

# Visual Localization in 3D Maps: Comparing Point Cloud, Mesh, and NeRF Representations

Lintong Zhang<sup>1\*</sup>, Yifu Tao<sup>1</sup>, Jiarong Lin<sup>2</sup>, Fu Zhang<sup>2</sup>, Maurice Fallon<sup>1</sup>

<sup>1</sup>Dept. of Engineering Science, Univ. of Oxford, Pok Fu Lam, Hong Kong, China.

<sup>2</sup>Dept. of Mechanical Engineering, Univ. of Hong Kong, Wellington Square, Oxford, UK.

\*Corresponding author(s). E-mail(s): [lintong@robots.ox.ac.uk](mailto:lintong@robots.ox.ac.uk);

Contributing authors: [yifu@robots.ox.ac.uk](mailto:yifu@robots.ox.ac.uk); [zivlin@connect.hku.hk](mailto:zivlin@connect.hku.hk); [fuzhang@hku.hk](mailto:fuzhang@hku.hk);  
[mfallon@robots.ox.ac.uk](mailto:mfallon@robots.ox.ac.uk);

## Abstract

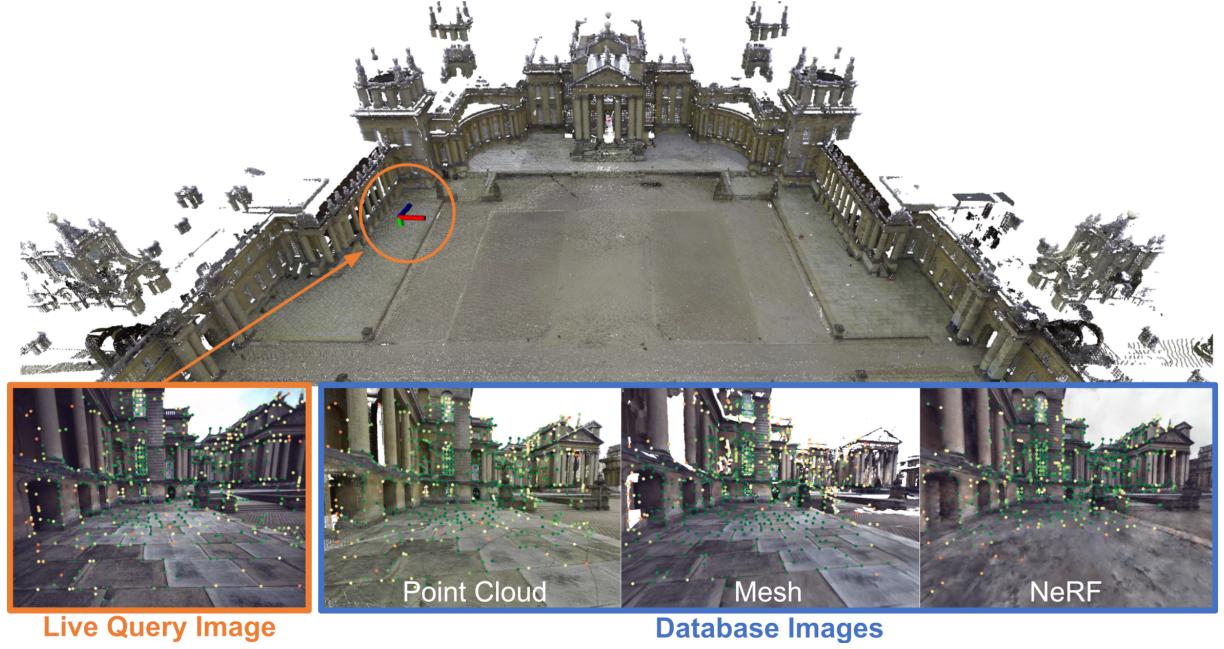
Recent advances in mapping techniques have enabled the creation of highly accurate dense 3D maps during robotic missions, such as point clouds, meshes, or NeRF-based representations. These developments present new opportunities for reusing these maps for localization. However, there remains a lack of a unified approach that can operate seamlessly across different map representations. This paper presents and evaluates a global visual localization system capable of localizing a single camera image across various 3D map representations built using both visual and lidar sensing. Our system generates a database by synthesizing novel views of the scene, creating RGB and depth image pairs. Leveraging the precise 3D geometric map, our method automatically defines rendering poses, reducing the number of database images while preserving retrieval performance. To bridge the domain gap between real query camera images and synthetic database images, our approach utilizes learning-based descriptors and feature detectors. We evaluate the system's performance through extensive real-world experiments conducted in both indoor and outdoor settings, assessing the effectiveness of each map representation and demonstrating its advantages over traditional structure-from-motion (SfM) localization approaches. The results show that all three map representations can achieve consistent localization success rates of 55 % and higher across various environments. NeRF synthesized images show superior performance, localizing query images at an average success rate of 72 %. Furthermore, we demonstrate an advantage over SfM-based approaches that our synthesized database enables localization in the reverse travel direction which is unseen during the mapping process. Our system, operating in real-time on a mobile laptop equipped with a GPU, achieves a processing rate of 1 Hz.

**Keywords:** Localization, Mapping, Sensor Fusion, RGB-D Perception

## 1 Introduction

Recognizing a previously visited place and estimating a robot/sensor's accurate metric pose is fundamental to the problem of robot localization. Localization not only addresses the question of

determining where a robot is, but it also enables other tasks such as loop detection in Simultaneous Localization and Mapping (SLAM), multi-robot or multi-session mapping, and augmented reality. Robot localization can be achieved using different sensor modalities, with camera [1–3] and



**Fig. 1:** Localization of a single query image to a database of images that are synthesized from either point cloud, mesh, or NeRF representations of Blenheim Palace. After identifying a matching image in the database, features are extracted with SuperPoint (as shown above) and matched with SuperGlue. (Displayed point cloud is not directly used.)

lidar-based approaches [4–6] the most commonly studied. While cameras offer a compact form factor and are often low-cost, visual localization and mapping systems often struggle to build large scale accurate 3D maps, especially in real time. Conversely, lidar systems are physically bigger, have higher power consumption and increased cost — factors which pose challenges for their widespread deployment on mobile robots. Because of this, researchers have proposed combining the advantages of both sensors by constructing a 3D dense map using lidar data (potentially aided by cameras), and then performing localization using only the camera [7, 8].

For any localization system, a prior representation of the scene—a prior map—is required. The prior map is usually built using the same sensor modality that will be used for later deployment. For example, autonomous industrial inspection has been demonstrated using legged platforms equipped with lidar [9], where a prior map is built by accumulating laser scans, and localization is performed against the map using of ICP [10]. However, lidar localisation is sensitive to physical scene changes and relies on a good initial

estimate for ICP registration. Conversely, purely visual localization methods typically build a map using a structure-from-motion pipeline (SfM) such as COLMAP [11], and then localize within it using visual place recognition followed pose refinement using methods such as perspective-n-points (PnP) [1]. SfM maps could, however, be inflexible as they are constrained to the local descriptors used to build them. The pose estimated achieved by visual localization within a SfM map are not scaled to real-world dimensions. As a result, additional information is required for effective deployment in robotic applications. Furthermore, these methods are generally not real-time as the SfM bundle adjustment process can take anywhere from a few hours to an entire day, depending on the size of the images or the map.

Achieving cross-modal localization is typically a more challenging problem due to the inherent differences in sensing modalities and map representations. While lidar-based localization often utilizes point cloud maps, some applications may only have access to a colored mesh-based building model [12]. In contrast, visual localization methods rely on a database of 3D points with

associated descriptors, specifically tailored to the feature detectors used during the map construction [2, 3]. Moreover, emerging scene representations such as NeRF [13] and Gaussian Splatting [14] are also dependent on the specifics of the original sensing setup. The variety of map representations poses challenges when applying existing localization approaches.

There has been many recent advances in high-quality 3D color mapping — using either mobile SLAM or fixed TLS data. This has creates new avenues for research into techniques which repurpose these maps for robotic localisation. We propose to create a unified synthetic map representation/database that accommodates the different types of maps produced by these new mapping systems. In particular, these 3D mapping systems enable the rendering of novel synthetic images from arbitrary rendering positions. In our approach, the visual localisation database is made up of rendered RGB and depth images at chosen poses and can be automatically synthesized from color point clouds, meshes, or NeRF maps. These 3D color maps can be built using vision and lidar data acquired from either industrial-grade Terrestrial Laser Scanners (TLS) or hand-held mapping devices. Using this unified representation, we achieve scalable global localization in varied environments, resilient to scene changes and lighting variations with visual learning based components. An overview of the system operation is shown in Fig. 1. In addition, we perform a direct comparison with SfM methods to demonstrate the capability of synthesizing views in the opposite direction of travel, as well as highlighting the advantage of reducing the number of required database images. All experiments are conducted in real-world environments, and the corresponding datasets and ground truth maps are available as part of the Oxford Spires Dataset <sup>1</sup>.

The key contributions of our work include:

- A versatile and unified approach for global visual localization of a single camera in dense 3D maps, using a database of synthetic RGB and depth images generated from point cloud, mesh, or NeRF representations.

---

<sup>1</sup><https://dynamic.robots.ox.ac.uk/datasets/oxford-spires/>

- A strategy for automatically determining the poses within the color 3D maps from which the synthetic images should be generated.
- An extensive evaluation across both indoor and outdoor environments comparing visual localisation performance across different map representations.
- A performance comparison between the our localization system with synthetic images and two (purely) visual localization systems with real images.

Our streamlined system operates on a mobile laptop at 1Hz — and is thus suitable for real-time operation.

## 2 Related Work

We define localization as the process of estimating the 6 DoF pose of a sensor within a 3D prior map or database. This is commonly achieved by means of a place recognition stage—which finds place candidates in the prior map—, and a registration step—that estimates a precise 6 DoF pose using the candidates. Given two sensing inputs of vision and lidar, the current literature can be categorized into four categories — based on sensing modality. These include (a) visual localization within a visual map, (b) lidar localization within a lidar prior map, (c) combined visual and lidar localization within a joint visual and lidar map, and (d) cross-modality visual localization within a lidar prior map or vice versa.

### 2.1 Visual Localization

Visual localization methods aim to obtain a 6 DoF pose by localizing against a map built from multiple images. In general, such maps are built using Structure-from-Motion (SfM) pipelines [11], which are able to recover the 3D structure from multiple views. Given that this reconstruction is accurate up to a scale factor, additional prior information, such as knowledge of a stereo baseline or direct depth sensing, is required for metric localization.

For localization, visual place recognition is implemented as an image retrieval problem. Classic methods such as FAB-MAP [15] and DBOW [16] solve the problem using local features, but more recent methods such as NetVLAD [17], PatchNetVLAD [18], EigenPlaces

[19], MixVPR [20] rely entirely on neural architectures to obtain global descriptors for retrieval. Metric pose estimation is then performed via local feature matching using approaches such as SIFT [21], SuperPoint [22], or R2D2 [23]. This type of pipeline has been implemented in methods such as HLoc [1] and [2], which employ a hierarchical approach to large-scale visual localization.

In addition to the aforementioned methods, alternative approaches have been explored in visual localization in more recent studies. MeshLoc [24] shows it is feasible to construct a dense 3D mesh model from multi-view stereo point clouds and to render synthetic images via an OpenGL pipeline to create a visual database. However, a key limitation of MeshLoc is that the retrieval step relies on a database of real images, again limiting the specific location and direction of the localisation. In another study, [25] opts for a different retrieval approach. Rather than using a global descriptor for a query image, they detect instances of buildings in outdoor environments and retrieve the best match from a database of buildings. They show improvement in long-term and large-scale localization datasets over classic hierarchical frameworks.

Works by Trivigno et al. [26] and Zhang et al. [27] employed an iterative strategy to render images and to refine the pose set based on a query image. While this is an effective approach for solving single query images, it requires an initial pose estimate and lacks the real-time capability needed for robotic applications.

## 2.2 Lidar Localization

Similar to visual localization, lidar localization also maintains a map database—in this case of 3D lidar scans. A hierarchical approach to retrieval and matching typically relies on generating global descriptors from each lidar scan. Early work started with handcrafted descriptors. Himstedt et al. [28] draw inspiration from the bag-of-words approach and transform 2D lidar scans into a histogram representation for place recognition. He et al. [29] project the 3D point cloud into multiple 2D planes, generating descriptors for each of the planes based on point density, and combining them into a global descriptor. More recent work has moved towards learned global descriptors. Vidanapathirana et al. [5] and Uy et al.

[6] both leverage deep learning networks and are able to produce robust global descriptors for loop closure in outdoor large-scale environments.

Recent work in lidar localization techniques has increasingly integrated semantics and extracted segments within the lidar point cloud. Kong et al. [30] introduce a place recognition approach based on a semantic graph, preserving the topological information of the point cloud. They show that by working on a semantic level, their method can be more robust to environmental changes. Aiming to address the same challenge, Segmap [31] introduces a segment-based map representation and generates descriptors for these segments using a learned network. These descriptors serve multiple purposes: localization, map reconstruction, and semantic information extraction. Building upon the segment concept, Locus [32] further extends the approach by combining descriptors of all segments with spatio-temporal high-order pooling to generate fixed-size global descriptors, which are effective for place recognition. This method was demonstrated to achieve robustness in challenging scenarios such as viewpoint changes and occlusions. However, there are unique challenges for indoor environments, as objects can clutter the room in confined spaces, leading to more occlusions. InstaLoc [33] demonstrated good performance for indoor localization by extracting object instances and generating a descriptor for each object before globally matching these objects to estimate the pose.

## 2.3 Combined Visual/Lidar Localization

In scenarios where multiple sensors are available on a mobile robotic platform, some approaches fuse information from both cameras and lidars for localization within a combined image and lidar database. Oneshot [34] customizes a network to generate descriptors from both lidar segments and their corresponding images from a camera. It estimates the sensor pose by extracting segments from a lidar scan and matching their descriptors to a database. The study demonstrates an enhanced retrieval rate when integrating visual information compared to lidar data alone. Bernreiter et al. [35] utilizes a spherical representation to generate descriptors for associated lidar scans and images. A notable advantage of their approach

is the flexibility in accommodating different camera and lidar sensor configurations during both training and querying stages.

In recent work, Adafusion [36] introduces a method that employs adaptive weighting on image and lidar pairs to generate descriptors through an attention network. An adaptive weight allows for different contributions from each sensor, and the results show improved retrieval rates and robustness to changing environments. LC2 [37] is an alternative method wherein 2D images and 3D lidar point clouds are both converted into 2.5D depth images. Then, a network is trained to extract global descriptors from disparity and range images. By reducing modality discrepancy, their method can perform well in very different lighting conditions across multiple missions.

## 2.4 Cross-Modal Localization

For cross-modal localization, most previous work uses a camera to localize in a lidar map. These works are the most relevant to our paper. EdgeMap [38] extracts straight lines from the point cloud map to build a 3D edge map, and applies an edge filter for the camera image. Based on a particle filter, the likelihood of each pose hypothesis is calculated through the filtered camera image and the edge map. Building upon the line extraction idea, Yu et al. [39] extract lines from both the camera image and the lidar map. With a predicted pose from a visual odometry (VO) system, the camera pose is iteratively optimized by minimizing projection error based on 2D-3D line correspondence. They demonstrate that their method can greatly reduce drift by registering live images to the lidar map. However, these methods primarily apply to structured environments where lines and edges are prominent features.

Within the context of autonomous driving, researchers have also been developing localization approaches utilizing inexpensive and readily available camera sensors. Wolcott et al. [7] propose a method to generate a greyscale synthetic view from a mesh map through shading. By minimizing the normalized mutual information between live and synthetic images, this method estimates a pose within the lidar prior map. Similarly, Pascoe et al. [8] synthesize color images from a color textured mesh and can further compensate for

camera calibration changes. Synthetic images produced by this (ten year old) work was rather limited, and localization success was constrained to the vehicle’s path with fixed viewing angles.

Very recently, more improvements have been made to enhance cross-modal image-to-lidar registration methods. Approaches like Zuo et al. [40] register dense stereo depth to a lidar map to correct for visual odometry drift. Based on semantic scene understanding, [41] utilizes the point cloud map and its semantic labels to generate a prior semantic map. During runtime, semantic cues are extracted from live images to localize within the prior semantic map. Both methods fall within the context of camera-lidar registration, which require an initial guess—a relative pose prior. In our work, we focus on global localization without any pose prior information.

## 3 Method

### 3.1 Problem Definition

Our goal is to estimate the position and orientation of a single live RGB camera in a prior map,  $\mathcal{M} = \{\mathcal{I}_1, \mathcal{D}_1, \mathcal{I}_2, \mathcal{D}_2, \dots, \mathcal{I}_n, \mathcal{D}_n\}$ , where  $\mathcal{I}_i, \mathcal{D}_i$  is the  $i$ -th synthesized RGB and depth image pair, that are rendered at  $\mathbf{T}_{WM_i}$ . The relevant frames are the earth-fixed world frame  $W$ , and the map image frame  $M_i$ .

Unless specified otherwise, the position  $\mathbf{P}_{WM_i}$  and orientation  $\mathbf{R}_{WM_i}$  of the map image are expressed in world coordinates, with  $\mathbf{T}_{WM} \in SE(3)$  as the corresponding homogeneous transform.

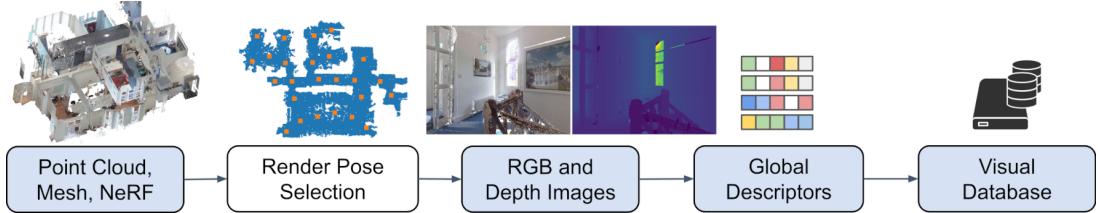
We aim to determine the pose of the live camera image  $C$  at a given time relative to a prior map image, defined as follows:

$$\mathbf{T}_{CM_i} \triangleq [\mathbf{t}_i, \mathbf{R}_i] \in SO(3) \times \mathbb{R}^3 \quad (1)$$

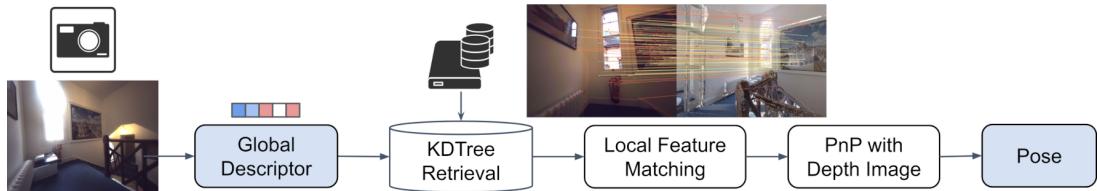
where  $\mathbf{t}_i \in \mathbb{R}^3$  is the translation and  $\mathbf{R}_i \in SO(3)$  is the orientation of  $C$  relative to  $M_i$ . Given  $\mathbf{T}_{WM_i}$  is known, we can find the live camera image pose in the world frame  $\mathbf{T}_{WC}$ .

### 3.2 System Overview

We propose a method that integrates learning-based approaches with classical visual geometry to achieve image localization within a color lidar 3D map. The system design is illustrated in Fig. 2, which shows that the system uses a color 3D map



(a) Step 1: Constructing a visual database from a color 3D map.



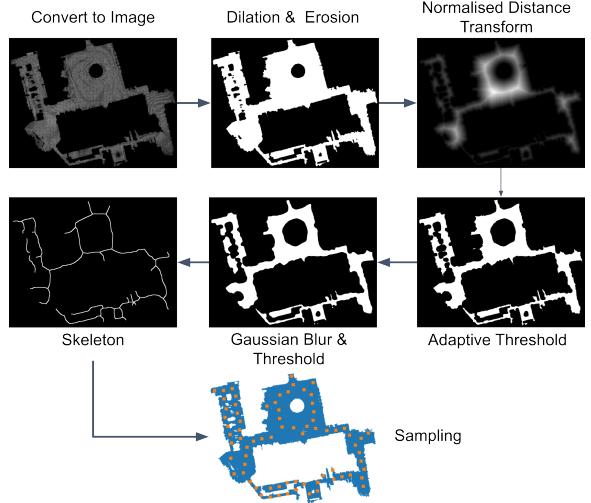
(b) Step 2: A live camera image being matched against a rendered image to obtain a relative pose estimate

**Fig. 2:** Overview of the system showing localization of a camera in the 3D prior map. The blue boxes represent data, and the white boxes represent algorithms.

(either point cloud, mesh or NeRF based) to generate synthetic RGB and depth images (Step 1). Using these images, we then construct a database of global descriptors. During live operation (Step 2), the system receives a query image and generates a descriptor. It then retrieves the closest match in the database. Learned local features are extracted from the images and matched. After retrieving the corresponding depth image, the camera pose can then be estimated based on the matched image features and their positions in the world frame.

### 3.3 Pose Selection when Rendering a Visual Database

A key step is how to decide the set of poses in the 3D prior map from which the artificial images should be synthesized. The goal is to render a reasonably low number of images (to keep the map database small) while achieving coverage across the point cloud or mesh. This step is crucial because it can directly affect both the database size and the available camera views, which influences the overall localization performance. Optimal place recognition can be compromised if the virtual camera is positioned in an unsuitable location—where the map is incomplete, or the camera does not visualize during



**Fig. 3:** Steps to establish a set of plausible rendering positions within a 3D map, which we call a “free path corridor”. The map is split into floors and top-down images are rendered containing all the upward-facing points (selected using their normals). The orange points indicate the final selected positions.

localization. Our approach involves automatically identifying free space within the map and strategically selecting rendering poses along a “free path corridor”.

To address this, we propose a straightforward yet effective method that utilizes geometry and image processing to select rendering poses.

For multi-floor building scans, we calculate normals using data from the TLS scans or the SLAM point cloud map. Planes are then extracted by filtering based on normal directions. Each floor plane can be isolated by constructing a histogram counting the number of points with normals facing upwards by point height. Subsequently, we convert each floor plane point cloud into a top-down image and subject this image to a sequence of image processing techniques to determine the reasonable walking area, as shown in Fig. 3. Dilation and erosion operations are applied to close small holes, followed by a normalized distance transform to identify the center of all the free spaces. A smoothed version of the primary “free path corridor” is produced using thresholding and Gaussian blurring. Finally, a thinning step is implemented to derive a skeleton representing the path of all the walkable space, which is then sampled to obtain a set of rendering positions. We generate four RGB/depth camera image pairs for every position along the path — forward, back, and two side-facing views. This method is used for all of the 3D prior map representations: point clouds, meshes, and NeRF representations.

### 3.4 Generating Synthetic Images

We aim to generate synthetic RGB and depth image pairs rendered from selected poses using one of our three prior map representations. This section describes the construction of each map representation which in turn facilitates the generation of synthetic images for localization.

#### 3.4.1 Render Images from Point Clouds

Point clouds are point samples represented within a 3D coordinate system, sampling an object’s or scene’s external surface. Point clouds are commonly produced by technologies such as lidar scanning and TLS. In our study, we do not utilize point clouds derived from a SLAM system (such as FastLIO [42]), we instead use our TLS-generated point clouds as they are also used to generate precise ground truth maps with millimeter-level measurement accuracy. Further details about the TLS scanning process are provided in Sec. 4.3.

To generate RGB and depth images from the point cloud, we utilize the rendering capabilities of Open3D [43], which leverages an OpenGL framework<sup>2</sup>. Optimal renders are achieved by strategically positioning several light sources around the scene to facilitate specular reflections and shininess. Cameras are placed and maneuvered within the point cloud according to the predefined rendering poses (from the previous subsection). For each pose, 3D points are projected into the camera’s field of view using a pinhole projection model, which allows the pixel size of each point to be adjusted.

Positioning the camera close to a wall causes gaps between points when using a small fixed pixel size, leading to a see-through effect in the rendered image, as illustrated on the left side of Fig. 4. To address this issue, the pixel size of rendered points is varied according to an inverse depth strategy , as detailed in Eq. (2). This approach ensures that the pixel size remains within the designated maximum and minimum splatting sizes.  $\mathbf{z}_{(u,v)}$  represents the depth of a point in 3D space relative to the camera frame.

$$\rho_{(u,v)} = \min[\max[\frac{\rho_{max}}{\mathbf{z}_{(u,v)}}, \rho_{min}], \rho_{max}] \quad (2)$$

Due to a limitation imposed by OpenGL, adjusting individual point sizes dynamically is not feasible, as the parallel processing architecture requires a uniform point size for effective splatting. To circumvent this limitation, we applied Eq. (2) by rendering multiple frames, each with a point size determined by the maximum and minimum point depths within the scene. These frames are then combined to compose the scene’s final RGB and depth image. This is illustrated in Fig. 5, showcasing how four images rendered with different point sizes are merged into a single RGB image.

#### 3.4.2 Render Images from Textured Meshes

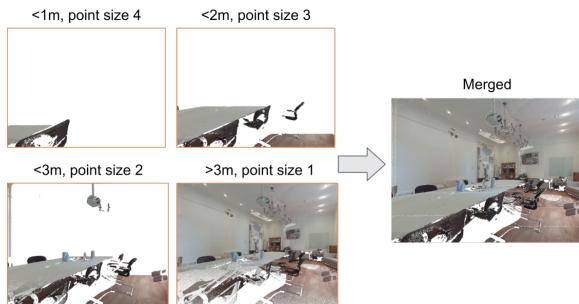
Meshes are widely used in robotics as dense representations for 3D scenes and objects. A mesh is a set of point vertices and the edges between

---

<sup>2</sup>OpenGL, Khronos Group, <https://www.opengl.org/>



**Fig. 4:** Images from the left and right sides illustrate the before and after of rendered images from the point cloud when utilizing an adjustable point size based on the distance to the virtual camera.



**Fig. 5:** Illustration of merging point cloud rendered images of different point sizes to get the final RGB and depth images.

them which together form a set of polygonal faces. Textured meshes are a popular method for representing static 3D scenes [44, 45], as they allow surfaces to be textured with color images, accurately capturing the scene’s appearance.

In our experiments, we extend our previous framework on mesh reconstruction, ImMesh [46], to reconstruct the triangular mesh of scenes captured using lidar scans. ImMesh initiates this process by estimating the pose of the lidar, followed by registering each scan to a global map. This map is partitioned into fixed-sized volumetric voxels. Mesh reconstruction is carried out online using an incremental, voxel-wise meshing algorithm. Initially, all points within a voxel are projected onto an estimated principal plane to reduce dimensionality. Subsequently, the triangular mesh is

reconstructed through a series of voxel-wise operations: mesh pull, commit, and push steps. For more detailed information about the mesh construction process, readers are referred to Section VI of [46]. ImMesh is designed to efficiently reconstruct the mesh of large-scale scenes in real time while maintaining high geometric accuracy.

However, further modifications were necessary to color and texture the mesh reconstruction. While ImMesh constructs a mesh of a scene with adjacent triangle facets connected by edges, these facets lack color information. To address this issue, we use color images captured by the visual camera to texture the facets. The camera poses and image exposure time can be estimated using R<sup>3</sup>LIVE++ [47]. To enhance the smoothness and natural appearance of the mesh texture, for each triangle facet  $\mathcal{T}_i$ , we blend images captured by the  $n$  nearest viewing cameras to form its texture  $\mathbf{I}_i$  (in our experiment,  $n$  is set to 5). The pixel value  $\mathbf{I}_i(u, v)$  at position  $(u, v)$  is blended as follows:

$$\mathbf{I}_i(u, v) = \frac{1}{\bar{\sigma}} \sum_{j=1}^n (\sigma_j \cdot \mathbf{I}_j(\boldsymbol{\pi}(\mathbf{R}_j, \mathbf{t}_j, \mathbf{P}))) \quad (3)$$

$$\bar{\sigma} = \frac{1}{n} \sum_{j=1}^n \sigma_j, \quad \mathbf{P} = \mathbf{A}^{-1}(u_i, v_j) \in \mathbb{R}^3 \quad (4)$$

where  $\boldsymbol{\pi}(\mathbf{R}_j, \mathbf{t}_j, \mathbf{P})$  is the camera projection function that projects a 3D point  $\mathbf{P}$  (on the triangle facet  $\mathcal{T}_i$ ) to the image plane using the estimated camera pose  $(\mathbf{R}_j, \mathbf{t}_j)$ .  $\sigma_j$  denotes the estimated exposure for the  $j$ -th image, and  $\mathbf{A}(\cdot)$  is the wrap transform that maps  $\mathbf{P}$  on facet  $\mathcal{T}_i$  to  $(u, v)$  in texture coordinates, which can be written as  $(u, v) = \mathbf{A}(\mathbf{P})$ .

After reconstructing the textured mesh, we generate synthetic images of the captured scene. This process is accomplished by rendering the mesh into RGB and depth images using OpenGL with the predetermined rendering poses (the approach described in 3.3).

### 3.4.3 Render Images from Neural Radiance Fields

NeRF is an emerging 3D representation that is particularly effective for photorealistic novel view synthesis. NeRF uses an implicit representation, usually a Multi-Layer Perceptron (MLP) [48], to

model a radiance field. The scene is represented as a function whose inputs include a 3D location  $\mathbf{x} = (x, y, z)$  and a 2D viewing direction  $(\theta, \phi)$ . The output of the function is a color  $\mathbf{c} = (r, g, b)$  and volume density  $\sigma$ .

The optimization of a NeRF is based on regulating the rendered image using the input image as the ground truth. Given the extrinsics and intrinsics of the camera, an image can be represented as a collection of rays  $r(t) = o + td$ . The expected color of each ray can be computed from the points sampled along the ray using volumetric integration with quadrature approximation [49, 50] as  $\hat{c}_u = \sum_{i=0}^N w_i c_i$ , where:

$$w_i = \exp \left( - \sum_{j=1}^{i-1} \delta_j \sigma_j \right) (1 - \exp(-\delta_i \sigma_i)). \quad (5)$$

Early NeRF formulations were very computationally intensive — with training typically taking days until convergence. Explicit representations such as voxels [51, 52] and 3D Gaussians [53] have been shown to accelerate training and rendering substantially. Nerfstudio [54] is a well-supported open-source project which incorporates these representations. It also includes features that are effective when working with real-world data, namely scene contraction [55] which can better represent unbounded scenes, and appearance encoding [56] which can model per-image appearance changes including lighting conditions and weather.

As with traditional visual 3D reconstruction systems such as Multi-view Stereo, it is difficult for NeRF to reconstruct textureless surfaces or locations with limited multi-view input. However, accurate depth is important for retrieval and matching, as described in Sec. 3.5. In addition, training a single NeRF for a large-scale scene is difficult due to relative small model size and limited computational hardware.

In this work we use our previous work on visual-lidar NeRF mapping called SiLVR (Scalable Lidar-Visual Reconstruction) [57]. SiLVR builds upon the vision-based Nerfstudio pipeline by adding strong geometry regularization using the lidar depth measurements. With volumetric rendering, one can render both the expected depth

and surface normal. SiLVR uses depth regularization [58] to encourage the ray distribution to follow a narrow normal distribution by minimizing the KL-Divergence between them:

$$\mathcal{L}_{\text{Depth}} = \sum_{\mathbf{r} \in \mathcal{R}} \text{KL}[\mathcal{N}(\mathbf{D}, \hat{\sigma}) \| w(t)] \quad (6)$$

Additionally, SiLVR adds surface normal regularization to enhance the surface reconstruction. We use the same loss function as MonoSDF [59], but instead of learning-based surface normal prediction for supervision, we estimate surface normals directly using the lidar range image. The rendered surface normal is computed as the negative gradient of the density field and is supervised using the following loss function:

$$\mathcal{L}_{\text{Normal}} = \sum_{\mathbf{r} \in \mathcal{R}} \|\hat{N}(\mathbf{r}) - \bar{N}(\mathbf{r})\|_1 + \left\| 1 - \hat{N}(\mathbf{r})^\top \bar{N}(\mathbf{r}) \right\|_1 \quad (7)$$

When training the NeRF model, there are several other considerations. Training large-scale scenes (such as Blenheim Palace in Fig. 1) usually leads to inferior results compared to a smaller scene. One reason for this is the limited model size — the representation power of our learnt model is determined partially by the number of parameters that can be optimised. A relatively small model trained with a large dataset can lead to under-fitting. Additionally, loading thousands of images from a large-scale scene into RAM memory is not always practical. Therefore, SiLVR adopts a submapping system to divide the scene into smaller overlapping submaps. Specifically, the submaps are created by applying Spectral Clustering [60] to divide the full global trajectory into smaller sections. The submaps overlap — with part of the global trajectory reused at the submap boundaries. This allows a smooth transition from one submap to another.

### 3.5 Retrieval and Matching

Upon completion of the synthetic image generation process and the subsequent creation of corresponding depth images, the resulting image database is then input into the NetVLAD [17] network. This network is built with a convolutional neural network and a special VLAD layer.

The central element of VLAD layer is inspired by the image representation technique known as the Vector of Locally Aggregated Descriptors. The network generates descriptors capable of accurately identifying the location of the query image, even amidst significant clutter (e.g., people, cars), variations in viewpoint, and stark differences in illumination, including day and night conditions. The descriptors of all the rendered RGB images are stored in the KDTree data structure for future indexing.

Upon receiving an incoming live camera image, our system initially undergoes a distortion correction process before being subjected to the same NetVLAD network. This step produces a descriptor for the live camera image. Afterward, the KD-Tree retrieves the virtual image that exhibits the closest match based on the descriptor values.

For the pair of matched cameras and virtual images obtained in the aforementioned step, local features are extracted using the SuperPoint algorithm. Due to the large domain gap between the synthetic-to-real images, we observe traditional, not learning-based feature detectors, such as SIFT [21], fail to detect common features. SuperPoint is first pre-trained on a synthetic dataset, boosting the domain adaptation performance. Then SuperPoint is further trained with a multi-scale, multi-transform augmentation, enabling self-supervised training of interest point detectors. This gives SuperPoint the properties of repeatable feature detection across different view angles.

Subsequently, these local features are matched through SuperGlue [61]. The SuperGlue network constitutes a Graph Neural Network coupled with an Optimal Matching layer, which is trained specifically to conduct matching on two sets of sparse image features.

Finally, with access to the matched local features and the depth image, the Perspective-n-Point (PnP)[62] algorithm is used to estimate the relative pose between the live camera image and the rendered reference image. Based on the known pose of the reference image, we can triangulate the live query image in the color 3D map. The pseudo algorithm describing the details of the matching process is shown in Algo. 1; note that we utilize the confidence score from the SuperGlue network to filter out bad matches.

---

**Algorithm 1** Query pose estimation with reference image.

---

```

Input: camera intrinsic  $f_x, f_y, c_x, c_y$ , depth
       image  $D$ , reference image pose in world frame
        $T_{WR}$ 
Output:  $T_{WQ}$  (query image camera pose in
       world frame)
1: for  $i = 1, 2, \dots, N$  matched keypoint do
2:    $q_u, q_v \leftarrow$  query image keypoint[i]
3:    $r_u, r_v \leftarrow$  reference image keypoint[i]
4:   depth  $P_z \leftarrow D[v][u]$ 
5:    $P_x \leftarrow (r_u - cx) * z / fx$ 
6:    $P_y \leftarrow (r_v - cy) * z / fy$ 
7:   if match confidence score  $\omega > 0$  then
8:     3d points  $\leftarrow [P_x, P_y, P_z]$ 
9:     image points  $\leftarrow [q_u, q_v]$ 
10:    end if
11:   end for
12:   if image points  $\leq 6$  then
13:      $T_{QR}, success \leftarrow$  perspective N points with
       (3D points, image points)
14:     if success then
15:       Return  $T_{WQ} = T_{WR} * T_{QR}^{-1}$ 
16:     end if
17:   end if

```

---

### 3.6 Implementation Details

For NeRF 3D reconstruction, we train each SiLVR map over 150,000 iterations, with 4096 rays sampled from the training set in each iteration. This takes around 5 hours on average for all experiments on a single Nvidia RTX 4090 GPU.

For the localization system, the input RGB images have  $720 \times 540$  pixels resolution, and we use off-the-shelf pre-trained networks. Specifically, we use a pre-trained NetVLAD backbone based on VGG16, trained on the Pitts30K [63] dataset. Additionally, the SuperGlue network we use is pre-trained on ScanNet data [64] for indoor scenes and MegaDepth data [65] for outdoor environments.

The localization part of the system (Step 2) is designed for real-time applications. Upon initialization, all rendered images are loaded, global descriptors are extracted for KDTree indexing, and local features are stored in memory. All algorithms are implemented on a Dell laptop with an Intel Core i7-9850H CPU running at 2.60GHz and a Nvidia Quadro T2000 GPU with 4GB of memory. The online localization pipeline processes each

input image in approximately 0.5 seconds, with the following upper limits for each step: global descriptor generation (200ms), KDTree retrieval (1ms), local feature detection (60ms), local feature matching (100ms), and pose estimation (60ms). These performance metrics enable the system to comfortably operate at 1 Hz.

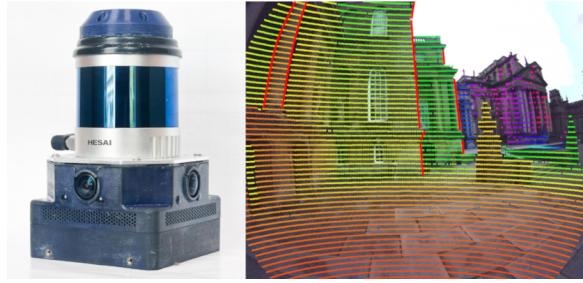
## 4 Experiment Setup and Hardware

### 4.1 Dataset

We collected two recordings at each location: one was used to build the prior map and to render the database images, and the second was used for localization testing. The recordings were collected with a walking speed of around 1m/s. The test locations had the following distinct characteristics:

- ORI [indoor] was collected at the Oxford Robotics Institute. The ORI dataset encompasses indoor environments such as office rooms, staircases, and a kitchen. This dataset is characterized by faster camera rotation movements than the outdoor dataset with the content in the images changing quickly as the recording device moves from room to room.
- Math [outdoor, medium scale] was collected outside the Oxford Mathematics Institute on a summer’s day. This dataset is characterized by substantial lighting variations, with transitions between areas of direct sunlight and shadows cast by the buildings. The presence of bushes, trees, and lawns increases the complexity of the image rendering.
- Blenheim [outdoor, large scale] was collected at Blenheim Palace, a 300-year-old country house in Woodstock, Oxfordshire. It captures walking sequences within the palace courtyard, partially extending into the palace’s main hall. This dataset is challenging as images often contain significant portions of ground and sky, while the geometric symmetry of the courtyard further adds complexity.

The test recording, ORI, was collected across 7 office rooms spread over 2 floors with a total trajectory length of 125 m. The Math trajectory is 451 m long, and the Blenheim trajectory is 386m long.



**Fig. 6:** Frontier multi-sensor unit consists of a 3D lidar, 3 orthogonal cameras and an IMU (left image). A fisheye image overlaid with lidar points (right image), demonstrating accurate intrinsic and extrinsic calibration (red lines are manually added to highlight the depth change).

Again please note that the datasets and ground truth maps used here are part of the Oxford Spires Dataset mentioned in Sec. 1 .

### 4.2 Hardware - Frontier

All experimental data was collected using a *Frontier*, Fig. 6, a self-developed multi-sensor unit consisting of a Sevensense Alphasense Multi-Camera kit with 3 orthogonal cameras of  $1440 \times 1080$  pixels resolution. The device contains a Hesai QT lidar with 64 beams and  $104.2^\circ$  vertical field of view. Note that the Hesai lidar is used to build the mesh and NeRF maps and generate ground truth poses.

The extrinsic calibration between the lidar and camera is established following the methodology outlined in [66], ensuring sub-millimeter accuracy in translation and sub-degree precision in rotations. Accurate calibration is crucial, as shown in Fig. 6, particularly for accurately overlaying lidar scans onto color images when reconstructing color 3D maps.

### 4.3 Ground Truth Maps and Trajectories

To generate ground truth point cloud maps, we used a (professional grade) Leica RTC360 Terrestrial Laser Scanner (TLS) to scan the Math and Blenheim sites and an (entry-level) Leica BLK360<sup>3</sup> to scan the ORI site. Views of these maps are shown in the left section of Tab 1. TLS scans are captured around 3 m apart for indoor

---

<sup>3</sup><https://leica-geosystems.com/products/laser-scanners>

**ORI** and 15 m to 20 m apart for outdoor **Math** and **Blenheim**. Note that ultra-dense distance between TLS scans is impractical due to the considerable time required to scan and process point clouds.

Each Hesai lidar scan was registered to the ground truth point cloud to obtain a 6 DoF pose. We refer readers to our previous paper, where we explain the details of the ground truth generation process in more detail [67], which we claim achieves centimeter precision.

These point clouds were also used to render the synthetic images for the point cloud map representation, as explained in Sec 3.4.1. Before rendering, we voxel-filtered the **ORI** point cloud to 5 mm resolution and the **Math** and **Blenheim** clouds to 1 cm.

## 5 Experiment Results

### 5.1 Comparison of Rendered Images

First we present a set of illustrative examples of rendered images obtained using the TLS point cloud, NeRF, and mesh reconstruction pipelines. Fig. 7 illustrates examples drawn from the three datasets. Each pipeline has the ability to generate reasonably lifelike synthetic images. However, we note that each also exhibits distinct limitations.

For the indoor **ORI** dataset, object occlusion and the relatively sparse coverage of the TLS scanner results in some visible gaps in the point cloud, leading to white regions in the rendered images. Mesh reconstruction faces challenges in accurately representing smaller objects or intricate details, primarily due to the fixed size of the triangular faces. This method is more suitable for objects with planar surfaces than objects with complex profiles. For example, the handrails are difficult to reconstruct with a mesh, as shown in (f). Conversely, NeRF demonstrates its superiority in outdoor environments, successfully reconstructing fine details such as bicycle rails where mesh reconstruction struggles, as shown in (b). However, NeRF often fails to capture intricate ground floor patterns, as shown in (g) and (h), while mesh reconstruction and point cloud techniques, which employ direct image projection, can easily accomplish this.

### 5.2 Full System Evaluation

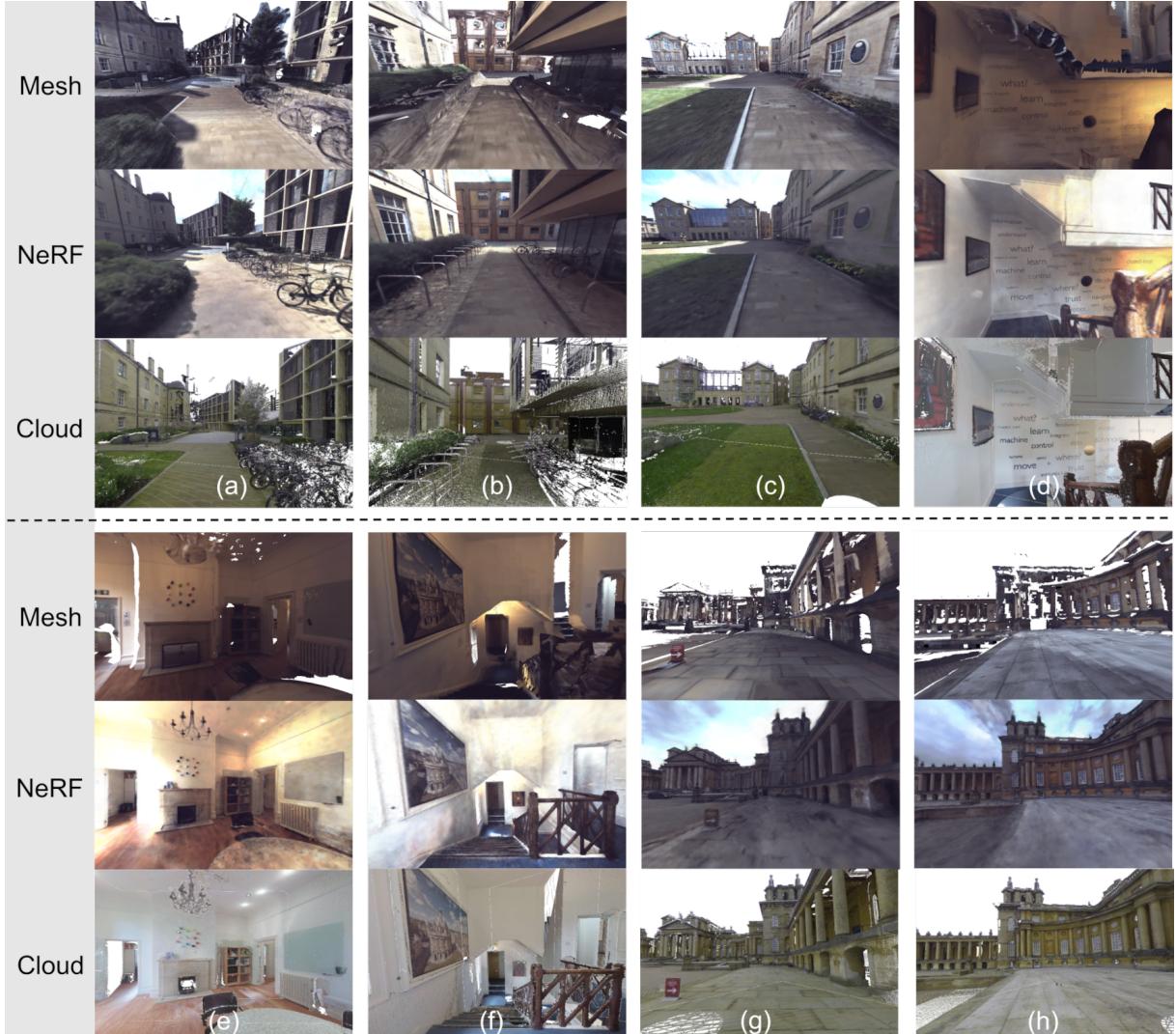
In this section, we quantify the localization performance of the three different rendering pipelines using these synthesized images, benchmarking performance against two entirely vision-based localization systems (which use real images), as detailed in Tab. 1. The systems we compare are HLoc and COLMAP, both based on Structure-from-Motion (SfM). We selected these two methods for comparison because they are widely adopted in the community and are designed for large-scale real-world reconstructions. Once the SfM step is completed, these methods allow for real-time localization of query images.

Tab. 2 presents a high-level analysis of the suitability of these systems for key robotics tasks. The SfM approaches are non-metric, meaning their pose estimates are not scaled to real-world dimensions. This limitation necessitates other information to be available to allow effective deployment in robotic applications. These methods are also not real-time, as the SfM bundle adjustment process would take a couple of hours to a day, depending on the image or map size.

In the following section, we will discuss the configurations of HLoc and COLMAP in our experiments and compare them with our methods of localizing using point cloud, mesh, and NeRF-based map representations. For a fair comparison, the primary performance metrics we use are *retrieval rate*, the proportion of images correctly retrieved as a percentage of the total number of queried images, and *localization rate*, the proportion of images correctly retrieved and localized within a threshold as a percentage of the total number of queried images. As we are evaluating global localization systems, the threshold is within 1 m and 30° of ground truth poses for indoors **ORI**. For outdoor **Blenheim** and **Math**, the threshold is within 2 m and 30°.

#### 5.2.1 Our Methods – Cloud, Mesh and NeRF

For a fair comparison, we render an equal number of images for each dataset at the same locations. For **ORI**, render poses are 2 m apart, generating 130 images. Since **Math** and **Blenheim** cover larger areas than **ORI**, render poses are 4 m apart, and 300 and 210 images were produced, respectively.



**Fig. 7:** A comparison of rendered images from mesh, NeRF, and point cloud in **Math** (a-c), **ORI** (d-f), and **Blenheim** (g-h).

When testing our approach of localizing with a database of rendered images, we found consistent performances for each of our three reconstruction techniques (point cloud, mesh, and NeRF) across three distinct test locations. On average, the localization rates for the point cloud and mesh approaches were 56 % and 58 %, respectively, while the NeRF-based approach achieved the best performance at 72 %. The relatively lower localization rates with point cloud and mesh systems can be attributed to the projection of color and textures from multiple camera views onto the 3D

space, occasionally resulting in occluded regions and color inconsistencies.

Fig. 8 illustrates the performance of pose estimation for query images localized in **Math** against a NeRF-rendered image database. The ground truth pose trajectory is drawn in blue, the estimated poses in orange dots, and the estimated poses that exceeded the threshold are highlighted in purple.

The notable advantage of point cloud and mesh mapping systems is their ability to generate color 3D maps online. The point cloud

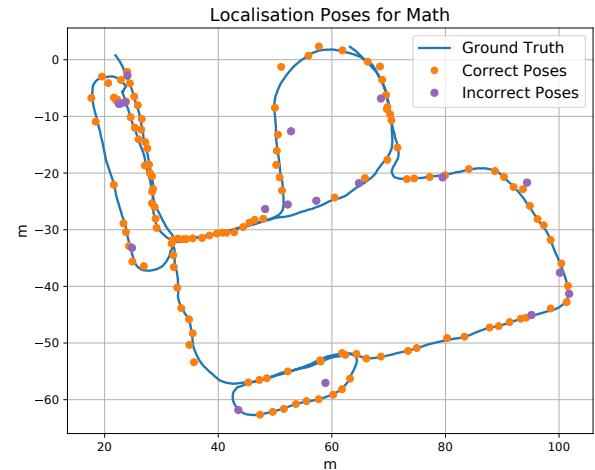
**Table 1:** Performance comparison of the different localization methods. (“DB size” is the number of images in the database. “Ret. Rate” refers to the retrieval rate. “Loc. Rate” means the localization rate. COLMAP (\*) required all the images from all three cameras to reconstruct the prior map accurately for the indoor ORI recording.)

		Dataset	Method	DB Size	Ret. Rate	Loc. Rate
ORI	HLoc	Sparse	130		N/A	Fail
	HLoc	Dense	254		N/A	86
	COLMAP	Dense	3020*		97	<b>96*</b>
	ORI	Cloud			68	54
	Ours	Mesh	130		81	61
	Ours	NeRF			86	<b>67</b>
Math	HLoc	Sparse	503		N/A	20
	HLoc	Dense	1019		N/A	39
	COLMAP	Dense	1569		66	<b>61</b>
	Math	Cloud			89	56
	Ours	Mesh	300		85	57
	Ours	NeRF			97	<b>75</b>
Blenheim	HLoc	Sparse	210		N/A	42
	HLoc	Dense	449		N/A	55
	COLMAP	Dense	449		N/A	<b>78</b>
	Blenheim	Cloud			77	57
	Ours	Mesh	210		77	56
	Ours	NeRF			86	<b>73</b>

**Table 2:** Comparative analysis of localization methods for use in different robotics tasks. ✓ means suitable, ✅ means partly suitable, and ✗ means unsuitable.

Method	Metric	Pose	Real Time	Planning
HLoc		✓		✓
COLMAP		✗	✗	✓
Ours-Cloud		✓		✓
Ours-NeRF		✓	✓	✓
Ours-Mesh				

and mesh results are available immediately following a SLAM run, making them useful for real-time applications that cannot support rendering delays. Conversely, NeRF reconstruction offers the most photorealistic rendering outcomes but requires training after data collection. Additionally, synthetic image generation with the NeRF reconstruction is executed through an MLP inference step, which also requires a slightly longer time (e.g.  $\sim 1$  s per image of resolution  $720 \times 540$  on a Nvidia RTX 4090 GPU). In contrast, the OpenGL-based rendering utilized in the point cloud and mesh method takes a few milliseconds per image.



**Fig. 8:** Estimated poses in Math from the NeRF synthesized image database. The purple dots show estimated poses exceeding the 2 m and  $30^\circ$  threshold compared to the ground truth poses.

### 5.2.2 HLoc

HLoc [1] is a 6 DoF visual localization toolkit. It employs a hierarchical localization approach that integrates image retrieval and feature-matching techniques. We first configured HLoc to build an SfM model using all real images and then used it to register query images to the SfM model. For an objective comparison, we used the same

pre-trained networks as used in our localization method — NetVLAD for image retrieval, SuperPoint for local feature detection, and SuperGlue for feature matching.

For HLoc and COLMAP, the experiments started with the same number of database images as our system. We increased the number of images until their performance improvement plateaued. In general, visual localization systems require a larger number of images to construct an accurate SfM.

To obtain query image pose estimates (up to the scale metric), we first refine the pose estimates by eliminating any erroneous matches and then use Evo<sup>4</sup> to align the query image poses with the ground truth poses to recover the scaling factor and the alignment transformation. Finally, we apply this scaling factor to all estimated poses. We can then calculate the localization rate.

During the initial SfM reconstruction step, HLoc performed well with indoor data but encountered difficulties in accurately constructing the larger outdoor environments of **MATH** and **Blenheim**. We found that constructing a robust SfM solution requires a dense set of images with substantial overlap and minimal lighting changes. For **ORI**, an image database of 130 images (as used for map rendering pipelines) was insufficient as the images were too sparse for the SfM solver to achieve sufficient overlap. Increasing the number of images (to 254 and later 500) enabled the creation of a complete reconstruction, albeit with some observable drift in the staircases connecting the two floors of the building. For **MATH**, the best results were achieved with 1019 images, and increasing the database (to 2040 images) did not noticeably improve the SfM model. Lastly, for **Blenheim**, HLoc achieved its best performance with 449 images, and doubling the image count did not improve the overall SfM accuracy.

Overall, HLoc worked well in the indoor setting, but the reconstruction was less successful in large outdoor scenes.

### 5.2.3 COLMAP

This system [11] is a widely-used SfM and Multi-View Stereo pipeline with graphical and command-line interfaces. It can construct visual

maps using ordered and unordered image collections. In our experiments, we adopted the recommended approach of utilizing SIFT as the feature detector and a vocabulary tree for image retrieval. Our results show that COLMAP requires a higher number of database images than HLoc. It achieves its best performance when incorporating images from the three cameras of the Frontier (front, left, and right facing) within the dataset. This was particularly the case for the indoor **ORI** dataset, where the camera motion is more abrupt/jerky. For **ORI**, COLMAP required all images from all three cameras — facing left, forward, and right, a total of 3020 images.

Once the SfM solution is accurately estimated, localizing new images is straightforward, especially if a query image closely resembles the database images. For **MATH**, the query and map images were captured in quite different lighting conditions. Hence, the performance of image retrieval is lower with COLMAP as it uses the non-learning-based image retrieval method. Across all three locations, COLMAP gave good localization performance, but constructing the SfM solution takes double the amount of time compared to HLoc, e.g., for **MATH**, COLMAP takes around 4.5 hours with 1569 images, and HLoc takes 2 hours for 1019 images.

## 5.3 Image Descriptor, Feature Detector and Matcher

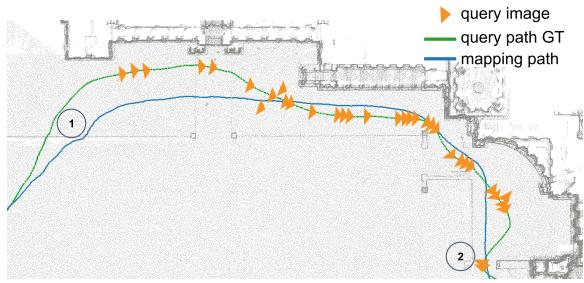
For a fair comparison, we adopted the same image descriptor and feature detector methods as HLoc. We also experimented with image descriptors such as EigenPlaces [19] and MixVPR [20], but NetVLAD marginally outperformed them when tested on real query images and synthetic database images. A more detailed ablation study comparing learning-based and non-learning-based feature detectors and matchers is provided in Sec. 5.6. It is important to note that our approach is not dependent on the specific learning-based detector and matcher — which can be swapped for improved methods as they emerge. The key point here is that by sampling a database of synthetic images, we can leverage dense 3D maps to provide a *unified representation* across different map types. This approach offers flexibility in generating poses within the 3D map and even allows for an iterative “render and compare”

---

<sup>4</sup><https://github.com/MichaelGrupp/evo>



(a) HLoc map: Although the SfM reconstruction is accurate, only four camera views could be correctly localized around the bend.



(b) Mesh map: 30 out of 90 query images could be correctly localized.

**Fig. 9:** Localizing when traveling in a direction opposite to the mapping phase. The mapping direction (blue) is from 1 to 2, and the query direction (green) is from 2 to 1. Orange triangles represent the localized camera view results.

strategy if needed, while also reducing the number of database images required, as compared to SfM-based approaches.

#### 5.4 Localization in Directions Unseen during Mapping

Our method has the powerful capability of generating images from any viewpoint. This is very useful when attempting to localize when traveling in directions opposite to that used to create the map. For instance, in the scenario where the map is constructed by traversing from left to right, as shown in Fig. 9, we can synthesize images from all four orthogonal directions. This capability facilitates localization even when traveling in the reverse direction, a challenge often encountered in conventional visual localization methodologies, which typically requires storing images from both travel directions in the database.

The initial recording was used to reconstruct a mapping database with only the front-facing camera, as shown in Fig. 9a. The map was generated by traversing from point 1 to point 2, where 98 images were captured (illustrated in blue). We then attempted to localize to the map during a reverse traversal from points 2 to 1, with the results depicted using orange-colored camera views. Of the 87 query images, only 4 were successfully localized (when turning around the corner) due to minimal overlap in the image views. A similar pattern was observed with COLMAP where it failed to localize any query images even though the localization module had an accurate SfM reconstruction. HLoc performed marginally better than COLMAP due to its learning-based feature matching approach.

NeRF generated blurrier images from reverse perspectives when trained exclusively with front-facing camera images, resulting in suboptimal localization performance. This issue is likely caused by insufficient training views when the path involves walking in a straight line. In contrast, the mesh-based pipeline demonstrated strong performance in this scenario. Mesh could effectively generate rendered images facing in the un-mapped direction (albeit with imperfections resulting in holes in the rendered images). Despite these visual limitations, mesh-rendered images achieved a localization rate of approximately 33%.

In a second experiment, we expanded the mapping database to include images from two cameras, the side and front camera. As expected, this enhanced the performance for COLMAP and HLoc, with side camera views available for reverse image matching. NeRF exhibited much better localization performance, matching 75% of the images. Note that the mesh reconstruction implementation only works with one camera.

To provide a broader comparison, we show that point clouds captured with an omnidirectional camera (both front and backward facing) can achieve a very high localization rate.

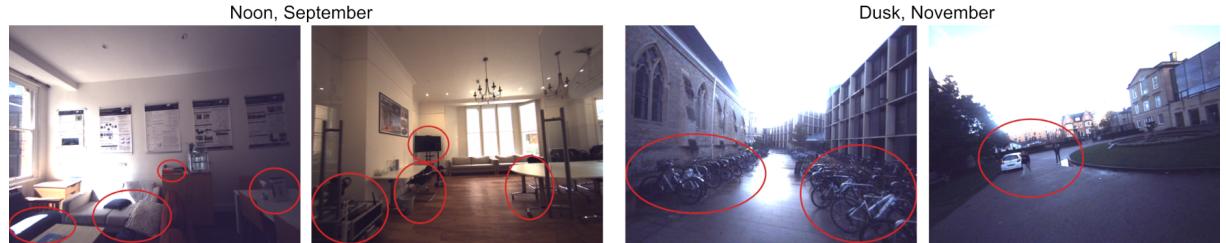
To conclude, this section demonstrates that our system can generate synthetic images in novel views, which aids the localization in a direction that is unseen during the mapping. In particular, mesh-based map representation can render images for localization even when there are very limited

**Table 3:** Localization when traveling in the opposite direction to the mapping trajectory with novel view synthesis.

Method	Camera	Images	Ret. Rate %	Loc. Rate %
COLMAP	front	98	N/A	0
HLoc	front	98	N/A	4
NeRF	front	30	10	7
Mesh	front	30	37	<b>33</b>
COLMAP	front + side	196	N/A	26
HLoc	front + side	196	N/A	39
NeRF	front + side	30	88	<b>75</b>
Mesh	front + side	-	-	-
Point Cloud	omni	30	98	<b>88</b>



(a) Scenes in the original maps.



(b) Changes after three months.

**Fig. 10:** Ablation study: assessing changes in **Math** and **ORI** 2-3 months apart. The top row displays images from the original dataset, while the bottom row shows corresponding images captured months later which reflect the typical movement of objects within the scene. The images on the right-half are from **Math**, while the bottom right ones were captured on a dark day. Please note that some images have been brightened for easier interpretation here.

views of the environment, while NeRF can render better images for localization with a moderate number of views.

### 5.5 Ablation Study: Changes in Environment

When using a prior map for localization, one would expect that changes to the scene would

affect algorithm performance. In indoor environments, furniture and decorations are often reconfigured. Meanwhile, outdoor environments are affected by seasonal and lighting changes, while dynamic objects such as parked cars or bicycles may be added or removed. Our method must be resilient to such changes.

**Table 4:** Localization performance amid scene changes in prior maps.

Image Source	Dataset	Time	Ret. Rate	Loc. Rate
Point Cloud	ORI	Original	68	<b>54</b>
		3 Months later	<b>70</b>	48
NeRF	Math	Original	<b>97</b>	<b>75</b>
		2 Months later	91	51
Mesh	Math	Original	<b>85</b>	<b>56</b>
		2 Months later	85	49

**Table 5:** Average performance of different feature detector combinations for matching a set of ten real and synthetic image pairs. An example is shown in Fig. 11. Note that many of the features matched for Akaze and SIFT are incorrect.

Method	Feature Detected	Matched
Akaze [68] + NN	410	18
SIFT [21] + KNN	841	21
R2D2 [23] + NN	1024	69
D2Net [69] + NN	2932	132
SuperPoint [22] + SuperGlue [61]	1024	201

To demonstrate the robustness of our approach to scene change, we tested our camera localization system with data collected 2-3 months after the initial construction of the maps. The right-hand side of Fig. 10 shows two images from MATH with changes (shown with red circles) in the vegetation color, the location of parked cars, and the configuration of the bicycle stand. On the left-hand side, images from ORI show rearranged furniture and equipment.

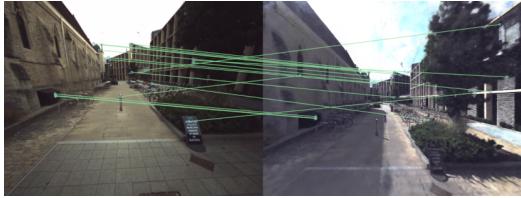
In both locations, lighting variation between the 2-3 month period was clearly observed. The detailed results of localization performance are presented in Tab. 4. All three map representations show a small drop in retrieval and location rates. However, they can still localize around 50 % of query images, demonstrating the robustness of matching against one unified synthetic image database, aided by learning-based visual feature detector SuperPoint and matcher SuperGlue.

## 5.6 Ablation Study: Feature Detection and Matching

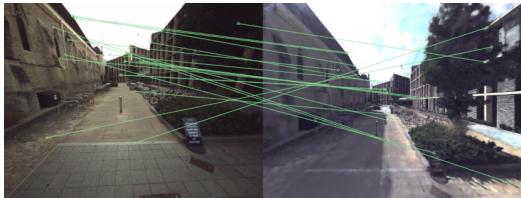
As shown in Fig. 7, there is a significant reality gap between images captured by real cameras and

synthetically rendered images. Each map representation has its own limitations when rendering the synthetic images. Images rendered from TLS point clouds often lack photorealism, exhibit a surreal quality, and have visible gaps corresponding to unscanned regions. Conversely, mesh-rendered images can contain areas with fragmented reconstruction and struggle to faithfully represent small and intricate geometric structures. For the NeRF-generated images, when the rendering poses differ largely from the training data viewpoints, one can often see a fog-like effect in the generated images. Perhaps learning-based feature detectors and matching techniques could help to bridge the domain gap between synthetic and real images and alleviate these issues.

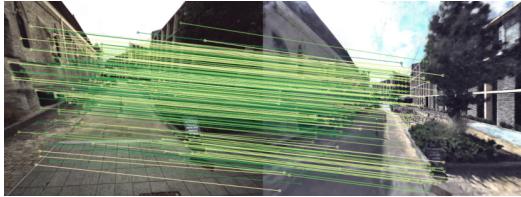
In this section, we conduct a comparative analysis of different feature detectors and matchers using a representative dataset of ten image pairs drawn from across the three datasets. One image is a live camera image, and the other is a rendered image from either the point cloud, mesh, or NeRF pipelines. Fig. 11 shows one scene with common challenges: vegetation, low-textured ground, and thin bike rails. Fig. 11c shows how learning-based approaches such as SuperPoint and SuperGlue can effectively identify and match numerous correct



(a) Detector: OpenCV Akaze, Matcher: Nearest Neighbor



(b) Detector: OpenCV SIFT, Matcher: Flann KNN



(c) Detector: SuperPoint, Matcher: SuperGlue

**Fig. 11:** Example of matching a camera image to a NeRF rendered image with different feature detectors.

features. In contrast, traditional feature detectors like Akaze (Fig. 11a) and SIFT (Fig. 11b) perform poorly in this scenario.

Tab. 5 presents quantitative results for this test, comparing several learning-based and classic feature detection and matching techniques. Among the learning-based feature detectors, SuperPoint, R2D2, and D2Net show promising performance by matching many features. However, traditional methods like SIFT and Akaze match fewer features, many of which are incorrect. Note that R2D2 produces descriptors of size 128, and D2Net generates descriptors of size 512. Given that the SuperGlue matcher is pre-trained with descriptors of size 256, we use the Nearest Neighbour (NN) matcher with a ratio threshold test for R2D2 and D2Net. SuperPoint and R2D2 are capped at 1024 features, while D2Net,

a dense image feature extractor, naturally yields more feature points.

In summary, leveraging learning-based feature detectors and matching techniques is crucial to mitigate the domain gap between camera-captured and synthetic images, enabling robust feature detection and matching across diverse image representations.

## 6 Conclusions and Future Work

### 6.1 Conclusions

Using readily available 3D color maps generated from SLAM missions or TLS scanners, we can repurpose these maps for localization tasks. In this paper, we introduce a localization system specifically designed to localize a single camera image within 3D color maps by sampling and generating a synthetic image database. We first demonstrated a pipeline to construct 3D prior maps using three distinct representations: point clouds, meshes, and NeRF. Each representation is then used to synthesize RGB and depth images. Following this, we proposed a strategy to define the set of rendering poses that optimize the creation of a representative 3D map while retaining a minimal set of database images. The visual localization pipeline can estimate the camera pose of a query image through a retrieval and matching process, leveraging learning-based descriptors and feature detectors. We conducted a comprehensive analysis of localization performance across these representations and discussed their respective merits. Additionally, we offered a benchmark comparison with two purely vision-based localization systems to situate our results within the wider field of visual localization. Notably, both point cloud and mesh representations achieve a localization accuracy of 55 % for query images, while the NeRF representation surpasses these, achieving a localization rate of 72 %.

### 6.2 Limitation and Future Work

#### 6.2.1 Scene Change

While we have studied the effect of scene changes within the context of a 3D color map, we acknowledge that there would likely be a performance decrease during gradual scene transitions over a

period of months and years. In future research, we aim to devise a detection algorithm capable of promptly identifying these changes in real time, thus facilitating the identification of out-of-date regions of the map. Additionally, in scenarios where the device is equipped with a lidar sensor, we propose to update the existing map following a remapping process to ensure its continued accuracy and relevance.

### 6.2.2 Synthetic Image

All three map representations encounter specific challenges in accurately rendering scenes with minimal texture details. This difficulty is particularly notable in 3D mapping and reconstruction, where effectively capturing the RGB texture details of ground and vegetation proves challenging. Furthermore, during the online localization of our system, images that predominantly contain low-texture objects often yield insufficient features for matching, resulting in inaccurate pose estimations. To address this, we could prune the database images during online localization to retain only the most useful images and synthesizing new ones on-the-fly as needed.

## References

- [1] Sarlin, P.-E., Cadena, C., Siegwart, R., Dymczyk, M.: From Coarse to Fine: Robust hierarchical localization at large scale. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR) (2019). <https://doi.org/10.1109/CVPR.2019.01300>
- [2] Sattler, T., Leibe, B., Kobbelt, L.: Efficient & effective prioritized matching for large-scale image-based localization. IEEE Trans. Pattern Anal. Mach. Intell. **39**(9), 1744–1756 (2017) <https://doi.org/10.1109/TPAMI.2016.2611662>
- [3] Zeisl, B., Sattler, T., Pollefeys, M.: Camera pose voting for large-scale image-based localization. In: Intl. Conf. on Computer Vision (ICCV), pp. 2704–2712 (2015). <https://doi.org/10.1109/ICCV.2015.310>
- [4] Kim, G., Kim, A.: Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud map. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 4802–4809 (2018). <https://doi.org/10.1109/IROS.2018.8593953>
- [5] Vidanapathirana, K., Ramezani, M., Moghadam, P., Sridharan, S., Fookes, C.: LoGG3D-Net: Locally guided global descriptor learning for 3D place recognition. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 2215–2221 (2022). <https://doi.org/10.1109/ICRA46639.2022.9811753>
- [6] Uy, M.A., Lee, G.H.: PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR) (2018). <https://doi.org/10.1109/CVPR.2018.00470>
- [7] Wolcott, R.W., Eustice, R.M.: Visual localization within lidar maps for automated urban driving. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 176–183 (2014). <https://doi.org/10.1109/IROS.2014.6942558>
- [8] Pascoe, G., Maddern, W., Newman, P.: Direct visual localisation and calibration for road vehicles in changing city environments. In: IEEE Intl. Conf. on Computer Vision Workshop (ICCVW), pp. 98–105 (2015). <https://doi.org/10.1109/ICCVW.2015.23>
- [9] Gehring, C., Fankhauser, P., Isler, L., Diethelm, R., Bachmann, S., Potz, M., Gerstenberg, L., Hutter, M.: ANYmal in the Field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot. In: Field and Service Robotics, pp. 247–260. Springer, Singapore (2021). [https://doi.org/10.1007/978-981-15-9460-1\\_18](https://doi.org/10.1007/978-981-15-9460-1_18)
- [10] Besl, P.J., McKay, N.D.: A method for registration of 3D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992) <https://doi.org/10.1109/34.121791>
- [11] Schönberger, J.L., Frahm, J.-M.: Structure-from-motion revisited. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR) (2016). <https://doi.org/10.1109/CVPR.2016.445>

- [12] Yin, H., Lin, Z., Yeoh, J.K.W.: Semantic localization on BIM-generated maps using a 3D lidar sensor. *Automation in Construction* **146**, 104641 (2023) <https://doi.org/10.1016/j.autcon.2022.104641>
- [13] Liu, J., Nie, Q., Liu, Y., Wang, C.: NeRF-Loc: Visual localization with conditional neural radiance field. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 9385–9392 (2023). <https://doi.org/10.1109/ICRA48891.2023.10161420>
- [14] Matsuki, H., Murai, R., Kelly, P.H.J., Davison, A.J.: Gaussian splatting SLAM. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR), pp. 18039–18048 (2024). <https://doi.org/10.1109/CVPR52733.2024.01708>
- [15] Cummins, M., Newman, P.: FAB-MAP: Probabilistic localization and mapping in the space of appearance. *Intl. J. of Robot. Res.* **27**(6), 647–665 (2008) <https://doi.org/10.1177/0278364908090961>
- [16] Galvez-López, D., Tardos, J.D.: Bags of binary words for fast place recognition in image sequences. *IEEE Trans. Robotics* **28**(5), 1188–1197 (2012) <https://doi.org/10.1109/TRO.2012.2197158>
- [17] Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **40**(6), 1437–1451 (2018) <https://doi.org/10.1109/TPAMI.2017.2711011>
- [18] Hausler, S., Garg, S., Xu, M., Milford, M., Fischer, T.: Patch-NetVLAD: Multi-scale fusion of locally-global descriptors for place recognition. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR), pp. 14141–14152 (2021). <https://doi.org/10.1109/CVPR46437.2021.01392>
- [19] Berton, G., Trivigno, G., Caputo, B., Masone, C.: EigenPlaces: Training viewpoint robust models for visual place recognition. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR), pp. 11080–11090 (2023). <https://doi.org/10.1109/ICCV51070.2023.01017>
- [20] Ali-bey, A., Chaib-draa, B., Giguère, P.: MixVPR: Feature mixing for visual place recognition. In: IEEE/CVF Winter Conf. on Applications of Computer Vision, pp. 2998–3007 (2023)
- [21] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Intl. J. of Computer Vision* **60**, 91–110 (2004) <https://doi.org/10.1023/B:VISI.0000029664.99615.94>
- [22] DeTone, D., Malisiewicz, T., Rabinovich, A.: SuperPoint: Self-supervised interest point detection and description. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 224–236 (2018). <https://doi.org/10.1109/CVPRW.2018.00060>
- [23] Revaud, J., Weinzaepfel, P., Souza, C.R., Humenberger, M.: R2D2: Repeatable and reliable detector and descriptor. In: Intl. Conf. on Neural Information Processing Systems (NeurIPS). Curran Associates Inc., Red Hook, NY, USA (2019)
- [24] Panek, V., Kukelova, Z., Sattler, T.: MeshLoc: Mesh-based visual localization. In: Eur. Conf. on Computer Vision (ECCV) (2022). [https://doi.org/10.1007/978-3-031-20047-2\\_34](https://doi.org/10.1007/978-3-031-20047-2_34)
- [25] Xue, F., Budvytis, I., Reino, D.O., Cipolla, R.: Efficient large-scale localization by global instance recognition. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR), pp. 17327–17336 (2022). <https://doi.org/10.1109/CVPR52688.2022.01683>
- [26] Trivigno, G., Masone, C., Caputo, B., Sattler, T.: The Unreasonable Effectiveness of Pre-Trained Features for Camera Pose Refinement (2024). <https://arxiv.org/abs/2404.10438>
- [27] Zhang, Z., Sattler, T., Scaramuzza, D.: Reference pose generation for long-term visual localization via learned features and view synthesis. *ijcv* **129**(4), 821–844 (2021) <https://doi.org/10.1007/s11862-021-01378-w>

<https://doi.org/10.1007/s11263-020-01399-8>

- [28] Himstedt, M., Frost, J., Hellbach, S., Böhme, H.-J., Maehle, E.: Large scale place recognition in 2D lidar scans using geometrical landmark relations. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 5030–5035 (2014). <https://doi.org/10.1109/IROS.2014.6943277>
- [29] He, L., Wang, X., Zhang, H.: M2DP: A novel 3D point cloud descriptor and its application in loop closure detection. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 231–237 (2016). <https://doi.org/10.1109/IROS.2016.7759060>
- [30] Kong, X., Yang, X., Zhai, G., Zhao, X., Zeng, X., Wang, M., Liu, Y., Li, W., Wen, F.: Semantic graph based place recognition for 3D point clouds. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 8216–8223 (2020). <https://doi.org/10.1109/IROS45743.2020.9341060>
- [31] Dubé, R., Cramariuc, A., Dugas, D., Sommer, H., Dymczyk, M., Nieto, J., Siegwart, R., Cadena, C.: Segmap: Segment-based mapping and localization using data-driven descriptors. *Intl. J. of Robot. Res.* **39**(2–3), 339–355 (2020) <https://doi.org/10.1177/0278364919863090>
- [32] Vidanapathirana, K., Moghadam, P., Harwood, B., Zhao, M., Sridharan, S., Fookes, C.: Locus: Lidar-based place recognition using spatiotemporal higher-order pooling. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 5075–5081 (2020). <https://doi.org/10.1109/ICRA48506.2021.9560915>
- [33] Zhang, L., Tejaswi Digumarti, S., Tinchev, G., Fallon, M.: InstaLoc: One-shot global lidar localisation in indoor environments through instance learning. In: Robotics: Science and Systems (RSS) (2023). <https://doi.org/10.15607/RSS.2023.XIX.070>
- [34] Ratz, S., Dymczyk, M., Siegwart, R.Y., Dubé, R.: OneShot Global Localization: Instant lidar-visual pose estimation. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 5415–5421 (2020). <https://doi.org/10.1109/ICRA40945.2020.9197458>
- [35] Bernreiter, L., Ott, L., Nieto, J., Siegwart, R., Cadena, C.: Spherical multi-modal place recognition for heterogeneous sensor systems. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 1743–1750 (2021). <https://doi.org/10.1109/ICRA48506.2021.9561078>. IEEE
- [36] Lai, H., Yin, P., Scherer, S.: AdaFusion: Visual-lidar fusion with adaptive weights for place recognition. *IEEE Robot. Autom. Lett. (RA-L)* **7**(4), 12038–12045 (2022) <https://doi.org/10.1109/LRA.2022.3210880>
- [37] Lee, A.J., Song, S., Lim, H., Lee, W., Myung, H.: (LC)<sup>2</sup>: Lidar-camera loop constraints for cross-modal place recognition. *IEEE Robot. Autom. Lett. (RA-L)* **8**(6) (2023) <https://doi.org/10.1109/LRA.2023.3268848>
- [38] Borges, P., Zlot, R., Bosse, M., Nuske, S., Tews, A.: Vision-based localization using an edge map extracted from 3D laser range data. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 4902–4909 (2010). <https://doi.org/10.1109/ROBOT.2010.5509517>
- [39] Yu, H., Zhen, W., Yang, W., Zhang, J., Scherer, S.: Monocular camera localization in prior lidar maps with 2D-3D line correspondences. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), 4588–4594 (2020) <https://doi.org/10.1109/IROS45743.2020.9341690>
- [40] Zuo, X., Ye, W., Yang, Y., Zheng, R., Vidal-Calleja, T., Huang, G., Liu, Y.: Multimodal localization: Stereo over LiDAR map. *J. Field Robot.* **37**(6), 1003–1026 (2020) <https://doi.org/10.1002/ROB.21936>
- [41] Liang, S., Zhang, Y., Tian, R., Zhu, D., Yang, L., Cao, Z.: SemLoc: Accurate and robust visual localization with semantic and structural constraints from prior maps. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 4135–4141 (2022). <https://doi.org/10.1109/ICRA46639.2022.9811925>

- [42] Xu, W., Cai, Y., He, D., Lin, J., Zhang, F.: FAST-LIO2: Fast direct lidar-inertial odometry. *IEEE Trans. Robotics* **38**(4), 2053–2073 (2022). <https://doi.org/10.1109/TRO.2022.3141876>
- [43] Zhou, Q.-Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018) <https://doi.org/10.48550/ARXIV.1801.09847>
- [44] Arvo, J.: Graphics Gems II. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1991)
- [45] Botsch, M., Kobelt, L., Pauly, M., Alliez, P., Lévy, B.: Polygon Mesh Processing. A K Peters, Natick, Massachusetts (2010). <https://doi.org/10.1201/b10688>
- [46] Lin, J., Yuan, C., Cai, Y., Li, H., Ren, Y., Zou, Y., Hong, X., Zhang, F.: ImMesh: An immediate lidar localization and meshing framework. *IEEE Trans. Robotics* (2023) <https://doi.org/10.1109/TRO.2023.3321227>
- [47] Lin, J., Zhang, F.: R<sup>3</sup>LIVE++: a robust, real-time, radiance reconstruction package with a tightly-coupled lidar-inertial-visual state estimator. arXiv preprint arXiv:2209.03666 (2022) <https://doi.org/10.48550/ARXIV.2209.03666>
- [48] 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: Eur. Conf. on Computer Vision (ECCV) (2020). [https://doi.org/10.1007/978-3-030-58452-8\\_24](https://doi.org/10.1007/978-3-030-58452-8_24)
- [49] Max, N.: Optical models for direct volume rendering. *IEEE Trans. on Visualization and Computer Graphics* **1**(2), 99–108 (1995) <https://doi.org/10.1109/2945.468400>
- [50] Kajiya, J.T., Von Herzen, B.P.: Ray tracing volume densities. *SIGGRAPH* **18**(3), 165–174 (1984) <https://doi.org/10.1145/800031.808594>
- [51] Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxtels: Radiance fields without neural networks. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR), pp. 5501–5510 (2022). <https://doi.org/10.1109/CVPR52688.2022.00542>
- [52] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. In: SIGGRAPH (2022). <https://doi.org/10.1145/3528223.3530127>
- [53] Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. on Graphics* **42**(4) (2023) <https://doi.org/10.1145/3592433>
- [54] Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., et al.: Nerfstudio: A modular framework for neural radiance field development. In: SIGGRAPH (2023). <https://doi.org/10.1145/3588432.3591516>
- [55] Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. *IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR)* (2022) <https://doi.org/10.1109/CVPR52688.2022.00539>
- [56] 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: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR) (2021). <https://doi.org/10.1109/CVPR46437.2021.00713>
- [57] Tao, Y., Bhalgat, Y., Fu, L.F.T., Mattamala, M., Chebrolu, N., Fallon, M.: SiLVR: Scalable lidar-visual reconstruction with neural radiance fields for robotic inspection. In: IEEE Intl. Conf. on Robotics and Automation (ICRA) (2024). <https://doi.org/10.1109/ICRA57147.2024.10611278>
- [58] Deng, K., Liu, A., Zhu, J.-Y., Ramanan, D.: Depth-supervised nerf: Fewer views and faster training for free. In: IEEE Intl. Conf.

- Computer Vision and Pattern Recognition (CVPR), pp. 12882–12891 (2022). <https://doi.org/10.1109/CVPR52688.2022.01254>
- [59] Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A.: MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. Intl. Conf. on Neural Information Processing Systems (NeurIPS) (2022)
- [60] Von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing **17**, 395–416 (2007) <https://doi.org/10.1007/S11222-007-9033-Z>
- [61] Sarlin, P.-E., DeTone, D., Malisiewicz, T., Rabinovich, A.: SuperGlue: Learning feature matching with graph neural networks. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR) (2020). <https://doi.org/10.1109/CVPR42600.2020.00499>
- [62] Marchand, E., Uchiyama, H., Spindler, F.: Pose estimation for augmented reality: A hands-on survey. IEEE Trans. on Visualization and Computer Graphics **22**(12), 2633–2651 (2016) <https://doi.org/10.1109/TVCG.2015.2513408>
- [63] Torii, A., Sivic, J., Okutomi, M., Pajdla, T.: Visual place recognition with repetitive structures. IEEE Trans. Pattern Anal. Mach. Intell. (2015) <https://doi.org/10.1109/TPAMI.2015.2409868>
- [64] Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR) (2017). <https://doi.org/10.1109/CVPR.2017.261>
- [65] Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR) (2018). <https://doi.org/10.1109/CVPR.2018.00218>
- [66] Fu, L.F.T., Chebrolu, N., Fallon, M.: Extrinsic calibration of camera to lidar using a differentiable checkerboard model. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), pp. 1825–1831 (2023). <https://doi.org/10.1109/IROS55552.2023.10341781>
- [67] Zhang, L., Camurri, M., Wisth, D., Fallon, M.: Multi-Camera LiDAR Inertial Extension to the Newer College Dataset. arXiv:2112.08854 (2021) <https://doi.org/10.48550/ARXIV.2112.08854>
- [68] Alcantarilla, P.F., Nuevo, J., Bartoli, A.: Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: British Machine Vision Conf. (BMVC) (2013). <https://doi.org/10.5244/C.27.13>
- [69] Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T.: D2-Net: A trainable cnn for joint detection and description of local features. In: IEEE Intl. Conf. Computer Vision and Pattern Recognition (CVPR) (2019). <https://doi.org/10.1109/CVPR.2019.00828>