MeshLoc: Mesh-Based Visual Localization

Vojtech Panek[1,2(✉)] , Zuzana Kukelova[3] , and Torsten Sattler[2]

[1] Faculty of Electrical Engineering, Czech Technical University (CTU) in Prague, Prague, Czech Republic
`vojtech.panek@cvut.cz`
[2] Czech Institute of Informatics, Robotics and Cybernetics CTU in Prague, Prague, Czech Republic
[3] Visual Recognition Group, Faculty of Electrical Engineering CTU in Prague, Prague, Czech Republic

Abstract. Visual localization, *i.e.*, the problem of camera pose estimation, is a central component of applications such as autonomous robots and augmented reality systems. A dominant approach in the literature, shown to scale to large scenes and to handle complex illumination and seasonal changes, is based on local features extracted from images. The scene representation is a sparse Structure-from-Motion point cloud that is tied to a specific local feature. Switching to another feature type requires an expensive feature matching step between the database images used to construct the point cloud. In this work, we thus explore a more flexible alternative based on dense 3D meshes that does not require features matching between database images to build the scene representation. We show that this approach can achieve state-of-the-art results. We further show that surprisingly competitive results can be obtained when extracting features on renderings of these meshes, without any neural rendering stage, and even when rendering raw scene geometry without color or texture. Our results show that dense 3D model-based representations are a promising alternative to existing representations and point to interesting and challenging directions for future research.

Keywords: Visual localization · 3D meshes · Feature matching

1 Introduction

Visual localization is the problem of estimating the position and orientation, *i.e.*, the camera pose, from which the image was taken. Visual localization is a core component of intelligent systems such as self-driving cars [27] and other autonomous robots [40], augmented and virtual reality systems [42,45], as well as of applications such as human performance capture [26].

In terms of pose accuracy, most of the current state-of-the-art in visual localization is structure-based [11–14,16,28,59,61,65,69,74,79,92]. These approaches
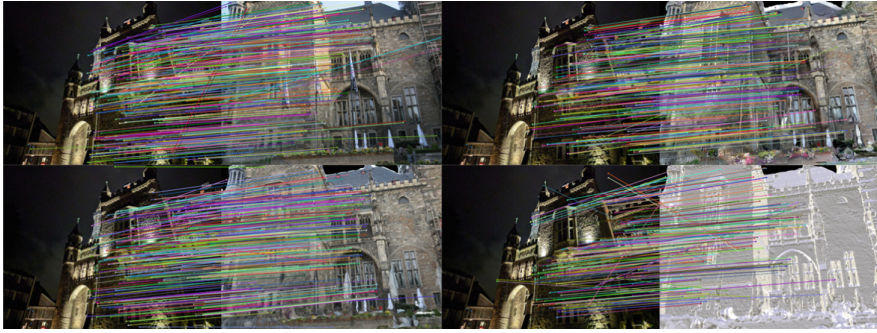
Fig. 1. Modern learned features such as Patch2Pix [94] are not only able to establish correspondences between real images (top-left), but are surprisingly good at matching between a real images and non-photo-realistic synthetic views (top-right: textured mesh, bottom-left: colored mesh, bottom-right: raw surface rendering). This observation motivates our investigation into using dense 3D meshes, rather than the Structure-from-Motion point clouds predominantly used in the literature

establish 2D-3D correspondences between pixels in a query image and 3D points in the scene. The resulting 2D-3D matches can in turn be used to estimate the camera pose, *e.g.*, by applying a minimal solver for the absolute pose problem [24,53] inside a modern RANSAC implementation [4–6,18,24,37]. The scene is either explicitly represented via a 3D model [28,29,38,39,59,62,63,65,79,92] or implicitly via the weights of a machine learning model [9–12,14–16,44,74,86].

Methods that explicitly represent the scene via a 3D model have been shown to scale to city-scale [63,79,92] and beyond [38], while being robust to illumination, weather, and seasonal changes [28,61,62,83]. These approaches typically use local features to establish the 2D-3D matches. The dominant 3D scene representation is a Structure-from-Motion (SfM) [1,70,76] model. Each 3D point in these sparse point clouds was triangulated from local features found in two or more database images. To enable 2D-3D matching between the query image and the 3D model, each 3D point is associated with its corresponding local features. While such approaches achieve state-of-the-art results, they are rather *inflexible*. Whenever a better type of local features becomes available, it is necessary to recompute the point cloud. Since the intrinsic calibrations and camera poses of the database images are available, it is sufficient to re-triangulate the scene rather than running SfM from scratch. Still, computing the necessary feature matches between database images can be highly time-consuming.

Often, it is possible to obtain a dense 3D model of the scene, *e.g.*, in the form of a mesh obtained via multi-view stereo [32,71], from depth data, from LiDAR, or from other sources such as digital elevation models [13,84]. Using a dense model instead of a sparse SfM point cloud offers more flexibility: rather than having to match features between database images to triangulate 3D scene points, one can simply obtain the corresponding 3D point from depth maps rendered from the model. Due to decades of progress in computer graphics research

and development, even large 3D models can be rendered in less than a millisecond. Thus, feature matching and depth map rendering can both be done online without the need to pre-extract and store local features. This leads to the question whether one needs to store images at all or could render views of the model on demand. This in turn leads to the question how realistic these renderings need to be and thus which level of detail is required from the 3D models.

This paper investigates using dense 3D models instead of sparse SfM point clouds for feature-based visual localization. Concretely, the paper makes the following contributions: (1) we discuss how to design a dense 3D model-based localization pipeline and contrast this system to standard hierarchical localization systems. (2) we show that a very simple version of the pipeline can already achieve state-of-the-art results when using the original images and a 3D model that accurately aligns with these images. Our mesh-based framework reduces overhead in testing local features and feature matchers for visual localization tasks compared to SfM point cloud-based methods. (3) we show interesting and promising results when using non-photo-realistic renderings of the meshes instead of real images in our pipeline. In particular, we show that existing features, applied out-of-the-box without fine-tuning or re-training, perform surprisingly well when applied on renderings of the raw 3D scene geometry without any colors or textures (*cf.* Fig. 1). We believe that this result is interesting as it shows that standard local features can be used to match images and purely geometric 3D models, *e.g.*, laser or LiDAR scans. (4) our code and data are publicly available at https://github.com/tsattler/meshloc_release.

Related Work. One main family of state-of-the-art visual localization algorithms is based on local features [13,28,38,59,61,63,65,69,79–81,92]. These approaches commonly represent the scene as a sparse SfM point cloud, where each 3D point was triangulated from features extracted from the database images. At test time, they establish 2D-3D matches between pixels in a query image and 3D points in the scene model using descriptor matching. In order to scale to large scenes and handle complex illumination and seasonal changes, a hybrid approach is often used [28,29,60,67,80,81]: an image retrieval stage [2,85] is used to identify a small set of potentially relevant database images. Descriptor matching is then restricted to the 3D points visible in these images. We show that it is possible to achieve similar results using a mesh-based scene representation that allows researchers to more easily experiment with new types of features.

An alternative to explicitly representing the 3D scene geometry via a 3D model is to implicitly store information about the scene in the weights of a machine learning model. Examples include scene coordinate regression techniques [9,10,12,14–16,74,86], which regress 2D-3D matches rather than computing them via explicit descriptor matching, and absolute [33,34,47,73,89] and relative pose [3,22,36] regressors. Scene coordinate regressors achieve state-of-the-art results for small scenes [8], but have not yet shown strong performance in more challenging scenes. In contrast, absolute and relative pose regressors

are currently not (yet) competitive to feature-based methods [68,95], even when using additional training images obtained via view synthesis [47,50,51].

Ours is not the first work to use a dense scene representation. Prior work has used dense Multi-View Stereo [72] and laser [50,51,75,80,81] point clouds as well as textured or colored meshes [13,48,93]. [48,50,51,72,75] render novel views of the scene to enable localization of images taken from viewpoints that differ strongly from the database images. Synthetic views of a scene, rendered from an estimated pose, can also be used for pose verification [80,81]. [47,48,93] rely on neural rendering techniques such as Neural Radiance Fields (NeRFs) [46,49] or image-to-image translation [96] while [50,72,75,93] rely on classical rendering techniques. Most related to our work are [13,93] as both use meshes for localization: given a rather accurate prior pose, provided manually, [93] render the scene from the estimated pose and match features between the real image and the rendering. This results in a set of 2D-3D matches used to refine the pose. While [93] start with poses close to the ground truth, we show that meshes can be used to localize images from scratch and describe a full pipeline for this task. While the city scene considered in [93] was captured by images, [13] consider localization in mountainous terrain, where only few database images are available. As it is impossible to compute an SfM point cloud from the sparsely distributed database images, they instead use a textured digital elevation model as their scene representation. They train local features to match images and this coarsely textured mesh, whereas we use learned features without re-training or fine-tuning. While [13] focus on coarse localization (on the level of hundreds of meters or even kilometers), we show that meshes can be used for centimeter-accurate localization. Compared to these prior works, we provide a detailed ablation study investigating how model and rendering quality impact the localization accuracy.

2   Feature-Based Localization via SfM Models

This section first reviews the general outline of state-of-the-art hierarchical structure-based localization pipelines. Section 3 then describes how such a pipeline can be adapted when using a dense instead of a sparse scene representation.

Stage 1: Image Retrieval. Given a set of database images, this stage identifies a few relevant reference views for a given query. This is commonly done via nearest neighbor search with image-level descriptors [2,25,55,85].

Stage 2: 2D-2D Feature Matching. This stage establishes feature matches between the query image and the top-k retrieved database images, which will be upgraded to 2D-3D correspondences in the next stage. It is common to use state-of-the-art learned local features [21,23,56,61,78,94]. Matches are established either by (exhaustive) feature matching, potentially followed by outlier filters such as Lowe's ratio test [41], or using learned matching strategies [57,58,61,94].

There are two representation choices for this stage: pre-compute the features for the database images or only store the photos and extract the features on-the-fly. The latter requires less storage at the price of run-time overhead. *E.g.*, storing SuperPoint [21] features for the Aachen Day-Night v1.1 dataset [66,67, 93] requires more than 25 GB while the images themselves take up only 7.5 GB (2.5 GB when reducing the image resolution to at most 800 pixels).

Stage 3: Lifting 2D-2D to 2D-3D Matches. For the i-th 3D scene point $\mathbf{p}_i \in \mathbb{R}^3$, each SfM point cloud stores a set $\{(\mathcal{I}_{i_1}, \mathbf{f}_{i_1}), \cdots (\mathcal{I}_{i_n}, \mathbf{f}_{i_n})\}$ of (image, feature) pairs. Here, a pair $(\mathcal{I}_{i_j}, \mathbf{f}_{i_j})$ denotes that feature $\mathbf{f}_{i_j}$ in image $\mathcal{I}_{i_j}$ was used to triangulate the 3D point position $\mathbf{p}_i$. If a feature in the query image matches $\mathbf{f}_{i_j}$ in database image $\mathcal{I}_{i_j}$, it thus also matches $\mathbf{p}_i$. Thus, 2D-3D matches are obtained by looking up 3D points corresponding to matching database features.

Stage 4: Pose Estimation. The last stage uses the resulting 2D-3D matches for camera pose estimation. It is common practice to use LO-RANSAC [17,24,37] for robust pose estimation. In each iteration, a P3P solver [53] generates pose hypotheses from a minimal set of three 2D-3D matches. Non-linear refinement over all inliers is used to optimize the pose, both inside and after LO-RANSAC.

Covisibility Filtering. Not all matching 3D points might be visible together. It is thus common to use a covisibility filter [38,39,60,64]: a SfM reconstruction defines the so-called visibility graph $\mathcal{G} = ((I, P), E)$ [39], a bipartite graph where one set of nodes I corresponds to the database images and the other set P to the 3D points. $\mathcal{G}$ contains an edge between an image node and a point node if the 3D point has a corresponding feature in the image. A set $M = \{(\mathbf{f}_i, \mathbf{p}_i)\}$ of 2D-3D matches defines a subgraph $\mathcal{G}(M)$ of $\mathcal{G}$. Each connected component of $\mathcal{G}(M)$ contains 3D points that are potentially visible together. Thus, pose estimation is done per connected component rather than over all matches [60,63].

3   Feature-Based Localization Without SfM Models

This paper aims to explore dense 3D scene models as an alternative to the sparse Structure-from-Motion (SfM) point clouds typically used in state-of-the-art feature-based visual localization approaches. Our motivation is three-fold:

(1) dense scene models are more flexible than SfM-based representations: SfM point clouds are specifically build for a given type of feature. If we want to use another type, *e.g.*, when evaluating the latest local feature from the literature, a new SfM point cloud needs to be build. Feature matches between the database images are required to triangulate SfM points. For medium-sized scenes, this matching process can take hours, for large scenes days or weeks. In contrast, once a dense 3D scene model is build, it can be used to directly provide the corresponding 3D point for (most of) the pixels in a database image by simply rendering a depth map. In turn, depth maps can be rendered highly efficiently

when using 3D meshes, *i.e.*, in a millisecond or less. Thus, there is only very little overhead when evaluating a new type of local features.

(2) dense scene models can be rather compact: at first glance, it seems that storing a dense model will be much less memory efficient than storing a sparse point cloud. However, our experiments show that we can achieve state-of-the-art results on the Aachen v1.1 dataset [66,67,93] using depth maps generated by a model that requires only 47 MB. This compares favorably to the 87 MB required to store the 2.3M 3D points and 15.9M corresponding database indices (for co-visibility filtering) for the SIFT-based SfM model provided by the dataset.

(3) as mentioned in Sect. 2, storing the original images and extracting features on demand requires less memory compared to directly storing the features. One intriguing possibility of dense scene representations is thus to not store images at all but to use rendered views for feature matching. Since dense representations such as meshes can be rendered in a millisecond or less, this rendering step introduces little run-time overhead. It can also help to further reduce memory requirements: *E.g.*, a textured model of the Aachen v1.1 [66,67,93] dataset requires around 837 MB compared to the more than 7 GB needed for storing the original database images (2.5 GB at reduced resolution). While synthetic images can also be rendered from sparse SfM point clouds [54,77], these approaches are in our experience orders of magnitude slower than rendering a 3D mesh.

The following describes the design choices one has when adapting the hierarchical localization pipeline from Sect. 2 to using dense scene representations.

Stage 1: Image Retrieval. We focus on exploring using dense representations for obtaining 2D-3D matches and do not make any changes to the retrieval stage. Naturally, use additional rendered views can be used to improve the retrieval performance [29,48,75]. As we are interested in comparing classical SfM-based and dense representations, we do not investigate this direction of research though.

Stage 2: 2D-2D Feature Matching. Algorithmically, there is no difference between matching features between real images and a real query image and a rendered view. Both cases result in a set of 2D-2D matches that can be upgraded to 2D-3D matches in the next stage. As such, we do not modify this stage. We employ state-of-the-art learned local features [21,23,56,61,78,94] and matching strategies [61]. We do not re-train any of the local features. Rather, we are interested in determining how well these features work out-of-the-box for non-photo-realistic images for different degrees of non-photo-realism, *i.e.*, textured 3D meshes, colored meshes where each vertex has a corresponding RGB color, and raw geometry without any color.

Stage 3: Lifting 2D-2D to 2D-3D Matches. In an SfM point cloud, each 3D point $\mathbf{p}_i$ has multiple corresponding features $\mathbf{f}_{i_1}, \cdots, \mathbf{f}_{i_n}$ from database images $\mathcal{I}_{i_1}, \cdots, \mathcal{I}_{i_n}$. Since the 2D feature positions are subject to noise, $\mathbf{p}_i$ will not precisely project to any of its corresponding features. $\mathbf{p}_i$ is computed such that it

minimizes the sum of squared reprojection errors to these features, thus averaging out the noise in the 2D feature positions. If a query feature $\mathbf{q}$ matches to features $\mathbf{f}_{i_j}$ and $\mathbf{f}_{i_k}$ belonging to $\mathbf{p}_i$, we obtain a single 2D-3D match $(\mathbf{q}, \mathbf{p}_i)$.

When using a depth map obtained by rendering a dense model, each database feature $\mathbf{f}_{i_j}$ with a valid depth will have a corresponding 3D point $\mathbf{p}_{i_j}$. Each $\mathbf{p}_{i_j}$ will project precisely onto its corresponding feature, *i.e.*, the noise in the database feature positions is directly propagated to the 3D points. This implies that even though $\mathbf{f}_{i_1}, \cdots, \mathbf{f}_{i_n}$ are all noisy measurements of the same physical 3D point, the corresponding model points $\mathbf{p}_{i_1}, \cdots, \mathbf{p}_{i_n}$ will all be (slightly) different. If a query feature $\mathbf{q}$ matches to features $\mathbf{f}_{i_j}$ and $\mathbf{f}_{i_k}$, we thus obtain multiple (slightly) different 2D-3D matches $(\mathbf{q}, \mathbf{p}_{i_j})$ and $(\mathbf{q}, \mathbf{p}_{i_k})$.

There are two options to handle the resulting multi-matches: (1) we **simply use all individual matches**. This strategy is extremely simple to implement, but can also produce a large number of matches. For example, when using the top-50 retrieved images, each query feature $\mathbf{q}$ can produce up to 50 2D-3D correspondences. This in turn slows down RANSAC-based pose estimation. In addition, it can bias the pose estimation process towards finding poses that are consistent with features that produce more matches.

(2) we **merge multiple 2D-3D matches into a single 2D-3D match**: given a set $\mathcal{M}(\mathbf{q}) = \{(\mathbf{q}, \mathbf{p}_i)\}$ of 2D-3D matches obtained for a query feature $\mathbf{q}$, we estimate a single 3D point $\mathbf{p}$, resulting in a single 2D-3D correspondence $(\mathbf{q}, \mathbf{p})$. Since the set $\mathcal{M}(\mathbf{q})$ can contain wrong matches, we first try to find a consensus set using the database features $\{\mathbf{f}_i\}$ corresponding to the matching points. For each matching 3D point $\mathbf{p}_i$, we measure the reprojection error *w.r.t.* to the database features and count the number of features for which the error is within a given threshold. The point with the largest number of such inliers[1] is then refined by optimizing its sum of squared reprojection errors *w.r.t.* the inliers. If there is no point $\mathbf{p}_i$ with at least two inliers, we keep all matches from $\mathcal{M}(\mathbf{q})$. This approach thus aims at averaging out the noise in the database feature detections to obtain more precise 3D point locations.

Stage 4: Pose Estimation. Given a set of 2D-3D matches, we follow the same approach as in Sect. 2 for camera pose estimation. However, we need to adapt covisibility filtering and introduce a simple position averaging approach as a post-processing step after RANSAC-based pose estimation.

Covisibility Filtering. Dense scene representations do not directly provide the co-visibility relations encoded in the visibility graph $\mathcal{G}$ and we want to avoid computing matches between database images. Naturally, one could compute visibility relations between views using their depth maps. However, this approach is computationally expensive. A more efficient alternative is to define the visibility graph on-the-fly via shared matches with query features: the 3D points visible in views $\mathcal{I}_i$ and $\mathcal{I}_j$ are deemed co-visible if there exists at least one pair of matches

---

[1] We actually optimize a robust MSAC-like cost function [37] not the number of inliers.

| original image | AC13-C textured | AC13 colored g | AC14 colored | AC15 colored |
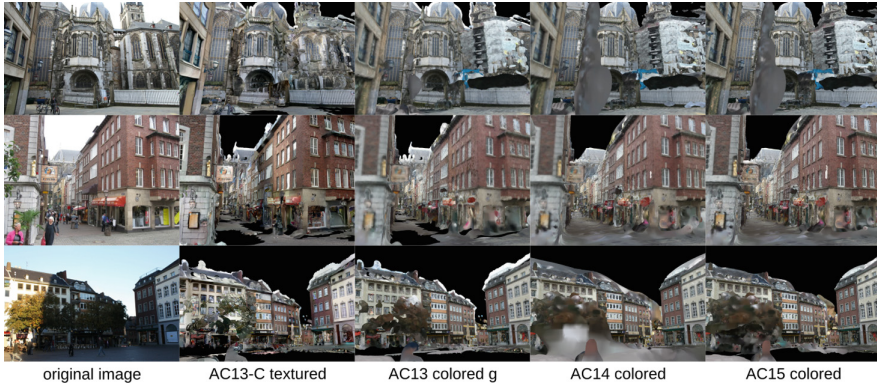
Fig. 2. Examples of colored/textured renderings from the Aachen Day-Night v1.1 dataset [66,67,93]. We use meshes of different levels of detail (from coarsest to finest: *AC13-C*, *AC13*, *AC14*, and *AC15*) and different rendering styles: a textured 3D model (only for *AC13-C*) and meshes with per-vertex colors (*colored*). For reference, the leftmost column shows the corresponding original database image.

$(\mathbf{q}, \mathbf{f}_i)$, $(\mathbf{q}, \mathbf{f}_j)$ between a query feature $\mathbf{q}$ and features $\mathbf{f}_i \in \mathcal{I}_i$ and $\mathbf{f}_j \in \mathcal{I}_j$. In other words, the 3D points from two images are considered co-visible if at least one feature in the query image matches to a 3D point from each image.

Naturally, the 2D-2D matches (and the corresponding 2D-3D matches) define a set of connected components and we can perform pose estimation per component. However, the visibility relations computed on the fly are an approximation to the visibility relations encoded in $\mathcal{G}$: images $\mathcal{I}_i$ and $\mathcal{I}_j$ might not share 3D points, but can observe the same 3D points as image $\mathcal{I}_k$. In $\mathcal{G}(M)$, the 2D-3D matches found for images $\mathcal{I}_i$ and $\mathcal{I}_j$ thus belong to a single connected component. In the on-the-fly approximation, this connection might be missed, *e.g.*, if image $\mathcal{I}_k$ is not among the top-retrieved images. Covisibility filtering using the on-the-fly approximation might thus be too aggressive, resulting in an over-segmentation of the set of matches and a drop in localization performance.

Position Averaging. The output of pose estimation approach is a camera pose R, $\mathbf{c}$ and the 2D-3D matches that are inliers to that pose. Here, $R \in \mathbb{R}^{3 \times 3}$ is the rotation from global model coordinates to camera coordinates while $\mathbf{c} \in \mathbb{R}^3$ is the position of the camera in global coordinates. In our experience, the estimated rotation is often more accurate than the estimated position. We thus use a simple scheme to refine the position $\mathbf{c}$: we center a volume of side length $2 \cdot d_{\text{vol}}$ around the position $\mathbf{c}$. Inside the volume, we regularly sample new positions with a step size d_{step} in each direction. For each such position $\mathbf{c}_i$, we count the number I_i of inliers to the pose R, $\mathbf{c}_i$ and obtain a new position estimate $\mathbf{c}'$ as the weighted average $\mathbf{c}' = \frac{1}{\sum_i I_i} \sum_i I_i \cdot \mathbf{c}_i$. Intuitively, this approach is a simple but efficient way to handle poses with larger position uncertainty: for these poses, there will be multiple positions with a similar number of inliers and the resulting position
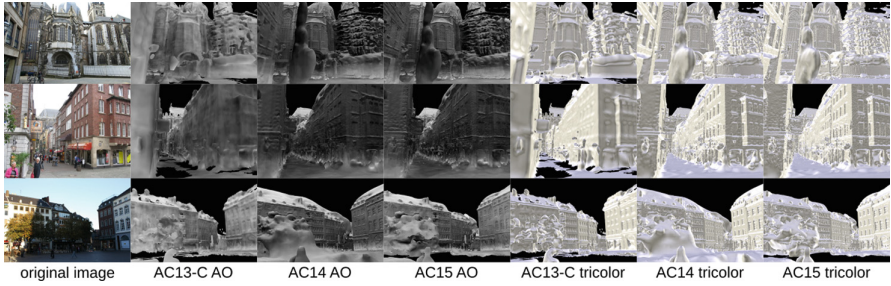
Fig. 3. Example of raw geometry renderings for the Aachen Day-Night v1.1 dataset [66,67,93]. We use different rendering styles to generate synthetic views of the *raw scene geometry*: *ambient occlusion* [97] (AO) and illumination from three colored lights (*tricolor*). The leftmost column shows the corresponding original database image.

Fig. 4. Example renderings for the 12 scenes dataset [86].

$\mathbf{c}'$ will be closer to their average rather than the position with the largest number of inliers (which might be affected by noise in the features and 3D points). Note that this averaging strategy is not tied to using a dense scene representations.

4   Experimental Evaluation

We evaluate the localization pipeline described in Sect. 3 on two publicly available datasets commonly used to evaluate visual localization algorithms, Aachen Day-Night v1.1 [66,67,93] and 12 Scenes [86]. We use the Aachen Day-Night dataset to study the importance (or lack thereof) of the different components in the pipeline described Sect. 3. Using the original database images, we evaluate the approach using multiple learned local features [56,61,78,91,94] and 3D models of different levels of detail. We show that the proposed approach can reach state-of-the-art performance compared to the commonly used SfM-based scene representations. We further study using renderings instead of real images to obtain the 2D-2D matches in Stage 2 of the pipeline, using 3D meshes of different levels of quality and renderings of different levels of detail. A main result is that modern features are robust enough to match real photos against non-photo-realistic renderings of raw scene geometry, even though they were never trained for such a scenario, resulting in surprisingly accurate pose estimates.

Datasets. The Aachen Day-Night v1.1 dataset [66,67,93] contains 6,697 database images captured in the inner city of Aachen, Germany. All database

Table 1. Statistics for the 3D meshes used for experimental evaluation as well as rendering times for different rendering styles and resolutions

| | Model | Style | Size [MB] | Vertices | Triangles | Render time [μs] | |
						800 px	full res.
	AC13-C	textured	645	$1.4 \cdot 10^6$	$2.4 \cdot 10^6$	1143	1187
	AC13-C	tricolor	47	$1.4 \cdot 10^6$	$2.4 \cdot 10^6$	115	219
Aachen v1.1	AC13	colored	617	$14.8 \cdot 10^6$	$29.3 \cdot 10^6$	92	140
	AC13	tricolor	558	$14.8 \cdot 10^6$	$29.3 \cdot 10^6$	97	152
	AC14	colored	1234	$29.4 \cdot 10^6$	$58.7 \cdot 10^6$	100	139
	AC14	tricolor	1116	$29.4 \cdot 10^6$	$58.7 \cdot 10^6$	93	205
	AC15	colored	2805	$66.8 \cdot 10^6$	$133.5 \cdot 10^6$	98	137
	AC15	tricolor	2538	$66.8 \cdot 10^6$	$133.5 \cdot 10^6$	97	160

images were taken under daytime conditions over multiple months. The dataset also contains 824 daytime and 191 nighttime query images captured with multiple smartphones. We use only the more challenging night subset for evaluation.

To create dense 3D models for the Aachen Day-Night dataset, we use Screened Poisson Surface Reconstruction (SPSR) [32] to create 3D meshes from Multi-View Stereo [71] point clouds. We generate meshes of different levels of quality by varying the depth parameter of SPSR, controlling the maximum resolution of the Octree that is used to generate the final mesh. Each of the resulting meshes, AC13, AC14, and AC15 (corresponding to depths 13, 14, and 15, with larger depth values corresponding to more detailed models), has an RGB color associated to each of its vertices. We further generate a compressed version of AC13, denoted as AC13-C, using [31] and texture it using [88]. Figure 2 shows examples.

The 12 Scenes dataset [86] consists of 12 room-scale indoor scenes captured using RGB-D cameras, with ground truth poses created using RGB-D SLAM [20]. Each scene provides RGB-D query images, but we only use the RGB part for evaluation. The dataset further provides the colored meshes reconstructed using [20], where each vertex is associated with an RGB color, which we use for our experiments. Compared to the Aachen Day-Night dataset, the 12 Scenes dataset is "easier" in the sense that it only contains images taken by a single camera that is not too far away from the scene and under constant illumination conditions. Figure 4 shows example renderings.

For both datasets, we render depth maps and images from the meshes using an OpenGL-based rendering pipeline [87]. Besides rendering colored and textured meshes, we also experiment with raw geometry rendering. In the latter case, no colors or textures are stored, which reduces memory requirements. In order to be able to extract and match features, we rely on shading cues. We evaluate two shading strategies for the raw mesh geometry rendering: the first uses ambient occlusion [97] (AO) pre-computed in MeshLab [19]. The second one uses three colored light sources (tricolor) (*cf.* supp. mat. for details). Figs. 3

and 4 show example renderings. Statistics about the meshes and rendering times can be found in Table 1 for Aachen and in the supp. mat. for 12 Scenes.

This paper focuses on dense scene representations based on meshes. Hence, we refer to the pipeline from Sect. 3 as MeshLoc. A more modern dense scene representations are NeRFs [7,43,46,82,90]. Preliminary experiments with a recent NeRF implementation [49] resulted in realistic renderings for the 12 Scenes dataset [86]. Yet, we were not able to obtain useful depth maps. We attribute this to the fact that the NeRF representation can compensate for noisy occupancy estimates via the predicted color [52]. We thus leave a more detailed exploration of neural rendering strategies for future work. At the moment we use well-matured OpenGL-based rendering on standard 3D meshes, which is optimized for GPUs and achieves very fast rendering times (see Table 1). See Sect. 6 in supp. mat. for further discussion on use of NeRFs.

Experimental Setup. We evaluate multiple learned local features and matching strategies: SuperGlue [61] (SG) first extracts and matches SuperPoint [21] features before applying a learned matching strategy to filter outliers. While SG is based on explicitly detecting local features, LoFTR [78] and Patch2Pix [94] (P2P) densely match descriptors between pairs of images and extract matches from the resulting correlation volumes. Patch2Pix+SuperGlue (P2P+SG) uses the match refinement scheme from [94] to refine the keypoint coordinates of SuperGlue matches. For merging 2D-3D matches, we follow [94] and cluster 2D match positions in the query image to handle the fact that P2P and P2P+SG do not yield repeatable keypoints. The supp. mat. provides additional results with R2D2 [56] and CAPS [91] descriptors.

Following [8,30,66,74,83,86], we report the percentage of query images localized within X meters and Y degrees of their respective ground truth poses.

We use the LO-RANSAC [17,37] implementation from PoseLib [35] with a robust Cauchy loss for non-linear refinement (*cf.* supp. mat. for details).

Experiments on Aachen Day-Night. We first study the importance of the individual components of the MeshLoc pipeline. We evaluate the pipeline on real database images and on rendered views of different level of detail and quality. For the retrieval stage, we follow the literature [59,61,78,94] and use the top-50 retrieved database images/renderings based on NetVLAD [2] descriptors extracted from the real database and query images.

Studying the Individual Components of MeshLoc. Table 2 presents an ablation study for the individual components of the MeshLoc pipeline from Sect. 3. Namely, we evaluate combinations of using all available individual 2D-3D matches (I) or merging 2D-3D matches for each query features (M), using the approximate covisibility filter (C), and position averaging (PA). We also compare a baseline that triangulates 3D points from 2D-2D matches between the query image and multiple database images (T) rather than using depth maps.

Table 2. Ablation study on the Aachen Day-Night v1.1 dataset [66,67,93] using real images at reduced (max. side length 800 px) and full resolution (res.), and depth maps rendered using the AC13 model. We evaluate different strategies for obtaining 2D-3D matches (using all individual matches (I), merging matches (M), or triangulation (T)), with and without covisibility filtering (C), and with and without position averaging (PA) for various local features. We report the percentage of nighttime query images localized within 0.25m and $2°/0.5$ m and $5°/5$ m and $10°$ of the ground truth pose. For reference, we also report the corresponding results (from visuallocaliztion.net) obtained using SfM-based representations (last row). Best results per feature are marked in bold

res.	2D-3D	C	PA	SuperGlue (SG) [61]	LoFTR [78]	Patch2Pix [94]	Patch2Pix + SG [94]
	I			72.8/**93.2**/99.0	77.0/92.1/**99.5**	70.7/89.0/95.3	72.3/91.6/100.0
	I	✓		72.3/92.7/99.0	76.4/92.1/**99.5**	72.3/**91.1**/97.4	73.3/91.1/99.5
	I		✓	74.3/93.2/99.0	**78.5**/**93.2**/**99.5**	73.8/89.5/95.3	73.8/92.1/99.5
800	I	✓	✓	73.3/92.1/99.0	77.5/92.7/**99.5**	73.3/**91.1**/97.4	73.8/91.1/99.5
	M		✓	75.4/92.7/99.5	77.0/92.7/**99.5**	70.7/89.5/96.3	73.8/92.7/99.5
	M	✓	✓	75.4/91.6/99.5	75.4/92.1/99.5	69.6/89.0/97.4	72.8/**93.2**/**100.0**
	T		✓	72.3/90.1/97.9	73.3/90.6/98.4	63.9/83.8/94.8	70.7/90.6/97.4
	T	✓	✓	71.7/89.5/97.9	73.8/90.6/98.4	62.8/82.2/94.2	72.3/90.6/97.9
full	I		✓	**77.0**/92.1/99.0		**74.3**/90.1/96.3	74.3/92.1/99.5
SfM				**77.0**/90.6/**100.0**	**78.5**/90.6/99.0	72.3/88.5/**97.9**	**78.0**/90.6/99.0

As can be seen from the results of using down-scaled images (with maximum side length of 800 px), using 3D points obtained from the AC13 model depth maps typically leads to better results than triangulating 3D points. For triangulation, we only use database features that match to the query image. Compared to an SfM model, where features are matched between database images, this leads to fewer features that are used for triangulation per point and thus to less accurate points. Preliminary experiments confirmed that, as expected, the gap between using the 3D mesh and triangulation grows when retrieving fewer database images. Compared to SfM-based pipelines, which use covisibility filtering before RANSAC-based pose estimation, we observe that covisibility filtering typically decreases the pose accuracy of the MeshLoc pipeline due to its approximate nature. Again, preliminary results showed that the effect is more pronounced when using fewer retrieved database images (as the approximation becomes coarser). In contrast, position averaging (PA) typically gives a (slight) accuracy boost. We further observe that the simple baseline that uses all individual matches (I) often leads to similar or better results compared to merging 2D-3D matches (M). In the following, we thus focus on a simple version of MeshLoc, which uses individual matches (I) and PA, but not covisibility filtering.

Comparison with SfM-Based Representations. Table 2 also evaluates the simple variant of MeshLoc on full-resolution images and compares MeshLoc against the corresponding SfM-based results from visuallocalization.net. Note

Table 3. Ablation study on the Aachen Day-Night v1.1 dataset [66,67,93] using real images at reduced resolution (max. 800 px) and full resolution with depth maps rendered from 3D meshes of different levels of detail (*cf.* Table 1). We use a simple MeshLoc variant that uses individual matches and position averaging, but no covis. filtering

Feature	res.	AC13-C	AC13	AC14	AC15
SuperGlue [61]		74.3/92.7/99.5	74.3/93.2/99.0	71.7/91.6/99.0	72.8/92.7/99.5
LoFTR [78]	800	77.5/92.7/99.5	78.5/93.2/99.5	76.4/92.1/99.5	78.0/92.7/99.5
Patch2Pix [94]		71.7/88.0/95.3	73.8/89.5/95.3	67.0/85.9/95.8	72.3/89.0/96.3
Patch2Pix+SG [61,94]		74.9/92.1/99.5	73.8/92.1/99.5	73.8/90.1/99.0	75.4/91.1/99.5
SuperGlue [61]		77.0/92.1/99.5	77.0/92.1/99.0	75.4/91.1/99.0	76.4/92.1/99.0
Patch2Pix [94]	full	74.3/90.1/96.9	74.3/90.1/96.3	71.2/86.9/95.3	72.3/88.0/96.9
Patch2Pix+SG [61,94]		73.3/92.1/99.5	74.3/92.1/99.5	73.3/91.1/99.5	74.3/92.7/99.5

that we did not evaluate LoFTR on the full-resolution images due to the memory constraints of our GPU (NVIDIA GeForce RTX 3060, 12 GB RAM). The simple MeshLoc variant performs similarly well or slightly better than its SfM-based counterparts, with the exception of the finest pose threshold (0.25 m, 2°) for Patch2Pix+SG. This is despite the fact that SfM-based pipelines are significantly more complex and use additional information (feature matches between database images) that are expensive to compute. Moreover, MeshLoc requires less memory at only a small run-time overhead (see supp. mat.). Given its simplicity and ease of use, we thus believe that MeshLoc will be of interest to the community as it allows researchers to more easily prototype new features.

Mesh Level of Detail. Table 3 shows results obtained when using 3D meshes of different levels of detail (*cf.* Table 1). The gap between using the compact AC13-C model (47 MB to store the raw geometry) and the larger AC13 model (558 MB for the raw geometry) is rather small. While AC14 and AC15 offer more detailed geometry, they also contain artefacts in the form of blobs of geometry (*cf.* supp. mat.). Note that we did not optimize these models (besides parameter adjustments) and leave experiments with more accurate 3D models for future work. Overall the level of detail does not seem to be critical for MeshLoc.

Using Rendered Instead of Real Images. Next, we evaluate the MeshLoc pipeline using synthetic images rendered from the poses of the database images instead of real images. Table 4 shows results for various rendering settings, resulting in different levels of realism for the synthetic views. We focus on Super-Glue [61] and Patch2Pix + SuperGlue [61,94]. LoFTR performed similarly well or better than both on textured and colored renderings, but worse when rendering raw geometry (*cf.* supp. mat.).

As Table 4 shows, the pose accuracy gap between using real images and textured/colored renderings is rather small. This shows that advanced neural rendering techniques, *e.g.*, NeRFs [46], have only a limited potential to improve the

Table 4. Ablation study on the Aachen Day-Night v1.1 dataset [66,67,93] using images rendered at reduced resolution (max. 800 px) from 3D meshes of different levels of detail (*cf.* Tab. 1) and different rendering types (textured/colored, raw geometry with ambient occlusion (AO), raw geometry with tricolor shading (tricolor)). For reference, the rightmost column shows results obtained with real images on AC13. MeshLoc uses individual matches and position averaging, but no covisibility filtering

AC13-C:	textured	AO	tricolor	real
SuperGlue [61]	72.3/91.1/99.0	0.5/3.1/24.6	7.3/23.0/53.9	74.3/92.7/99.5
Patch2Pix+SG [61,94]	70.7/90.6/99.5	1.0/4.2/27.7	9.4/25.1/57.6	74.9/92.1/99.5

AC13:	colored	AO	tricolor	real
SuperGlue [61]	68.1/90.1/97.4	6.3/19.9/45.5	22.0/50.8/74.3	74.3/93.2/99.0
Patch2Pix+SG [61,94]	71.7/91.1/97.9	6.8/26.2/49.2	23.0/55.0/78.5	73.8/92.1/99.5

AC14:	colored	AO	tricolor	real
SuperGlue [61]	70.2/90.1/96.3	23.6/44.5/63.9	33.0/65.4/79.1	71.7/91.6/99.0
Patch2Pix+SG [61,94]	72.3/92.1/96.9	26.7/48.2/68.1	39.3/68.6/80.6	73.8/90.1/99.0

AC15:	colored	AO	tricolor	real
SuperGlue [61]	75.4/89.5/98.4	24.1/47.1/63.4	37.2/60.7/77.5	72.8/92.7/99.5
Patch2Pix+SG [61,94]	72.8/92.1/98.4	25.1/51.3/70.2	40.3/66.0/80.1	75.4/91.1/99.5

results. Rendering raw geometry results in significantly reduced performance since neither SuperGlue nor Patch2Pix+SG were trained on this setting. AO renderings lead to worse results compared to the tricolor scheme as the latter produces more sharp details (*cf.* Fig. 3). Patch2Pix+SuperGlue outperforms SuperGlue as it refines the keypoint detections used by SuperGlue on a per-match-basis [94], resulting in more accurate 2D positions and reducing the bias between positions in real and rendered images. Still, the results for the coarsest threshold (5 m, 10°) are surprisingly competitive. This indicates that there is quite some potential in matching real images against renderings of raw geometry, *e.g.*, for using dense models obtained from non-image sources (laser, LiDAR, depth, *etc.*.) for visual localization. Naturally, having more geometric detail leads to better results as it produces more fine-grained details in the renderings.

Experiments on 12 Scenes. The meshes provided by the 12 Scenes dataset [86] come from RGB-D SLAM. Compared to Aachen, where the meshes were created from the images, the alignment between geometry and image data is imperfect.

We follow [8], using the top-20 images retrieved using DenseVLAD [85] descriptors extracted from the original database images and the original pseudo ground-truth provided by the 12 Scenes dataset. The simple MeshLoc variant with SuperGlue, applied on real images, is able to localize 94.0% of all query images within 5 cm and 5° threshold on average over all 12 scenes. This is comparable to state-of-the-art methods such as Active Search [65], DSAC* [12], and DenseVLAD retrieval with R2D2 [56] features, which on average localize more than 99.0% of all queries within 5 cm and 5°. The drop is caused by

a visible misalignment between the geometry and RGB images in some scenes, *e.g.*, apt2/living (see supp. mat. for visualizations), resulting in non-compensable errors in the 3D point positions. Using renderings of colored meshes respectively the tricolor scheme reduces the average percentages of localized images to 65.8% respectively 14.1%. Again, the color and geometry misalignment seems the main reason for the drop when rendering colored meshes, while we did not observed such a large gap for Aachen dataset (which has 3D meshes that better align with the images). Still, 99.6%/92.7%/36.0% of the images can be localized when using real images/colored renderings/tricolor renderings for a threshold of 7 cm and 7°. These numbers further increase to 100%/99.1%/54.2% for 10 cm and 10°. Overall, our results show that using dense 3D models leads to promising results and that these representations are a meaningful alternative to SfM point clouds. Please see the supp. mat. for more 12 Scenes results.

5 Conclusion

In this paper, we explored dense 3D model as an alternative scene representation to the SfM point clouds widely used by feature-based localization algorithms. We have discussed how to adapt existing hierarchical localization pipelines to dense 3D models. Extensive experiments show that a very simple version of the resulting MeshLoc pipeline is able to achieve state-of-the-art results. Compared to SfM-based representations, using a dense scene model does not require an extensive matching step between database images when switching to a new type of local features. Thus, MeshLoc allows researchers to more easily prototype new types of features. We have further shown that promising results can be obtained when using synthetic views rendered from the dense models rather than the original images, even without adapting the used features. This opens up new and interesting directions of future work, *e.g.*, more compact scene representations that still preserve geometric details, and training features for the challenging tasks of matching real images against raw scene geometry. The meshes obtained via classical approaches and classical, *i.e.*, non-neural, rendering techniques that are used in this paper thereby create strong baselines for learning-based follow-up work. The rendering approach also allows to use techniques such as database expansion and pose refinement, which were not included in this paper due to limited space. We released our code, meshes, and renderings.

References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building Rome in a day. In: ICCV 2009, pp. 72–79 (2009)
2. Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. In: CVPR (2016)
3. Balntas, V., Li, S., Prisacariu, V.: RelocNet: continuous metric learning relocalisation using neural nets. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 782–799. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_46
4. Barath, D., Ivashechkin, M., Matas, J.: Progressive NAPSAC: sampling from gradually growing neighborhoods. arXiv preprint arXiv:1906.02295 (2019)
5. Barath, D., Matas, J.: Graph-cut RANSAC. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6733–6741 (2018)
6. Barath, D., Noskova, J., Ivashechkin, M., Matas, J.: MAGSAC++, a fast, reliable and accurate robust estimator. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1304–1312 (2020)
7. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-NeRF: a multiscale representation for anti-aliasing neural radiance fields. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 5835–5844 (2021)
8. Brachmann, E., Humenberger, M., Rother, C., Sattler, T.: On the limits of pseudo ground truth in visual camera re-localisation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6218–6228 (2021)
9. Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., Rother, C.: DSAC - differentiable RANSAC for camera localization. In: CVPR (2017)
10. Brachmann, E., Rother, C.: Learning less is more - 6D camera localization via 3D surface regression. In: CVPR (2018)
11. Brachmann, E., Rother, C.: Expert sample consensus applied to camera relocalization. In: ICCV (2019)
12. Brachmann, E., Rother, C.: Visual camera re-localization from RGB and RGB-D images using DSAC. TPAMI **44**, 5847–5865 (2021)
13. Brejcha, J., Lukáč, M., Hold-Geoffroy, Y., Wang, O., Čadík, M.: LandscapeAR: large scale outdoor augmented reality by matching photographs with terrain models using learned descriptors. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12374, pp. 295–312. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58526-6_18
14. Cavallari, T., Bertinetto, L., Mukhoti, J., Torr, P., Golodetz, S.: Let's take this online: adapting scene coordinate regression network predictions for online RGB-D camera relocalisation. In: 3DV (2019)
15. Cavallari, T., Golodetz, S., Lord, N.A., Valentin, J., Di Stefano, L., Torr, P.H.S.: On-the-fly adaptation of regression forests for online camera relocalisation. In: CVPR (2017)
16. Cavallari, T., et al.: Real-time RGB-D camera pose estimation in novel scenes using a relocalisation cascade. TPAMI **42**, 2465–2477 (2019)
17. Chum, O., Matas, J.: Randomized RANSAC with $T_{d,d}$ test. In: British Machine Vision Conference (BMVC) (2002)
18. Chum, O., Perdoch, M., Matas, J.: Geometric min-hashing: finding a (thick) needle in a haystack. In: ICCV (2007)

19. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: MeshLab: an open-source mesh processing tool. In: Eurographics Italian Chapter Conference (2008)

20. Dai, A., Nießner, M., Zollöfer, M., Izadi, S., Theobalt, C.: BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. TOG **36**, 1 (2017)

21. DeTone, D., Malisiewicz, T., Rabinovich, A.: SuperPoint: self-supervised interest point detection and description. In: CVPR Workshops (2018)

22. Ding, M., Wang, Z., Sun, J., Shi, J., Luo, P.: CamNet: coarse-to-fine retrieval for camera re-localization. In: ICCV (2019)

23. Dusmanu, M., et al.: D2-Net: a trainable CNN for joint detection and description of local features. In: CVPR (2019)

24. Fischler, M.A., Bolles, R.C.: Random sampling consensus: a paradigm for model fitting with application to image analysis and automated cartography. CACM (1981)

25. Gordo, A., Almazan, J., Revaud, J., Larlus, D.: End-to-end learning of deep visual representations for image retrieval. Int. J. Comput. Vision **124**(2), 237–254 (2017)

26. Guzov, V., Mir, A., Sattler, T., Pons-Moll, G.: Human POSEitioning system (HPS): 3D human pose estimation and self-localization in large scenes from body-mounted sensors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4318–4329 (2021)

27. Heng, L., et al.: Project AutoVision: localization and 3D scene perception for an autonomous vehicle with a multi-camera system. In: ICRA (2019)

28. Humenberger, M., et al.: Robust image retrieval-based visual localization using kapture. arXiv:2007.13867 (2020)

29. Irschara, A., Zach, C., Frahm, J.M., Bischof, H.: From structure-from-motion point clouds to fast location recognition. In: CVPR (2009)

30. Jafarzadeh, A., et al.: CrowdDriven: a new challenging dataset for outdoor visual localization. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9825–9835 (2021)

31. Jakob, W., Tarini, M., Panozzo, D., Sorkine-Hornung, O.: Instant field-aligned meshes. ACM Trans. Graph. **34**(6), 189–1 (2015)

32. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Trans. Graph. **32**(3) (2013)

33. Kendall, A., Cipolla, R.: Geometric loss functions for camera pose regression with deep learning. In: CVPR (2017)

34. Kendall, A., Grimes, M., Cipolla, R.: PoseNet: a convolutional network for real-time 6-DOF camera relocalization. In: ICCV (2015)

35. Larsson, V.: PoseLib - minimal solvers for camera pose estimation (2020). https://github.com/vlarsson/PoseLib

36. Laskar, Z., Melekhov, I., Kalia, S., Kannala, J.: Camera relocalization by computing pairwise relative poses using convolutional neural network. In: ICCV Workshops (2017)

37. Lebeda, K., Matas, J.E.S., Chum, O.: Fixing the locally optimized RANSAC. In: BMVC (2012)

38. Li, Y., Snavely, N., Huttenlocher, D., Fua, P.: Worldwide pose estimation using 3D point clouds. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7572, pp. 15–29. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33718-5_2

39. Li, Y., Snavely, N., Huttenlocher, D.P.: Location recognition using prioritized feature matching. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010.

LNCS, vol. 6312, pp. 791–804. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15552-9_57

40. Lim, H., Sinha, S.N., Cohen, M.F., Uyttendaele, M.: Real-time image-based 6-DOF localization in large-scale environments. In: CVPR (2012)

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

42. Lynen, S., Sattler, T., Bosse, M., Hesch, J., Pollefeys, M., Siegwart, R.: Get out of my lab: large-scale, real-time visual-inertial localization. In: RSS (2015)

43. Martin-Brualla, R., Radwan, N., Sajjadi, M.S.M., Barron, J.T., Dosovitskiy, A., Duckworth, D.: NeRF in the wild: neural radiance fields for unconstrained photo collections. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7206–7215 (2021)

44. Massiceti, D., Krull, A., Brachmann, E., Rother, C., Torr, P.H.: Random forests versus neural networks - what's best for camera relocalization? In: ICRA (2017)

45. Middelberg, S., Sattler, T., Untzelmann, O., Kobbelt, L.: Scalable 6-DOF localization on mobile devices. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8690, pp. 268–283. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10605-2_18

46. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 405–421. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_24

47. Moreau, A., Piasco, N., Tsishkou, D., Stanciulescu, B., de La Fortelle, A.: LENS: localization enhanced by neRF synthesis. In: CoRL (2021)

48. Mueller, M.S., Sattler, T., Pollefeys, M., Jutzi, B.: Image-to-image translation for enhanced feature matching, image retrieval and visual localization. ISPRS Ann. Photogram. Remote Sens. Spatial Inf. Sci. (2019)

49. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4), 102:1–102:15 (2022). https://doi.org/10.1145/3528223.3530127

50. Naseer, T., Burgard, W.: Deep regression for monocular camera-based 6-DoF global localization in outdoor environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017)

51. Ng, T., Rodriguez, A.L., Balntas, V., Mikolajczyk, K.: Reassessing the limitations of CNN methods for camera pose regression. CoRR abs/2108.07260 (2021)

52. Oechsle, M., Peng, S., Geiger, A.: UNISURF: unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)

53. Persson, M., Nordberg, K.: Lambda twist: an accurate fast robust perspective three point (P3P) solver. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11208, pp. 334–349. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01225-0_20

54. Pittaluga, F., Koppal, S.J., Kang, S.B., Sinha, S.N.: Revealing scenes by inverting structure from motion reconstructions. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

55. Revaud, J., Almazán, J., Rezende, R.S., Souza, C.R.D.: Learning with average precision: training image retrieval with a listwise loss. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5107–5116 (2019)

56. Revaud, J., Weinzaepfel, P., de Souza, C.R., Humenberger, M.: R2D2: repeatable and reliable detector and descriptor. In: NeurIPS (2019)

57. Rocco, I., Arandjelović, R., Sivic, J.: Efficient neighbourhood consensus networks via submanifold sparse convolutions. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12354, pp. 605–621. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58545-7_35

58. Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., Sivic, J.: Neighbourhood consensus networks. Adv. Neural Inf. Process. Syst. **31** (2018)

59. Sarlin, P.E., Cadena, C., Siegwart, R., Dymczyk, M.: From coarse to fine: robust hierarchical localization at large scale. In: CVPR (2019)

60. Sarlin, P.E., Debraine, F., Dymczyk, M., Siegwart, R., Cadena, C.: Leveraging deep visual descriptors for hierarchical efficient localization. In: Conference on Robot Learning (CoRL) (2018)

61. Sarlin, P.E., DeTone, D., Malisiewicz, T., Rabinovich, A.: SuperGlue: learning feature matching with graph neural networks. In: CVPR (2020)

62. Sarlin, P.E., et al.: Back to the feature: learning robust camera localization from pixels to pose. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3247–3257 (2021)

63. Sattler, T., Havlena, M., Radenovic, F., Schindler, K., Pollefeys, M.: Hyperpoints and fine vocabularies for large-scale location recognition. In: ICCV (2015)

64. Sattler, T., Leibe, B., Kobbelt, L.: Improving image-based localization by active correspondence search. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7572, pp. 752–765. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33718-5_54

65. Sattler, T., Leibe, B., Kobbelt, L.: Efficient & effective prioritized matching for large-scale image-based localization. PAMI **39**, 1744–1756 (2017)

66. Sattler, T., et al.: Benchmarking 6DOF urban visual localization in changing conditions. In: CVPR (2018)

67. Sattler, T., Weyand, T., Leibe, B., Kobbelt, L.: Image retrieval for image-based localization revisited. In: BMVC (2012)

68. Sattler, T., Zhou, Q., Pollefeys, M., Leal-Taixé, L.: Understanding the limitations of cnn-based absolute camera pose regression. In: CVPR (2019)

69. Schönberger, J.L., Pollefeys, M., Geiger, A., Sattler, T.: Semantic visual localization. In: CVPR (2018)

70. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)

71. Schönberger, J.L., Zheng, E., Frahm, J.-M., Pollefeys, M.: Pixelwise view selection for unstructured multi-view stereo. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 501–518. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_31

72. Shan, Q., Wu, C., Curless, B., Furukawa, Y., Hernandez, C., Seitz, S.M.: Accurate geo-registration by ground-to-aerial image matching. In: 3DV (2014)

73. Shavit, Y., Ferens, R., Keller, Y.: Learning multi-scene absolute pose regression with transformers. In: ICCV (2021)

74. Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in RGB-D images. In: CVPR (2013)

75. Sibbing, D., Sattler, T., Leibe, B., Kobbelt, L.: SIFT-realistic rendering. In: 3DV (2013)

76. Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from internet photo collections. IJCV **80**, 189–210 (2008)

77. Song, Z., Chen, W., Campbell, D., Li, H.: Deep novel view synthesis from colored 3D point clouds. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV

2020. LNCS, vol. 12369, pp. 1–17. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58586-0_1

78. Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X.: Loftr: detector-free local feature matching with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)

79. Svärm, L., Enqvist, O., Kahl, F., Oskarsson, M.: City-scale localization for cameras with known vertical direction. PAMI **39**(7), 1455–1461 (2017)

80. Taira, H., et al.: InLoc: indoor visual localization with dense matching and view synthesis. In: CVPR (2018)

81. Taira, H., et al.: Is this the right place? geometric-semantic pose verification for indoor visual localization. In: The IEEE International Conference on Computer Vision (ICCV) (2019)

82. Tancik, M., et al.: Block-NeRF: scalable large scene neural view synthesis. ArXiv abs/2202.05263 (2022)

83. Toft, C., et al.: Long-term visual localization revisited. TPAMI 1 (2020). https://doi.org/10.1109/TPAMI.2020.3032010

84. Tomešek, J., Čadík, M., Brejcha, J.: CrossLocate: cross-modal large-scale visual geo-localization in natural environments using rendered modalities. In: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 2193–2202 (2022)

85. Torii, A., Arandjelović, R., Sivic, J., Okutomi, M., Pajdla, T.: 24/7 place recognition by view synthesis. In: CVPR (2015)

86. Valentin, J., et al.: Learning to navigate the energy landscape. In: 3DV (2016)

87. Waechter, M., Beljan, M., Fuhrmann, S., Moehrle, N., Kopf, J., Goesele, M.: Virtual rephotography: novel view prediction error for 3D Reconstruction. ACM Trans. Graph. **36**(1) (2017)

88. Waechter, M., Moehrle, N., Goesele, M.: Let there be color! large-scale texturing of 3D reconstructions. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 836–850. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_54

89. Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S., Cremers, D.: Image-based localization using LSTMs for structured feature correlation. In: ICCV (2017)

90. Wang, Q., et al.: IBRNet: learning multi-view image-based rendering. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4688–4697 (2021)

91. Wang, Q., Zhou, X., Hariharan, B., Snavely, N.: Learning feature descriptors using camera pose supervision. arXiv:2004.13324 (2020)

92. Zeisl, B., Sattler, T., Pollefeys, M.: Camera pose voting for large-scale image-based localization. In: ICCV (2015)

93. Zhang, Z., Sattler, T., Scaramuzza, D.: Reference pose generation for long-term visual localization via learned features and view synthesis. IJCV **129**, 821–844 (2020)

94. Zhou, Q., Sattler, T., Leal-Taixe, L.: Patch2pix: epipolar-guided pixel-level correspondences. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)

95. Zhou, Q., Sattler, T., Pollefeys, M., Leal-Taixé, L.: To learn or not to learn: visual localization from essential matrices. In: ICRA (2019)

96. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017)

97. Zhukov, S., Iones, A., Kronin, G.: An ambient light illumination model. In: Rendering Techniques (1998)