-based Visual Localization using Kapture

Martin Humenberger Yohann Cabon Nicolas Guerin Julien Morat Jérôme Revaud
Philippe Rerole Noé Pion Cesar de Souza Vincent Leroy
Gabriela Csurka
NAVER LABS Europe, 38240 Meylan, France
https://europe.naverlabs.com
firstname.lastname@naverlabs.com

Abstract

In this paper, we present a versatile method for visual localization. It is based on robust image retrieval for coarse camera pose estimation and robust local features for accurate pose refinement. Our method is top ranked on various public datasets showing its ability of generalization and its great variety of applications. To facilitate experiments, we introduce kapture, a flexible data format and processing pipeline for structure from motion and visual localization that is released open source. We furthermore provide all datasets used in this paper in the kapture format to facilitate research and data processing. Code and datasets can be found at `https://github.com/naver/kapture`*, more information, updates, and news can be found at* `https://europe.naverlabs.com/research/3d-vision/kapture`*.*

1. Introduction

Visual localization The goal of visual localization is to estimate the accurate position and orientation of a camera using its images. In detail, correspondences between a representation of the environment (map) and query images are utilized to estimate the camera pose in 6 degrees of freedom (DOF). The representation of the environment can be a structure from motion (SFM) reconstruction [43, 52, 19, 49, 34], a database of images [56, 53, 41], or even a CNN [24, 28, 7, 48]. Structure-based methods [43, 32, 44, 30, 53, 49] use local features to establish correspondences between 2D query images and 3D reconstructions. These correspondences are then used to compute the camera pose using perspective-n-point (PNP) solvers [25] within a RANSAC loop [17, 10, 29]. To reduce the search range in large 3D reconstructions, image retrieval methods can be used to first retrieve most relevant images from the SFM model. Second, local correspondences are established in the area defined by those images. Scene point regression methods [51, 8] establish the 2D-3D correspondences using a deep neural network (DNN) and absolute pose regression methods [24, 28, 7, 48] directly estimate the camera pose with a DNN. Furthermore, also objects can be used for visual localization, such as proposed in [58, 40, 11, 4].

Challenges Since in visual localization correspondences between the map and the query image need to be established, environmental changes present critical challenges. Such changes could be caused by time of day or season of the year, but also structural changes on house facades or store fronts are possible. Furthermore, the query images can be taken under significantly different viewpoints than the images used to create the map.

Long term visual localization To overcome these challenges, researchers proposed various ways to increase robustness of visual localization methods. Most relevant to our work are data-driven local [33, 15, 16, 38, 13] and global [1, 36, 37] features. Instead of manually describing how keypoints or image descriptions should look like, a large amount of data is used to train an algorithm to make this decision by itself. Recent advances in the field showed great results on tasks like image matching [35] and visual localization [41, 16, 39]. [45] provide an online benchmark which consists of several datasets covering a variety of the mentioned challenges.

In this paper, we present a robust image retrieval-based visual localization method. Extensive evaluations show that it reports top results on various public datasets which highlights its versatile application. We implemented our algorithm using our newly proposed data format and toolbox named *kapture*. The code is open source and all datasets from the website mentioned above are provided in this format.

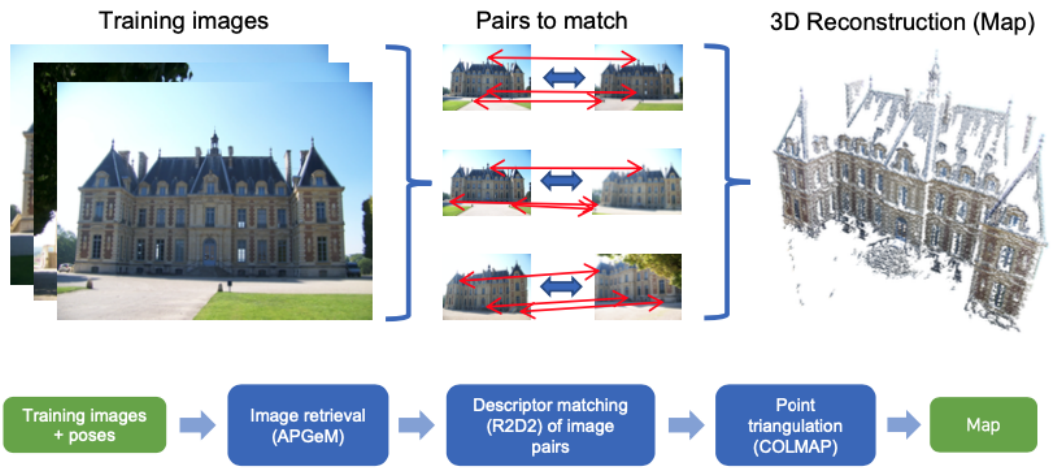

Figure 1. Overview of the structure from motion (SFM) reconstruction of the map from a set of training (mapping) images. Photos: Sceaux Castle image dataset[2]

2. Visual Localization Method

As a reminder, visual localization is the problem of estimating the 6DOF pose of a camera within a known 3D space representation using query images. There are several ways to tackle this problem including structure-based methods [43, 32, 44, 30, 53, 49], pose [24, 28, 7, 48] and scene point regression-based [51, 8] methods or image retrieval-based methods [55, 60, 56]. Our approach follows the workflow of image retrieval as well as structure-based methods and combines functionalities provided by the COLMAP SFM library[3] [49] as well as our local features R2D2 [38] and our global image representation AP-GeM [37]. The method consists of two main components: the SFM-based mapping pipeline (shown in Figure 1) and the localization (image registration) pipeline (shown in Figure 2).

Mapping SFM is one of the most popular strategies for reconstruction of a 3D scene from un-ordered photo collections [52, 19, 49, 34]. The main idea is to establish 2D-2D correspondences between local image features (keypoints) of mapping[4] image pairs, followed by geometric verification to remove outliers. By exploiting transitivity, observations of a keypoint can be found in several images allowing to apply relative pose estimation for initialization of the reconstruction followed by 3D point triangulation [23] and image registration for accurate 6DOF camera pose estimation. RANSAC [17, 10, 29] can be used to increase robustness of several steps in this pipeline and bundle adjustment [57] can be used for global (and local) optimization of the model (3D points and camera poses). Since the camera poses of the training images for all datasets used in this paper are known, our mapping pipeline can skip the camera pose estimation step of SFM. For geometric verification of the matches and triangulation of the 3D points, we use COLMAP. Figure 1 illustrates our mapping workflow.

Localization Similarly to the reconstruction step, 2D-2D local feature correspondences are established between a query image and the database images used to generate the map. In order to only match relevant images, we use image retrieval to obtain the most similar images (e.g. 20 or 50) from the database. Since many keypoints from the database images correspond to 3D points of the map, 2D-3D correspondences between query image and map can be established. These 2D-3D matches are then used to compute the 6DOF camera pose by solving a PNP problem [25, 26, 27] robustly inside a RANSAC loop [17, 10, 29]. We again use COLMAP for geometric verification and image registration.

Local descriptors We can see that both pipelines (mapping and localization) heavily rely on local image descriptors and matches. Early methods used handcrafted local feature extractors, notably the popular SIFT descriptor[5] [31]. However, those keypoint extractors and descriptors have several limitations including the fact that they are not necessary tailored to the target task. Therefore, several data-driven learned representations were proposed recently (see the evolution of local features in [13, 50]).

Our method uses R2D2 [38], which is a sparse keypoint extractor that jointly performs detection and description but separately estimates keypoint reliability and keypoint repeatability. Keypoints with high likelihoods on both aspects are chosen which improves the overall feature match-

[3] https://colmap.github.io

[3] https://github.com/openMVG/ImageDataset_SceauxCastle

[4] Also referred to as training images.

[5] as used in COLMAP

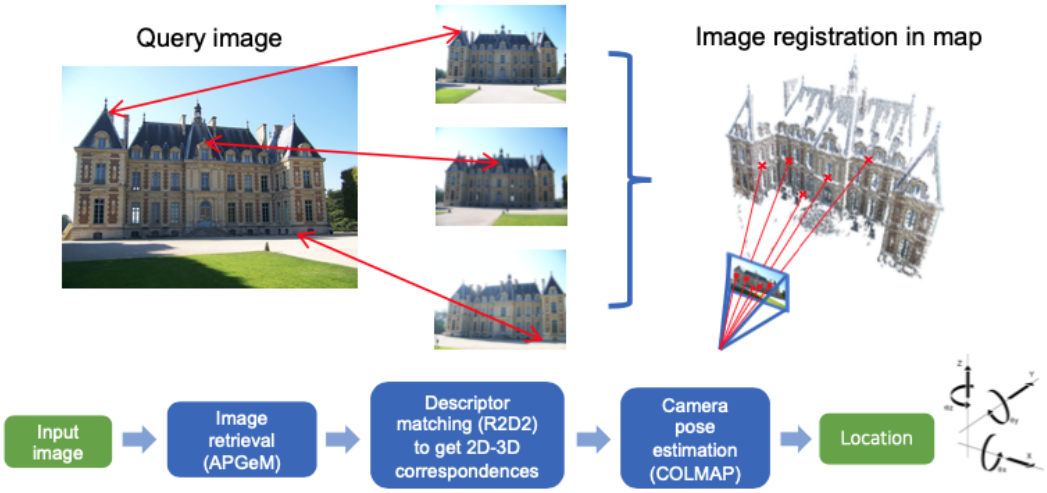

Figure 2. Overview of the localization pipeline which registers query images in the SFM map. Photos: Sceaux Castle image dataset[3]

ing pipeline. R2D2 uses a list-wise loss that directly maximises the average precision to learn reliability. Since a very large amount of image patches (only one is correct) is used per batch, the resulting reliability is well suited for the task of matching. Since reliability and patch descriptor are related, the R2D2 descriptor is extracted from the reliability network. The R2D2 model was trained with synthetic image pairs generated by known transformations (homographies) providing exact pixel matches as well as optical flow data from real image pairs. See Section 4 for details about the model.

Image retrieval In principle, mapping and localization can be done by considering all possible image pairs. However, this approach does not scale well to visual localization in real-world applications where localization might need to be done in large scale environments such as big buildings or even cities. To make visual localization scaleable, image retrieval plays an important role. On the one hand, it makes the mapping more efficient, on the other hand, it increases robustness and efficiency of the localization step [20, 42, 53]. This is achieved in two steps: First, the global descriptors are matched in order to find the most similar images which form image pairs (e.g. reference-reference for mapping and query-reference for localization). Second, these image pairs are used to establish the local keypoint matches.

Localization approaches based on image retrieval typically use retrieval representations designed for geo-localization [1, 54, 2]. However, our initial experiments have not shown superiority of these features compared to our off-the-shelf deep visual representations Resnet101-AP-GeM [37]. Note that our model was trained for the landmark retrieval task on the Google Landmarks (GLD) dataset [33]. The model considers a generalized mean-pooling (GeM) layer [36] to aggregate the feature maps into a compact, fixed-length representation which is learned by directly optimizing the mean average precision (mAP). Section 5 contains more details about AP-GeM as well as comparisons of various global image representations and combinations of them.

3. Kapture

3.1. Kapture format and toolbox

When running a visual localization pipeline on several datasets, one of the operational difficulties is to convert those datasets into a format that the algorithm and all the tools used can handle. Many formats already exist, notably the ones from Bundler[6], VisualSFM[7], OpenMVG[8], OpenSfM[9], and COLMAP[10], but none meets all our requirements. In particular we need a format that can handle timestamps, shared camera parameters, multi-camera rigs, but also reconstruction data (keypoints, descriptors, global features, 3D points, matches...) and that would be flexible and easy to use for localization experiments. Furthermore, it should be easy to convert data into other formats supported by major open source projects such as OpenMVG and COLMAP.

Inspired by the mentioned open source libraries, kapture started as pure data format that provids a good representation of all the information we needed. It then grew into a Python toolbox and library for data manipulation (conversion between various popular formats, dataset merging/s-

[6] https://www.cs.cornell.edu/~snavely/bundler/bundler-v0.4-manual.html#S6
[7] http://ccwu.me/vsfm/doc.html#nvm
[8] https://openmvg.readthedocs.io/en/latest/software/SfM/SfM_OutputFormat/
[9] https://www.opensfm.org/docs/dataset.html/
[10] https://colmap.github.io/format.html

plitting, trajectory visualization, etc.), and finally it became the basis for our mapping and localization pipeline. More precisely, the kapture format can be used to store sensor data: images, camera parameters, camera rigs, trajectories, but also other sensor data like lidar or wifi scans. It can also be used to store reconstruction data, in particular local descriptors, keypoints, global features, 3D points, observations, and matches.

We believe that the kapture format and tools are useful for the community, so we release them as open-source at `https://github.com/naver/kapture`. We also provide major public datasets of the domain in this format to facilitate future experiments for everybody.

3.2. Kapture pipeline

We implemented our visual localization method, described in Section 2, on top of the kapture tools and libraries. In particular, the mapping pipeline consists of the following steps:

1. Extraction of local descriptors and keypoints (e.g. R2D2) of training images
2. Extraction of global features (e.g. AP-GeM) of training images
3. Computation of training image pairs using image retrieval
4. Computation of local descriptor matches between these image pairs
5. Geometric verification of the matches and point triangulation with COLMAP

The localization steps are similar:

1. Extraction of local and global features of query images
2. Retrieval of similar images from the training images
3. Local descriptor matching
4. Geometric verification of the matches and camera pose estimation with COLMAP

4. Evaluation

For evaluation of our method, we use the datasets provided by the online visual localization benchmark[11] [45]. Each of these datasets is split into a training (mapping) and a test set. The training data, which consists of images, corresponding poses in the world frame as well as intrinsic camera parameters, is used to construct the map, the test data is used to evaluate the precision of the localization method. Intrinsic parameters of the test images are not always provided.

We converted all datasets to kapture and we used the publicly available models for R2D2[12] and AP-GeM[13] for all datasets and evaluations. If not indicated differently, we used the top 20k keypoints extracted with R2D2.

Parameters We experimented with three COLMAP parameter settings which are presented in Table 1. For map generation we always used *config1*.

Metrics All datasets used are divided into different conditions. These conditions could be different times of day, differences in weather such as snow, or even different buildings or locations within the dataset. In order to report localization results, we used the online benchmark[11] which computes the percentage of query images which where localized within three pairs of translation and rotation thresholds.

4.1. Aachen Day-Night

The Aachen Day-Night dataset [45, 47] represents an outdoor handheld camera localization scenario where all query images are taken individually with large changes in viewpoint and scale, but also between daytime and nighttime. In detail, the query images are divided into the classes *day* and *night* and the two classes are evaluated separately. We evaluated our method in two settings: (i) we used the *full* dataset to construct a single map using the provided reference poses and localized all query images within this map, and (ii) we used the *pairs*[14] provided for the local features evaluation task on the online benchmark[11], which cover nighttime images only. Recently, an updated version Aachen Day-Night v.1.1 [61], which contains more training images and more accurate poses of the query images (not public), was released. Table 2 presents the results for both versions of the dataset.

4.2. InLoc

InLoc [53, 59] is a large indoor dataset for visual localization. It also represents a handheld camera scenario with large viewpoint changes, occlusions, people and even changes in furniture. Contrary to the other datasets, InLoc also provides 3D scan data, i.e. 3D point clouds for each training image. However, since the overlap between the training images is quite small, the resulting structure from motion models are sparse and, according to our experience, not suitable for visual localization. Furthermore, the InLoc

[11] `http://visuallocalization.net`

[12] r2d2_WASF_N8_big from `https://github.com/naver/r2d2`

[13] Resnet101-AP-GeM-LM18 from `https://github.com/almazan/deep-image-retrieval`

[14] `https://github.com/tsattler/visuallocalizationbenchmark`

Table 1. Parameter configurations.

COLMAP image_registrator	config1	config2	config3
−−Mapper. ba_refine_focal_length	0	0	1
−−Mapper. ba_refine_principal_point	0	0	0
−−Mapper.ba_refine_extra_params	0	0	1
−−Mapper.min_num_matches	15	4	4
−−Mapper.init_min_num_inliers	100	4	4
−−Mapper.abs_pose_min_num_inliers	30	4	4
−−Mapper. abs_pose_min_inlier_ratio	0.25	0.05	0.05
−−Mapper.ba_local_max_num_iterations	25	50	50
−−Mapper.abs_pose_max_error	12	20	20
−−Mapper. filter_max_reproj_error	4	12	12

Table 2. Results on Aachen Day-Night. In *pairs* we used the top 40k R2D2 keypoints. Day: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°), Night: (0.5m, 2°) / (1m, 5°) / (5m, 10°)

v1 setting	day	night
full (config2)	88.7 / 95.8 / 98.8	81.6 / 88.8 / 96.9
pairs (config1, R2D2 40k)	-	76.5 / 90.8 / 100.0
v1.1 setting	**day**	**night**
full (config2)	90.0 / 96.2 / 99.5	72.3 / 86.4 / 97.9
pairs (config1, R2D2 40k)	-	71.2 / 86.9 / 97.9

Table 3. Results on InLoc using different 3D point distance thresholds for mapping. (0.25m, 10°) / (0.5m, 10°) / (5m, 10°)

setting	DUC1	DUC2
config2, 0.5mm	36.4 / 52.0 / 64.1	30.5 / 53.4 / 58.0
config2, 0.1mm	36.9 / 53.0 / 65.7	34.4 / 52.7 / 59.5

environment is very challenging for global and local features because it contains large textureless areas and many repetitive, non-unique areas. To overcome these problems, the original InLoc localization method [53] introduced various dense matching and pose verification methods which make use of the provided 3D data.

Mapping We constructed our SFM map using the provided 3D data and the camera poses, which differs from the mapping described in Section 2. We first assign a 3D point to each local feature in the training images. Second, we generate matches based on 3D points. In detail, we look for local features which are the projection of the same 3D point in different images. To decide whether or not a 3D point is the same for different keypoints, we use an Euclidean distance threshold (0.5mm and 0.1mm). This results in a very dense 3D map (Figure 3) where each 3D point is associated with a local descriptor and can, thus, be used in our method.

Localization We ran the localization pipeline (Figure 2) for all provided query images. Table 3 presents the results.

4.3. RobotCar Seasons

RobotCar Seasons [45] is an outdoor dataset captured in the city of Oxford at various periods of a year and in different conditions (rain, night, dusk, etc.). The images are taken from a car with a synchronized three-camera rig pointing in three directions (rear, left and right). The data was captured at 49 different non-overlapping locations and several 3D models are provided. Training images were captured with a reference condition (overcast-reference), while test images were captured in different conditions. For each test image, the dataset provides its condition, the location where it was captured (one of the 49 location used in the training data), its timestamp, and the camera name.

Mapping Since the different locations are not overlapping, there is no benefit in building a single map. For our experiments, we used the individual models for each 49 locations that are provided in the COLMAP format. We converted the COLMAP files into the kapture format to recover trajectories (poses and timestamps) and created 49 individual maps using our mapping pipeline (Figure 1). For this step, we used the provided camera parameters (pinhole model) and considered each camera independently without using the rig information.

Localization Since the location within the dataset is given for each query image, we can directly use it during localization. Otherwise, we would have first selected the correct map, e.g. by using image retrieval. We tested both, COLMAP config1 and config2.

For the images that could not be localized we ran two additional steps. First, we leverage the fact that images are captured synchronously with a rig of three cameras for which the calibration parameters are provided. Hence, if one image taken at a specific timestamp is localized, using the provided extrinsic camera parameters we can compute

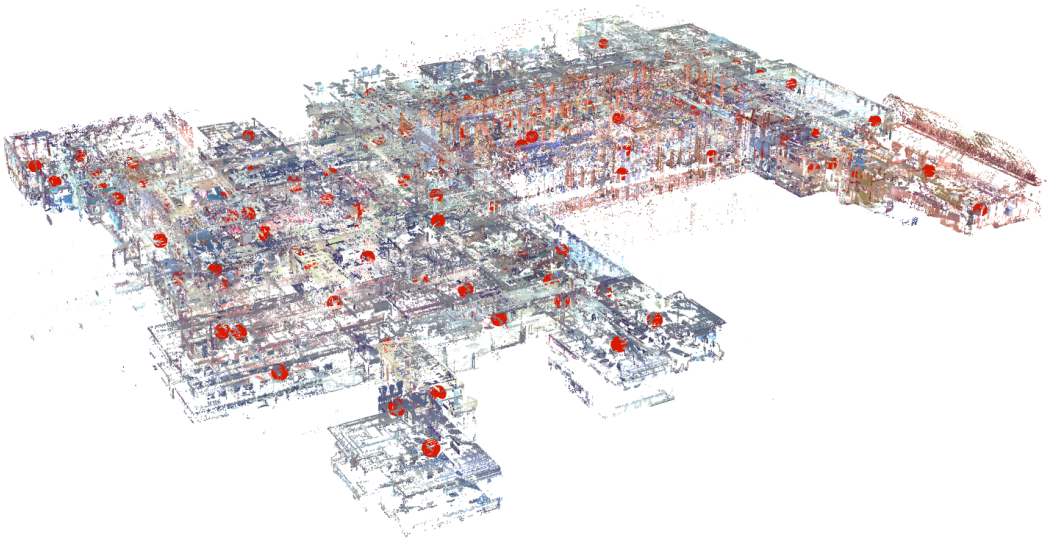

Figure 3. InLoc map generated by assigning a 3D point to each R2D2 feature in the training images (viewed in COLMAP).

the pose for all images of the rig (even if they were not successfully localized). We used this technique to find the missing poses for all images for which this can be applied.

However, there are still timestamps for which no pose was found for any of the three cameras. In this case, we leverage the fact that query images are given in sequences (e.g. 6 to 12 images in most cases). Sequences can be found using image timestamps. When the gap between two successive timestamps is too large (i.e. above a certain threshold), we start a new sequence. Once the sequences are defined, we look for non-localized image triplets in these sequences and estimate their poses by linear interpolation between the two closest successfully localized images. If this is not possible, we use the closest available pose. Note that for real-world applications, we could either only consider images of the past or introduce a small latency if images from both directions (before and after) are used. These steps increase the percentage of localized images to 97.2%. Table 4 presents the results of the configurations tested. Interestingly, even if config2 could localize all images and config1 only 90%, applying the rig and sequence information on config1 led to overall better results.

Table 4. Results on RobotCar Seasons. Thresholds: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°)

setting	day	night
config2	55.2 / 82.0 / 97.1	28.1 / 59.0 / 82.7
config1	55.1 / 82.1 / 96.9	26.9 / 55.6 / 78.4
config1 + rig	55.1 / 82.1 / 97.2	28.7 / 58.3 / 83.4
config1 + rig + seq	55.1 / 82.1 / 97.3	28.8 / 58.8 / 89.4

4.4. Extended CMU-Seasons

The Extended CMU-Seasons dataset [45, 5] is an autonomous driving dataset that contains sequences from urban, suburban, and park environments. The images were recorded in the area of Pittsburgh, USA over a period of one year and thus contain different conditions (foliage/mixed-foliage/no foliage, overcast, sunny, low sun, cloudy, snow). The training and query images were captured by two front-facing cameras mounted on a car, pointing to the left and right of the vehicle at approximately 45 degrees with respect to the longitudinal axis. The cameras are not synchronized. This dataset is also split into multiple locations. Unlike RobotCar Seasons, there is some overlap between the locations. However, we did not leverage this in our experiments.

Mapping For our experiments, we used the individual models for each location. We converted the ground-truth-database-images-sliceX.txt files into the kapture format to recover trajectories (poses and timestamps). We then created 14 individual maps (the slices that were provided with queries 2-6/13-21) using the pipeline described above. For this step, we used the provided camera parameters (OpenCV[15] pinhole camera), and considered each camera independently, i.e. without using the rig information.

Localization We ran the localization pipeline described above on all images listed in the test-images-sliceX.txt files with config1. We then ran two post-processing steps: rig and sequence. For rig, we first estimated a rig configuration from the slice2 training poses. For all images that

[15] https://opencv.org

failed to localize, we computed the position using this rig if the image from the other camera with the closest timestamp was successfully localized. Finally, we applied the same sequence post-processing as we did for RobotCar Seasons (see Section 4.3). Table 5 presents the results on this dataset and the improvements we get from each of the post-processing steps.

Table 5. Results on Extended CMU-Seasons. All conditions: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°)

setting	urban	suburban	park
config2	95.9 / 98.1 / 98.9	89.5 / 92.1 / 95.2	78.3 / 82.0 / 86.4
config1	95.8 / 98.1 / 98.8	88.9 / 91.1 / 93.4	75.5 / 78.4 / 82.0
config1 + rig	96.5 / 98.8 / 99.5	94.3 / 96.7 / 99.1	83.1 / 87.9 / 92.8
config1 + rig + seq	96.7 / 98.9 / 99.7	94.4 / 96.8 / 99.2	83.6 / 89.0 / 95.5

4.5. SILDa Weather and Time of Day

SILDa Weather and Time of Day [6] is an outdoor dataset captured over a period of 12 months (clear, snow, rain, noon, dusk, night) which covers 1.2km of streets around Imperial College in London. It was captured using a camera rig composed of two back-to-back wide-angle fisheye lenses. The geometry of the rig as well as the hardware synchronization of the acquisition could be leveraged, e.g. to reconstruct spherical images.

Mapping The dataset provides camera parameters corresponding to a fisheye model that is not available in COLMAP. For the sake of simplicity, we chose to estimate the parameters of both cameras using a camera model supported by COLMAP, namely the FOV model (we still use the provided estimation of the principal point).

Localization Similarly to the RobotCar Seasons dataset, we applied the image sequences and camera rig configuration to estimate camera poses of images which could not be localized. As the rig geometry is not given for SILDa, we estimated an approximation. Table 6 presents the results of the configurations used. As can be seen, leveraging the sequence did not improve the results.

Table 6. Results on SILDa. Thresholds: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°)

setting	evening	snow	night
config1	31.8 / 66.3 / 89.4	0.3 / 3.9 / 64.9	30.0 / 53.4 / 77.5
config1 + rig	31.9 / 66.6 / 92.5	0.5 / 5.8 / 89.2	30.5 / 54.2 / 78.5
config1 + rig + seq	31.9 / 66.6 / 92.5	0.5 / 5.8 / 89.2	30.5 / 54.2 / 78.5

5. Experiments with late fusion of multiple image representations

The motivation of these experiments is finding a late fusion strategy where the combination of different global descriptors outperforms each descriptor individually. To do so, we replaced the global descriptor AP-GeM, we used for image retrieval described in Section 3.2, with combinations of 4 different global descriptors:

DenseVLAD [54] To obtain the DenseVLAD representation for an image, first RootSIFT [3] descriptors are extracted on a multi-scale, regular, densely sampled grid, and then, aggregated into an intra-normalized VLAD [22] descriptor followed by PCA compression, whitening, and L2 normalization [21]. DenseVLAD is often used in structure-based visual localization methods to scale them up to large scenes [54, 46].

NetVLAD [1] The main component of the NetVLAD architecture is a generalized VLAD layer that aggregates mid-level convolutional features extracted from the entire image into a compact single vector representation for efficient indexing similarly to VLAD [22]. The resulting aggregated representation is then compressed using PCA to obtain a final compact descriptor for the image. NetVLAD is trained with geo-tagged image sets consisting of groups of images taken from the same locations at different times and seasons, allowing the network to discover which features are useful or distracting and what changes should the image representation be robust to. This makes NetVLAD very interesting for the visual localization pipeline. Furthermore, NetVLAD has already been used in state-of-the-art localization pipelines [41, 18] and in combination with D2-Net [16].

AP-GeM [37] This model, similarly to [36], uses a generalized-mean pooling layer (GeM) to aggregate CNN-based descriptors of several image regions at different scales. Instead of contrastive loss, it directly optimizes the Average Precision (AP) approximated by histogram binning to make it differentiable. It is currently one of the best performing methods of image representation on popular landmark retrieval benchmarks such as $\mathcal{R}$Oxford and $\mathcal{R}$Paris [35].

DELG [9] DELG is a 2-in-1 local and global features extraction CNN. After a common backbone, the model is split into two parts (heads), one to detect relevant local features and one to describe the global content of the image as a compact descriptor. The two networks are jointly trained on Google Landmark v1 [33] in an end-to-end manner using the ArcFace [14] loss for the compact descriptor. The method is originally designed for image search where the local features enable geometric verification and re-ranking.

5.1. Fusion

Late fusion means that we first compute the similarities for each descriptor individually and then apply a fusion operator, such as:

Generalized harmonic mean (GHarm) [12] GHarm is a generalization of the weighted harmonic mean (WHarm) [16]. It can be obtained using the generalized f-mean:

$$M_f(x_1, ..., x_n) = f^{-1}(\sum_{i=1}^{n} \frac{f(x_i)}{n}) \quad (1)$$

with $f = \frac{1}{\gamma + x}, x_i = \alpha_i sim_i, \sum_{i=1}^{n} \alpha_i = 1$.

round_robin[17] Most similar images are picked from each individual descriptor in equal portions and in circular order.

mean_and_power (WMP) [12]

$$M_{WMP} = \gamma \cdot \sum_{i=1}^{n} \alpha_i sim_i + (1 - \gamma) \cdot \prod_{i=1}^{n} sim_i^{\alpha_i} \quad (2)$$

min_and_max [12]

$$M_{Min\&Max} = (1 - \alpha) \max_{i=1}^{n}(sim_i) + \alpha \min_{i=1}^{n}(sim_i) \quad (3)$$

min [12]

$$M_{Min} = \min_{i=1}^{n}(sim_i) \quad (4)$$

max [12]

$$M_{Max} = \max_{i=1}^{n}(sim_i) \quad (5)$$

5.2. Parameters

For AP-GeM[18], we used the Resnet101-AP-GeM model trained on Google Landmarks v1 [33]. For DELG[19], the model is also trained on Google Landmarks v1. [33]. For NetVLAD[20], we used the VGG-16-based NetVLAD model trained on Pitts30k. DenseVLAD is available at `http://www.ok.ctrl.titech.ac.jp/~torii/project/247/`. We used GHarm [12] with $\alpha_i = \frac{1}{n}$ $\gamma = 0.5$ and for all other operators we used equal weights.

[16] `https://en.wikipedia.org/wiki/Harmonic_mean`

[17] `https://en.wikipedia.org/wiki/Round-robin_scheduling`

[18] AP-GeM code at `https://github.com/almazan/deep-image-retrieval`

[19] DELG code at `https://github.com/tensorflow/models/tree/master/research/delf/delf/python/delg`

[20] NetVLAD code at `https://github.com/Relja/netvlad`

5.3. Aachen Day-Night v1.1

As we already did for the Aachen Day-Night experiments in Section 4.1, we evaluated our method with two settings: (i) we used the *full* dataset to construct a single map using the provided reference poses and localized all query images within this map, and (ii) we used the *pairs*[21] provided for the local features evaluation task on the online benchmark[11], which cover nighttime images only. Table 7 presents the results.

Table 7. Results on Aachen Day-Night v1.1. In all experiments but *pairs* and *gharm top50*, we used the top 20k R2D2 keypoints. All conditions: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°)

setting	day	night
pairs (r2d2 40k) config1	-	71.2 / 86.9 / 97.9
gharm top50 (r2d2 40k) config2	**90.9** / 96.7 / **99.5**	**78.5** / **91.1** / 98
gharm top20 config2	90.5 / **96.8** / 99.4	74.9 / 90.1 / **98.4**
AP-GeM-LM18 top20 config2	89.9 / 96.5 / **99.5**	71.2 / 86.9 / 97.9
DELG top20 config2	90.0 / 96.0 / 99.2	73.3 / 87.4 / 97.4
netvlad_vd16pitts top20 config2	88.7 / 95.1 / 98.1	74.3 / 89.5 / 97.9
densevlad top20 config2	88.2 / 94.2 / 97.3	62.8 / 79.1 / 88.0
round_robin top20 config2	89.6 / 96.6 / 99.4	72.8 / 87.4 / **98.4**
mean_and_power top20 config2	90.7 / 96.8 / 99.3	73.3 / 89.5 / **98.4**
min_and_max top20 config2	89.9 / 96.4 / 99.2	73.3 / 88.0 / 97.4
min top20 config2	89.1 / 95.0 / 98.2	71.2 / 84.8 / 92.1
max top20 config2	89.2 / 96.0 / 99.0	73.8 / 88.0 / 97.9

5.4. InLoc

We ran the localization pipeline described in Section 4.2 for all provided query images. Table 8 presents the results.

Table 8. Results on InLoc using 0.1mm 3D point distance threshold for mapping. (0.25m, 10°) / (0.5m, 10°) / (5m, 10°). In all experiments we used the top 40k R2D2 keypoints.

setting	DUC1	DUC2
gharm top50 config2	**41.4** / **60.1** / **73.7**	47.3 / **67.2** / **73.3**
AP-GeM-LM18 top50 config2	37.4 / 55.6 / 70.2	36.6 / 51.9 / 61.1
DELG top50 config2	38.4 / 56.1 / 71.7	37.4 / 59.5 / 67.9
netvlad_vd16pitts top50 config2	36.9 / 58.1 / 70.2	38.2 / 62.6 / 70.2
densevlad top50 config2	33.8 / 51.5 / 67.7	45.0 / 65.6 / 72.5
round_robin top50 config2	33.8 / 51.5 / 66.2	29.8 / 48.1 / 53.4
mean_and_power top50 config2	39.9 / 57.6 / 71.7	**48.9** / **67.2** / **73.3**
min_and_max top50 config2	37.9 / 56.6 / 70.7	44.3 / 62.6 / 69.5
min top50 config2	32.8 / 54.5 / 67.7	42.7 / 65.6 / 71.8
max top50 config2	39.9 / 55.1 / 69.2	38.9 / 58.0 / 64.1

5.5. RobotCar Seasons v2

For this experiment, we used the newly released v2 of the RobotCar Seasons dataset. The new version is based on the same data, but uses a different split for training and testing. In addition, the training data (images and poses) of v2 contains images from various conditions (contrary to the orig-

[21] `https://github.com/tsattler/visuallocalizationbenchmark`

inal dataset where only condition overcast-reference was available). We created 22 individual maps (the locations that were provided with queries 4-6/23-36,44,45,47/49) using the pipeline described above. Table 9 presents the results.

Table 9. Results on RobotCar Seasons v2. Thresholds: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°). In all experiments we used the top 20k R2D2 keypoints.

setting	day	night
gharm top20 config1 + rig + seq	**66.0 / 95.1 / 100.0**	46.2 / 76.5 / 91.4
AP-GeM-LM18 top20 config1 + rig + seq	65.7 / **95.1 / 100.0**	43.6 / 76.7 / 93.9
DELG top20 config1 + rig + seq	65.6 / 94.6 / 99.6	37.8 / 64.6 / 78.8
netvlad_vd16pitts top20 config1 + rig + seq	65.6 / **95.1 / 100.0**	35.7 / 70.4 / 90.9
densevlad top20 config1 + rig + seq	65.7 / **95.1 / 100.0**	41.3 / 74.1 / 92.8
round_robin top20 config1 + rig + seq	65.9 / **95.1 / 100.0**	42.4 / 75.1 / **94.2**
mean_and_power top20 config1 + rig + seq	65.9 / **95.1 / 100.0**	**46.4 / 79.7** / 92.5
min_and_max top20 config1 + rig + seq	65.7 / **95.1 / 100.0**	41.3 / 72.5 / 86.2
min top20 config1 + rig + seq	65.8 / **95.1 / 100.0**	40.8 / 72.0 / 91.6
max top20 config1 + rig + seq	65.6 / 94.7 / 99.7	36.6 / 63.2 / 80.0

5.6. Extended CMU-Seasons

We created 14 individual maps (the slices that were provided with queries 2-6/13-21) using the pipeline described above. Table 10 presents the results on this dataset.

Table 10. Results on Extended CMU-Seasons. All conditions: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°). In all experiments we used the top 20k R2D2 keypoints.

setting	urban	suburban	park
gharm top20 config1 + rig + seq	97.0 / 99.1 / **99.8**	**95.0** / 97.0 / 99.4	**89.2 / 93.4** / 97.5
AP-GeM-LM18 top20 config1 + rig + seq	96.7 / 98.9 / 99.7	94.4 / 96.8 / 99.2	83.6 / 89.0 / 95.5
DELG top20 config1 + rig + seq	96.6 / 98.8 / 99.7	94.1 / 96.7 / 99.1	84.7 / 89.6 / 95.7
netvlad_vd16pitts top20 config1 + rig + seq	**97.1** / 99.1 / **99.8**	93.8 / 96.3 / 99.1	88.1 / 92.7 / 97.5
densevlad top20 config1 + rig + seq	96.1 / 98.4 / 99.4	94.2 / 96.7 / 99.1	87.9 / 92.3 / 96.8
round_robin top20 config1 + rig + seq	96.9 / 99.1 / 99.7	94.8 / 97.0 / 99.4	88.8 / 93.1 / **97.6**
mean_and_power top20 config1 + rig + seq	97.0 / **99.2** / 99.7	94.7 / 97.0 / **99.5**	**89.2 / 93.4** / 97.5
min_and_max top20 config1 + rig + seq	96.7 / 98.9 / 99.6	**95.0 / 97.1 / 99.5**	88.6 / 93.2 / **97.6**
min top20 config1 + rig + seq	96.3 / 98.5 / 99.4	94.2 / 96.6 / 99.2	88.0 / 92.4 / 97.3
max top20 config1 + rig + seq	96.8 / 98.9 / 99.7	94.4 / 97.0 / 99.3	84.1 / 89.2 / 96.0

5.7. SILDa Weather and Time of Day

Table 11 presents the results of the configurations used.

Table 11. Results on SILDa. Thresholds: (0.25m, 2°) / (0.5m, 5°) / (5m, 10°). In all experiments we used the top 20k R2D2 keypoints

setting	evening	snow	night
gharm top20 config1 + rig + seq	**32.4** / 67.4 / 93.3	0.2 / 4.1 / 88.9	30.4 / 54.2 / 81.1
AP-GeM-LM18 top20 config1 + rig + seq	31.9 / 66.6 / 92.5	**0.5** / 5.8 / 89.2	**30.5** / 54.2 / 78.5
DELG top20 config1 + rig + seq	31.3 / 66.4 / 92.1	0.2 / **7.5** / 85.6	30.2 / 54.1 / 77.4
netvlad_vd16pitts top20 config1 + rig + seq	31.6 / 66.5 / 91.0	0.0 / 2.7 / **89.6**	28.9 / 52.1 / 78.5
densevlad top20 config1 + rig + seq	27.9 / 59.7 / 75.6	0.0 / 1.9 / 59.1	29.3 / 52.9 / 79.7
round_robin top20 config1 + rig + seq	31.8 / 66.9 / 92.0	0.0 / 3.4 / **89.6**	29.3 / 54.0 / 79.5
mean_and_power top20 config1 + rig + seq	31.7 / 67.0 / **93.5**	0.0 / 2.7 / 88.5	30.4 / **54.4** / 80.7
min_and_max top20 config1 + rig + seq	32.1 / **67.6** / 93.4	0.2 / 3.3 / 88.7	**30.5** / 54.0 / 80.2
min top20 config1 + rig + seq	30.1 / 63.9 / 82.5	0.0 / 1.9 / 76.2	28.1 / 54.1 / **81.4**
max top20 config1 + rig + seq	31.8 / 65.9 / 92.4	0.3 / 2.7 / 84.6	30.4 / 54.1 / 77.8

5.8. Discussion

In summary, the improvement over the individual features, especially AP-GeM, is not too large. The reason is that AP-GeM already performs quite well and it is sufficient for our localization pipeline to retrieve just a few good images. However, we observe a consistent improvement for all of our datasets using the GHarm operator. As expected, the improvement is more significant for cases where the performance with individual descriptors is lower, such as nighttime or indoor images.

6. Conclusion and Future Work

We presented a versatile method for visual localization based on robust global features for coarse localization using image retrieval and robust local features for accurate pose computation. We evaluated our method on multiple datasets covering a large variety of application scenarios and challenging situations. Our method ranks among the best methods on the online visual localization benchmark[11]. We implemented our method in Python and ran the experiments using kapture, a unified SFM and localization data format which we released open source. Since all datasets are available in this format, we hope to facilitate future large scale visual localization and structure from motion experiments using a multitude of datasets. Finally, we showed that late fusion of global image descriptors is a promising direction to improve our method.

References

[1] Relja Arandjelović, Petr Gronát, Akihiko Torii, Tomáš Pajdla, and Josef Sivic. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In *CVPR*, 2016.

[2] Relja Arandjelović, Josef Sivic, Masatoshi Okutomi, and Tomáš Pajdla. Dislocation: Scalable Descriptor Distinctiveness for Location Recognition. In *ACCV*, 2014.

[3] Relja Arandjelović and Andrew Zisserman. Three Things Everyone Should Know to Improve Object Retrieval. In *CVPR*, 2012.

[4] Shervin Ardeshir, Amir Roshan Zamir, Alejandro Torroella, and Mubarak Shah. Gis-assisted object detection and geospatial localization. In *European Conference on Computer Vision*, pages 602–617. Springer, 2014.

[5] Hernan Badino, Daniel Huber, and Takeo Kanade. The CMU Visual Localization Data Set. `http://3dvis.ri.cmu.edu/data-sets/localization`, 2011.

[6] Vasileios Balntas, Duncan Frost, Rigas Kouskouridas, Axel Barroso-Laguna, Arjang Talattof, Huub Heijnen, and Krystian Mikolajczyk. Silda: Scape imperial localisation dataset. `https://www.visuallocalization.net/`, 2019.

[7] Vassileios Balntas, Shuda Li, and Victor Prisacariu. RelocNet: Continuous Metric Learning Relocalisation Using Neural Nets. In *ECCV*, 2018.

[8] Eric Brachmann and Carsten Rother. Learning Less Is More - 6D Camera Localization via 3D Surface Regression. In *CVPR*, 2018.

[9] Bingyi Cao, André Araujo, and Jack Sim. Unifying Deep Local and Global Features for Efficient Image Search. *arXiv*, 2001.05027, 2020.

[10] Ondřej Chum and Jiří Matas. Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1472–1482, 2008.

[11] Andrea Cohen, Johannes L. Schönberger, Pablo Speciale, Torsten Sattler, Jan-Michael Frahm, and Marc Pollefeys. Indoor-outdoor 3d reconstruction alignment. In *European Conference on Computer Vision*, pages 285–300. Springer, 2016.

[12] Gabriela Csurka and Stephane Clinchant. An empirical study of fusion operators for multimodal image retrieval. In *2012 10th International Workshop on Content-Based Multimedia Indexing (CBMI)*.

[13] Gabriela Csurka, Christopher R. Dance, and Martin Humenberger. From Handcrafted to Deep Local Invariant Features. *arXiv*, 1807.10254, 2018.

[14] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. 01 2018.

[15] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised Interest Point Detection and Description. In *CVPR*, 2018.

[16] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: a Trainable CNN for Joint Description and Detection of Local Features. In *CVPR*, 2019.

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

[18] Hugo Germain, Guillaume Bourmaud, and Vincent Lepetit. Sparse-to-Dense Hypercolumn Matching for Long-Term Visual Localization. In *International Conference on 3D Vision (3DV)*, 2019.

[19] J. Heinly, J. L. Schönberger, E. Dunn, and J. M. Frahm. Reconstructing the world* in six days. In *CVPR*, 2015.

[20] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From Structure-from-Motion Point Clouds to Fast Location Recognition. In *CVPR*, 2009.

[21] Hervé Jégou and Ondřej Chum. Negative Evidences and Co-occurrences in Image Retrieval: the Benefit of PCA and Whitening. In *ECCV*, 2012.

[22] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating Local Descriptors Into a Compact Image Representation. In *CVPR*, 2010.

[23] Lai Kang, Lingda Wu, , and Yee-Hong Yang. Robust multiview L2 triangulation via optimal inlier selection and 3D structure refinement. *PR*, 47(9):2974–2992, 2014.

[24] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: a Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*.

[25] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. A Novel Parametrization of the Perspective-three-point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *CVPR*, 2011.

[26] 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.

[27] Viktor Larsson, Zuzana Kukelova, and Yinqiang Zheng. Making Minimal Solvers for Absolute Pose Estimation Compact and Robust. In *ICCV*, 2017.

[28] Zakaria Laskar, Iaroslav Melekhov, Surya Kalia, and Juho Kannala. Camera Relocalization by Computing Pairwise Relative Poses Using Convolutional Neural Network. 2017.

[29] Karel Lebeda, Juan E. Sala Matas, and Ondřej Chum. Fixing the Locally Optimized RANSAC. In *BMVC*, 2012.

[30] Liu Liu, Hongdong Li, and Yuchao Dai. Efficient Global 2D-3D Matching for Camera Localization in a Large-Scale 3D Map. In *ICCV*, 2017.

[31] David G. Lowe. Distinctive Image Features from Scale-invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[32] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion. In *ICCV*, 2013.

[33] Hyeonwoo Noh, André Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-Scale Image Retrieval with Attentive Deep Local Features. In *ICCV*, 2017.

[34] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305364, 2017.

[35] Filip Radenović, Asmet Iscen, Giorgos Tolias, and Ondřej Avrithis, Yannis Chum. Revisiting Oxford and Paris: Large-scale Image Retrieval Benchmarking. In *CVPR*, 2018.

[36] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-Tuning CNN Image Retrieval with no Human Annotation. *PAMI*, 41(7):1655–1668, 2019.

[37] Jérôme Revaud, Jon Almazan, Rafael Sampaio de Rezende, and Cesar Roberto de Souza. Learning with Average Precision: Training Image Retrieval with a Listwise Loss. In *ICCV*, 2019.

[38] Jérôme Revaud, Philippe Weinzaepfel, César De Souza, and Martin Humenberger. R2D2: Reliable and Repeatable Detectors and Descriptors. In *NeurIPS*, 2019.

[39] Jérôme Revaud, Philippe Weinzaepfel, César De Souza, Noé Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2D2: Reliable and Repeatable Detectors and Descriptors for Joint Sparse Keypoint Detection and Local Feature Extraction. *CoRR*, (arXiv:1906.06195), 2019.

[40] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013.

[41] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From Coarse to Fine: Robust Hierarchical Localization at Large Scale. In *CVPR*, 2019.

[42] Torsten Sattler, Michal Havlena, Filip Radenovic, Konrad Schindler, and Marc Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition. In *ICCV*, 2015.

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

[44] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization. *PAMI*, 39(9):1744–1756, 2017.

[45] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6DoF Outdoor Visual Localization in Changing Conditions. In *CVPR*, 2018.

[46] Torsten Sattler, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi, and Tomas Pajdla. Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization? In *CVPR*, 2017.

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

[48] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixé. Understanding the Limitations of CNN-based Absolute Camera Pose Regression. In *CVPR*, 2019.

[49] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion Revisited. In *CVPR*, 2016.

[50] Johannes L. Schönberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative Evaluation of Hand-Crafted and Learned Local Features. In *CVPR*, 2017.

[51] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *CVPR*, 2013.

[52] N. Snavely, S.M. Seitz, and R. Szeliski. Modeling the World from Internet Photo Collections. *IJCV*, 80(2):189–210, 2008.

[53] 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. *PAMI*, pages 1–1, 2019.

[54] Akihiko Torii, Relja Arandjelović, Josef Sivic, Masatoshi Okutomi, and Tomáš Pajdla. 24/7 Place Recognition by View Synthesis. In *CVPR*, 2015.

[55] Akihiko Torii, Josef Sivic, and Tomáš Pajdla. Visual Localization by Linear Combination of Image Descriptors. In *ICCV-W*, 2011.

[56] A. Torii, H. Taira, J. Sivic, M. Pollefeys, M. Okutomi, T. Pajdla, and T. Sattler. Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization? *PAMI*, pages 1–1, 2019.

[57] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[58] Philippe Weinzaepfel, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. Visual localization by learning objects-of-interest dense match regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[59] Erik Wijmans and Yasutaka Furukawa. Exploiting 2d floorplan for building-scale panorama rgbd alignment. In *Computer Vision and Pattern Recognition, CVPR*, 2017.

[60] Amir Roshan Zamir and Mubarak Shah. Accurate Image Localization Based on Google Maps Street View. In *ECCV*, 2010.

[61] Zichao Zhang, Torsten Sattler, and Davide Scaramuzza. Reference Pose Generation for Visual Localization via Learned Features and View Synthesis. *arXiv*, 2005.05179, 2020.