論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Large-Scale Visual Localization Based on Image Retrieval and Local 3D Reconstruction
著者(和文)	田平創
Author(English)	Hajime Taira
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第11421号, 授与年月日:2020年3月26日, 学位の種別:課程博士, 審査員:奥冨 正敏,蜂屋 弘之,倉林 大輔,大山 真司,塚越 秀行
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第11421号, Conferred date:2020/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Doctor Thesis 2019

Large-Scale Visual Localization Based on Image Retrieval and Local 3D Reconstruction

Tokyo Institute of Technology
Dept. of Systems and Control Engineering, School of Engineering
Hajime Taira

December, 2019

Supervisor: Masatoshi Okutomi

Abstract

Finding an accurate location and direction of a robot in the 3D real world is a key ability of robotic intelligent system and autonomous navigation. One approach for solving the localization problem is to use a camera attached to the target device to estimate the current pose of the target (visual localization). Because of the wide spreading of image data and internet sharing, the scale (reference scenes) for the problem becomes larger in both indoor (including multiple rooms, floors, and buildings) and outdoor (including the whole cities, regions) scenarios. At the same time, new applications, such as Mixed Reality (MR) including Augmented Reality (AR), often require an accurate 6-Degree-of-Freedom (6DoF) camera pose of the input image. Previous visual localization approaches that require a 3D database capturing the whole scene to estimate the camera pose are impractical or sometimes infeasible for such large-scale visual localization problem, regarding memory consumption and maintenance/updates of the database.

In this thesis, we introduce a visual localization pipeline designed for large-scale environments. In contrast to other approaches using the large 3D database, we first seek the relevant locations of an input image using an image database, which allows scaling to the larger scene. We next compute an accurate 6DoF camera pose using local geometric information around the relevant location. Contributions of this thesis can be divided into three parts; 1) We propose two feature matching algorithms for local image features. They are designed for the general purpose of wide-baseline image matching, but can also be beneficial for database construction and camera pose estimation for visual localization. 2) We construct a large-scale image dataset in an urban city-scale scene while preparing precise 6DoF camera poses of input images to evaluate the performance of visual localization. Using this new dataset, we evaluate state-of-the-art approaches for visual localization and show the large-scale 3D model is actually not necessary for accurate camera pose estimation in a large-scale scene. 3) We also evaluate our pipeline on a large-scale indoor scenario. To deal with several difficulties raised in indoor scenes such as weakly textured appearances and repetitive structures, we propose a new algorithm verifying the estimated camera pose using a pre-captured local 3D point cloud. Our method achieves state-of-the-art performance on our indoor dataset. Altogether, we tackled for scaling visual localization problem to much larger environments. We confirmed our approach effectively ad-

dresses the issue based on efficient image retrieval and local 3D reconstruction. Also, two new datasets we constructed through the thesis depict several challenging situations for visual localization induced in large-scale scenarios and will be beneficial for further researches in this area.

Acknowledgement

First of all, the author would like to express the sincerest thank to the supervisor, Professor Masatoshi Okutomi, for providing the opportunities and environments to focus on this research topic. The author would also like to thank Professor Torii for his kindful and suggestive advice related to the project.

Other members in Okutomi-Tanaka laboratory, including Professor Masayuki Tanaka, Doctor Yusuke Monno, and master/bachelor students, always allowed the author to spend joyful and constructive times in the laboratory. The author would like to thank all of the colleagues for their understandings and kindnesses.

This project has been supported by many collaborators, from Chalmers University of Technology, Inria, Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, and ETH Zürich. The author would like to thank Professor Torsten Sattler, Professor Josef Sivic, Professor Tomas Pajdla, Professor Marc Pollefeys, Doctor Mircea Cimpoi, Doctor Jiri Sedlar, and Ignacio Rocco, for having insightful discussions and many supports, during these beautiful times to co-author. The author would also like to express the deep appreciation to Professor Yasutaka Furukawa for his arrangement to capture query photographs at Washington University in St. Louis, for composing a new dataset that takes on an essential role of this thesis.

Finally, the author would like to thank his families, Masato, Sachiko, Riko, Niko, and Koko, in all aspects.

This work was supported by JSPS KAKENHI Grant Number 17J05908.

Contents

Chapter 1

Introduction

1.1 Background

Finding accurate location and direction of a robot in the 3D real world is a key ability of robotic intelligent system [1], *e.g.*, autonomous driving in an urban city. Also, localizing a mobile device can benefit recent Mixed Reality (MR) applications, including Augmented Reality (AR) [2, 3], *e.g.*, for a smartphone, Google Glass or any other types of a wearable device.

Fig. 1.1 shows several known approaches for the localization problem. One common approach to localize the target device is Global Positioning System (GPS) that determines the position by triangulating signals from multiple satellites. Similarly, Wifi signals from calibrated transmitters can be used for localization mainly in indoor environments [4–8]. In addition, Inertial Measurement Unit (IMU) is often used to find an accurate pose of the device [9, 10]. Other approaches use observation of the environments surrounding the device, *e.g.*, color images from a camera, 3D points captured by LIDAR. One of the popular fields here is Simultaneous Localization and Mapping (SLAM) that incrementally estimates current 6-Degree-of-Freedom (6DoF) pose of the device and 3D environments, assuming a sequence of the input images or 3D points from LIDAR [11–13]. In contrast, **visual localization** approach assumes a single image input and estimates the 6DoF pose of the target. Assuming a database that represents a known 3D space, *e.g.*, image collection that has taken from known viewpoints, one can estimate the current location by registering the query (input) image to the database. As well as the use of other modality sensors like GPS and IMU, which are sometimes unavailable or provide erroneous measures, this purely image-based approach is also highly relevant for applications above, and is beneficial regarding device costs.

In the last decade, the database which is available for visual localization become larger [14–22] because of wide spread of image data and internet sharing, *e.g.*, user photos uploaded to Flickr, and city or region scale image database on Google Street View. Furthermore, thanks to recent multi-modal sensors, additional information that can contribute to find 6DoF pose

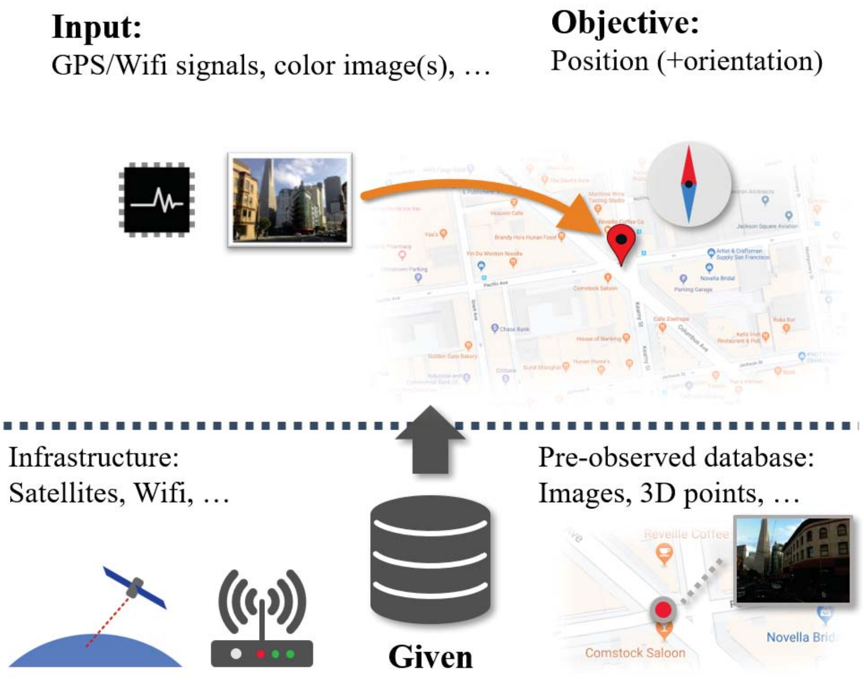


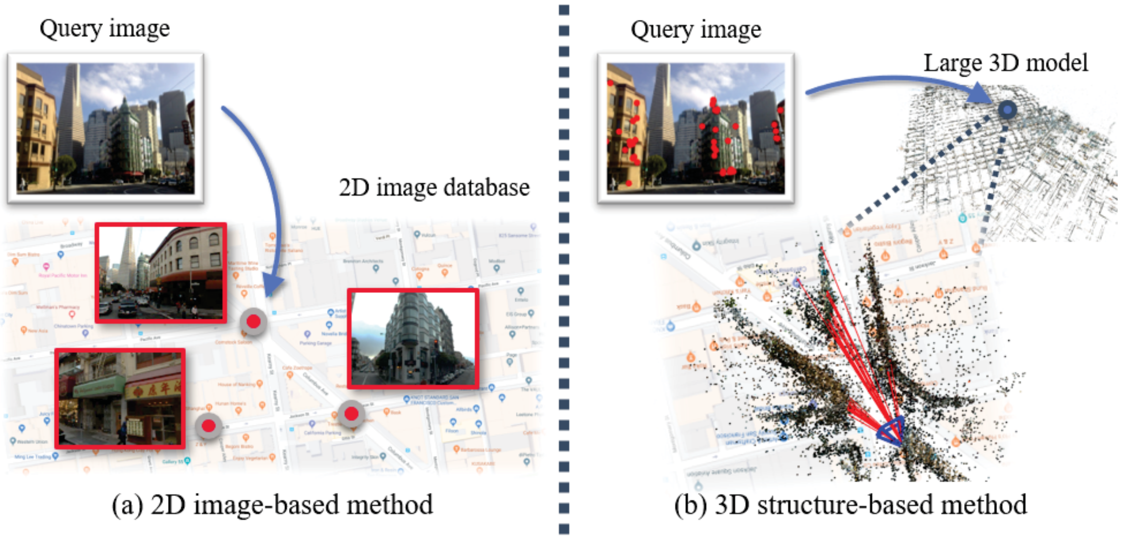
Figure 1.1: **Localization problem.** The goal for localization is to find an accurate location (and orientation) for an input consisting of sensor signals or observations for the surrounding environments, *e.g.*, GPS signals, IMU information, and color images. As a key of the location, sensor signal-based localization generally requires pre-constructed infrastructures, *e.g.*, GPS satellites, calibrated Wifi. On the other hand, visual localization approaches determines the position and orientation of the camera using pre-observed database in the format of image collection or 3D structures.

of the camera, *e.g.*, accurate and dense 3D map captured by a laser range scanner, become also be available. However, large-scale database often includes many challenging scenarios for visual localization, *e.g.*, visually similar places that obviously hurt the uniqueness of the images, and less textured or highly symmetric scenes that makes queries difficult to be localized to the correct positions (Typical image examples are also shown in Fig. 1.5). At the same time, effective usage of additional information such as 3D model along with image database is not trivial, thus is a opening issue to be investigated.

1.2 Large-scale visual localization

1.2.1 Overview

As illustrated in Fig. 1.2, existing visual localization approaches can mainly be divided into two streams; 2D image-based methods and 3D structure-based methods, both assume a pre-collected



(a) 2D image-based method

(b) 3D structure-based method

Figure 1.2: **Two main approaches for visual localization**. (a) 2D image-based methods cast the localization problem as the ranking among the geotagged database images (red highlighted images). For a given query image (Top left image), they perform image retrieval and approximate the query location by the location of the most relevant database image. (b) On the other hand, 3D structure-based methods assume a large 3D model (point cloud) depicting the whole scene and directly register the input image into the model. Registration is done by establishing 2D-3D correspondences between the query and the 3D model (red dots in the image and red lines in the 3D model) and estimating 6DoF camera pose of the query (blue cone in the 3D model).

Table 1.1: System-level summary of visual localization approaches.

	2D image-based localization	3D structure-based localization
Database	Database of geotagged images	3D points with associated image descriptors
Approach	Image retrieval	Descriptor matching followed by pose estimation
Output	Set of database images related to query, coarse position estimate	6DoF camera pose of the query image (position and orientation)
Advantage	Easy to maintain / update database	Directly provides pose estimates
Disadvantage	Requires extra post-processing to obtain 6DoF poses	Needs to construct a consistent 3D model

database for the target environment but in different styles (also see Tab. 1.1). 2D approaches (Fig. 1.2 (a)) cast the localization problem as a ranking problem among the geometrically annotated (geotagged) image database and employ image-based similarity measurements such as image retrieval and feature matching. On the other hand, 3D methods (Fig. 1.2 (b)) assume a 3D model of the scene as the database and estimate a 6DoF camera pose of the input image regarding the model. This section summarizes key components of these approaches, feature matching (Sec. 1.2.2), image retrieval (Sec. 1.2.3), and camera pose estimation (Sec. 1.2.4), together with the database construction process (Sec. 1.2.5), which is another fundamental part of both methods.

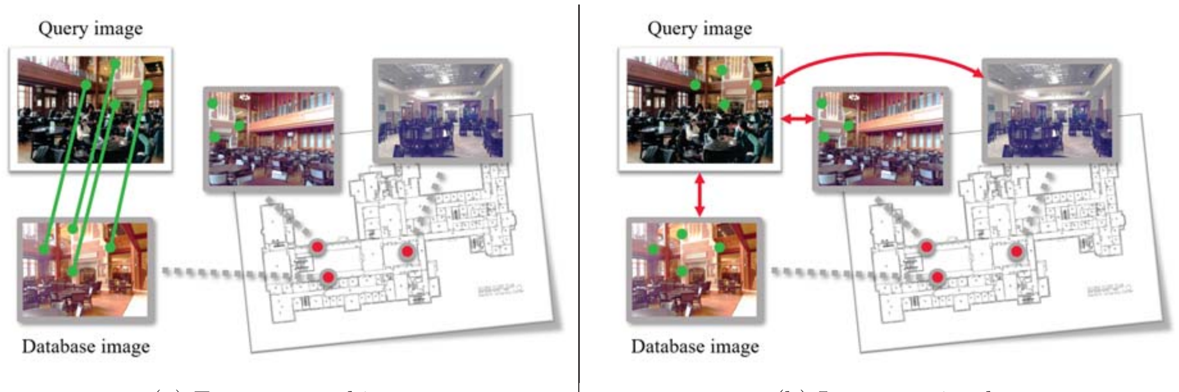
1.2.2 Feature matching

Visual localization is often be treated as image ranking problem, *i.e.*, assuming a database of geotagged images $\{\mathcal{I}_{DB}\}$, the (approximated) location of an input image $\mathcal{I}_Q$ can be presented by the geotag of a database image $\hat{\mathcal{I}}_{DB}$ that is visually similar to the query. One of the simplest approaches for ranking the database images is to perform *feature matching* M that finds a set of local correspondences between $\mathcal{I}_Q$ and each $\mathcal{I}_{DB}$ and to count the number of the matches n (illustrated in Fig. 1.3 (a)).

$$M(\mathcal{I}_Q, \mathcal{I}_{DB}) = \{\mathcal{I}_Q(\boldsymbol{x}_i), \mathcal{I}_{DB}(\boldsymbol{y}_i)\}_i^n \tag{1.1}$$

$\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ represent the i–th corresponding keypoint found in $\mathcal{I}_Q$ and $\mathcal{I}_{DB}$, respectively. $\hat{\mathcal{I}}_{DB}$ is then selected as the most matched images. Also, establishing local correspondences between given images is a fundamental step in many computer vision tasks, including Structure from Motion (SfM) [23–25], image retrieval [26, 27], and camera pose estimation [28–30], which are used to construct an image database for visual localization.

Most algorithms of feature matching compose of three steps: keypoint detection, feature description, and matching, where each of them basically relies on the consistency (invariance) of local regions in the images. The first step extracts a set of keypoint from each of $\mathcal{I}_Q$ and $\mathcal{I}_{DB}$ that potentially be matched to the ones extracted from the other image regarding the characteristic regions in the image, *e.g.*, edges and corners [31–35]. Each keypoint (region) is then described as a feature vector that represents the image point in the high-dimensional feature space. It is usually done using the image intensities in the local patches or regions around the keypoint, *e.g.*, computing image gradients in the image patch [32]. Finally, feature matching is performed for each feature in the feature space by searching the nearest feature extracted from the other image. Those potential correspondences are verified and filtered by fitting any geometrical model, *i.e.*, via Random Sample Consensus (RANSAC) [28], for the tasks that require an accurate set of matches, *e.g.*, SfM [23–25].



(a) Feature matching

(b) Image retrieval

Figure 1.3: **Place recognition using image database**. For a given query image, visual localization has often been performed by seeking the most relevant database image to the query. (a) One of the simplest approaches for ranking the database images is to build local correspondences (green lines) between database and query images by feature matching. The number of matches can represent the relevance of each database image to the query. (b) On the other hand, image retrieval approaches provide a more efficient solution for ranking by measuring per-image similarities (red arrows) rather than measuring the similarity of each local region. Feature matching is often performed as a stricter ranking process on top of the image retrieval.

Challenges. Feature matching seeks image matches relying on the invariance of local regions in the images. However, this assumption often is violated when the viewpoint (image location) has significantly changed. Substantial view changes can cause appearance changes even in the local regions and lead keypoints miss-detection and feature variations, which prevent matching algorithms from finding accurate correspondences. At the same time, as partly shown in Fig. 1.3 (a), view changes between pre-captured database images and the query are actually inevitable regarding the difference in capturing times and camera devices [20].

1.2.3 Image retrieval

Rather than counting the number of local matches, image retrieval-based approaches [27,36–41] provide a more efficient solution for ranking database images w.r.t. the similarity to the input image (illustrated in Fig. 1.3 (b)). An encoder F converts an image to a feature vector that presents the image in the feature space. The most relevant database image $\hat{\mathcal{I}}_{DB}$ is selected as the nearest one in the space to the feature of query:

$$\hat{\mathcal{I}}_{DB} = \operatorname*{argmin}_{\mathcal{I}_{DB}} d(F(\mathcal{I}_Q), F(\mathcal{I}_{DB})) \tag{1.2}$$

where d is the distance function for high-dimensional feature vector, *e.g.*, the Euclid distance. The encoder F is usually composed of some efficient representations of a set of local features, such as Bug-of-Words (BoW) [42,43] and Vector of Locally Aggregated Descriptors (VLAD) [44, 45].

One general technique to improve the ranking is geometric verification [42] that employs feature matching as post-processing for image retrieval, for proving a geometric relationship between the query and the database image, rather than relying only on the per-image metric provided by image retrieval. For the set of top-retrieved N database images, it performs pairwise feature matching against the query image and verifies correspondences by fitting homography or any type of geometric model via RANSAC [28]. Then the database images are reranked regarding the number of inlier matches:

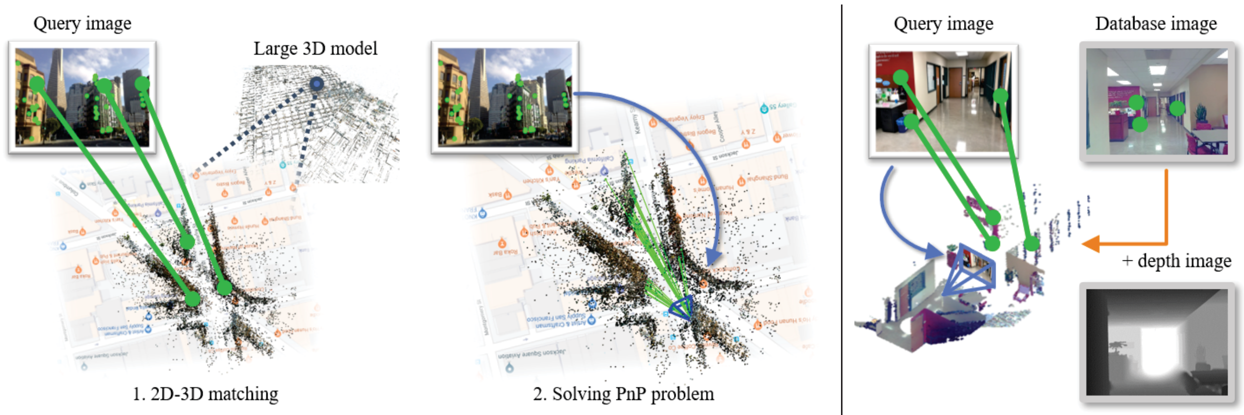
$$\hat{\mathcal{I}}_{DB} = \underset{\mathcal{I}_{DB} \in N}{\operatorname{argmax}} \|M(\mathcal{I}_Q, \mathcal{I}_{DB})\| \tag{1.3}$$

where $\|M(\mathcal{I}_Q, \mathcal{I}_{DB})\|$ is the number of matches, which is equal to n in Eq. (1.1).

This purely image-based ranking process carries a key technology of **2D image-based localization** approaches (Fig. 1.2). The location where the input image $\mathcal{I}_Q$ has been taken can be approximated by the location where $\hat{\mathcal{I}}_{DB}$ has been taken, *i.e.*, the geotag of the database image (summarized in Fig. 1.2 (a)).

Image-based localization methods are often adopted to place recognition tasks in urban scenarios [27, 36–40]. Assuming a large-scale (city-scale or more) image database annotated to a known city map, *i.e.*, a set of geotagged images, image retrieval provides an efficient location searching since it basically relies on a simple distance computation of the feature vectors (Eq. (1.2)). Also, it can naturally be scaled to a database expansion (an expansion of the target scene by storing new images) since it only requires to store geotags and feature vectors as the database, which allows manageable and maintenance-able database construction.

Challenges. Image retrieval-based localization methods provide only the relevant location of the image. If one can assume that a query image follows a spatial distribution quite similar to that of database images, *i.e.*, taken from the same street and from very similar viewpoint, the approximated location (the location of the database image) can present a sufficiently relevant position of the query (the bottom example in Fig. 1.5). This approximation, however, leads a limitation to the accuracy of the estimated locations when query image and database images have been captured from far distant spatial distribution, *e.g.*, assuming queries taken by users on pedestrians, whereas database images have been collected as street-view dataset captured by car-mounted cameras. The top examples in Fig. 1.5 shows a typical example of such situations. The query (a) and retrieved database image (b) certainly see the same object (building). Still, the structures in the image are large, and query captures it from a relatively far viewpoint, inducing significant spatial uncertainties of the estimated location. As a result, partly due to sparse distributions of the database images, image retrieval-based localization [27] gives a street-level inaccurate position. In addition, some applications of visual localization require an



(a) Camera pose estimation using an SfM model

(b) Camera pose estimation using an RGBD image

Figure 1.4: **Camera pose estimation using 3D database**. (a) Assuming a 3D point cloud obtained via SfM, 3D structure-based localization methods firstly build correspondences between image features and 3D points by feature matching. They then compute a 6DoF camera pose of the image (blue cone in the 3D model) by solving a PnP problem. (b) An RGBD image captured by a depth sensor can build another type of the 3D database. Matches between the query and the database image are translated as 2D-3D matches using the depth information and formulate the camera pose of the query.

accurate position and orientation (6DoF pose) of the query, rather than the approximated one. Additional post-processing for estimating the accurate camera pose is therefore required while it costs additional computation resources.

1.2.4 Camera pose estimation

Theoretically, the minimal solution of a 6DoF camera pose is provided by three correspondences between 2D image points (keypoints) and 3D scene points [28, 47] (Perspective-three-points (P3P) problem, which is in a series of Perspective-n-Point (PnP) problem). This pose estimation is usually done in incremental SfM pipeline [23–25] that estimates 3D scene points and camera poses using a set of images input. They first perform pairwise feature matching among the input images and initialize the 3D model (3D points and relative camera pose of a selected pair of images) by epipolar fitting and triangulation for local feature correspondences between the images. The model is then incrementally updated by registering other images also using local feature matches. A 2D image point match between a registered image and a new image can be interpreted to a 2D-3D correspondence if the feature in the registered image associates a 3D point. Subsequently, the registration can be robustly done by solving the P3P problem in the RANSAC [28, 29, 48] scheme. At the same time, new 3D points are added to the model via

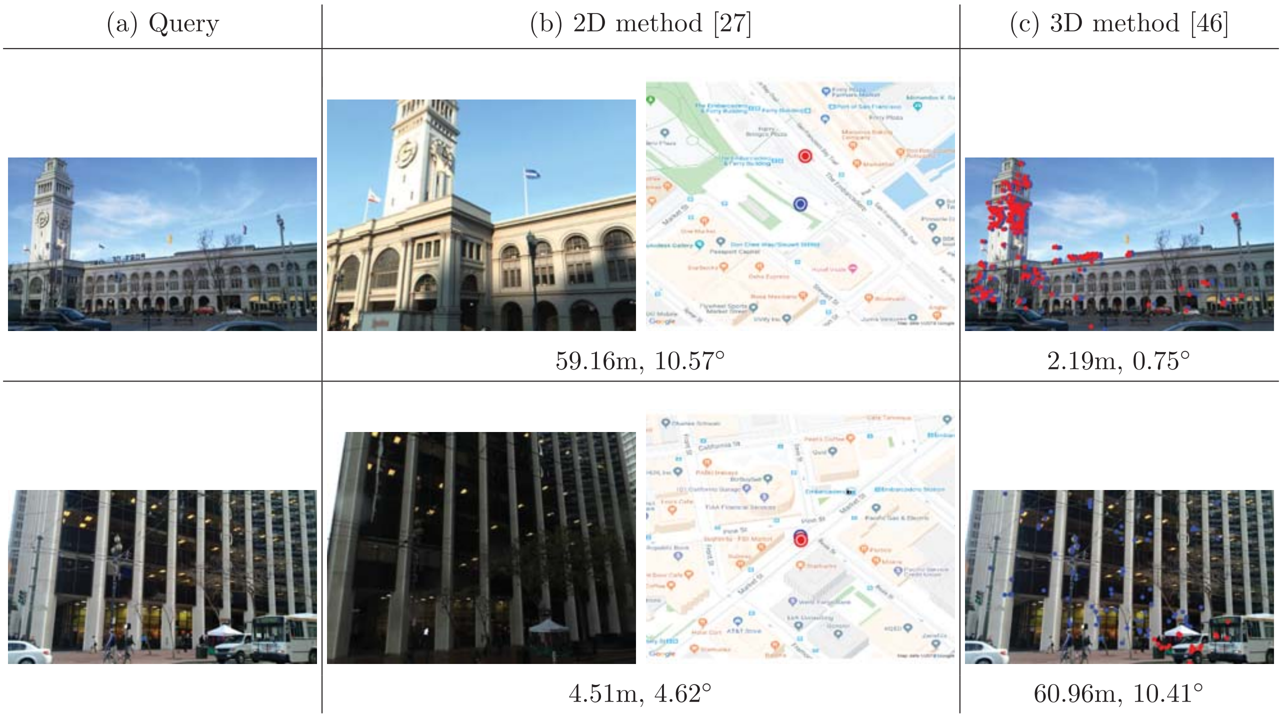


Figure 1.5: **Examples of challenging situations for visual localization in an outdoor urban environment [21]**. Each column shows: (a) Query image fed to 2D image-based [27] and 3D structure-based [46] localization methods. (b) The best matched database image selected by image retrieval [27] (left) and its location (red dot in right map), together with query ground-truth location (blue dot). (c) 3D points projected to the query with respect to an estimated camera pose obtained by [46] (red dots in the image) and its corresponding 2D image points (blue dots). Numbers in (b) and (c) present the positional and orientational errors of the camera pose obtained by each method.

point triangulation with respect to the relative camera poses and 2D matches, resulting in a large 3D point cloud at the end.

Similarly, assuming a pre-constructed database of 3D points, one can estimate a 6DoF camera pose of a query using the 3D point cloud. Several works [46,48–55] provide such solutions consisting of the other visual localization stream, **3D structure-based localization**, namely (illustrated in Fig. 1.2 (b)). These (hand-crafted) methods offer an accurate 6DoF camera pose, assuming a pre-constructed 3D point cloud depicting the street-level target scene. 3D points are usually constructed via SfM reconstruction using database images. As illustrated in Fig. 1.4 (a), the 6DoF camera pose of a given image can be estimated by collecting the correspondences between 2D local features and 3D points via feature matching and solving the PnP problem.

As well as the sparse 3D points obtained via SfM, a 3D scene can also be represented by an

RGB image together with pixel-wise depth information (RGBD image) captured by a multi-modal sensor such as LIDAR [22, 56–58]. As illustrated in Fig. 1.4 (b), dense 3D structures depicted by the RGBD image can benefit to the camera pose estimation tasks. Since each pixel of the RGBD image corresponds to a 3D coordinate, local feature matches between a database image and a query (RGB) image are translated as 2D-3D matches between the target 3D scene and the query. The 6DoF camera pose of a query can subsequently be computed by solving the PnP problem.

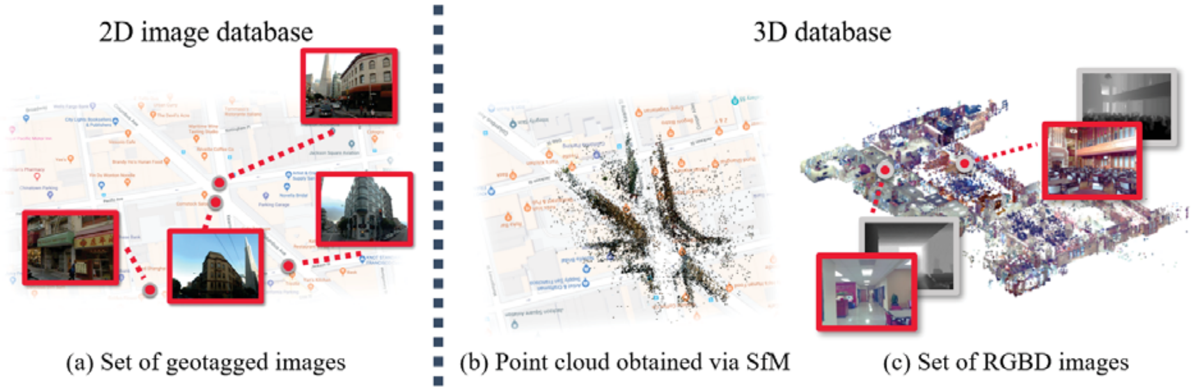
Alternatively, recent deep learning technology using the convolutional neural network (CNN) architectures allows directly predicting the 2D-3D correspondences [59, 60] or camera pose [55, 61, 62], regarding a known 3D database.

Challenges. While providing accurate 6DoF pose estimation in the query time, storing 3D points and its descriptors as a database for large-scale (city-scale) visual localization requires an extra memory consumption compared to image retrieval-based methods. CNN-based representation of a 3D database can potentially offer a more efficient database [61]. Yet, scaling CNN-based approaches to large-scale scenes is still an open challenge [60, 63, 64]. Also, updating a large 3D database requires more effort than constructing a 2D image database. The incompleteness of the 3D point cloud (or sparsity of the 3D points) can be a severe limitation for pose estimation since it relies on explicit 2D-3D matches, thus cannot be generalized to unknown scenes, *i.e.*, the scenes that are not associated by 3D database.

Another challenge, induced with the difficulties of local feature matching, is shown in an example at the bottom of Fig. 1.5. The query includes highly repetitive and symmetric appearances, *e.g.*, structures of a building, occlusions induced by moving objects, *e.g.*, cars, and reflective surfaces, *e.g.*, windows, which often appear in large-scale urban environments, and make local features difficult to be matched correctly. As a result, the 3D-based method (c) fails to find corresponding 3D scene points from the (large) 3D database and obtains a completely wrong camera pose.

1.2.5 Dataset

As mentioned in previous parts (Sec. 1.2.3 and Sec. 1.2.4) and summarized in Tab. 1.1, existing visual localization methods assume two types of the database regarding the required output; 2D image database with geotags for seeking a location where the query has been taken, and 3D database for estimating accurate 6DoF camera pose of the query. As illustrated in Fig. 1.6 (a), the 2D image database consists of RGB color images and its locations, *i.e.*, positions and orientations. Database capturing for urban scenes is often collected using a car-mounted camera and a GPS device [17, 37], resulting in a large-scale geotagged image database representing a city-scale visual localization scenario. The feature vector for image retrieval $F(\mathcal{I}_{DB})$ usually



(a) Set of geotagged images (b) Point cloud obtained via SfM (c) Set of RGBD images

Figure 1.6: **Three types of the database for visual localization**. The database for visual localization can mainly be divided into two types of domains: 2D image database and 3D database. The image database (a) for a place recognition task (Fig. 1.3) consists of a set of RGB images annotated by geotags regarding a known scene geometry, *i.e.*, map. On the other hand, the 3D database for camera pose estimation task (Fig. 1.4) depicts known 3D structures of the target scene, as a 3D point cloud obtained via SfM (b) or a set of RGBD images (c).

associates the images and scales the database to a reasonable size[1].

On the other hand, the SfM using database images can provide the 3D database for visual localization [23–25], which builds a 3D point cloud representing the scene structures (Fig. 1.6 (b)). Each 3D point is estimated based on feature matching between database images and triangulation [25], thus be associated with two or more local features in the database images. As the constructed 3D model is scale-indefinite, additional registration to a real-scale coordinate system such as the GPS coordinate system is performed as the post-processing [65].

Mainly in indoor scenarios, new multi-modal sensors such as LIDAR and laser range finder enable to capture scene structures directly, along with color information. They provide further accurate and dense 3D point cloud for the scene (Fig. 1.6 (c)), represented as an RGBD image [19, 56, 66–69]. Since one-time capturing of these devices can only depict a small local area, *e.g.*, a single room or section in a building, a large-scale RGBD database (depicting the whole floor) consists of a set of RGBD images, together with relative transformations to each other [56, 69], or registration to a known floor map [19].

Challenges. A general problem of the dataset for visual localization is the lack of input images for evaluation. To validate the performance of localization in practical situations such as autonomous navigation, one should use the query images representing several challenging situations, *e.g.*, captured differently from the database, in terms of viewpoints, devices, and timings. Also, accurate ground-truth for the query image is essential for evaluation. Constructing an SfM model using both database and query images can potentially provide an accurate

[1] For example, storing 4096-dimensional feature vectors for entire San Francisco city requires 17.4GB or less

6DoF camera poses for evaluation. However, this workaround still is limited for challenging scenarios, *e.g.*, weakly textured scene, because SfM is also challenging such scenarios due to the lack of local features, thus can provide an inaccurate reconstruction of the camera poses.

RGBD sensors can provide more accurate 3D representations than SfM 3D point clouds even in weakly textured scenes. They also serve denser 3D points since each pixel in the image associates a 3D scene point defined by depth information, whereas the SfM model represents the scene only by sparse local features. However, adapting those dense structures to visual localization is still an opening issue, which has been only partly addressed [70, 71].

1.3 Related works

1.3.1 Feature matching

Scale-invariant features. In the pioneering work related to local feature detection, scale-invariant keypoint detectors using Laplacian-of-Gaussian (LoG) or Difference-of-Gaussian (DoG) scale space representation were proposed [31,32], which were designed for seeking scale-invariant regions from multiple-scales representation of the image. Other approaches include Harris-Laplacian or Hessian-Laplacian [33], which measure the cornerness of the keypoint also in a multiple-scale images.

For describing the local regions/patches detected by the keypoint detector, SIFT [32] is a popular feature descriptor that computes histograms of gradients in the scaled patch around the keypoint. Variants include acceleration or approximation of SIFT [34, 35] or binarized description [72, 73] to improve memory efficiency. These approaches based on scaled patch representation are less effective when matching largely skewed image pairs, as they are not designed to handle them.

Affine covariant region estimation. One approach that can provide accurate matching in cases with relatively large viewpoint changes is to detect affine covariant regions. Harris (or Hessian)-Affine [33] are popular methods to estimate elliptical regions around the initial keypoints as the additional process for the scaled patch detectors, *i.e.*, post-processing of scale invariant regions detection by Harris-Laplacian or Hessian-Laplacian. Detected elliptical regions are then warped to circular regions to describe affine invariant feature vectors. These approaches still have limitations when the views are markedly different, *e.g.*seeing a plane from an extremely slanted view direction, because of the weakness of the initial keypoint detector.

View synthesis. Rather warping each local region, ASIFT [74] warps the whole image to match, resulting in an image matching framework that is robust to changes in viewpoints. It synthesizes multiple slanted views covering full affine space and extracts SIFT features from every synthesized image. The full affine simulation for both keypoint detection and feature description significantly improves the performance against large view changes. However, this brute-force method has low computational efficiency, because it requires full image synthesis and use a greater number of features. Moreover, it might generate numerous mismatches because of its heuristic feature generation. Several works have attempted to decrease its computational cost by balancing the image synthesis and matching performance. MODS [75] iteratively synthesizes warped images and performs feature matching. In each iteration, it checks the reliability of matches and stops iteration if a sufficient number of matches is obtained.

Metric learning. Feature matching is performed by computing similarities among feature descriptors. Numerous metrics have been proposed for improving discriminability and matchability. PCA-SIFT [76] is a widely known metric learning strategy for SIFT features. It projects feature descriptors onto a pre-trained low-dimensional space learned with principal component analysis (PCA), which fundamentally preserves the discriminability of features. To provide a strategy that is more specific to the image retrieval problem, [77] proposed descriptor projection learned by automatically generated training data. Recently, RootSIFT [26] has been widely used because of its simplicity and effectiveness. RootSIFT computes the Hellinger distance, which is accomplished by computing the Euclidean distances among L1 normalized and square rooted SIFT descriptors.

The Mahalanobis metric is used for adopting feature variations into similarity measurements [78]. [79] proposed Relevant Component Analysis (RCA), which learns a common covariance from all features in a training set. As a different approach to the feature matching problem, [80] proposed a method to learn the Mahalanobis metric for every patch without explicitly generating synthesized images.

1.3.2 Image retrieval-based localization

Image-based approaches (Left column in Tab. 1.1) model localization as an image retrieval problem. The location of a given query image is predicted by transferring the geotag of the most similar image retrieved from database [27, 36–41]. This approach scales to entire cities thanks to compact image descriptors and efficient indexing techniques [26, 44, 45, 81–85] and can be further improved by spatial re-ranking [42], informative feature selection [81, 86] or feature weighting [37,39,87,88]. Most of the above methods are based on image representations using sparsely sampled local invariant features. While these representations have been very successful, outdoor image-based localization has recently also been approached using *densely sampled* local descriptors [38] or (densely extracted) descriptors based on CNN [40, 41, 89, 90].

1.3.3 Camera pose estimation in large-scale environments

3D Structure-based localization methods (Right column in Tab. 1.1) assume that a scene is represented by a 3D model. The map is usually composed of a 3D point cloud constructed via Structure-from-Motion (SfM) [18], where each 3D point is associated with two or more local feature descriptors. The query pose is then obtained by feature matching and solving a Perspective-n-Point problem (PnP) [46, 48–54].

Descriptor matching quickly becomes a bottleneck and three (partially orthogonal) approaches exist to accelerate this stage: i) Prioritized search strategies [16, 49, 51] terminate correspondence search early on, ii) model compression schemes use only a subset of all 3D points [16, 48, 54], iii) retrieval-based approaches restrict matching to the 3D points visible in the top-ranked database images [46, 48, 52, 91, 92].

Alternatively, pose estimation can be formulated as a learning problem, where the goal is to train a regressor from the input RGB(D) space to camera pose parameters [55, 93] or 3D point positions [56, 59, 60, 64]. However, these methods do not yet achieve the same pose accuracy as structure-based methods on outdoor scenes [93]. In addition, scaling these methods to large-scale datasets is still an open challenge [60, 63]. Rather than reformulating the whole localization problem, several works replace each individual component of it with learned feature extractor or matcher. Traditional hand-crafted local features for feature matching can be replaced with learned alternatives [94–98]. There are also some works aiming to directly establish matches between images [99–101]. NetVLAD [40] provides a learned image representation for image retrieval that revisits the dense feature extraction and aggregation process with a CNN architecture. In general, those approaches are better generalized compared to learned direct pose estimator, and can potentially be incorporated to large-scale scenarios.

1.3.4 Dataset

In the last decade, image dataset for visual localization are mostly provided for urban environments [17, 20, 37]. San Franciso Landmark dataset [17] provides about 1.06M database images capturing the entire San Francisco city by car-mounted camera, and its accurate positions in the GPS coordinate system. They also include 803 query images taken by hand-held cameras, which have different properties (image resolution, camera intrinsics, and viewpoints) from database images. However, those image database only provide coarse locations or poor GPS positions of the queries. Therefore, evaluation on them are limited to place recognition task, *i.e.*, finding street-level location where the input image has been taken. One workaround to get an accurate 6DoF ground-truth camera pose for a query image is to construct a large 3D model via SfM using image database and to divide images to database and validation set. [16] provides two dataset, Dubrovnik6K and Rome16K, namely, which consists of internet photos and associating 3D point clouds. While providing an accurate camera pose for each query, those dataset are sometimes limited in terms of image distribution in the scene, *i.e.*, variety of the image appearances.

Compared to outdoor urban situations, visual localization in indoor environments has received less attentions in the last years, while the increased availability of new sensor devices such as laser range scanners and time-of-flight (ToF) sensors allows to get additional depth data besides RGB images [66–68,102–105], which gives more complete and accurate 3D database. Some datasets also also provide reference camera poses registered into the 3D point cloud [67,68,104], though their focus is not on localization. Datasets focused specifically on indoor localization [56, 57, 106] have so far captured fairly small spaces such as a single room (or a single floor at largest) and have been constructed from densely-captured sequences of RGBD images. More recent datasets [19,69] provide larger scale (multi-floor) indoor 3D maps containing RGBD images registered to a global floor map. However, they are designed for object retrieval, 3D reconstruction, or training deep-learning architectures. Most importantly, they do not contain query images taken from viewpoints far from database images, which are necessary for evaluating whether visual localization can generalize beyond the database images.

1.4 Thesis overview and contributions

Scenario. In this thesis, we introduce our research on visual localization for large-scale environments. From a single input image, we aim to find an accurate 6DoF camera pose using a large-scale database. While existing methods usually assume 3D point cloud capturing the whole of the huge target environment for pose estimation [46, 49, 50, 92], it requires huge memory consumption and is hard to maintain/update. Considering computational efficiency and potential expansions to further large-scale environments, we motivate to use a more efficient format of the database that provides sufficient information to determine the camera pose of the query.

Also, large-scale environments naturally lead to a diversity of input images, *e.g.*, changes in viewpoints, timings, and camera devices. Therefore, evaluation from the perspective of the robustness to the variations of the input images is essential to seek a general pipeline for visual localization in large-scale environments. At the same time, evaluating the accuracy of the estimated poses requires accurate camera poses given separately from the method, whereas that is sometimes impossible to observe ground-truth camera pose for practical input image, *e.g.*, the input from hand-held devices or past pictures archive. Therefore, the quantitative evaluation protocol for practical situations assuming general images input is one of the less focused areas in existing works [17, 20, 37, 56].

Contributions. Our contributions in this thesis can be divided into four parts (and be

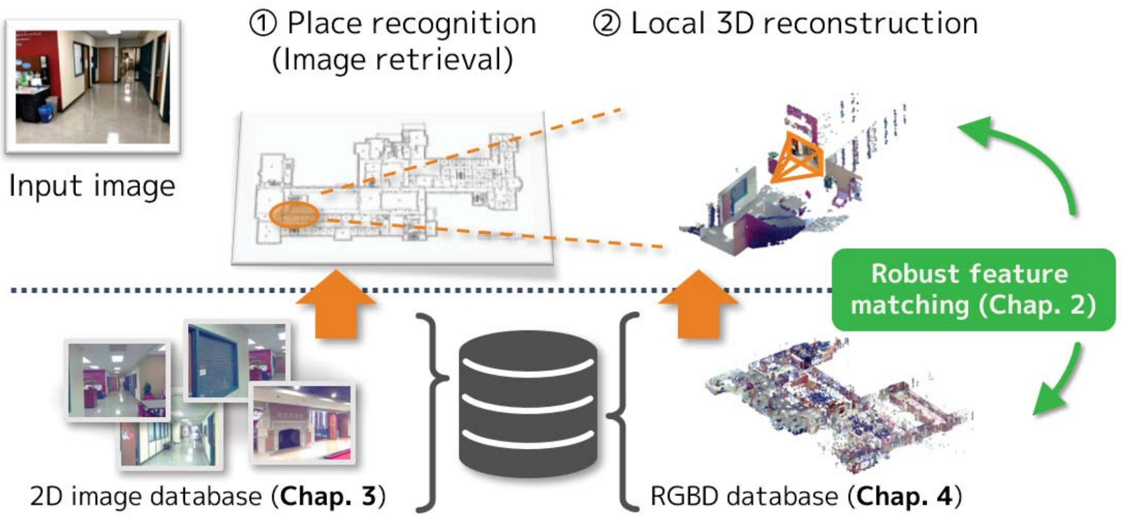

① Place recognition
(Image retrieval)
② Local 3D reconstruction
Input image
Robust feature matching (Chap. 2)
2D image database (Chap. 3)
RGBD database (Chap. 4)

Figure 1.7: **Proposed localization pipeline and contributions**. For a given input image, our localization pipeline firstly performs image retrieval using a 2D image database to determine the coarse relevant location(s) of the query. We next estimate 3D camera pose by reconstructing the local 3D scene around the query location using retrieved images. If available, a 3D database depicting accurate 3D structures of the scene, *e.g.*, RGBD database, provides a more accurate reconstruction than using only 2D images. We test this pipeline both in outdoor (Chap. 3) and indoor (Chap. 4) scenarios that assume a 2D image database and an RGBD database, respectively. In addition, we also propose robust feature matching algorithms (Chap. 2) that are beneficial to database construction and camera pose estimation for visual localization.

visually abstracted in Fig. 1.7);

1. We firstly propose **two feature matching methods** for general purposes but can benefit to visual localization and database construction. The first one is designed for feature matching of spherical panoramic images, which are often used to construct a large 3D model. While spherical panoramic image captures the scene efficiently by representing the entire spherical view in a single image, non-negligible distortion on the image is produced and affects the keypoint detection and local feature description for matching. Our method mitigates the distortion by synthesizing multiple undistorted images and achieves accurate feature matching between two panoramic images. We also show that the proposed matching algorithm can benefit 3D reconstruction using panoramic images. The second method is proposed for a pair of images that have been taken from far-distant viewpoints, *e.g.*, feature matching between database and query images. Wide viewpoint changes produce large appearance changes between images, which also hurts the performance of feature matching. We again exploit the use of synthetic images that imitate the potential appearance variation on the images due to viewpoint changes. Furthermore,

we show that the simple metric learning using the Mahalanobis distances can efficiently improve the performance of local feature matching.

2. We next focus on visual localization in large-scale urban environments. Because of the wide spreads of the 2D image database, outdoor environments are often represented by a set of database images (and its geometric locations). An efficient place recognition for an input image can be approximately done by image retrieval among those database images (image retrieval-based methods). On the other hand, aiming to find a 6DoF camera pose for the input image, existing 3D structure-based methods additionally require a large 3D model, *i.e.*, constructed via SfM, which subsequently depend upon much efforts to maintain/update the model. To tackle this trade-off between the efficiency and the quality of estimated locations, we propose **a general pipeline for visual localization in large-scale environments**, which takes advantage of both image retrieval-based and 3D structure-based approaches (illustrated in Fig. 1.7). Our pipeline firstly performs an efficient image retrieval-based localization to determine the relevant location of the query, *i.e.*, set of best-matched database images and its locations in the known map. We then estimate an accurate location (6DoF pose) of the query using those retrieved database images and the geometric properties around them, *e.g.*, feature correspondences, relative poses of database images in the 3D space, and 3D points (depth images), if available. While providing 6DoF camera pose as the 3D structure-based methods do, the proposed pipeline does not necessarily require a large 3D database that contains the whole referenced scene since it uses the 3D information only in the local (relatively small) area around the retrieved locations. Our method based purely on 2D images outperforms 3D-based methods that rely on a large 3D model, and computational burdens are negligible.

3. We also generalize the proposed pipeline for large-scale indoor environments. Compared to the urban scenarios, the indoor environment consists of relatively small structures, *e.g.*, rooms, corridors, and pieces of furniture and decoration placed in them.Thus, visual localization in an indoor environment can take advantage of recent ranging sensors such as laser range scanners, together with color sensors (camera devices). Large-scale indoor environments are often be represented by a set of high-quality 3D scans, *i.e.*, RGBD images, registered to a known floor map. While quite accurate and dense 3D structures are available, visual localization in indoor scenes is still challenging; weakly textured scene natures and highly repetitive structures make it difficult to determine a unique location of a query. To address these uncertainties, we propose **a localization method that takes advantage of the high-quality 3D structures** to estimate and validate the 6DoF camera pose of the query. The whole flow of the proposed method is constructed in the same manner as in outdoor scenarios. We first employ image retrieval to determine coarse locations of a query and then estimate (candidates of) camera pose using known

3D structures of each local area. We also propose **a new ranking method (pose verification) that select the best-matched camera pose using a rich 3D database**, which gives a significant performance gain when combined with the proposed pipeline.

4. In addition, through the validations of our localization pipeline, we introduce **two new datasets for outdoor and indoor scenarios**. The dataset for the outdoor scene includes a 2D image database, which represents a city-scale scenario of visual localization. On the other hand, the indoor dataset consists of multiple rooms, floors, and buildings and represents a remarkably large-scale scenario compared to existing datasets [56, 57]. It includes an accurate and dense 3D point clouds for each local area captured by laser range finder, thus presents a 3D database composed of RGBD images. Most importantly, both datasets contain sets of input images (query images) captured in various conditions separately from the database collection, **depicting the several challenging situations arisen from the diversity of the input**. Furthermore, we carefully prepare a manually annotated camera pose for each of these queries that can be used to evaluate the performance of visual localization. To the best of our knowledge, our datasets are the first to **accomplish the quantitative evaluation for visual localization in large-scale environments**, which provide further new insights of its performance on the diverse inputs.

This thesis is organized as follows. In Chap. 2, we propose two feature matching algorithms that can potentially contribute to visual localization. Sec. 2.2 introduce a feature matching for spherical panoramic images which capture the scene efficiently despite of large image distortion. Our method synthesizes multiple undistorted images for each panoramic image and perform feature matching on them to mitigate the side effect of image distortion. We show that our method provides accurate and rich correspondences between panoramic images and can contribute to build a rich 3D model via SfM pipeline. We also propose a feature matching designed for image pairs taken from very different viewpoints (Sec. 2.3). We again generate multiple synthetic images that approximate potential appearance changes on the image induced by view changes, to collect the variations of keypoints and feature vectors. Chap. 3 gives a detailed description of our contributions for outdoor city-scale visual localization. We first create a new dataset for visual localization in an urban situation that includes a set of accurate camera poses for evaluation (Sec. 3.2). We propose a localization pipeline that computes 6DoF camera pose of a query using only 2D image database (Sec. 3.4), which constructs our main contribution in this thesis. Sec. 3.5 provides various evaluation of our method and others on our new dataset, and shows the benefits of our approach on large-scale outdoor scenarios. Chap. 4 describes our contributions for large-scale indoor visual localization, on which the rich 3D point clouds and additional information are available. Because of the lack of existing dataset for indoor scenario, we again create a dataset composed of RGBD database images and user photos taken by smartphone, along with its accurate camera poses (Sec. 4.2). In Sec. 4.3, We propose a localization pipeline exploiting rich 3D database to estimate and verify the camera pose for the input image. Sec. 4.4 additionally provides details of our investigations for pose verification step. We validate our method effectively scales to a large-scale indoor environment and achieves state-of-the-art performance (Sec. 4.5). Finally, we add concluding remarks and potential future works of this area in Chap. 5.

In what follows, we provide a list of our publications related to this thesis.

- H. Taira, Y. Inoue, A. Torii, and M. Okutomi, "Robust feature matching for distorted projection by spherical cameras," *IPSJ Computer Vision and Applications*, vol. 7, pp. 84-88, 2015.
- H. Taira, A. Torii, and M. Okutomi, "Robust Feature Matching by Learning Descriptor Covariance with Viewpoint Synthesis," in *Proc. ICPR*, 2016.
- T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla, "Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization?" in *Proc. CVPR*, 2017.
- H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, "InLoc: Indoor Visual Localization with Dense matching and View synthesis," in *Proc. CVPR*, 2018.
- H. Taira, I. Rocco, J. Sedlar, M. Okutomi, J. Sivic, T. Pajdla, T. Sattler, and A. Torii, "Is This The Right Place? Geometric-Semantic Pose Verification for Indoor Visual Localization," in *Proc. ICCV*, 2019.
- A. Torii, H. Taira, J. Sivic, M. Pollefeys, M. Okutomi, T. Pajdla, and T. Sattler, "Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization?" *IEEE PAMI*, 2019 (Early access).
- H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, "InLoc: Indoor Visual Localization with Dense matching and View synthesis," *IEEE PAMI*, 2019 (Early access).

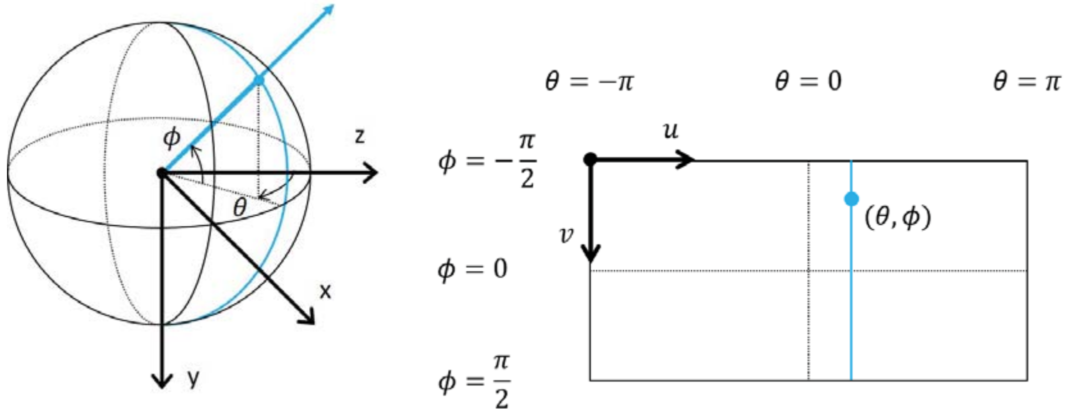
Chapter 2

Robust feature matching

2.1 Robust feature matching for image deformations

Establishing visual correspondence is a fundamental step in many computer vision applications, including SfM, image retrieval, and place recognition. Especially, public incremental SfM software such as Bundler [15] and VisualSFM [24] have made it possible to build a great 3D model using standard perspective images alone, which can also be used as a 3D database for visual localization composed of 3D points and associating image descriptors. These softwares require input images to have many common view fields, *i.e.*, requires to use many images (with a narrow field of view camera), to achieve a high-quality 3D model for a large-scale scene. One way to reduce the number of images required is to use spherical imaging systems such as Point Grey Ladybug or RICOH THETA. These cameras allow the user to capture images with large amounts of common views with little effort. However, one fatal drawback of using these cameras is the large distortions (changes of appearances) in the captured images as a result of the spherical projection, shown in Fig. 2.1 and Fig. 2.2. This distortion makes feature matching difficult, which could critically affect the qualities of the 3D model since all the geometric estimation (and RANSAC) in SfM relies on the local feature correspondences among input images collection.

Another source of error in feature matching is large viewpoint changes between image pairs to be matched (Fig. 2.7 (a) and (b)) because it again induces significant changes of appearances, which hurt commonnesses (repeatabilities) of local keypoints and feature descriptors (Fig. 2.7 (c) and (d)). Still, the large view changes often happen when building 3D points via SfM or estimating a 6DoF pose of a camera with respect to a pre-built 3D model. A well-known approach for viewpoint-invariant representation is to approximate changes in the appearance via an affine transformation. ASIFT [74] establishes a large number of correspondences by simulating affine warping for both feature detection and description. A shortcoming of this framework is that it requires additional computational costs for affine warped view synthesis

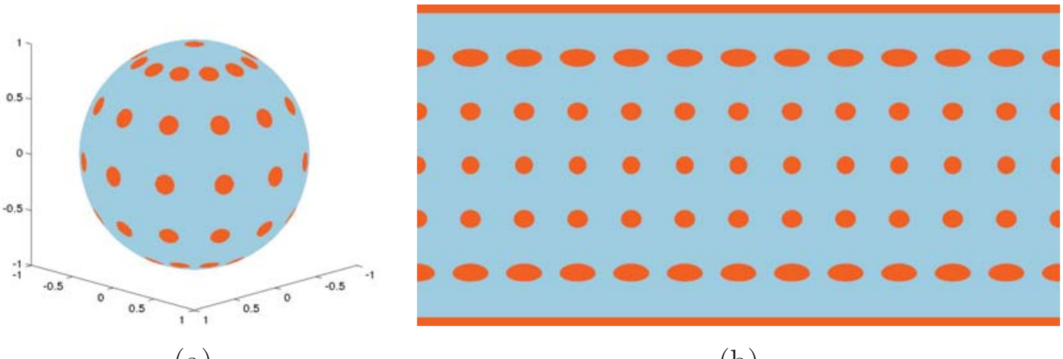


(a) (b)

Figure 2.1: Imaging with equirectangular projection. (a) Spherical coordinates. (b) Image coordinates of equirectangular projection image.

and matching. Several works have improved the framework's computational efficiency by limiting the range of view synthesis [75, 107]; However, additional affine warped view synthesis must still be performed for each image.

To address these common challenges, we propose two new feature matching framework exploiting view synthesis and efficient metric learning for feature descriptors. Sec. 2.2 describes a feature matching designed for spherical panoramic images, namely, equirectangular images. The key idea is to synthesize less-distorted images by rotating the spherical coordinate system. We then compute features only on the less distorted area in the rectified panorama. The method can be used as a preprocessing step for most standard feature extraction algorithms. Through experiments on synthetic and real images, we show our method improves the matching performance on equirectangular images and benefits to build a rich 3D model via SfM. Sec. 2.3 introduces another feature matching framework for standard perspective images that have been taken from largely different viewpoints. It again conducts view synthesis (but for either one of the two images) and extracts features from synthetic views to improve keypoint detection performance. Furthermore, we learn how the feature descriptor varies according to viewpoint changes using those synthesized images, and incorporate this knowledge into the similarity measurement scheme. As shown in Fig. 2.7, our trained Mahalanobis metric improves matchability by ignoring the descriptor variations caused by viewpoint changes. We show the performance of our matching framework on several standard image matching datasets. We finally summarize this chapter and add discussions in Sec. 2.4.



(a) (b)

Figure 2.2: Image distortion by equirectangular projection. The orange discs have the same size on the sphere (a) but mapping to the equirectantular image severely distorts them (b).

2.2 Feature matching for spherical panoramic images

In this section, we first describe the properties of image distortion induced by spherical panoramic projection (Sec. 2.2.1). We next provide the details of proposed method for feature detection and description with panoramic image rectification (Sec. 2.2.2). Finally, we validate our method on synthetic and real images (Sec. 2.2.3).

2.2.1 Properties of image distortion on spherical cameras

This section formulates the problems on image distortions induced by spherical cameras associated with the camera motion. We focus on the equirectuangular images captured by spherical cameras but the same approach can be applied to any wide FoV cameras such as omnidirectional and fisheye cameras.

An equirectangular image taken by a spherical camera is represented as

$$u = c(\theta + \pi), \quad v = c(\phi + \pi/2) \tag{2.1}$$

where u and v are the coordinates on the image (Fig. 2.1 (b)), θ and ϕ denote the longitude and latitude in the spherical coordinates (Fig. 2.1 (a)), and c is a constant defined by the image width w and height $h = w/2$ such that $c = w/2\pi = h/\pi$. This equirectangular projection has the property that the lines of longitude and latitude are evenly spaced. Therefore, as shown in Fig. 2.2, if we map circular regions (discs) of same size on the sphere to an equirectangular image plane, the discs close to the equators are less distorted but the ones close to the poles are heavily distorted. In other word, the image region close to the equator ($|\phi| < \beta$, where β is a small constant) is less distorted, *i.e.*, similar to perspective projection, whereas the region close to the poles is severely elongated. The region of interest looks very different in two images when features move from the equator to the pole as a result of camera rotation and/or translation. This reduces the matchability of local features.

We tackle this problem by introducing a simple algorithm:

1. We generate a set of rectified images by rotating the camera (spherical) coordinate system along x-axis (Fig. 2.3). We detect and describe features only from weakly distorted region in the rectified images.

2. When matching features between a pair of images, we perform step (1) for both images and use nearest neighbor search with Lowe's ratio test [32].

We next describe each step in more detail.

2.2.2 Panoramic image rectification and matching

Panoramic image rectification (PR). We generate n rectified images by rotating around an axis pointing at the equator (x-axis throughout the experiments) by the angle $\alpha = m\pi/n$ for $m = 0, 1, \ldots, n-1$, with respect to the relationship between a point on the sphere and a point on the equirectangular image in Eq. (2.1). As Figure 2.3 (a) shows, this rotation transfers the positions of features in the image, and all the features will eventually appear nearby the equator in the range of $\beta = \pi/2n$.

We next define masks for detecting features in the rectified images. We detect features in the region D of each rectified equirectangular image;

$$\left\{(\theta, \phi)^\top \in D \;\middle|\; -\beta < \arctan\left(\tan\phi/\cos\theta\right) \leq \beta\right\} \tag{2.2}$$

where $\beta = \pi/2n$ is the angle determined by the number of rectifications n as described above. Furthermore, this feature detection with panoramic image rectification can find the features from the entire sphere without duplication, as illustrated in Fig. 2.3 (b). Finally, we assemble the features detected from the region D for every rectification angle α by back-rotating the keypoint positions to the original coordinate system. Note that the keypoints are detected strictly in the masked areas to exclude duplicated detections but the masks are not used for descriptions of local patches to isolate effects by mask boundaries.

Feature matching. Feature matching for a pair of images is fairly simple. For each image, we detect and describe features using the panoramic image rectification described in Sec. 2.2.2. Although not a necessary condition, it is reasonable to use the same division number n for both images. We match the features using the descriptors by searching nearest neighbors followed by Lowe's ratio test (we use the threshold of 0.7 throughout the experiments). Note that any post-processing to refine the feature matches can also be applied such as RANSAC or its variants [108].

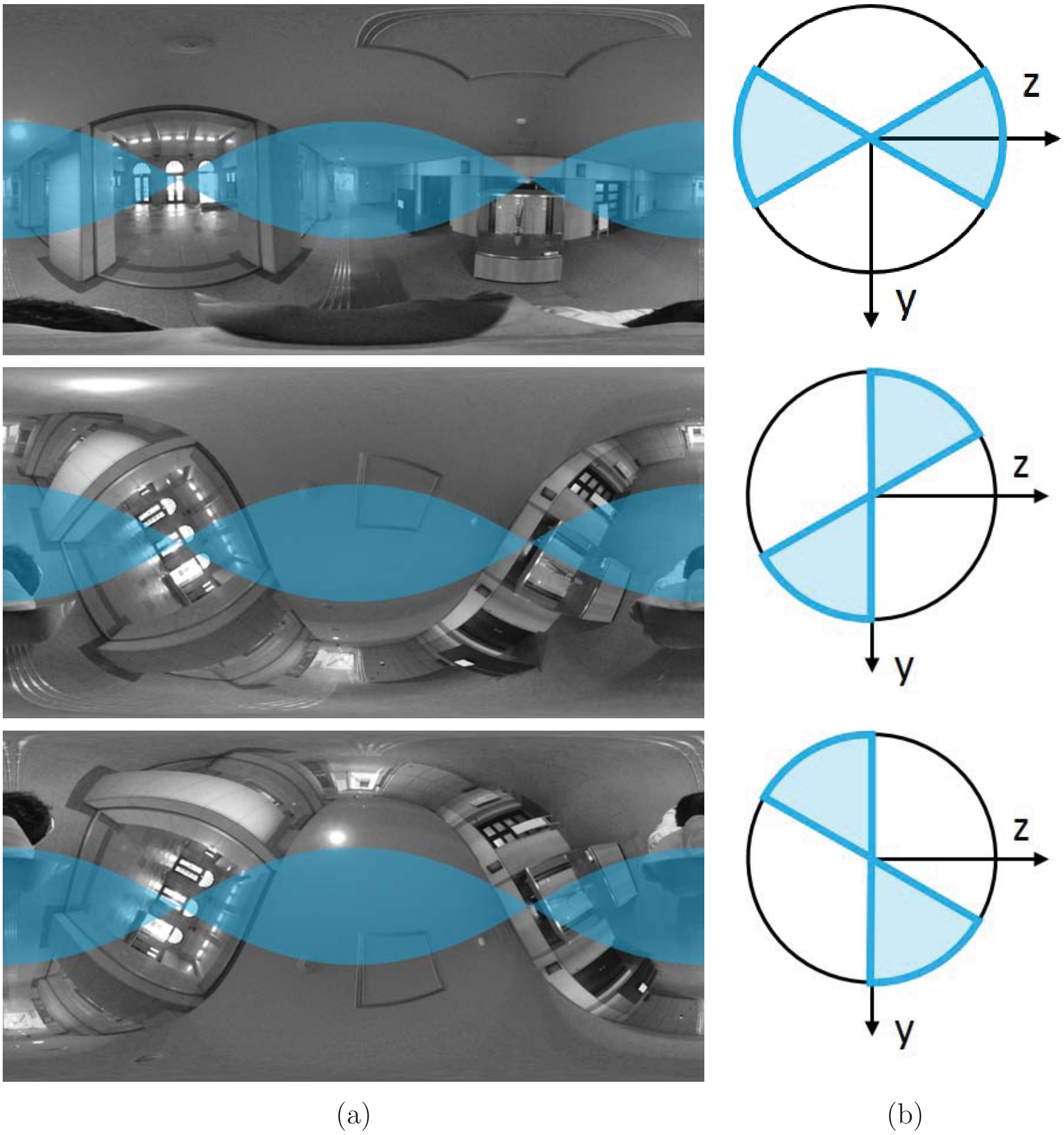
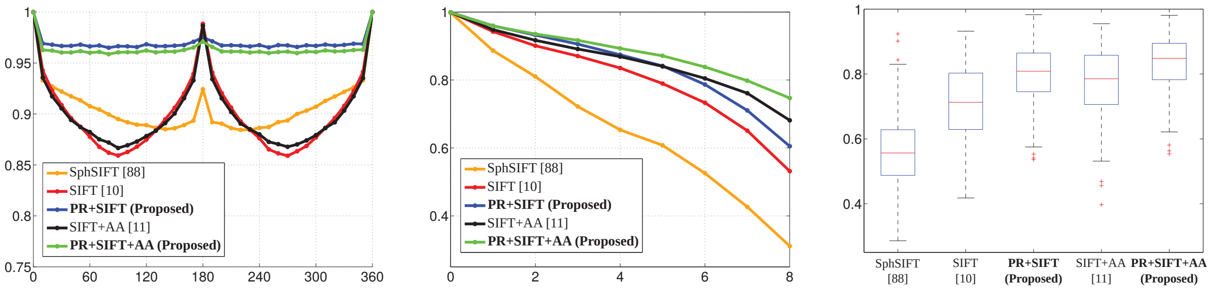


Figure 2.3: Example of panoramic image rectification. The rectified equirectangular images (a) obtained by rotation of 0°, 60°, and 120° w.r.t. x-axis. We detect features from masked regions (cyan) in each rectified image. The masked regions on the rectified image correspond to the colored region (cyan) on the spherical coordinates (b).



(a) Camera rotation (b) Camera translation (c) Random camera motion

Figure 2.4: Experimental result of feature matching. Graphs show the fraction of matching precision on y-axis for each rotation angle (a) or distance of cameras (b) on x-axis. The boxplot shows the statistics (y-axis) of the fraction of matching precision for each method (x-axis).

2.2.3 Experiments

First, we evaluate performance of the proposed method combined with a standard feature matching algorithm for perspective images (DoG keypoint detection and SIFT description). We compare it on both real and synthetic data with the state-of-the-art baseline methods. We next evaluate the performance when combined with affine adaptation. Finally, we evaluate the benefit of the proposed method in the incremental SfM using real images in the Pittsburgh Research dataset.

Dataset. We use 50 equirectangular images of 2896 × 1448 pixels. 25 of them are indoor and outdoor scenes captured by ourselves using a commercial spherical camera, RICOH THETA[1] and 25 images randomly chosen from the Google Pittsburgh Research Data Set[2].

Evaluation protocol. For correspondences (y, y') obtained by the nearest neighbor search followed by the ratio test, we define an evaluation function $f(y, y')$ with threshold T. We deem the features as correctly matched if $f(y, y') < T$. We then measure the performance by *precision* defined as

$$precision = \frac{\#correct\ matches}{\#feature\ matches} \tag{2.3}$$

Comparisons. We select SIFT [32] as the baseline comparison, which is designed for standard perspective images. Another comparison is spherical SIFT [109] which is a state-of-the-art feature matching pipeline for spherical images. It converts images on the sphere and describes SIFT-like features using the spherical scale-space. We use VLfeat [110] for SIFT, and the implementation of the original authors for spherical SIFT.

Feature matching with pure rotation. We first simulate a set of motion by rotating the

[1] https://theta360.com

[2] Provided and copyrighted by Google.

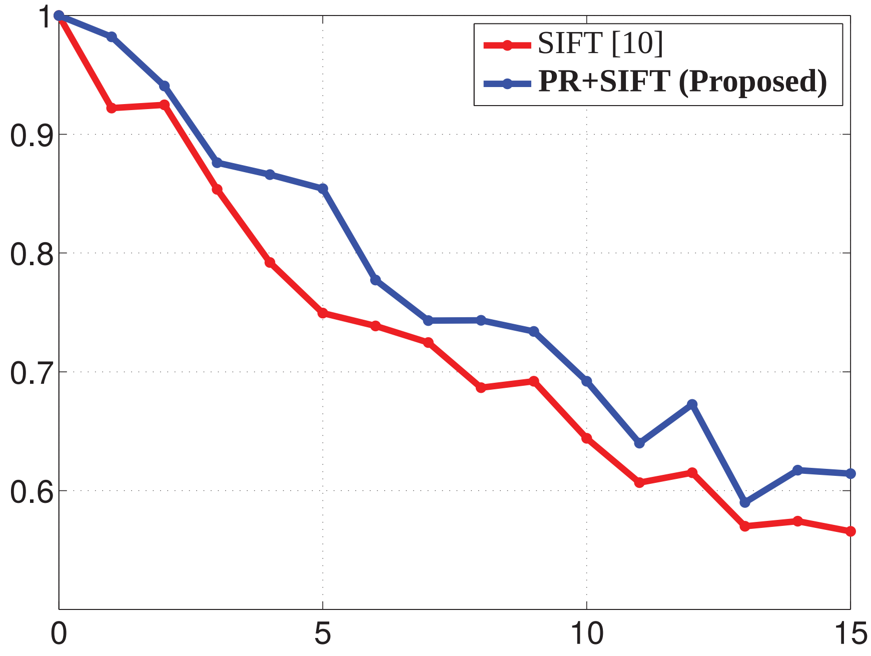


Figure 2.5: A comparison of the average fraction of cameras reconstructed ($y - axis$) when SIFT (red) and the proposed method (PR+SIFT) (blue) are used in incremental SfM. x-axis indicates the number of frames skipped from the original sequence.

camera around z-axis at the interval of $10°$. Notice that this axis differs to the x-axis used for the proposed panoramic image rectification. For each rotation angle, we generate 50 pairs of images by coupling the original image and the image after the rotation. Using these pairs of images, we evaluate matching precision. To measure the correctness of the match we define f as $f_{rot}(y, y') = \angle(y, R^{-1}(y'))$ where R is the rotation matrix obtained from the ground truth rotation. If $f(y, y') < 0.1°$, we deem the features as correctly matched.

Fig. 2.4(a) show the results of matching precision, respectively, for SIFT (red), spherical SIFT (orange), and SIFT with the proposed panoramic rectification (PR) (blue). Our method maintains high precision regardless of the rotation angles. whereas the baseline methods drops in performance when the appearance changes by rotation are large.

Feature matching with camera translation. To validate matching performances against camera translation, we prepare virtual cubes of 10^3 to render perspective cutouts taken from indoor and outdoor scenes. We simulate a set of translation by moving the camera from the initial position $(5, 5, 1)$ to $(5, 5, 9)$ at the interval of 1.

For each step of translation, we test 50 pairs of images by coupling the image at the initial position and the image after the motion. We evaluate precision with camera translation in the same way as camera rotation. To measure the correctness, we define f as $f_{trans}(y, y') = \|Y(y) - Y'(y')\|_2$ where Y and Y' are the 3D points associated to y and y' obtained from the

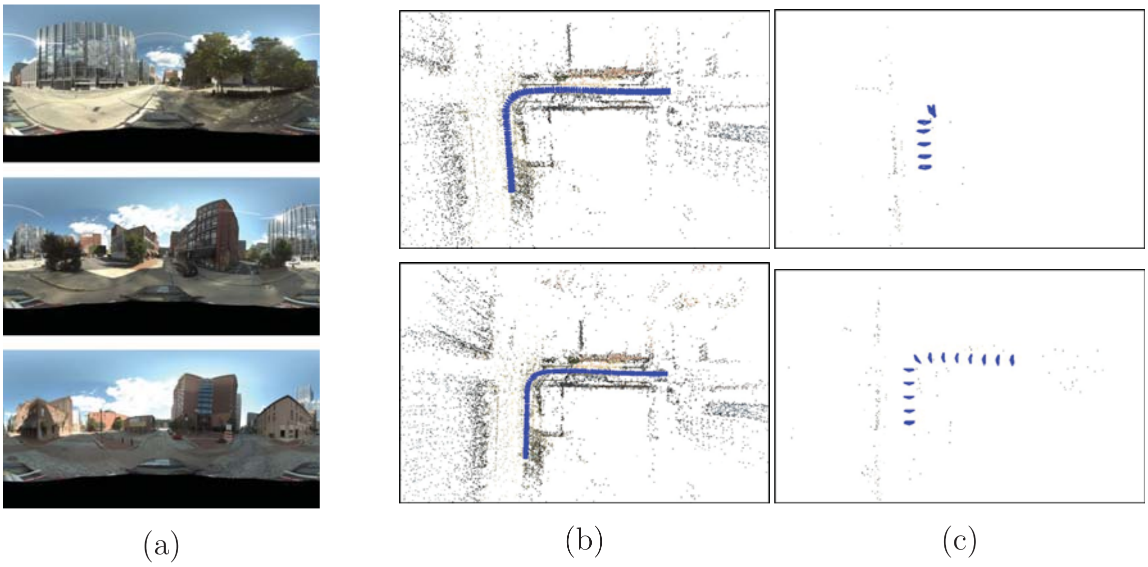


Figure 2.6: Experimental results with SfM. (a) shows examples of input equirectangular images used for incremental SfM. The SfM results reconstructed with standard SIFT (top) and the proposed method with panoramic image rectification (bottom) using all the frames (b) and every 7 frames (c).

ground truth values. We deem the matching is correct if $f_{trans}(y, y') < \sqrt{0.1}$.

The results in Fig. 2.4(b) show that the feature matching with camera translation is more challenging for all comparisons. However, similar to the rotation results, our method maintains higher precision than the baseline methods as the proposed method removes artifacts caused by distorted projection.

Next, for evaluation with random camera motion, we generate 100 pairs of images by generating 2 random rotations and translations for 50 scenes, then match them with the initial panorama where the camera position is $(5, 5, 5)$ with zero rotation. The results in Fig. 2.4(c) shows that the proposed method is the most effective (PR+SIFT) in comparison with baseline methods (sph SIFT and SIFT).

Proposed method combined with affine adaptation. We also evaluate the performance of the proposed method when combined with affine adaptation [33, 110], which has often been used for matching perspective images with large viewpoint changes. Fig. 2.4(a) shows that feature matching combined with affine adaptation (black) have no superiority to baseline SIFT (red) at pure camera rotation. This means that affine adaptation is not able to deal with distorted projection. Fig. 2.4 (b) and (c) show the results of matching precision in camera translation. Affine adaptation constantly outperforms all the baseline methods when the camera is translated. Notice that the proposed method combined with affine adaptation gives the best performance and the difference is significant on larger motions.

Proposed method for 3D reconstruction by SfM. We finally evaluate the performance of the proposed feature matching when used as a component of incremental SfM. We generated 50 sets of 100 consecutive frames randomly chosen from the 8999 equirectangular panoramic images of 3328×1664 pixels in the Google Pittsburgh Research Data Set. We implemented an incremental SfM pipeline similar to [111, 112] and evaluate the performance by measuring the numbers of cameras recovered in each set.

Fig. 2.5 shows the fraction of average numbers of recovered cameras over the input cameras (y-axis). To compare the performance on different baseline lengths, we skip k frames in each subset (x-axis) and run the SfM. The results clearly show that the incremental SfM combined with the proposed method (blue) gives consistently better performance than the one with the standard SIFT matching (red).

Fig. 2.6 shows examples of 3D reconstructed model. Figure 2.6(a) shows examples of input images. Figure 2.6(b) and (c) show 3D points and camera poses reconstructed by incremental SfM with standard SIFT (top) and the proposed robust matching with panoramic image rectification (bottom), respectively.

2.3 Feature matching robust to large viewpoint changes

2.3.1 Overview

In this section, we introduce our feature matching algorithms robust to large viewpoint changes. Fig. 2.10 shows the outline of the proposed feature matching. Key idea of proposed method can be divided into two phases;

1. The most failures of feature matching when matches images taken from very different views arise from the variations of keypoints and feature vectors, *i.e.*, one feature in the image can be appeared very differently in another image (illustrated in Fig. 2.7 (e)), or can even be lost in other images. This can be addressed by fully simulating image warping against possible viewpoint changes [74]. Assuming the either one of the two images is pre-processable, *e.g.*, feature matching between input and database images for visual localization, we synthesize multiple slanted views of the image in the same manner as in [74]. To observe the variations of feature vectors, we extract features from all synthesized views and group repeated features.

2. Storing all keypoints and descriptors extracted from multiple synthesized views and using them in matching require high computational cost compared to standard matching using features only from original views. Instead, we aim to learn the feature variations using synthetic features, and use them as the weighting factors for similarity metric between

correspondences. The variations are represented by a covariance matrix, and used for Mahalanobis metric (illustrated in Fig. 2.7 (f)) in our feature matching stage.

Sec. 2.3.2 provides details of our feature sampling scheme exploiting slanted view synthesis. Using sampled features, we learn how the descriptors vary regarding view changes (Sec. 2.3.3). Details of our feature matching algorithm are described in Sec. 2.3.4. We finally demonstrate the performance of the proposed feature matching framework in Sec. 2.3.5.

2.3.2 Feature extraction and grouping

Slanted views sampling. One common formulation of image deformations induced by viewpoint changes is the approximation via affine transformation [33, 74, 113, 114]. Assuming as affine transformation matrix $\boldsymbol{A}$, deformed image $\mathcal{I}'_{\boldsymbol{A}}$ can be represented using original image $\mathcal{I}$:

$$\mathcal{I}'_{\boldsymbol{A}}(\boldsymbol{x}) = \mathcal{I}(\boldsymbol{A}^{-1}\boldsymbol{x}) \tag{2.4}$$

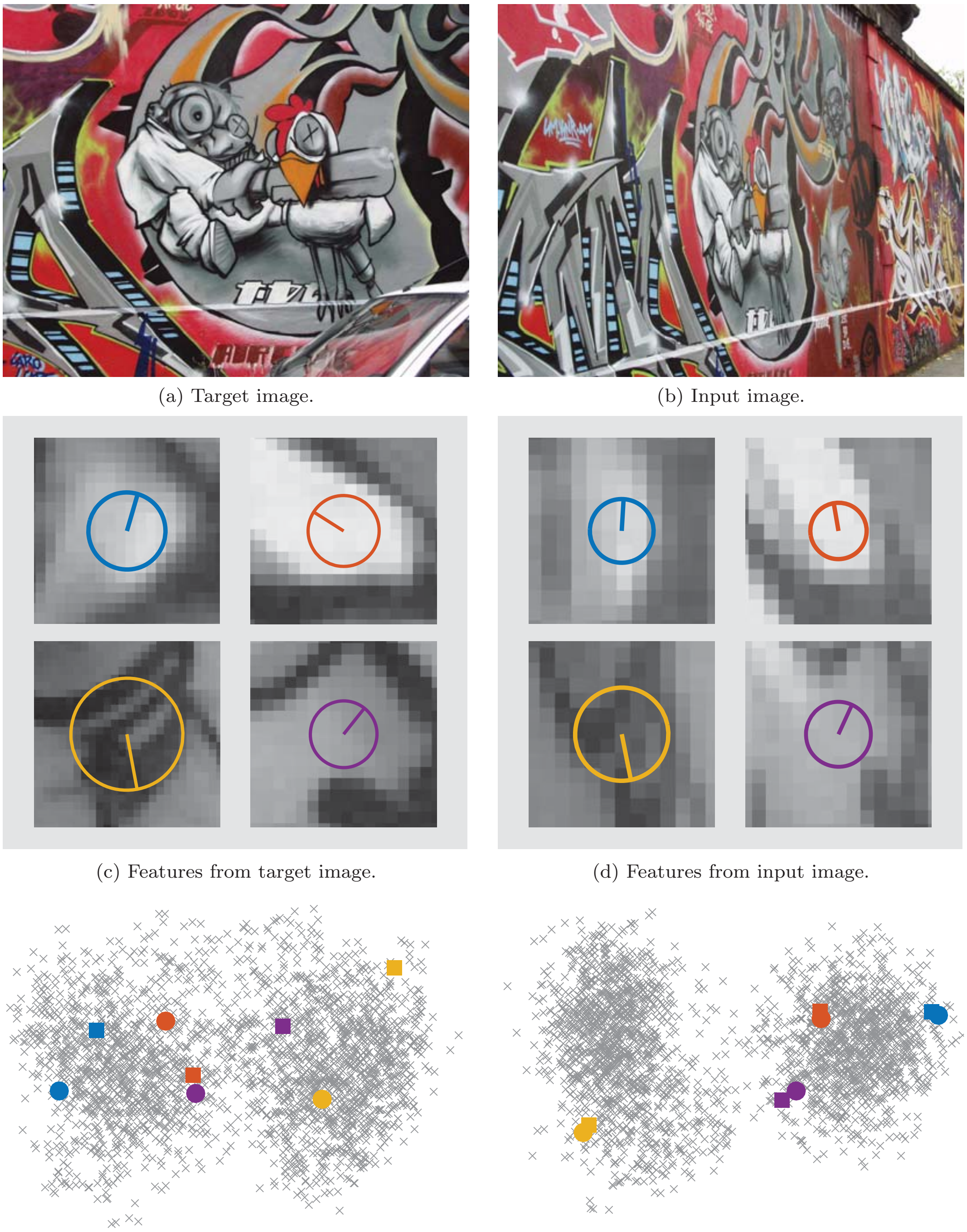
where $\boldsymbol{x} = [u, v]^{\mathrm{T}}$ is the image point in $\mathcal{I}'$. Regarding viewpoint moving around the original view, $\boldsymbol{A}$ can be decomposed into four parameters:

$$\boldsymbol{A} = \lambda \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{2.5}$$

Therein, $\lambda t > 0$ denotes the scale parameter, $\psi \in [0, 2\pi)$ and $\phi \in [0, \pi)$ represent the camera rotation around the optical axis and optical axis longitude, and $t > 1$ signifies the parameter for absolute tilt. Since most local features are designed to be invariant to ψ and λ, we sample absolute tilt t and optical axis longitude ϕ to generate several slanted views. We set the sampling steps of each parameter as $t = \{1, \sqrt{2}, 2, 2\sqrt{2}, 4, 4\sqrt{2}\}$ and $\Delta\phi = \phi_{k+1} - \phi_k = 2\pi/5t$, resulting in 47 affine transformed images generated for one original image.

Feature extraction. From each synthesized image, we extract SIFT [32] features which are associated by keypoint position $\boldsymbol{u}_k \in \mathbb{R}^2$, scaling factor $s_k \in \mathbb{R}$ indicating circular region around the keypoint, and orientation vectors $\boldsymbol{v}_k \in \mathbb{R}^2$. To associate all features by the keypoints on the original image, we warp them onto the original image plane. In other word, each feature is represented in original image by geometric information G_k, which is composed of warped keypoint $\boldsymbol{u}'_k = \boldsymbol{A}^{-1}\boldsymbol{u}_k$, covariance matrix $E_k = s^2(\boldsymbol{A}_k^{\top}\boldsymbol{A}_k)^{-1}$ indicating the elliptical region around the keypoint, and orientation vector $\boldsymbol{v}'_k = \boldsymbol{A}^{-1}\boldsymbol{v}_k$.

Feature grouping with respect to geometric information. Detecting features from multiple synthesized views often produces multiple features originating from the same keypoint, which have very similar but varied descriptors. In contrast to the other works [74, 75] that remove such repeated features, we employ the feature grouping process to sample varied descriptors.



(a) Target image.

(b) Input image.

(c) Features from target image.

(d) Features from input image.

(e) Descriptors in Euclidean space.

(f) Descriptors in Mahalanobis space.

Figure 2.7: **Learning Mahalanobis metric for feature matching.** (a, b) Images captured from different views. (c, d) Some typical SIFT features. Matching colors indicate corresponding features. (e) 2D PCA projection of SIFT descriptors for visualization. Highlighted points represent the descriptors of the features shown in (c) and (d) (circle: left features, square: right features). Feature descriptors vary according to changes in appearance. (f) We learn the descriptor variation caused by viewpoint changes in order to improve feature matching. Our metric clearly decreases the relative distances between the correct pairs of descriptors.

Algorithm 1 Feature grouping algorithm.

```
Require: $N_f$ features $\{f_k\}_{k=1}^{N_f}$ from affine warped images
Ensure: Set of feature chunklets $\{C_i\}_{i=1}^{N_C}$
Project features onto the original image plane: $\{f_k\}_{k=1}^{N_f} \rightarrow G = \{\boldsymbol{u}'_k, E_k, \boldsymbol{v}'_k\}_{k=1}^{N_f}$
$i \leftarrow 0$
while $G \neq \varnothing$ do
  $i \leftarrow i + 1$
  Select one feature from $G$ randomly: $\{\boldsymbol{u}'_i, E_i, \boldsymbol{v}'_i\}$
  for all $\{\boldsymbol{u}'_j, E_j, \boldsymbol{v}'_j\}$ in $G$ do
    if $\{\boldsymbol{u}'_i, E_i, \boldsymbol{v}'_i\}$ and $\{\boldsymbol{u}'_j, E_j, \boldsymbol{v}'_j\}$ satisfy Eq. (2.6) then
      Add $f_j$ to the $i$-th chunklet $C_i$
    else
      Add $\{\boldsymbol{u}'_j, E_j, \boldsymbol{v}'_j\}$ to $G'$
    end if
  end for
  Update $G$: $G \leftarrow G'$
end while
```

The workflow of our feature grouping process is shown in Alg. 1. We evaluate the consistencies between the geometric information of the features, and group similar features into the same set. The i-th and j-th features are grouped when their geometric information sets (G_i and G_j) have similar properties:

$$\|\boldsymbol{u}'_i - \boldsymbol{u}'_j\| < T_{\boldsymbol{u}}, \quad \sqrt{\left|E_i E_j^{-1}\right|} > T_s, \quad \angle(\boldsymbol{v}'_i, \boldsymbol{v}'_j) < T_{\boldsymbol{v}} \tag{2.6}$$

We set thresholds $T_{\boldsymbol{u}}, T_s, T_{\boldsymbol{v}}$ for each of geometric information by 3 pixels, 0.7, and $\pi/8$, respectively.

Fig. 2.8 shows examples of grouped features extracted from slanted views. Features pointing to similar regions are grouped and linked to the new keypoint at the center of their position $\boldsymbol{u}$. We finally obtain a set of grouped features (chunklet) for each keypoint that contains multiple feature vectors described in slanted views.

2.3.3 Learning descriptor covariance through view synthesis

We next learn how feature vector (descriptor) will be varied by viewpoint changes, using a set of features (chunklet) seeing a keypoint from many aspects. We represents the predicted variation as the covariance matrix. Fig. 2.9 illustrates two types of our training methods. We first estimate the covariance matrix for each keypoint representing the per-feature descriptor variation. We also propose an approach based on the Relevant Component Analysis (RCA) algorithm [79], which estimates one covariance matrix representing the characteristics common to all features. Learned covariances are used to improve similarity measurements in the matching stage.

Per-feature covariance matrices. The i-th chunklet contains N_i descriptors $\{\boldsymbol{x}_{ij}\}_{j=1}^{N_i}$ capturing the same keypoint from multiple viewpoints. Using these descriptors, the descriptor

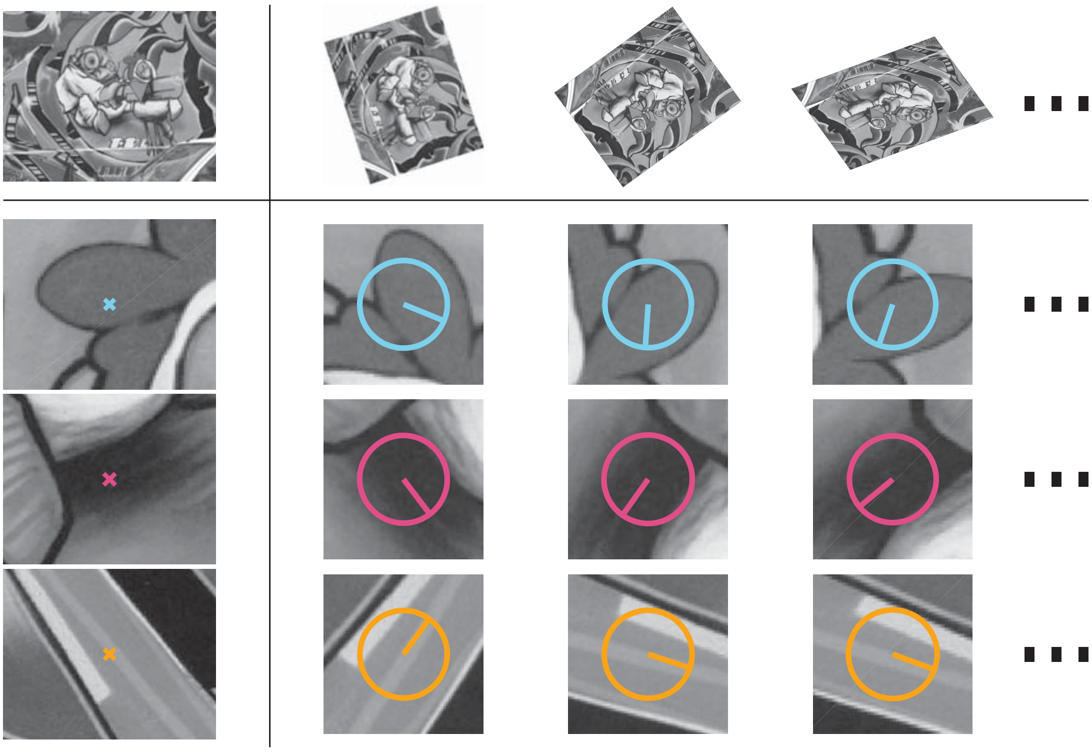


Figure 2.8: Some results of the feature grouping process. The top row shows the original image (left column) and affine warped images (right three columns). The bottom three rows present feature groups obtained by our feature grouping. The left column shows the new keypoint associated with each feature group; the right three columns show the features extracted from the image in each column.

variation in the i-th chunklet is formulated by covariance matrix $\boldsymbol{\Sigma}_i$ as shown below.

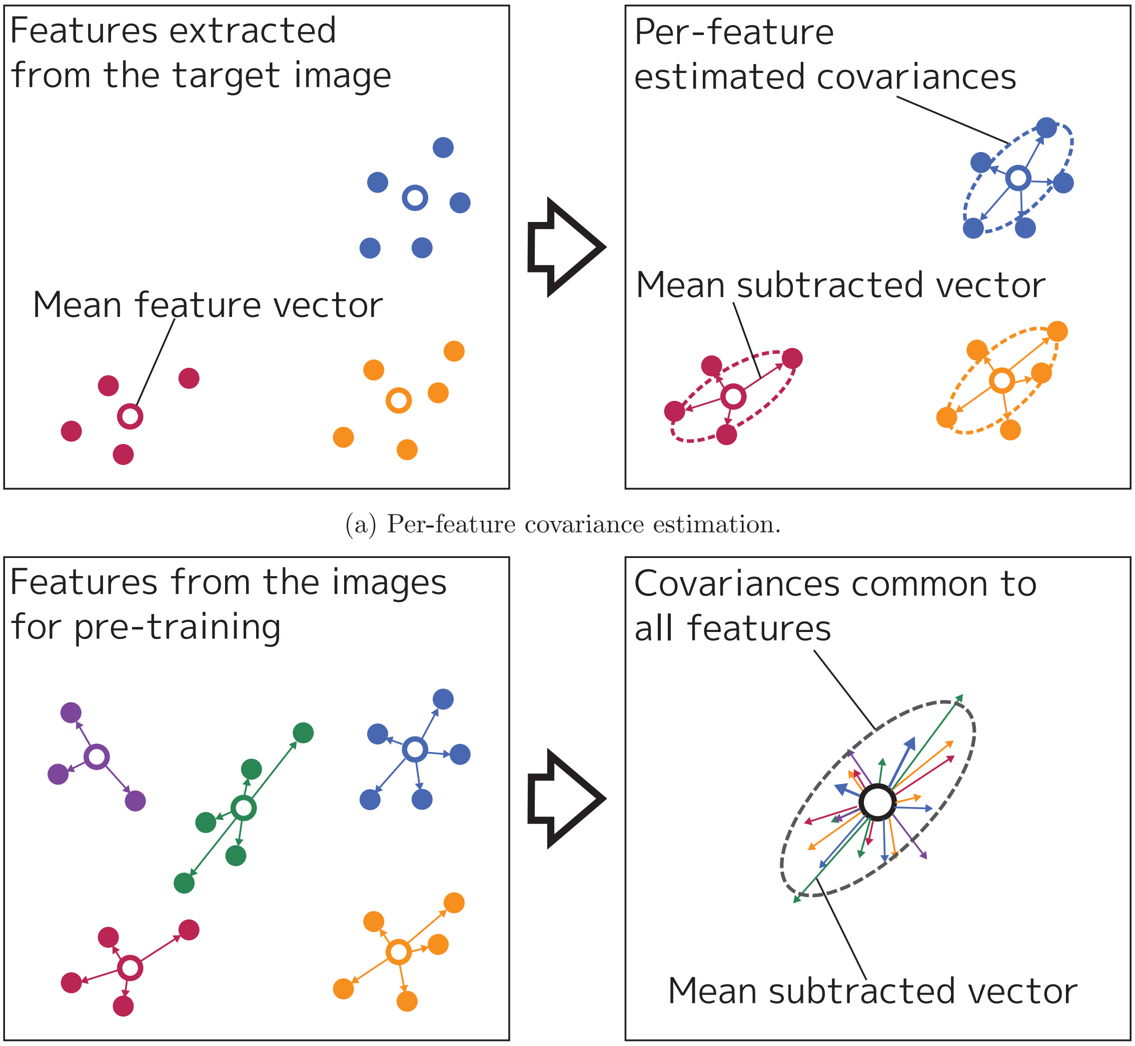
$$\boldsymbol{\Sigma}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)^\top \tag{2.7}$$

Therein, $\bar{\boldsymbol{x}}_i$ denotes the mean descriptor of the i-th chunklet. $\boldsymbol{\Sigma}_i$ indicates the covariances of descriptors that extract the same object from different views.

RCA-based common covariance matrix. The per-feature approach incurs high computational costs for estimating the metric of each feature because it computes the descriptor variation for each feature. We also propose another training method that estimates the descriptor variations common to all features.

Assuming that K is the number of chunklets, the RCA algorithm calculates a covariance matrix $\boldsymbol{\Sigma}_{common}$ by:

$$\boldsymbol{\Sigma}_{common} = \frac{1}{N}\sum_{i=1}^{K}\sum_{j=1}^{N_i}(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)^\top = \frac{1}{N}\sum_{i=1}^{K} N_i\boldsymbol{\Sigma}_i \tag{2.8}$$


Features extracted from the target image
Mean feature vector
Per-feature estimated covariances
Mean subtracted vector
Features from the images for pre-training
Covariances common to all features
Mean subtracted vector

(a) Per-feature covariance estimation.

(b) RCA-based common covariance estimation.

Figure 2.9: Our two approaches for metric learning. Descriptor variation is estimated as the covariance matrices. (a) In our per-feature approach, we estimate a covariance matrix for each feature group using the mean subtracted vectors of feature groups sampled from the target image. (b) In another approach, we conduct pre-training by gathering the mean subtracted vectors from the set of images to estimate a covariance matrix, which represents the common descriptor variation.

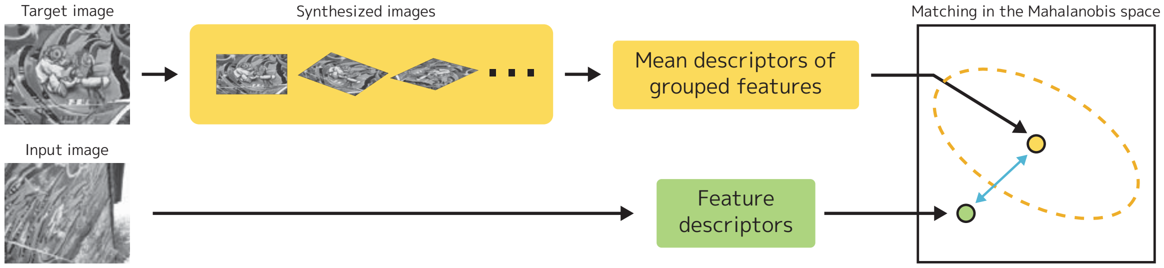

Target image
Synthesized images
Input image
Mean descriptors of grouped features
Feature descriptors
Matching in the Mahalanobis space

Figure 2.10: Overview of our feature matching framework designed for matching largely view-changed images.

where N denotes the total number of features extracted. The mean subtracted vector $(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)$, which is also used in Eq. (2.7) to calculate covariance matrix $\boldsymbol{\Sigma}_i$ for the i-th chunklet, is used here to indicate the sampled variations of descriptors that relate to a common characteristic of all features.

2.3.4 Feature matching using descriptor variations

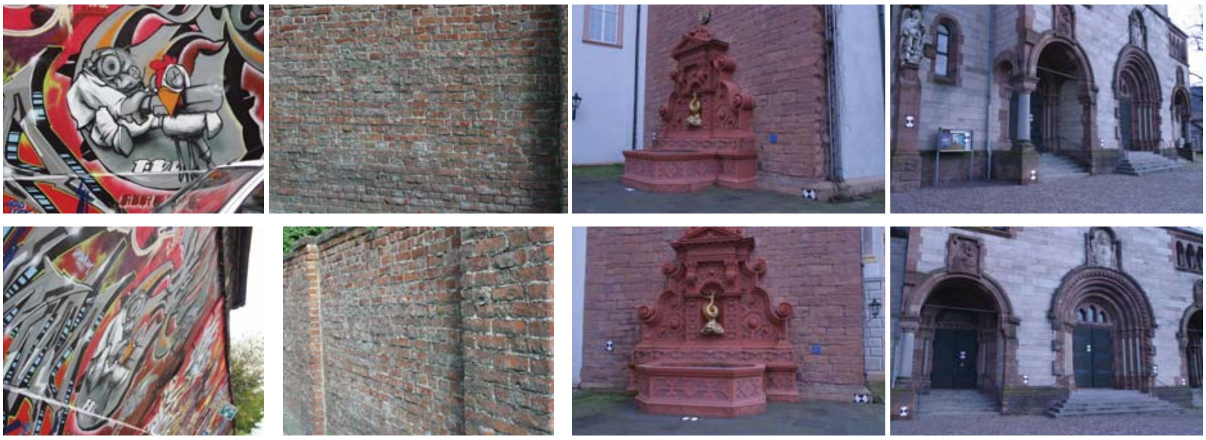
We propose a new feature matching framework that considers descriptor variations associated with viewpoint changes; the framework is summarized in Fig. 2.10. Because we are proposing two approaches for covariance learning, the proposed framework has two property types (Per-feature and Common). Both methods follow the diagram shown in Fig. 2.10; however, they use different similarity measurement metrics, according to their learning approaches.

We first extract feature chunklets from the target image as described in Sec. 2.3.2. The i-th chunklet associating the i-th keypoint in the target image has properties of descriptors $\{\bar{\boldsymbol{x}}_i, \boldsymbol{\Sigma}_i\}$ where $\bar{\boldsymbol{x}}_i$ denotes the mean descriptor of the chunklet and $\boldsymbol{\Sigma}_i$ denotes the estimated covariance matrix for the descriptor variation. The proposed method that uses the per-feature approach for covariance learning (Proposed (Per-feature)) calculates the covariance matrix $\boldsymbol{\Sigma}_i$ for each chunklet according to Eq. (2.7). The other approach is the common approach (Proposed (Common)), which assumes the same properties $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_{common}$ for all features and computes $\boldsymbol{\Sigma}_{common}$ using all chunklets.

When the input image for matching appears, we extract features only from the original image. The similarity between the i-th target chunklet $\{\bar{\boldsymbol{x}}_i, \boldsymbol{\Sigma}_i\}$ and the j-th query feature $\boldsymbol{x}_j$ is computed using the Mahalanobis distance.

$$D_M\left(\boldsymbol{x}_j, \{\bar{\boldsymbol{x}}_i, \boldsymbol{\Sigma}_i\}\right) = \sqrt{\left(\boldsymbol{x}_j - \bar{\boldsymbol{x}}_i\right)^\top \boldsymbol{\Sigma}_i^{-1} \left(\boldsymbol{x}_j - \bar{\boldsymbol{x}}_i\right)} \tag{2.9}$$

High-variance elements are suppressed, whereas low-variance elements are emphasized. We therefore upweight the elements having lower variances, which can be regarded as the non-sensitive elements associated with viewpoint changes.



(a) Graffiti (b) Wall (c) Fountain (d) Herzjesu

Figure 2.11: Examples of image pairs used in performance evaluation: (Top) target image and (Bottom) one of the input images for each scene.

Table 2.1: Details of image datasets used for performance evaluation.

Scene	Pixels	Input images
Graffiti	800 × 640	5 frames
Wall	880 × 680	5 frames
Fountain	3072 × 2048	2 frames
Herzjesu	3072 × 2048	2 frames

$\mathbf{\Sigma}_i$ sometimes becomes a singular matrix, such that an exact inverse matrix $\mathbf{\Sigma}_i^{-1}$ cannot be determined. A psuedo-inverse matrix that neglects the dimensions having lower singular values is often used in such case. However, this modification does not fit to our case because we aim to emphasize the dimensions having lower variances. Instead, we modify the covariance matrix by rounding the eigenvalues with minimum threshold ϵ so that $\mathbf{\Sigma}_i^{-1}$ can be determined. We set $\epsilon = 10^{-5}$ experimentally.

After finding correspondences by nearest neighbor searching, tentative correspondences are built using a standard distance ratio test [32], which evaluates whether the distance between the first nearest target feature and the query feature is sufficiently less than the distance between the second nearest neighbor and the query feature.

2.3.5 Experiments

We evaluate the performance of our feature matching framework with respect to matching precision and computational time. We match images captured from different viewpoints and evaluate the accuracy of correspondences. We also measure the matching time required by each method, to demonstrate the computational simplicity of our feature matching framework.

Implementation details. The DoG detector and SIFT descriptor were provided by VLFeat [110]. We implemented the SIFT matching framework and the proposed matching framework with MATLAB (**SIFT**, **Proposed (Per-feature)**, **Proposed (Common)**). We compared our methods with **ASIFT**, which was implemented by the authors [74] using C++. The computational time for each method was then evaluated on a machine equipped with an Intel Core i7-5930K CPU at 3.5 GHz and 64 GB RAM. We also evaluated our methods using the RootSIFT descriptor, which showed more accurate matching results (**Proposed (Per-feature, RootSIFT descriptor)**, **Proposed (Common, RootSIFT descriptor)**). We compared them against the ASIFT implementation, which was modified to use the RootSIFT descriptor (**ASIFT (RootSIFT descriptor)**).

Datasets. Tab. 2.1 and Fig. 2.11 provide information regarding the image pairs we used in the experiments. We selected four image sequences from widely known datasets. Each scene contains one target image and multiple input images. The Graffiti and Wall [115] scenes contain images capturing planar walls from a fronto-parallel viewpoint (for the target image) and from slanted viewpoints (for the input images). Fountain and Herzjesu [116], which are subsets of fountain-P11 and Herz-jesu-P8 [117], capture more three-dimensional scenes. We selected these scenes because they are more challenging situations for features that capture local affine regions.

Matching performance. Table 2.2 shows the average matching precision and the number of correct matches for each scene, and for all 14 image pairs. Figure 2.12 shows some typical feature matching results. **SIFT** and **RootSIFT** generate a lower matching precision and number of matches, because they do not provide a view synthesizing process for both target and input images. **ASIFT** generates a larger number of correct matches because it extracts numerous features from affine warped images for both target and input images. However, it also produces a large number of false matches. The proposed methods with the RootSIFT descriptor outperform ASIFT for all scenes, including challenging scenes such as Fountain and Herzjesu.

Figure 2.13 shows more detailed feature matching results obtained by each method, using the SIFT descriptor for the scene "Graffiti." The matching precision of **SIFT** decreases as the viewpoints of the input images deviate further from the viewpoint of the target image. The view synthesis process clearly prevents the decline in precision (**ASIFT**). Furthermore, our Mahalanobis metric effectively improves feature matching (**Proposed (Per-feature, Common)**). **Proposed (Per-feature)** provides the highest precision values for all images. It indicates that our per-feature Mahalanobis metric effectively improves matchability without requiring a view synthesis process for the input image. **Proposed (Common)** gives constantly comparable results with **Proposed (Per-feature)**. That is to say, our common covariance learning method successfully estimates the descriptor variations associated with viewpoint changes, using pre-

Table 2.2: Quantitative matching results obtained by each method. The top four methods use the SIFT descriptor and the bottom four use RootSIFT. The average matching precision (and number of correct matches) is shown for each scene. The best precision value is highlighted.

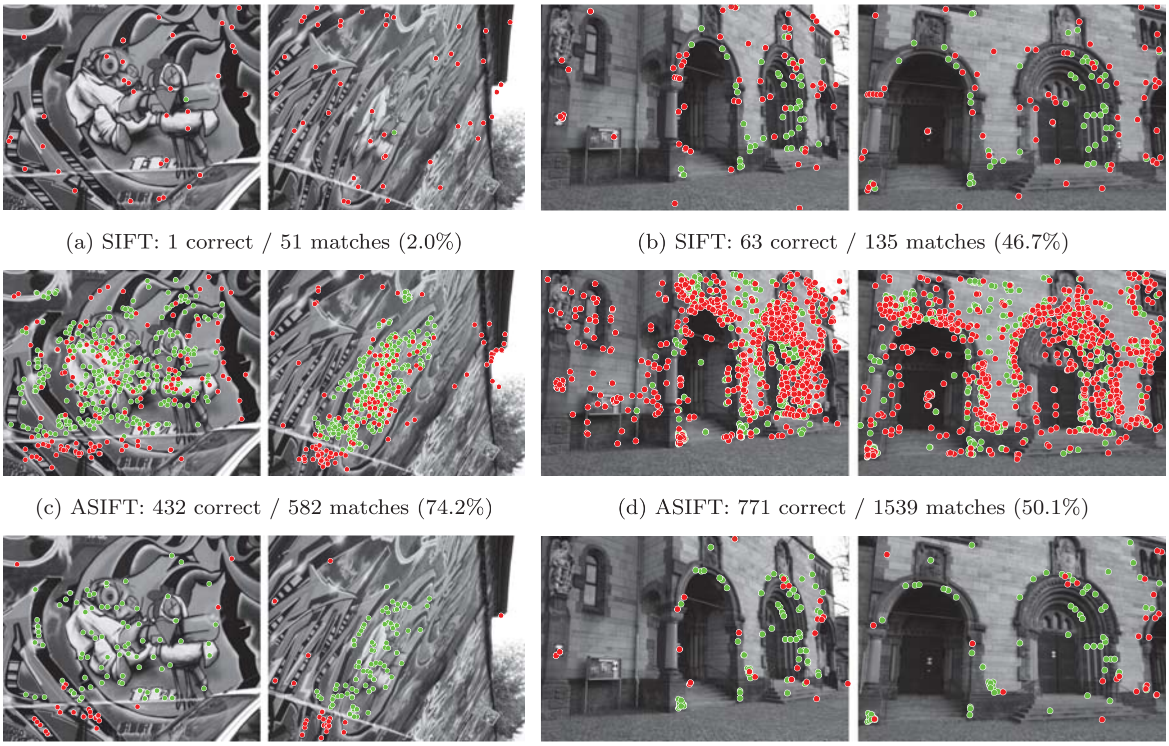
	Graffiti	Wall	Fountain	Herzjesu	All
SIFT	44.5 (169)	82.5 (382)	68.4 (274)	62.8 (340)	64.1 (284)
ASIFT	83.1 (1704)	96.0 (5055)	**80.9** (6863)	59.2 (3284)	84.0 (3863)
Proposed (Per-feature)	**91.4** (290)	97.3 (476)	79.7 (304)	**80.9** (376)	**90.3** (370)
Proposed (Common)	89.1 (253)	**98.2** (474)	80.0 (294)	77.0 (346)	89.3 (351)
RootSIFT	48.3 (173)	90.0 (386)	72.0 (281)	68.2 (351)	69.4 (290)
ASIFT (RootSIFT descriptor)	85.0 (1796)	96.3 (5081)	83.6 (7279)	67.2 (3877)	86.3 (4050)
Proposed (Per-feature, RootSIFT descriptor)	**90.7** (291)	97.4 (479)	**83.9** (313)	**86.8** (380)	**91.6** (374)
Proposed (Common, RootSIFT descriptor)	90.1 (263)	**97.8** (476)	83.6 (306)	80.7 (371)	90.6 (361)

Table 2.3: Average computational time (seconds) required by each feature matching method to process the input images for each scene.

	Graffiti	Wall	Fountain	Herzjesu
SIFT (w/o FLANN)	0.306	0.360	1.641	1.722
SIFT (w/ FLANN)	0.268	0.305	1.618	1.622
ASIFT (w/o FLANN)	7.527	15.267	96.128	142.186
Proposed (Common, w/o FLANN)	0.770	0.675	2.114	2.802
Proposed (Common, w/ FLANN)	0.271	0.311	1.623	1.632

training that is separated from the matching process.

Computational time. Table 2.3 shows the average computational time, *i.e.*, the time period between the appearance of a new input image and the establishment of the correspondences. Notice that **SIFT** and **Proposed (Common)** were implemented using MATLAB; **ASIFT** was implemented using C++. **ASIFT** still incurs a significant computational cost to perform full view synthesis and matching. On the other hand, our method extracts features only from the original input image. The computational overhead of **Proposed (Common, w/o FLANN)** added to the baseline **SIFT (w/o FLANN)** is incurred by the similarity measurements for target features extracted from multiple synthesized images. Results show that the computational time required for these measurements is only fractional. We also evaluate feature matching combined with a fast approximate nearest neighbor technique [118]. Our feature matching framework accelerated with FLANN (**Proposed (Common, w/ FLANN)**) shows comparable computational time to the original SIFT (**SIFT (w/ FLANN)**).



(a) SIFT: 1 correct / 51 matches (2.0%)

(b) SIFT: 63 correct / 135 matches (46.7%)

(c) ASIFT: 432 correct / 582 matches (74.2%)

(d) ASIFT: 771 correct / 1539 matches (50.1%)

(e) Proposed (Per-feature):

100 correct / 126 matches (79.4%)

(f) Proposed (Per-feature):

86 correct / 114 matches (75.4%)

Figure 2.12: Matching results obtained using SIFT (top), ASIFT (middle), and our method (bottom) for the scenes “Graffiti” (left) and “Herzjesu” (right). For each scene, the left image is a target and the right image is an input. Green dots represent correctly matched features; red dots represent incorrect matches.

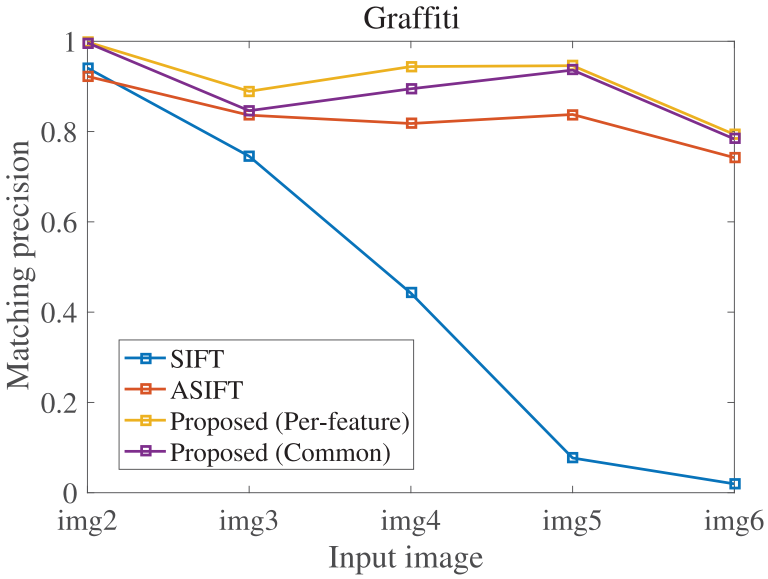


Figure 2.13: Matching precision for each image pair in the scene “Graffiti.” A higher image index indicates a more slanted input image.

2.4 Summary

In this chapter, we discussed about view synthesis-based techniques for matching images which sometimes look very differently from each although seeing the same scene. We focused on two types of deformations; induced by image projection format and camera viewpoint changing. We addressed both situations by simulating possible deformations on the images and combining them with standard feature matching algorithms [32]. As shown in experiments, this approach enables us to generalize our methods to other feature matching frameworks [26,33] and to easily combine our methods with existing applications exploiting feature matching such as SfM.

Sec. 2.2 discussed about our feature matching algorithm focusing on image distortion induced by spherical panoramic projection. We simulated potential deformations by generating multiple synthetic images that rotates around an axis vertical to polar axis. By extracting features only from less distorted areas in the synthetic images, we succeeded to collect a set of stable features for each image to be matched. Several experiments on synthetic data (Fig. 2.4) revealed the robustness of our feature matching against to the camera rotation and translation. We also demonstrated the benefit of our method when combined with SfM reconstruction, which provide more rich and complete 3D point cloud of the scene (Fig. 2.6) that could be useful as the 3D database for visual localization.

Sec. 2.3 discussed about feature matching for images taken from largely distant from its target image. While standard local features are not generalized to deformations induced by such images, view changing is often inevitable in practical applications including visual localization, *e.g.*, geometric verification [42] and 3D-based localization methods [46, 48–50, 52, 54] requires matches between database and input image, and the accuracy of correspondences is crucial for its reliabilities. We addressed the issue again exploiting view synthesis. We sampled local features while simulating full affine space and gathered multiple descriptors that see the same keypoint from different aspects. Rather performing exhaustive matching among all features, we achieved an efficient feature matching by learning the Mahalanobis metric to describe the feature variations against to the potential camera motions. Experiments on several public image sets confirmed the robustness and accuracy of our pipeline on severe conditions for matching. Proposed pipeline provides image correspondences with high accuracy rate in comparable computational times to the baseline method [32]. Since our matching pipeline (illustrated in Fig. 2.10) designed to reduce computation processes for input image, *i.e.*, view synthesis and metric learning is performed only for target image, it can naturally be combined with some visual localization pipelines, *e.g.*, geometric verification.

Chapter 3

Visual localization in a large-scale urban environment

3.1 Background

Determining the location from which a photo was taken is a key challenge in the navigation of autonomous vehicles such as self-driving cars and drones [1], robotics [119], mobile Augmented Reality (AR) [2, 3], and Structure-from-Motion (SfM) [18, 25, 120, 121]. In addition, solving the visual localization problem enables a system to determine the content of a photo. This can be used to develop interesting new applications, *e.g.*, virtual tourism [15] and automatic annotation of photos [122, 123].

Currently, vision-based localization problems have often been addressed by *visual place recognition* approaches [17, 20, 27, 39, 88, 124, 125], which represent a scene as a database of geotagged images and cast the problem as an image retrieval task. Given a query photo, they employ **2D image-based localization** methods that operate purely on an image level to determine a set of database images similar to the query. The geotag of the most relevant retrieved photo then often serves as an approximation to the position from which the query was taken. On the other hand, some applications, including autonomous driving and AR, require an accurate 6DoF pose, *i.e.*, position and orientation, of the query image, rather than rough location of a query. *Image-based localization* approaches [46, 49–52, 126] cast the localization problem as a camera resectioning task while assuming a database of the scene represented via an SfM model, *i.e.*, a 3D point cloud with associating image descriptors [127, 128]. **3D structure-based localization** methods then use the database (set of image descriptors) to establish a set of 2D-3D matches. In turn, these matches are used to recover the full 6DoF camera pose [129, 130]. Tab. 1.1 provides the system-level summaries of these two approaches.

A common perception is that 2D image-based approaches can be used by 3D structure-based methods to determine which parts of a scene might be visible in the query [46, 52, 91, 131].

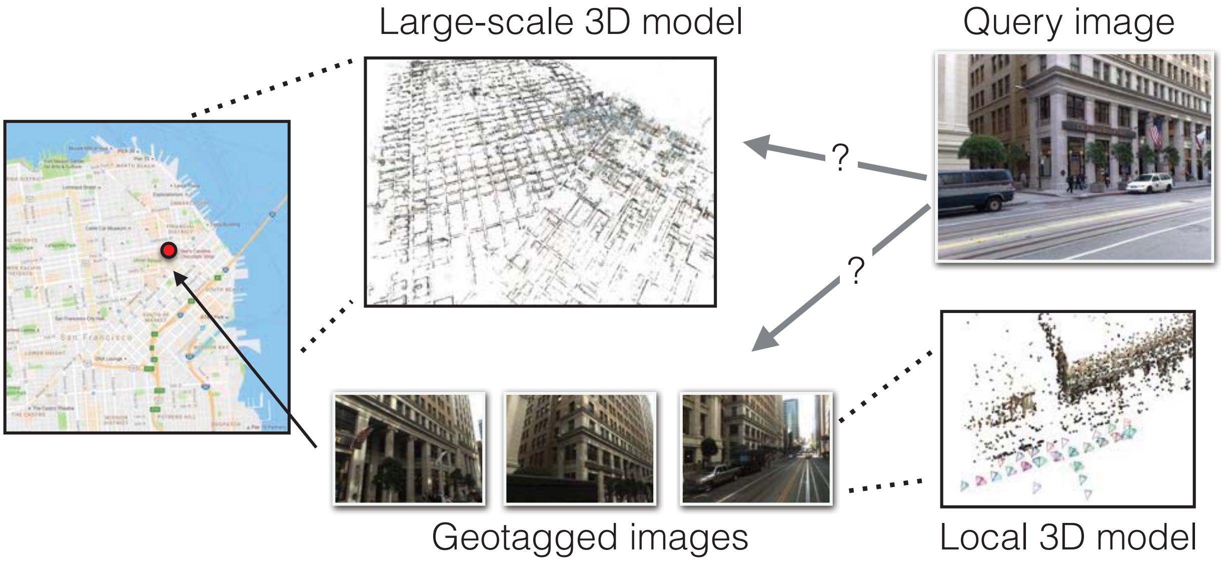


Figure 3.1: **The state-of-the-art for large-scale visual localization.** 2D image-based methods (bottom) use image retrieval and return the pose of the most relevant database image. 3D structure-based methods (top) use 2D-3D matches against a 3D model for camera pose estimation. Both approaches have been developed largely independently of each other and never compared properly before.

Purely 2D-based techniques are considered unsuited for accurate visual localization due to only approximating the true camera position of the query. Consequently, 2D- and 3D-based localization methods are only compared in terms of place recognition performance [39, 46, 126]. However, this ignores the fact that a more accurate position, together with the camera orientation, can be computed if two or more related database images can be retrieved [132–134]. This naturally leads to the question of whether 2D image-based localization approaches can achieve the same pose accuracy as 3D structure-based methods. This is a compelling question due to the way both types of methods represent scenes: especially for large-scale scenes, building and maintaining the 3D models required by structure-based techniques is a non-trivial task. At the same time, image-based techniques just require a database of geotagged images, which is significantly easier to maintain.

Scenario. In this chapter, we want to answer whether large-scale 3D models are actually necessary for accurate visual localization or whether sufficiently precise pose estimates can already be obtained from a database of geotagged images. Fig. 3.1 illustrates the general scenario of visual localization in city-scale urban environments. The objective of visual localization is to determine the 6DoF pose of the camera using known 2D or 3D database depicting the target environment. While several approaches assumes sequential images input to incrementally seek the current locations, *e.g.*, SLAM [11–13], we focus on a single image input as it is a basic problem and also extendable for sequential inputs. At the same time, since large-scale environments naturally contain various circumstances, *e.g.*, urban or residential areas, richly or weakly textured scenes, quiet narrow streets or crowded downtown, visual localization for large-scale environments are required to suit for more general situations, rather than fitting to a specific situation.

Contributions. Considering the outdoor scenario, our work makes the following contributions: i) We generate reference camera pose annotations for the query images of the San Francisco Landmarks dataset [17], resulting in the first city-scale dataset with such information. ii) We use this new dataset for the first comparison of 2D- and 3D-based localization approaches regarding their pose accuracy. To this end, we combine 2D image-based methods with an SfM-based post-processing step for pose estimation. Our results clearly show that 2D image-based methods can achieve a similar or even better positional accuracy than 3D structure-based methods. As such, we refute the notion that purely 2D image-based approaches are inaccurate. iii) We demonstrate that the previously used strategy of evaluating localization methods via a landmark recognition task is unsuitable for predicting pose accuracy. Also, we show that pose precision results obtained on smaller landmark datasets do not translate to large-scale localization. Thus, our new benchmark closes a crucial gap in the literature and will help to drive research on accurate and scalable visual localization.

This chapter is organized as follows; Sec. 3.2 reviews existing dataset for visual localization and provides details of our new dataset including manually annotated 6DoF camera poses for queries which can be used for validation; Sec. 3.3 next reviews existing visual localization methods in the spirit of 2D image-based and 3D structure-based approaches, which in this chapter we aim to compare; We also propose an efficient camera pose estimation technique for 2D image-based methods that provides an accurate 6DoF camera pose for the query based on SfM pipeline, *i.e.*, using only 2D images (Sec. 3.4); Sec. 3.5 gives several validations on our dataset. We compare methods in each of 2D- and 3D-based methods, including 2D-based methods combined with our camera pose estimation method as post-processing; Sec. 3.6 concludes this chapter.

3.2 Dataset for visual localization in an urban environment

In this section, we first motivate our new pose dataset by reviewing the currently used evaluation protocols. Next, we review the San Francisco dataset before detailing how we generate reference poses for some of its query images.

3.2.1 Review for current evaluation protocols

3D structure-based localization approaches are typically evaluated by counting how many query images have an estimated pose with at least X inliers, where X is some threshold. This is based on the observation, made on smaller datasets, that wrong pose estimates are rarely supported by many inliers. However, this observation does not transfer to large-scale datasets [39,46,126]. At scale, repetitive structures and sheer size increase the chance of finding more wrong matches that are geometrically consistent [46,126]. Simply counting the query images with at least X inliers thus overestimates the performance of structure-based methods. As such, it is necessary to also consider pose accuracy.

The datasets commonly used to evaluate the localization accuracy of structure-based approaches, 7 Scenes [56], Arts Quad [121], Cambridge Landmarks [135], Dubrovnik [16], and the recent Aachen Day-Night, RobotCar Seasons, and CMU Seasons benchmarks [136], all depict small- to medium-scale scenes with significant texture. Consequently, it is often possible to find many matches, which aids pose accuracy. Richly textured scenes are less frequent in urban environments due to the prevalence of reflecting or texture-less surfaces. This creates a need to also evaluate pose accuracy for truly large-scale datasets characterized by more ambiguous structures. Creating a benchmark for such a scene is one of the contributions of this work.

2D image-based localization methods are mostly evaluated in the context of landmark or place recognition [17,20,27,39,40,124,125,133,137]. For landmark recognition, the goal is to retrieve at least one database image that depicts the same landmark or scene element as the query photo [17]. Vision is a long-range sensor and as such, a relevant database image might depict the same landmark while being taken tens or hundreds of meters away from the position of the query image. Thus, the geotag of such an image is not necessary a good approximation to the position of the query. Still, it might be possible to accurately determine this position through camera pose estimation (*c.f.* Sec. 3.3.1). One of the contributions of this work is to evaluate to what extend landmark recognition performance translates to accurate localization.

In terms of place recognition, image-based localization methods are tasked to find a database image whose geotag is within a certain radius of the query's GPS position [125,133]. The fact that vision is a long-range sensor again causes problems in this setting as it is can be hard to distinguish between database images depicting the same part of the scene taken close to or far away from the query position [39]. In addition, the GPS positions associated with the query images can be rather inaccurate, especially in urban environments [17], requiring the use of a high threshold of tens or even hundreds of meters.

The San Francisco dataset. The publicly available San Francisco (SF) dataset, originally presented in [17], consists of $1,062,468$ street view images, cutout of panoramas using perspective projection from panoramas and denoted as PCI images, taken from the top of a car and 803 query images taken with cell phones by pedestrians. All photos depict downtown San

Francisco (see the gray points in Fig. 3.2 for the distribution of the PCI images). Each PCI is associated with an accurate GPS position and a building ID, generated by back-projecting a 3D model of the city into the image [17]. Similarly, most query images have a GPS position and a list of IDs of the buildings visible in them. Unfortunately, the GPS coordinates of the query photos are not very precise and thus cannot be used as ground truth to measure localization accuracy.

There exist two SfM reconstructions of the San Francisco models [50]. The *SF-0* version of the dataset contains around 30M 3D points, associated with SIFT descriptors [32], reconstructed from $610,773$ images. To create the *SF-1* variant, the database images were histogram-equalized before extracting upright SIFT features, resulting in a model containing roughly 75M points reconstructed from $790,409$ images. For both 3D models, each 3D point can be associated with the building IDs from the database photos it was reconstructed from. Thus, the SF dataset is commonly used to evaluate and compare structure- and image-based localization methods in the context of landmark recognition. However, both models do not provide reference poses for the query images. Using the reference poses we generated enables to evaluate camera pose accuracy on the SF dataset.

3.2.2 Generating reference poses

Without any precise geotags, which are hard to obtain in downtown areas due to multi-path effects, the easiest way to obtain ground truth poses at scale is to use SfM algorithms. We follow this approach. Yet, instead of adding the query images to an existing model, which would require us to solve the localization problem, we generate local reconstructions around the queries which we subsequently geo-register. While we took great care to ensure the accuracy of our pose estimates, there is still a certain error in them. We thus use the term “reference poses” rather than “ground truth poses” to indicate that our poses are a rather precise reference than a centimeter accurate ground truth.

Generating local reconstructions. The first step is to generate SfM reconstructions from the database images around the query images. Unfortunately, the GPS coordinates for the query images provided by the SF dataset are inaccurate with errors up to hundreds of meters [17]. Thus, we determine relevant PCI images for each query photo by exploiting the readily available building IDs. For each query, we perform feature matching against all database photos with a relevant building ID, followed by approximate geometric verification [42, 125]. We visually inspect the 20 images with the largest number of inliers, as long as they have at at least 5 inliers, and select the photo that is visually most similar to the query image for a later consistency check.

There is a strong change in viewpoint between the PCI (taken from the road) and query (taken mostly from sidewalks) images. Thus, finding sufficient matches to include the query

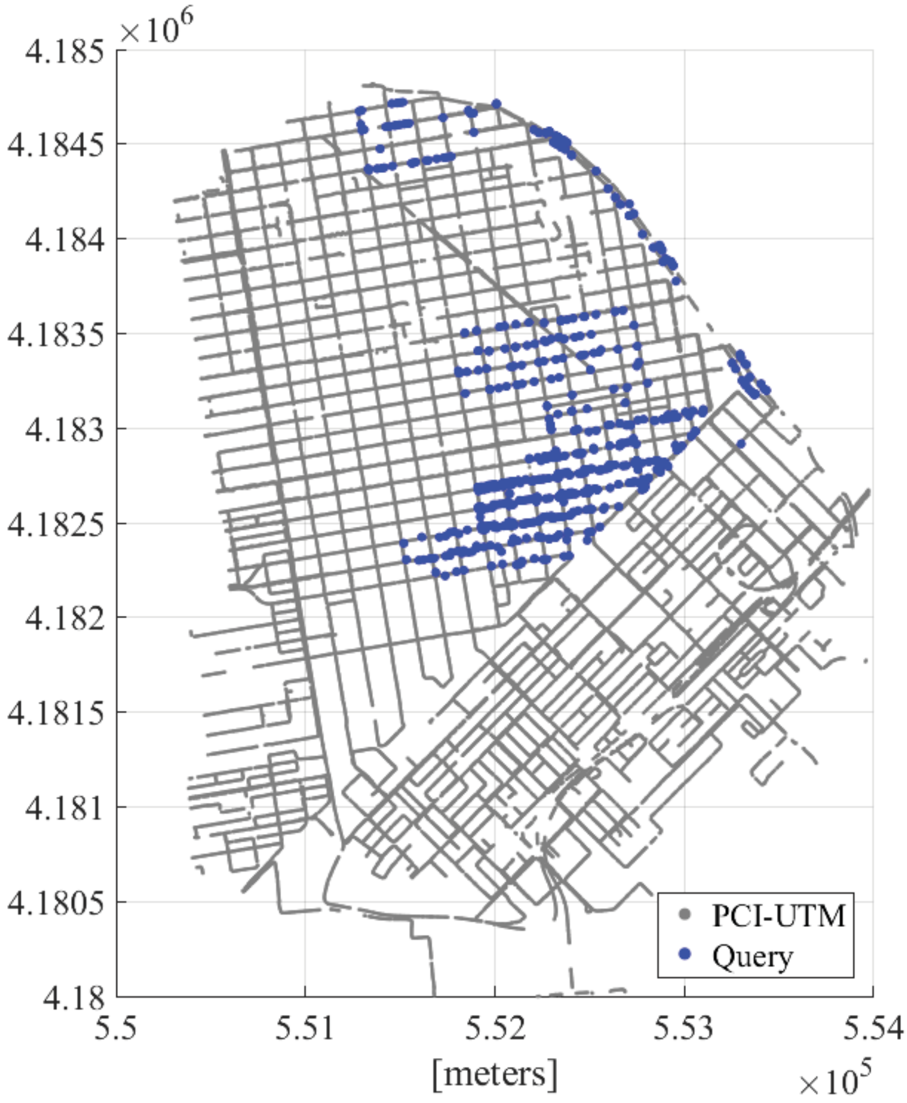


Figure 3.2: **The San Francisco dataset with the reference poses of query images.** We provide the reference poses of query images (blue) which can be used as the ground truth for large-scale localization benchmarks on the SanFrancisco dataset.

image in the local reconstruction can be challenging. To increase the chance of finding enough matches, we thus include additional images that are not part of the original SF dataset. More precisely, we use Google Street View Time Machine (GSVTM) data [20, 40, 88] that covers the same area as the San Francisco dataset. We chose GSVTM instead of GSV images as they provide a denser spacial sampling. GSVTM provides panoramas of $13,312 \times 6,656$ pixels associated with geotag information, more precisely, longitude, latitude, and orientation (heading to the north)[1]. Similar to [17, 125], we cut 24 perspective images of $1,920 \times 1,440$ pixels, with a 60° field-of-view and 50% overlap to the neighboring views, from each panorama. For each query image, we use the 240 perspective images corresponding to the 10 GSVTM panoramas spatially closest to the most relevant PCI image.

We run SfM on the query, PCI images, and GSVTM perspective images. For redundancy, both COLMAP [25] and VisualSFM [23, 24] are used to obtain two SfM reconstructions.

Geo-registration with gravity constraint. In order to obtain the global positions and orientations of the cameras in each local reconstruction, we transformed the local model coordinate system to UTM coordinates. We first convert the GPS tags of the PCI and GSVTM images to UTM. Since the geotags do not include the height of the cameras above ground, we set it to zero. We then estimate the similarity transform between the camera positions in the model and their UTM coordinates.

A naive approach to geo-registration is to calculate a 7DOF similarity transform (scale, rotation, translation) between the SfM camera positions and the UTM coordinates of the database images [65]. However, this results in unstable estimates in degenerated camera configurations. A common degenerate configuration in our setting is that the cameras used for an SfM reconstruction align on a straight line since the car drives down a road.

This degenerated camera configuration problem can be addressed using the gravity direction computed from the SfM model. We first assume that all the PCI and GSVTM perspective images are aligned with respect to the gravity direction in UTM coordinates. Each camera pose in the SfM model and its pitch angle in UTM give a mapping of the gravity direction in UTM to the SfM coordinates. We determine the gravity direction in the SfM coordinates by taking a median of all the mapped gravity directions. Using this median gravity direction, we rotate the SfM model and finally compute a 5DOF similarity transform (1D scale, 1D rotation, and 3D translation), using LO-RANSAC with a 1 meter tolerance threshold[2].

Verification. Besides not being able to register the query image in the model, there are multiple ways a SfM reconstruction might provide an inaccurate estimate for a query's camera

[1] GSVTM provides two location and orientation estimates, the original GPS information and geotags obtained via some alignment. We use the former as they are better aligned with the geotags of the PCI images.

[2] We experimented with three registration approaches based on the geotags of (1) only the PCI images, (2) only the GSVTM images, and (3) all images. All variants show a similar pose accuracy, but the last version registers the largest number of query images.

pose. For example, only few matches might be found or the correspondences might be in an unstable configuration, *e.g.*, all matches are situated in a small region of the query image. Consequently, we verify the poses after the registration process through consistency checks.

Let $\mathcal{Q}$ be the query image for which we want to verify the estimated pose and let $\mathcal{D}$ be the PCI image we selected for it. Using $\mathcal{D}$ and the SF-0 model, also registered to UTM coordinates, we generate a set of 2D-3D matches for the query image $\mathcal{Q}$. From the SF-0 model, we obtain a list of 3D points visible in $\mathcal{D}$. We project these 3D points into $\mathcal{D}$ to obtain 2D pixel positions, which we use to manually annotate the corresponding image positions in the query image. This results in a set of 2D-3D matches and, as a side product, also produces a set of 2D-2D correspondences between $\mathcal{Q}$ and $\mathcal{D}$. To obtain additional correspondences, we manually annotate 20 to 50 2D-2D matches between $\mathcal{D}$ and $\mathcal{Q}$. We use all these 2D-2D matches to compute the relative pose between the two images and use the 2D-3D matches to determine the scale of the translation. The resulting pose in UTM coordinates is then refined using bundle adjustment [138]. Ideally, this procedure should result in a precise estimate of $\mathcal{Q}$'s camera pose. However, it is hard to obtain accurate manually annotated pixel matches, resulting in some inaccuracy on the pose. We thus use it for a consistency check on the *absolute* camera pose. The check accepts a SfM pose if it is inside 10 meters of the position and within 15° of the view angle of the pose obtained from the manual matches.

A second consistency check employs the manually annotated 2D-2D matches between $\mathcal{D}$ and $\mathcal{Q}$. From each of the two SfM models, we extract the essential matrix $\mathtt{E}$ describing the relative pose between the two images. For a given 2D-2D match $(\boldsymbol{x}_{\mathcal{Q}}, \boldsymbol{x}_{\mathcal{D}})$, we measure the pixel distances from the epipolar lines defined by $\mathtt{E}$ and $\mathtt{E}^{-1}$. $\mathtt{E}$ is considered to be consistent with the match if both errors are less than 3 pixels each. We consider a pose obtained by SfM to be consistent with this *relative* check if $\mathtt{E}$ is consistent with at least 10 of the manually annotated matches.

For each query image, a pose obtained by COLMAP or VisualSFM is accepted as a reference pose if it passes one of the two consistency checks. If poses from both COLMAP and VisualSFM pass this test, we select the one estimated by COLMAP.

Statistics. We created manual annotations for 687 out of the 803 query images from the SF dataset. Tab. 3.1 shows statistics on how many SfM poses, obtained with either COLMAP or VisualSFM, pass the two consistency checks. Finally, we obtain **598 reference poses** that are consistent with our manual annotations. For comparison, we only obtained 442 reference poses without using the additional GSVTM images.

Reference pose accuracy. In the next section, we present the image- and structure-based localization methods that we evaluate using our reference poses in Sec. 3.5. In order to draw valid conclusions, it is however necessary to understand the accuracy of these poses. We thus measure the uncertainty of the estimated poses as follows: Using RANSAC, we compute

Table 3.1: Statistics on the consistency of the reconstructed SfM poses with our manual annotations.

Method \Consistency Test	Absolute	Relative	Both
COLMAP	311	553	306
VisualSFM	269	279	170
COLMAP & VisualSFM	228	245	142

multiple poses from a subset of the 2D-3D matches used for each reference pose. From the resulting 100 poses, those supported by more than 80% of the 2D-3D matches are used to measure the differences to the reference pose. The mean median position and orientation errors are 1.01 meters and 2.19°.

The accuracy of any pose estimation approach, including SfM, that minimizes reprojection errors depends on the distance of the camera to the scene. More precisely, the uncertainty of the estimated pose grows roughly quadratically with the distance to the scene. On average, a query image is 10.5 meters away from its selected PCI image and 35.6 meters away from the 3D structure estimated during SfM. We thus consider the reference poses to be reasonably accurate. We also measured the positional gap between the geotags of the PCI and the reconstructed PCI cameras registered in UTM coordinates. The mean average positional discrepancy is 0.39 meters, *i.e.*, the UTM coordinates estimated by SfM reconstruction and registration are consistent with the geotags of the PCI images.

3.3 Existing visual localization methods

The introduction posed the question whether 2D image-based localization approaches can achieve the same pose accuracy as structure-based methods. In other words, we are interested in determining whether an underlying 3D representation is necessary for high localization precision or whether a database of geotagged images can be sufficient.

This section gives summaries of existing localization methods designed in the spirit of 2D image-based approaches (Sec. 3.3.1) or 3D structure-based approaches (Sec. 3.3.2). We evaluate these methods using our new dataset in Sec. 3.5.

3.3.1 2D image-based localization

Disloc [**27, 39**] is a state-of-the-art method based on the BoW paradigm and Hamming embedding [82]. During the voting stage of the retrieval pipeline, Disloc takes the density of the Hamming space into account to give less weight to features found on repeating structures while emphasizing unique features.

We also use the combination of Disloc with the geometric burstiness weighting scheme recently proposed in [39]. Given a list of spatially verified database images found by Disloc, the weighting strategy clusters these photos into places based on their geotags. It identifies features in the query image that are inliers to database photos coming from different places, *i.e.*, features found on repeating structures. Finally, the strategy performs a second re-ranking step where such features have less influence, which has been shown to improve landmark recognition performance.

DenseVLAD [20]. Disloc is based on the BoW paradigm and thus needs to store one entry per each image feature in an inverted file. This quickly leads to high memory requirements for large-scale scenes such as San Francisco. The DenseVLAD descriptor [20] is an example for a state-of-the-art localization algorithm based on compact image representations. Each image is represented by a single VLAD vector [44, 45], resulting in a more compact database representation. The DenseVLAD descriptor is constructed by aggregating RootSIFT [26] descriptors densely sampled on a regular grid in each image. As such, the method foregoes the feature detection stage, which has been shown to lead to more robust retrieval results, especially in the presence of strong illumination changes [20, 136].

NetVLAD [40]. The DenseVLAD descriptor is based on hand-crafted RootSIFT descriptors. In contrast, the NetVLAD representation uses a convolutional neural network to learn the descriptors that are aggregated into a VLAD vector. Training this representation in an end-to-end manner using a weakly supervised triplet loss has been shown to improve place recognition performance over DenseVLAD and other compact image descriptors.

3.3.2 3D structure-based localization

Camera Pose Voting (CPV) [126]. Following [139], CPV assumes that the gravity direction, both in the local coordinate system of the camera and the global coordinate frame of the 3D model, is known together with a rough prior on the camera's height above the ground and its intrinsic calibration. In this setting, knowing the height of the camera directly defines the distance $\text{dist}(p)$ of the camera to a matching 3D point p up to $\pm\varepsilon$, where ε is a small distance modeling the fact that the point might not re-project perfectly into the image. Thus, the camera's center falls into a circular band with minimum radius $\text{dist}(p) - \varepsilon$ and maximum radius $\text{dist}(p) + \varepsilon$ around p. As shown in [126], fixing the final[3] orientation angle of the camera also fixes the position of the camera inside the circular band.

The last observation directly leads to the camera pose voting scheme from [126]: Iterating over a set of discrete camera heights (defined by the coarse height prior) and a set of discrete camera orientations, each 2D-3D match votes for a 2D region[4] in which the camera needs to

[3] The other angles are already fixed by knowing the gravity direction.

[4] Regions account for the discretization of the pose parameters.

be contained. The matches voting for the cell receiving the most votes define a set of putative inliers and the position of the cell, together with the corresponding height and orientation, provides an approximation to the camera pose. This approximation is then refined by applying RANSAC with a 3-point-pose (P3P) solver on these matches. If available, a GPS prior can be used to further restrict the set of plausible cells and thus possible camera positions.

CPV was selected for our evaluation as [126] report state-of-the-art pose accuracy on the Dubrovnik dataset [16] and the state-of-the-art landmark recognition performance on the San Francisco dataset among structure-based localization methods.

Hyperpoints (HP) [46]. Rather than using Lowe's ratio test, which enforces *global uniqueness* of a match in terms of descriptor similarity, the HP method searches for *locally unique* matches [46]. It uses a fine visual vocabulary of 16M words [140] to define the similarity between the descriptor $\mathbf{d}(f)$ of a query image feature f and the descriptor $\mathbf{d}(p)$ of a 3D point p based on a ranking function: p has rank $\mathrm{r}(p, f) = i$ if $\mathbf{d}(p)$ falls into the i-th nearest visual word of $\mathbf{d}(f)$. The point's rank is $\mathrm{r}(p, f) = \infty$ if $\mathbf{d}(p)$ does not fall into any of the $k = 7$ nearest words of $\mathbf{d}(f)$. A 2D-3D match (f, p) is locally unique if there exists no other 3D point p' that is co-visible with p and has $\mathrm{r}(p', f) \leq \mathrm{r}(p, f)$. Two points are co-visible if they are observed together in one of the database images used to reconstruct the model.

Each locally unique 2D-3D match (f, p) votes for all database images observing p and the top-N images with the most votes are considered for pose estimation. Let $\mathcal{D}$ be one of these database images. All matches whose 3D point is visible in $\mathcal{D}$ as well as all matching points visible in nearby images are used for RANSAC-based pose estimation. Two images are considered nearby if they share at least one jointly observed 3D point in the SfM model. Considering points outside $\mathcal{D}$ increases the chance of obtaining more correct matches. Restricting the additional matches to nearby cameras avoids considering unrelated matches, thus avoiding high outlier ratios in RANSAC.

After computing a camera pose for each retrieved database image, the pose with the highest effective inlier count is selected. The effective inlier count takes both the number of inliers and their spatial distribution into account [52].

HP was selected as it represents a hybrid between 2D image-based and 3D structure-based localization methods. In addition, HP also outperforms other structure-based approaches employing retrieval techniques [48, 52, 91] at large scale.

Active Search (AS) [49]. CPV and HP have been designed to operate at large-scale. We compare their performance with Active Search, a state-of-the-art method for efficient localization at small-to-medium scale [49, 93]. AS relies on Lowe's ratio test to identify and reject ambiguous matches. As the ratio test rejects more and more correct matches at scale [50], we expect AS to localize significantly fewer images than CPV and HP. The comparison with AS thus serves to demonstrate the challenges encountered when scaling to larger, more complex scenes.

3.4 Image-based visual localization providing accurate 6DoF camera pose

As mentioned in Sec. 3.1 and summarized in Tab. 1.1, 2D methods are based on image retrieval which only provide a list (ranking) of database images that are relevant to the input image. Therefore, they mostly serve a coarse position estimation of the query approximated by the location of retrieved database images, instead of the accurate camera pose consistent to the 3D model.

In this section, we break down this limitation by showing that the camera pose of a query can be estimated using only 2D image database. We first review baseline location estimation techniques for 2D image-based methods which suggest query location approximated by relevant database images. During experiments in Sec. 3.5, we regard them as baseline methods of pose estimation. We next describe proposed method based on SfM pipeline, which can provide an accurate camera pose of a query using only the query image and a set of retrieved database images.

3.4.1 Approximation for camera location

Nearest neighbor (NN). Traditionally, most 2D image-based methods approximate the pose of the query image by the pose of the most relevant database image, *i.e.*, the database photo with the most similar BoW or VLAD descriptor. We use this strategy as a baseline and refer to it as *Nearest Neighbor (NN) pose*.

Spatial re-ranking (SR). Re-ranking the retrieved database images after spatial verification is known to improve image retrieval performance. As a second baseline, we use the pose of the best-matching database image after verification and refer to this strategy as *Spatial Re-ranking (SR) pose*. We perform spatial verification [42] for the top-200 retrieved images. For Disloc, we exploit the matches computed during the retrieval process while we extract and match RootSIFT features for both VLAD-based methods. For the VLAD-based methods, we re-rank based on the raw number of inliers. For Disloc, we also experiment with re-rank based on the geometric burstiness score.

3.4.2 Local 3D reconstruction for camera pose estimation

The previous two pose estimation strategies only consider the top-ranked database image. They ignore that each 2D-based approach typically retrieves multiple database images depicting the same place. In addition, the geotags of the database photos can also be used to identify a larger set of potentially relevant images. The relative knowledge of database and query images, *e.g.*, local keypoint correspondences and relative camera pose estimated using them, can therefore serve a 3D reconstruction of the local area around the query, which includes 6DoF camera pose of the query.

SfM-on-the-fly (SfM). Inspired by [120], who generate a SfM model from a single photo by repeatedly querying an image database, we use small-scale SfM to obtain a local 3D model around the query image. Poses in the local model can then be converted into global poses by registering the SfM reconstruction into UTM coordinates based on the geotags of the database images. We refer to this strategy as *SfM-on-the-fly (SfM)*. For 2D image-based methods, we generate a small subset from the top-200 retrieved images which are located within 25 meters from the pose obtained via the NN or SR strategy. We use COLMAP on the selected photos to obtain the 3D reconstruction. If COLMAP fails to recover the pose of a query camera, *e.g.*, when the reconstruction fails, we resort to the NN or SR pose.

A naive implementation of SfM-on-the-fly constructs a local model from scratch and ignores the fact that the poses of the database images are available. We thus also evaluate a version (*SfM init.*) that uses these known poses for initialization. This accelerates the reconstruction process and also makes it more stable. Compared to 3D-based methods, this approach achieves a higher pose accuracy.

Another technique to accelerate the local SfM process is to avoid exhaustive matching between all images. In order to reconstruct the query pose, database images with feature matches to the query image are most relevant. We thus also experiment with a *transitive matching strategy (trans.)*. We first match the query image against all retrieved database images. We then match two database images against each other only if each has a sufficient number of matches with the query image.

3.5 Experiments

This section uses our new reference poses to compare the localization accuracy of 2D image- and 3D structure-based methods. After describing the experimental setup and the evaluation protocol, we quantitatively evaluate the different approaches. We then discuss the results and their relevance.

3.5.1 Experimental setup

For Disloc [27, 39], DenseVLAD [20], and NetVLAD [40], we use source code provided by the authors for our evaluation. Disloc uses a visual vocabulary of 200k words trained on a subset of all database images. DenseVLAD uses a dictionary with 128 words also trained on the SF dataset, while NetVLAD uses 64 words. Unfortunately NetVLAD does not provide a version fine-tuned on San Francisco. Instead, we use the variant trained on the Pitts30k dataset [40]. Both DenseVLAD and NetVLAD generate 4,096 dimensional descriptors. For Hyperpoints (HP) [46], Camera Pose Voting (CPV) [126], and Active Search (AS) [49], we use poses estimated on the SF-0 dataset [50] as all methods use an large SfM model to represent the scene. We run AS with vocabularies with 10k and 100k words, trained on the 3D point descriptors of the SF-0 dataset. We denote structure-based models by "(3D)" in the tables and legends.

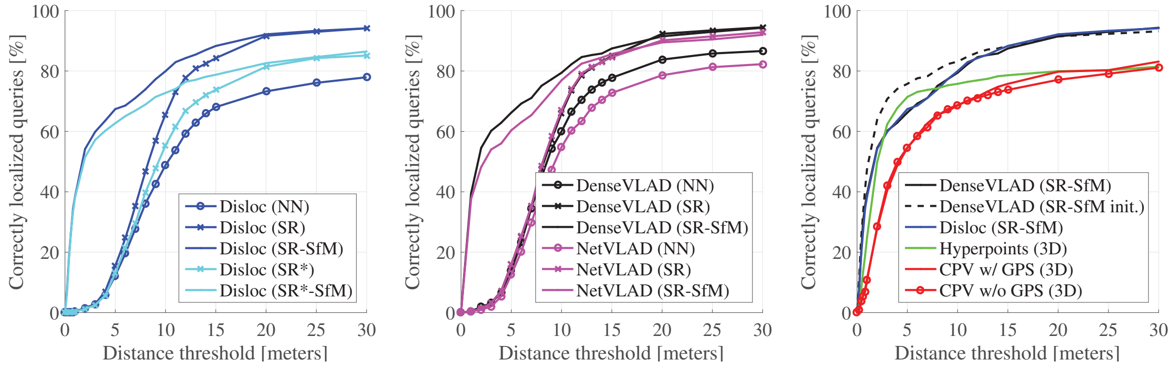
Evaluation metric. We are mostly concerned with the pose accuracy achieved by the different methods. We measure the positional error in UTM coordinates since the local models used to construct the reference poses and the SF-0 reconstruction are registered to this coordinate system. However, the SF dataset only provides geodetic latitudes and longitudes of the cameras and not the altitudes. Thus, there is one degree of freedom in these registrations, namely the height above the plane defined by graticule. Accordingly, we measure the position error in 2D coordinates and evaluate how many images can be correctly localized by the different methods within a certain distance threshold.

In addition to the positional error, measured in meters, we also measure the orientation error. Given the reference camera orientation $\mathtt{R}_{\text{ref}}$ and the estimated query orientation $\mathtt{R}_Q$, both expressed as rotation matrices, we measure the angular error $|\alpha|$ between the two orientations as $2\cos(|\alpha|) = \text{trace}(\mathtt{R}_{ref}^{\top}\mathtt{R}_Q) - 1$ [141].

3.5.2 Quantiative Evaluation

We first evaluate the positional and orientational accuracy achieved by the 2D image-based methods. We compare the accuracy obtained when using the pose of the best-matching database image after retrieval (NN), the pose of the best-matching image after spatial verification (SR), and after local SfM reconstruction (SfM). The latter resorts to the NN (NN-SfM) and SR (SR-SfM) strategies if a pose cannot be estimated from the local model.

Fig. 3.3 and Fig. 3.4 show results for BoW-based methods (a) and VLAD-based methods (b). As can be seen, spatial re-ranking (SR) increases the chance that the top-ranked database image is related to the query, *i.e.*, that the position and orientation of the retrieved database photo is close to the reference pose of the query. As a result, more query images can be correctly localized for larger distance thresholds. However, SR does not improve performance much in the high-accuracy regime. The reason is that the database images of the SF dataset were captured from a car driving on the road while the query photos were taken by pedestrians on the sidewalks. Thus, there is a certain minimal distance between their respective locations. A much better estimate can be obtained when using local SfM models (SfM), boosting the



(a) 2D (BoW) (b) 2D (VLAD) (c) 2D vs. 3D

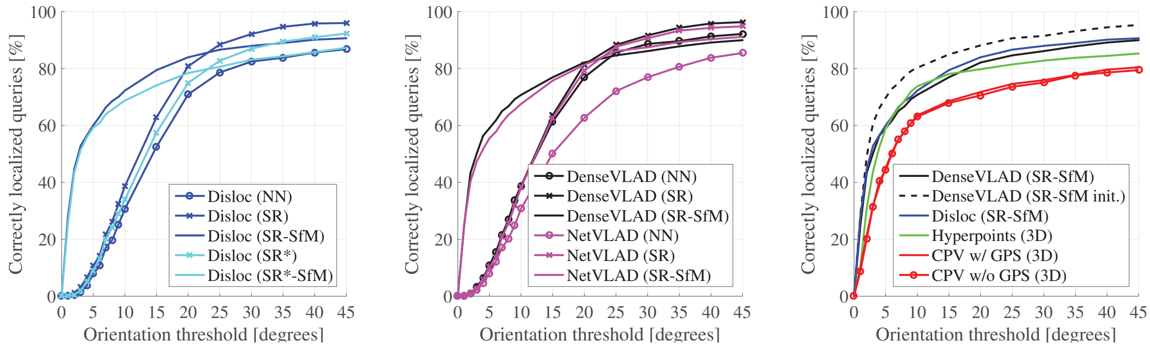
Figure 3.3: **Evaluation of the positional localization accuracy** for BoW-based methods (a), VLAD-based approaches (b), and when comparing 2D- and 3D-based methods (c). Each plot shows the fraction of correctly localized queries (y-axis) within a certain distance to the reference pose (x-axis). As can be seen, using local SfM reconstructions (SfM) to estimate the camera poses allows 2D-based methods (Disloc, DenseVLAD) to achieve a positional accuracy similar or superior to 3D-based methods (Hyperpoints (HP), Camera Pose Voting (CPV)).

percentage of queries localized correctly within 5m from below 20% to about 60%. Similarly, local SfM improves the orientation accuracy by a large margin for smaller thresholds (0° to 20°). Interestingly, SR-SfM degrades the orientation accuracy compared to SR for angular errors above 20° (*c.f.* Fig 3.4). This indicates that the orientation estimates provided by SfM can sometimes be rather inaccurate. As can be seen from the results in (c), using known poses for the database images (SfM init.) rater than computing them from scratch improves pose accuracy.

We observe that Disloc with inter-place geometric burstiness [39] (Disloc (SR*)) shows no additional improvement in comparison with the original Disloc (SR) [27] in this dataset. Disloc (SR*-SfM) uses relatively smaller subsets of images because Disloc (SR*) clusters relevant images by their geotags. This changes the quality of local reconstructions. This is an interesting result as [39] showed that accounting for geometric burstiness leads to a better location recognition performance. Our result thus indicates that better location recognition performance does not automatically translate to better camera pose accuracy.

For the VLAD-based representations, we notice that NetVLAD with the NN strategy performs worse than DenseVLAD (NN). DenseVLAD has the advantage that its vocabulary was trained on SF, while NetVLAD was trained on another dataset. However, their performance is virtually the same in combination with spatial re-ranking and local SfM. In the following, we focus on DenseVLAD and do not use NetVLAD for further experiments.

Fig. 3.3 (c) and Fig. 3.4 (c) compare the positional and orientational accuracy of the best-

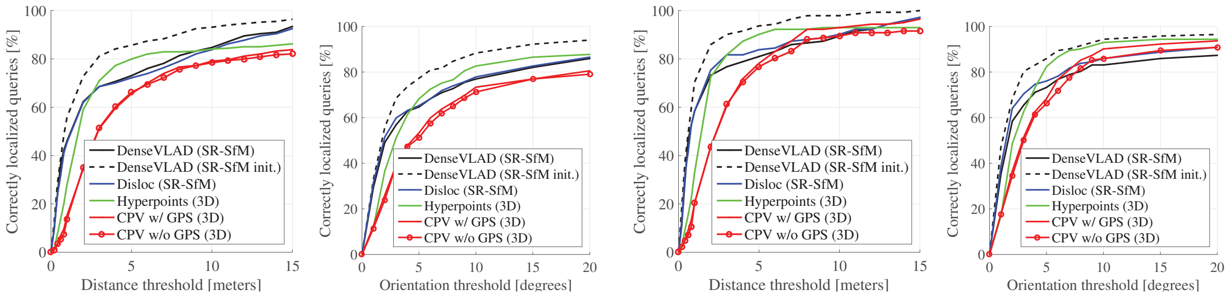


(a) 2D (BoW) (b) 2D (VLAD) (c) 2D vs. 3D

Figure 3.4: **Evaluation of the orientational localization accuracy** for BoW-based methods (a), VLAD-based approaches (b), and when comparing 2D- and 3D-based methods (c). Each plot shows the fraction of correctly localized queries (y-axis) within a certain angular distance to the reference orientation (x-axis). Using local SfM reconstructions (SfM) to estimate the camera poses also allows 2D-based methods (Disloc, DenseVLAD) to achieve a orientational accuracy similar or superior to 3D-based methods (Hyperpoints (HP), Camera Pose Voting (CPV)).

performing 2D-based approaches with the two structure-based methods, Hyperpoints (HP) and Camera Pose Voting (CPV). As can be seen, both Disloc and DenseVLAD perform as good as HP for queries with an smaller error (2m and 5°) when using *SfM* as a post-processing step. 2D-based approaches are able to localize more images overall. If a pose cannot be estimated via local SfM, the 2D-based methods resort to reporting the position of the highest-ranking database image. The overall lower percentage of localized images observed for HP and CPV comes from such cases. For these images, their 2D-3D matching stage fails to produce enough matches for pose estimation. The interesting implication is that it is still possible to find relevant database images even when pose estimation itself fails due to a lack of matches.

Many interesting applications, *e.g.*, self-driving cars, require highly accurate poses. In order to better understand the behavior of 2D-based and 3D-based methods in the high-precision regime, we compare their performance on two subsets of our reference poses. The first subset, containing 334 poses, is constructed from all reference poses for which either COLMAP or VisualSFM provides a pose that passes both consistency checks explained in Sec. 3.2. This subset represents the more accurate among all of our reference poses. The second subset contains all 142 poses where both reconstructed poses pass both tests, thus containing the reference poses most likely to be highly accurate. Fig. 3.5 depicts the performance of the different methods on both subsets. We again observe that HP performs better in the error range 2.5m to 9.0m than DenseVLAD (SfM) and Disloc (SfM). Yet, the best performance is again obtained using the SfM init. strategy. This clearly demonstrates that using known



(a) 334 subset.

(b) 142 subset.

Figure 3.5: **Localization accuracy for subsets of the reference poses**, selected to include more accurate camera poses: (a) reference poses from either COLMAP or VisualSFM passing both consistency checks (334 reference poses) and (b) reference poses where both reconstructions pass both checks (142 poses). For each subset, we evaluate both positional (left) and orientational (right) accuracy for 2D- and 3D-based localization methods.

database poses for initialization increases the robustness of the SfM process. Based on the results, we conclude that large-scale 3D models are not really necessary for highly accurate visual localization.

Other metrics. The previous experiment considered the orientation and position errors separately. Following [56,136], we next jointly consider both errors. Tab. 3.2 shows the percentage of query images that are localized within certain thresholds on the position and orientation error. The best results are clearly obtained by DenseVLAD (SR-SfM init.). The transitive matching strategy accelerates the local SfM process, but also decreases pose accuracy slightly. For more relaxed thresholds (25-30 meters, 20 degrees), 2D-based methods (DenseVLAD / Disloc (SR-SfM)) show better a recall than 3D-based methods (HP and CPV). Overall, image-based method can achieve a similar or higher performance than structure-based methods without a single consistent 3D model.

We next evaluate positional and orientational accuracy in a single unit, *i.e.*, pose accuracy, by computing reprojection errors. As described in Sec. 3.2, the accuracy of any approach that estimates a camera pose by minimizing a reprojection error depends on the scene. Consequently, we can expect larger errors if the image is taken rather far away from the scene. In contrast, the mean reprojection error does not depend the distance to the scene.

To compute the reprojection errors, we retrieve 2D-3D correspondences associated with the query reference pose. The reprojection errors are calculated by projecting 3D points to the image plane of the estimated query pose and computing the distances to the reference 2D points. Tab. 3.3 shows the mean reprojection error for each method. Not surprisingly, the patterns of results are similar to the positional and orientational accuracy evaluations. The higher errors of DenseVLAD / Disloc (SR-SfM) for the 75% quantile result from resorting to

Table 3.2: **Localization performance depending on the positional and orientational errors.** For each pair of thresholds, we provide the percentage of queries that are localized within the thresholds by each method.

	Time	Thresholds [meters, degrees]					
Method	[sec]	5, 5	10, 5	15, 10	20, 10	25, 20	30, 20
DenseVLAD (SR-SfM)	18.38	57.02	58.03	68.56	69.73	80.43	80.77
DenseVLAD (SR-SfM init.)	9.08	**66.89**	**67.39**	**77.76**	**78.43**	**85.79**	**86.12**
+trans.	7.26	63.71	64.55	**77.26**	77.93	85.62	85.95
Disloc (SR-SfM)	18.38	57.19	58.53	69.06	70.40	81.94	82.11
AS, 10K w/o GPS (3D)	0.62	27.42	27.76	33.44	33.44	35.79	35.79
AS, 10K w/ GPS (3D)	0.66	35.79	36.62	43.81	43.98	44.98	45.15
AS, 100K w/o GPS (3D)	0.09	29.43	29.93	35.95	36.12	37.46	37.46
AS, 100K w/ GPS (3D)	0.12	34.28	35.28	42.64	42.64	43.81	43.81
Hyperpoints (3D)	∼3	55.85	57.53	72.24	72.41	76.76	76.92
CPV w/ GPS (3D)	∼3	38.63	43.14	62.21	62.71	70.23	70.74
CPV w/o GPS (3D)	∼3	38.63	42.98	61.20	62.04	69.06	69.23

Table 3.3: **Quantiles for mean reprojection errors**.

Method	25%	50%	75%
DenseVLAD (SR)	100.09	155.75	233.56
DenseVLAD (SR-SfM)	10.92	33.78	165.30
DenseVLAD (SR-SfM init.)	**9.04**	**22.35**	**85.43**
DenseVLAD (SR-SfM init.)+trans.	9.90	24.03	86.95
Disloc (SR)	100.09	157.22	246.18
Disloc (SR-SfM)	10.46	38.60	160.02
Hyperpoints (3D)	14.93	32.21	124.82
CPV w/ GPS (3D)	21.05	46.87	116.99
CPV w/o GPS (3D)	21.18	44.74	122.36

image retrieval if local SfM fails.

Using positional priors. At large scale, it is often reasonable to assume that some coarse positional prior is given, *e.g.*, via GPS / WiFi localization. This prior can then be used to simplify the localization problem by restricting the search space. For example, image-based methods can restrict the search for relevant images to a certain radius around the positional prior of a given query [17]. Similarly, CPV can use such a regional prior to restrict the voting space [126]. We extend AS to use a position prior by restricting 2D-to-3D matching to points within 200 meters of the prior.

As can be seen in Tab. 3.2 and Tab. 3.3, using a pose prior for CPV has little impact on camera pose accuracy. This is an interesting observation and its relevance will be discussed in detail in Sec. 3.5.3. In contrast, AS clearly benefits from the prior, which is due to its use of Lowe's ratio test. The prior allows AS to ignore some 3D points during matching. This results

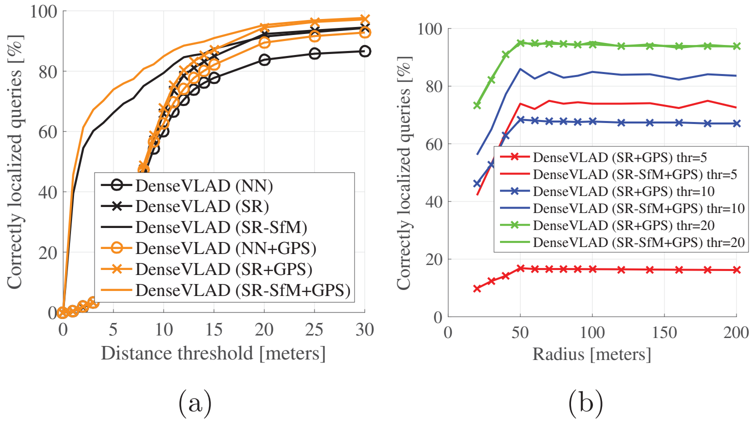


Figure 3.6: **2D image-based localization with and without positional priors.** (a) Each plot shows the percentage of query images (y-axis) localized within a certain distance to the reference pose (x-axis). "+GPS" indicates restricting the search to a 100 meter radius around the given GPS prior. (b) The percentage of queries localized within 10, 20, 30 meters of the reference position (y–axis) obtained by DenseVLAD when varying the search radius (x–axis) around the GPS prior.

in a sparser descriptor space and thus decreases the chance that the ratio test rejects correct matches. Still, AS localizes significantly fewer images than CPV, even with the prior.

The original GPS measurements provided with the SF dataset are rather noisy, with errors of up to 150 meters [17]. These measurements were obtained with rather old hardware and software. We thus generate a more accurate positional prior by randomly sampling a position from a region which centers our reference pose and has a radius of 50 meters. We then incorporate the GPS priors into the 2D image-based methods.

Fig. 3.6 (a) evaluates the localization performances for a search radius of 100 meters. As can be seen from the plots, using a GPS prior improves the localization performance of DenseVLAD.

We next evaluate the robustness of the localization to the search radius. In Fig. 3.6 (b), both DenseVLAD (SR+GPS) and DenseVLAD (SR-SfM+GPS) show the best performances with the 50 meters radius, which is equal to the GPS uncertainty. This result is in line with the observation reported by Chen *et al.* [17]. DenseVLAD (SR-SfM) robustly retains its localization rate at the larger searching radius as it accurately estimates the pose of the query image. In contrast, we notice a significant drop in performance for DenseVLAD (SR) as it approximates the pose of the query through the pose of the top-retrieved database image.

Memory requirements. When dealing with large-scale datasets, the memory required to represent the scene is no longer a negligible issue. In the following, we summarize the memory requirements of each method. Here, we focus on the amount of data that needs to be kept in main memory during processing. For example, the image-based methods need access to the original images for the local SfM stage. However, these images could be read from disk after the initial retrieval step.

Storing 4096-dimensional VLAD [20, 40] for all 1,062,468 database images requires 17.4GB, and the requirements can be reduced further, *e.g.*, using product quantization [142], with negligible loss in performance [20]. For comparison, the DisLoc implementation from [39] requires about 20GB, although its memory footprint could be reduced to about 9.4GB by storing quantized feature geometry [86]. As reported in [46], the Hyperpoints approach requires 4.9GB to store the 3D model information and the visual vocabulary. In contrast, camera pose voting [126] uses all 149.3M SIFT descriptors of the SF-0 model for matching, thus requiring more than 18GB of memory. The memory footprint could be reduced to about 4.9GB by storing a single mean descriptor for each of the 30M 3D points, although this might reduce the localization performance. The two best-performing methods (Disloc and Hyperpoints) show similar localization accuracy but Hyperponts has five times smaller memory footprint. AS requires about 15GB of storage space and the difference in memory requirements for different vocabulary sizes is negligible.

Timings. Tab. 3.5 shows timings for the online components of the different algorithms, evaluated on Dubrovnik dataset. As can be seen, DenseVLAD (SR-SfM) is significantly slower than all other methods. This is unsurprising as it avoids the need for generating and maintaining a single 3D model by computing small 3D models on the fly. It thus trades flexibility for run-time. The corresponding run-times for the San Francisco dataset are shown in Tab. 3.2. Note that local SfM is less efficient on the Dubrovnik dataset due to larger image resolutions, resulting in more extracted features.

Computing the DenseVLAD and NetVLAD descriptors for Dubrovnik's 6044 database images took 2.4h and 0.85h, respectively. While we use existing 3D models for Dubrovnik and SF-0, we expect that reconstructing the datasets from scratch takes less than 1 day and about 1-2 weeks, respectively. HP requires about 3s per image for online processing on SF-0.

Tab. 3.4 provides more detailed timings for the different variants of SfM-on-the-fly discussed in Sec. 3.4, together with the impact of the modifications on pose accuracy. These timings were obtaining for COLMAP, using a GPU for feature extraction and parallelization for matching and reconstruction. As can be seen, most of the time is spent on the incremental reconstruction process. Using known database poses for initialization decreases the reconstruction time by a factor of about 3 while also increasing localization performance. Using transitive matching improves the matching times at a slight reduction of pose accuracy at the stricter thresholds. Feature extraction could be accelerated at the price of memory by pre-extracting features for the database images.

Table 3.4: **Impact of different variations of SfM-on-the-fly on timings and performance.** We provide the average computation time for each component of SfM-on-the-fly, and the percentage of queries that are localized within the thresholds by each method.

	Time [sec]			Thresholds [meters, degrees]					
Method	feature ext.	feature match.	reconstruction	5, 5	10, 5	15, 10	20, 10	25, 20	30, 20
DenseVLAD (SR-SfM)	1.13	2.23	15.03	57.02	58.03	68.56	69.73	80.43	80.77
DenseVLAD (SR-SfM init.)	1.13	2.23	5.73	66.89	67.39	77.76	78.43	85.79	86.12
DenseVLAD (SR-SfM init.)+trans.	1.13	1.80	4.33	63.71	64.55	77.26	77.93	85.62	85.95

3.5.3 Relevance of the Results

To put the results obtained at large scale with our references poses into context, we provide results on the medium-scale Dubrovnik dataset [16]. The 3D model consists of 1.9M 3D points reconstructed from 6044 database images.

Tab. 3.5 compares the DenseVLAD variants with CPV and AS. In addition, we also provide results for PoseNet [55, 135], a learning-based approach. HP is not applicable for this dataset as it was designed for larger-scale scenes where memory consumption and matching quality are issues. On the Dubrovnik dataset, the fine vocabulary of 16M words used by HP already requires more memory than the complete dataset.

As can be seen from Tab. 3.5, combining DenseVLAD with local SfM results in a localization accuracy comparable to Active Search but worse than CPV. The opposite is the case for the larger SF-0 model, where DenseVLAD (SR-SfM) is clearly more precise. The reason is that finding good matches is easy on the smaller and well-textured Dubrovnik dataset while it is extremely challenging for the significantly larger SF-0 model. This is evident when comparing CPV's median positional accuracy on Dubrovnik (0.56m) and SF-0 (>2m). The matching step of local SfM is able to recover matches lost by CPV, enabling more accurate poses at large scale. The pose accuracy of DenseVLAD (SR-SfM) strongly depends on the quality of the local 3D models. Here, the SF-0 model is better suited due to the regular spatial distribution of its database images. In contrast, the spatial density of Dubrovnik's database photos varies strongly, making it harder to obtain good local models for some query images.

An interesting observation can be made from the relative performance between HP and CPV on SF-0. Previously, the SF dataset was used to evaluate the performance of structure-based localization methods in a landmark recognition scenario [46, 50, 126]. In this scenario, an image was considered correctly localized if it observed the correct building as specified by the building IDs provided by the SF dataset. Methods are evaluated based on their recall@95% precision, *i.e.*, based on the percentage of correctly localized images if the algorithm is allowed to make a mistake in 5% of all cases. In this scenario, CPV achieves a recall of 67.5% and 74.2% without and with a GPS prior, respectively. In contrast, HP only obtains a recall of 63.5%. This shows that good performance on the landmark recognition task does not necessarily translate to pose

Table 3.5: Additional comparison on the Dubrovnik dataset [16].

	Time	Quantile errors [m]		
Method	[sec]	25%	50%	75%
DenseVLAD [20] (NN)	1.42	1.4	3.9	11.2
DenseVLAD [20] (SR)	1.43	0.9	2.9	9.0
DenseVLAD [20] (SR-SfM)	∼200	0.3	1.0	5.1
DisLoc [27] (NN)	11.28	1.1	3.7	11.1
DisLoc [27] (SR)	11.29	0.9	2.9	8.9
DisLoc [27] (SR-SfM)	∼200	0.5	1.9	9.4
Camera Pose Voting (CPV) (3D) [126]	3.78	**0.19**	**0.56**	**2.09**
Active Search (3D) [49]	0.16	0.5	1.3	5.0
PoseNet [55, 135]	∼0.005	-	7.9	-

accuracy.

Another interesting observation can be made from the results obtained with CPV and a pose prior. In [126], including a pose prior improved landmark recognition performance. Yet, we observe no improvement in pose accuracy. This behavior is due to a peculiarity of the landmark recognition protocol: In order to increase the recall@5% precision, a large margin between the scores of correct and incorrect results is desirable. CPV uses the GPS prior to restrict the camera pose voting space, thus reducing the number of votes for incorrect poses. This, in turn, allows CPV to better distinguish between correct and incorrect poses. As such, our new dataset closes a crucial gap in the literature as it enables measuring pose accuracy at a large scale.

Regarding the performance of PoseNet, we observe that simply approximating the pose of a query image via the pose of the top-retrieved database image (DenseVLAD (NN)) already provides more accurate pose estimates. This shows that pose regression techniques such as PoseNet currently do not scale well. This is in line with reports from other work, which reports problems when trying to train such methods in more complex scenes [22, 97, 136].

Comparing the results obtained with Active Search on SF-0 (Tab. 3.2) and Dubrovnik (Tab. 3.5), we observe that AS scales well in terms of run-time. Part of this is due to the fact that the query images for Dubrovnik contain about five times more features on average, which compensates for the larger model size of SF-0. Yet, AS localizes significantly fewer images on the larger dataset. This demonstrates the need to also consider pose accuracy when evaluating the scalability of localization methods.

3.5.4 Qualitative Results

We next show some qualitative examples to visually investigate when and how 2D image- and 3D structure-based methods work. Based on the quantitative results, we choose the methods for 2D image- and 3D structure-based localization as Disloc (SR-SfM) and Hyperpoints (HP). We chose (SR-SfM) rather than (SR-SfM init.) to illustrate potential failure cases and since the former only requires coarse geotags for the database images.

Fig. 3.7 and Fig. 3.8 show examples of query images correctly localized within 5m of the reference position by Disloc (SR-SfM) but not HP, respectively localized within 5m by HP but not by Disloc. Similarly, Fig. 3.9 and Fig. 3.10 show examples for which Disloc respectively HP provide pose estimates within 30m whereas the other method is less accurate. HP uses quantized decriptors for matching. As a result, it has problems handling scenes with dominantly repetitive and similar structural elements. In contrast, Disloc (SR-SfM) uses the full feature descriptors and thus handles these scenes better.

However, Disloc (SR-SfM) has problems with accurately localizing images taken rather far away from the scene. This problem is compounded if the scenes are weakly textured. In this scenario, the local 3D models build from a few PCI images via SfM are less precise than the global model build used by HP. This reflects in the localization accuracy of Disloc (SR-SfM). If the viewpoint change between the retrieved PCI images and the query image is too large, the local SfM reconstruction process often fails to register the query image. In these cases, Disloc (SR-SfM) defaults to the pose provided by the (SR) strategy.

3.6 Summary

In this chapter, we have presented the first comparison of 2D image-based and 3D structure-based localization methods regarding their localization accuracy at a large scale. To facilitate this comparison, we have created reference poses for some query images from the San Francisco dataset [17]. As shown in Sec. 3.2, the camera poses in our dataset (reference poses) were computed using relative relationships between query and database images, *i.e.*, via SfM for local areas around the query, and verified using manually annotated correspondences. Therefore, the reference poses can provide reliable 6DoF camera properties which can be used for evaluating visual localization results regarding the general indices to both 2D and 3D methods, *i.e.*, errors of estimated camera poses. To the best of our knowledge, ours is the first dataset that can be used to measure the pose estimation accuracy on a large, complex dataset. Our results show that our dataset closes a crucial gap in the literature as this case is not covered by previous benchmarks and evaluation protocols. For reproducing our results and for further researches in this area, we make our reference poses together with all data and evaluation scripts publicly available[5].

Through the benchmarking on this new dataset (*c.f.* Fig. 3.3 (c), Fig. 3.4 (c), and Fig. 3.5), we observed a principal about the performances of existing 2D and 3D methods in an urban

[5] http://www.ok.sc.e.titech.ac.jp/~torii/project/vlocalization/

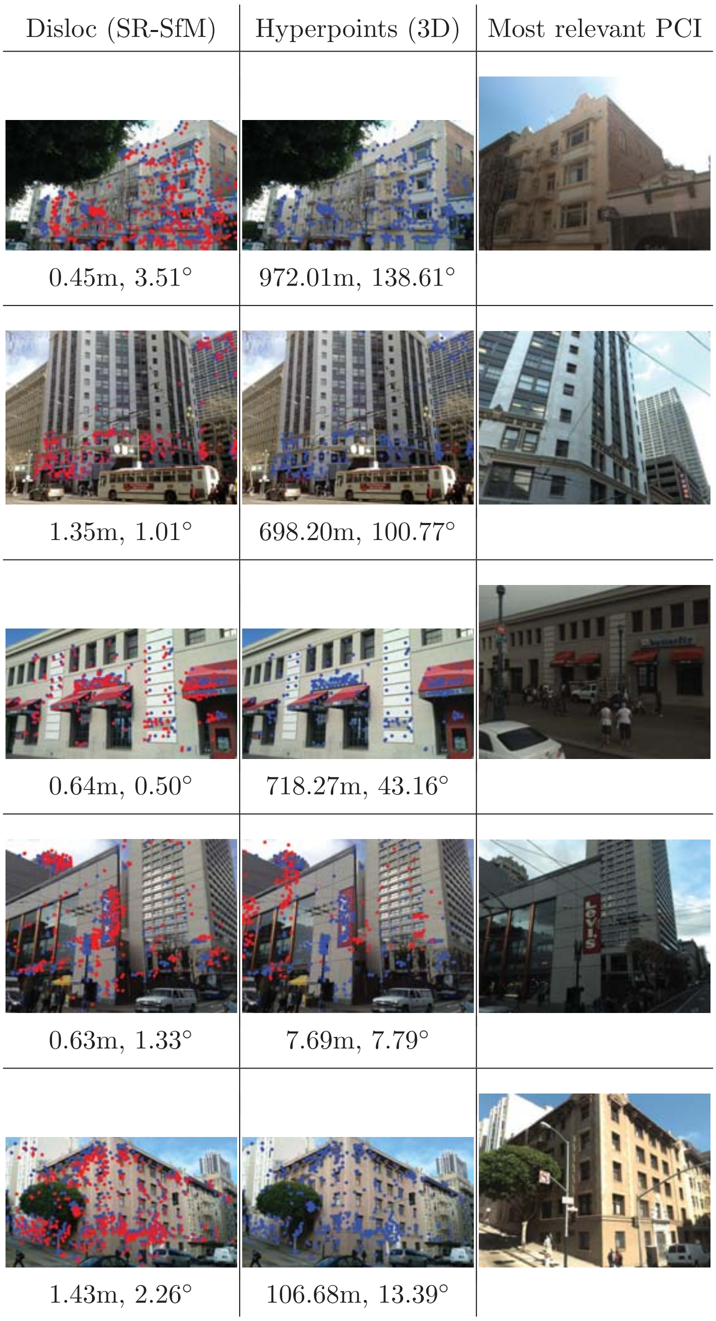


Figure 3.7: **Examples of query images localized within 5m of the reference poses by a 2D image-based method (Disloc (SR-SfM)) (left) but not by a 3D structure-based method (Hyperpoints) (middle).** Colored dots are the reference 2D points used for computing the reference pose (blue) and the 3D points, associated to the reference 2D points, reprojected at the pose estimated by each method (red). The numbers below the images show the positional and orientational errors. The right column shows manually selected database PCI images that are most relevant to the queries.

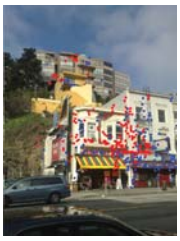
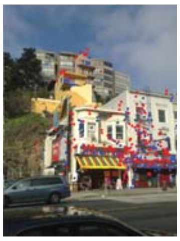
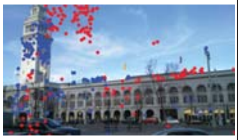
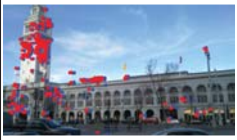
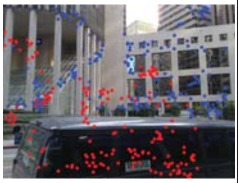
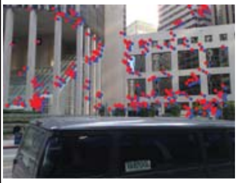
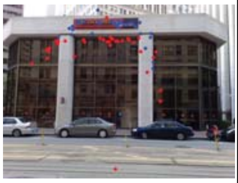
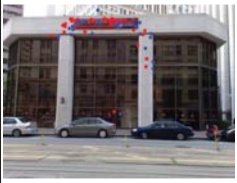
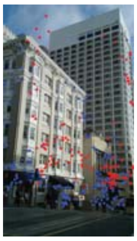
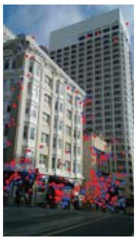
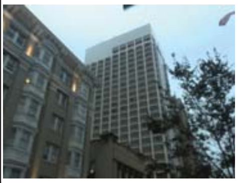
Disloc (SR-SfM)	Hyperpoints (3D)	Most relevant PCI
 5.82m, 5.04°	 1.18m, 2.20°	
 59.16m, 10.57°	 2.19m, 0.75°	
 11.36m, 21.53°	 1.19m, 0.71°	
 5.79m, 5.22°	 1.64m, 8.75°	
 6.59m, 12.98°	 3.84m, 5.28°	

Figure 3.8: **Examples of query images localized within 5m of the reference position by a 3D structure-based method (Hyperpoints) (middle) but not by a 2D image-based method (Disloc (SR-SfM)) (left).** See caption of Fig. 3.7 for details.

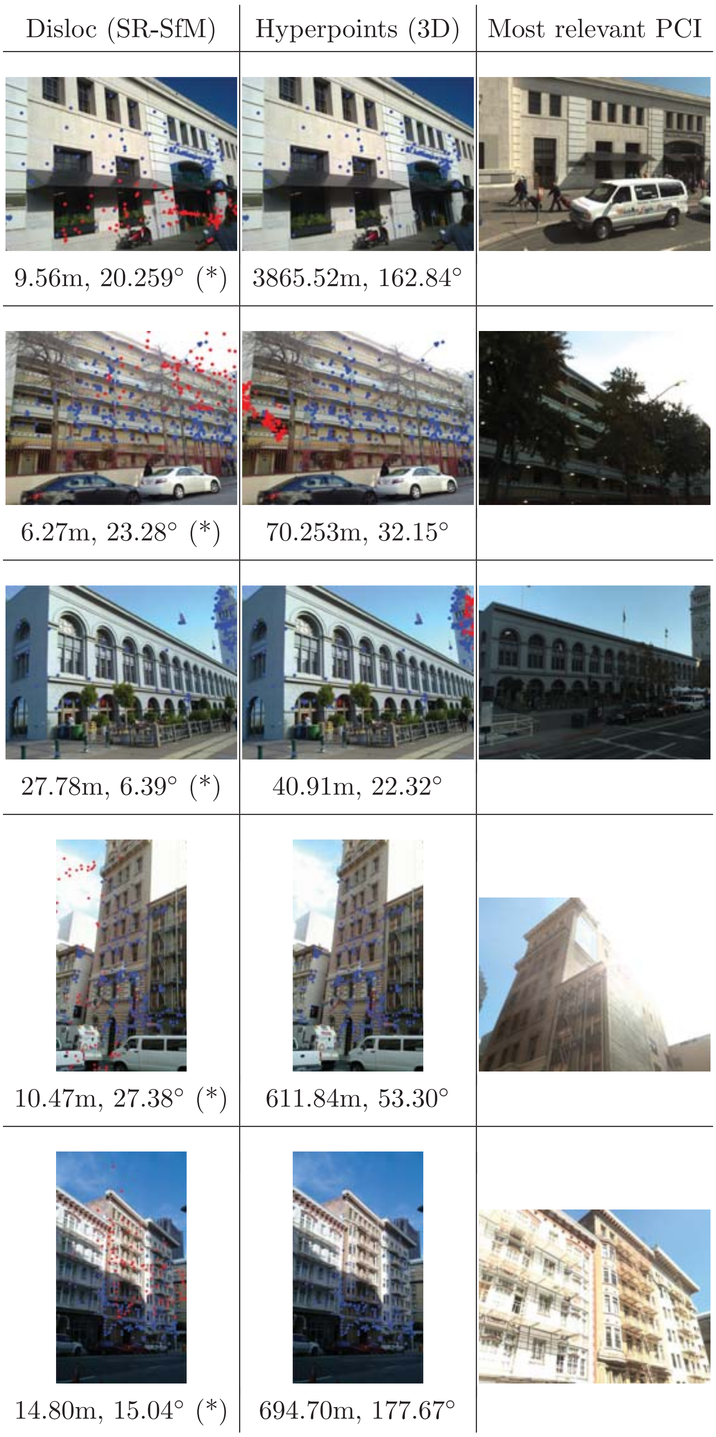


Figure 3.9: **Examples of query images localized within 30m of the reference position by a 2D image-based method (Disloc (SR-SfM)) (left) but not by a 3D structure-based method (Hyperpoints) (middle).** The right column shows manually selected database PCI images that are most relevant to queries. "(*)" besides the results for Disloc (SR-SfM) indicate that local SfM fails so the results are the same as Disloc (SR). See also the caption of Fig. 3.7.

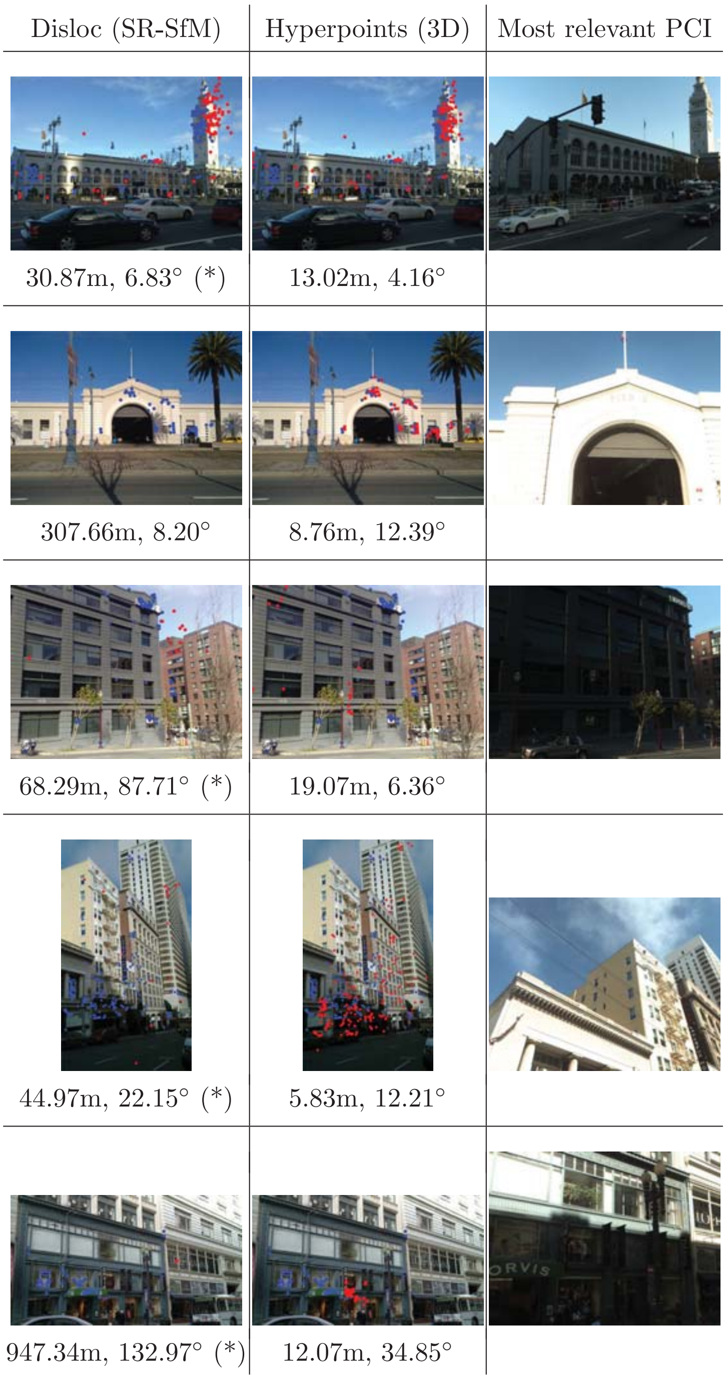


Figure 3.10: **Examples of query images localized within 30m of the reference position by a 3D structure-based method (Hyperpoints) (middle) but not by a 2D image-based method (Disloc (SR-SfM)) (left).** See the caption of Fig. 3.9 for details.

large-scale environment; While 3D structure-based methods give good recall at severe thresholds, *i.e.*, provide accurate camera poses for queries, 2D image-based methods show better recall at coarser thresholds, *i.e.*, 2D methods can provide precise location robustly to image conditions. This is partly because of most 3D structure-based methods (assuming 3D database constructed via SfM) require a set of 2D-3D correspondences for the query, which is sometimes violated in urban environments, *e.g.*, repetitive structures shown in Fig. 3.7 and Fig. 3.9 can cause the lack of local features in images, which results in inaccurate camera pose estimates due to the sparity of 3D database or inaccurate 2D-3D correspondences. To make matters worse, the reference environments depicted in the database can be sometimes changed in query times, *e.g.*, texture changes of buildings, occlusions caused by moving objects such as pedestrians and driving cars. Updating 3D database for these conditions requires high efforts or sometimes just infeasible. On the other hand, 2D image-based methods based on image retrieval (in BoW or VLAD spirits) also use local features in images but rely on more robust image representations (as the histogram of local features or the aggregated feature vector) rather than explicit correspondences. This property can contribute more robust localization performance (*c.f.* Fig. 3.3, Fig. 3.7, and Fig. 3.9) in terms of the place recognition task, *i.e.*, regarding the recall at rough thresholds. Also, 2D image database is generally much easier to maintain / updating against the scene changes, compared to 3D database. However, they only provide an approximation of query location.

The fact led us to design a visual localization pipeline combining both 2D and 3D approaches to take the advantages of both worlds. Our pipeline (discussed in Sec. 3.4) estimates the query camera pose in coarse-to-fine manner; we estimates the coarse location of the queries at first, and then computes fine camera pose using 3D properties in relatively small areas around the location. Inspired by the successes of the local feature matching-based reranking (SR in Fig. 3.3 (a) and (b)) employed as the post-processing for image retrieval, we proposed a camera pose estimation technique for image-based methods purely based on image correspondences. For a image set consisted of a query and set of retrieved database images, we build a small 3D model via SfM pipeline that serves a camera pose estimation using local feature matching and 3D points triangulation. By registering the local 3D model to real-scale map coordinate system using geotags of database images, we got a query location in the 6DoF camera pose format. We can further reduce the additional computational cost for 3D reconstruction and make the 3D model stable assuming the camera pose of the database images are also available.

Several validations in Sec. 3.5 (Fig. 3.3, Fig. 3.4, Fig. 3.5, Tab. 3.2, Tab. 3.3, and Tab. 3.5) demonstrate that our proposed pipeline (SR-SfM) achieves the state-of-the-art localization performance in middle- to large-scale urban environments, at the price of longer run-times during the localization process. Furthermore, an alternative of our pipeline that initializes local 3D model via known camera poses of database images (SR-SfM init.) gives the best performance in total, while preserving the additional computational timings in marginal ranges (*c.f.* Tab. 3.2, Tab. 3.4, and Tab. 3.5).

Altogether, we conclude this chapter by answering the question in Sec. 3.1; A large 3D model depicting the whole scene is actually not necessarily needed to estimate an accurate 6DoF pose of a query. Our pipeline exploiting image retrieval and 3D reconstruction for local areas around the query can achieve comparable (or higher) performance compared to existing methods assuming the large 3D database, while preserving low-cost 2D image database. Still, there is room for improvement in terms of pose accuracy (*c.f.* Fig. 3.8 and Fig. 3.10). This can potentially be addressed by applying robust camera pose estimation using reliable image information, *e.g.*, point-line correspondences [143], or employing recent deep learning-based pose estimation techniques [61, 64, 144] for our local 3D reconstruction stage.

Chapter 4

Visual localization in a large-scale indoor environment

4.1 Indoor visual localization scenarios with dense 3D database

Autonomous navigation inside buildings is a key ability of intelligent robotic systems [1, 145]. Successful navigation requires both to localize a robot and to determine a path to its goal. One approach for solving the localization problem is to build a 3D map of the building and then use a camera[1] to estimate the current position and orientation of the robot (Fig. 4.1). Imagine also the benefit of an intelligent indoor navigation system that helps you find your way, for example, at Chicago airport, Tokyo Metropolitan station, or a convention center. Besides intelligent systems, the *visual localization* problem is also highly relevant for any type of Mixed Reality applications, including Augmented Reality [2, 146, 147].

Due to the availability of datasets, *e.g.*, obtained from Flickr [50] or captured from autonomous vehicles [17, 148], large-scale localization in urban environments has been an active field of research [2, 17, 27, 37–40, 46, 48–53, 55, 65, 88, 90, 126, 133, 136, 139]. In contrast, indoor localization [1, 56, 57, 59, 70, 71, 106, 149] has received less attention in the last years. At the same time, indoor localization is, in many ways, a harder problem than urban localization:

1. Due to the ***short distance to the scene***, even small changes in viewpoint lead to large changes in image appearance. For the same reason, occluders, such as movable furniture, people, and pillars, often have a stronger impact compared to urban scenes. Thus, indoor localization approaches have to handle significantly larger changes in appearance between

[1] While RGBD sensors could also be used indoors, they are often too energy-consuming for mobile scenarios or have only a short-range to scan close-by objects (faces). Thus, purely RGB-based localization approaches are also relevant in indoor scenes. Obviously, indoor scenes are GPS-denied environments.

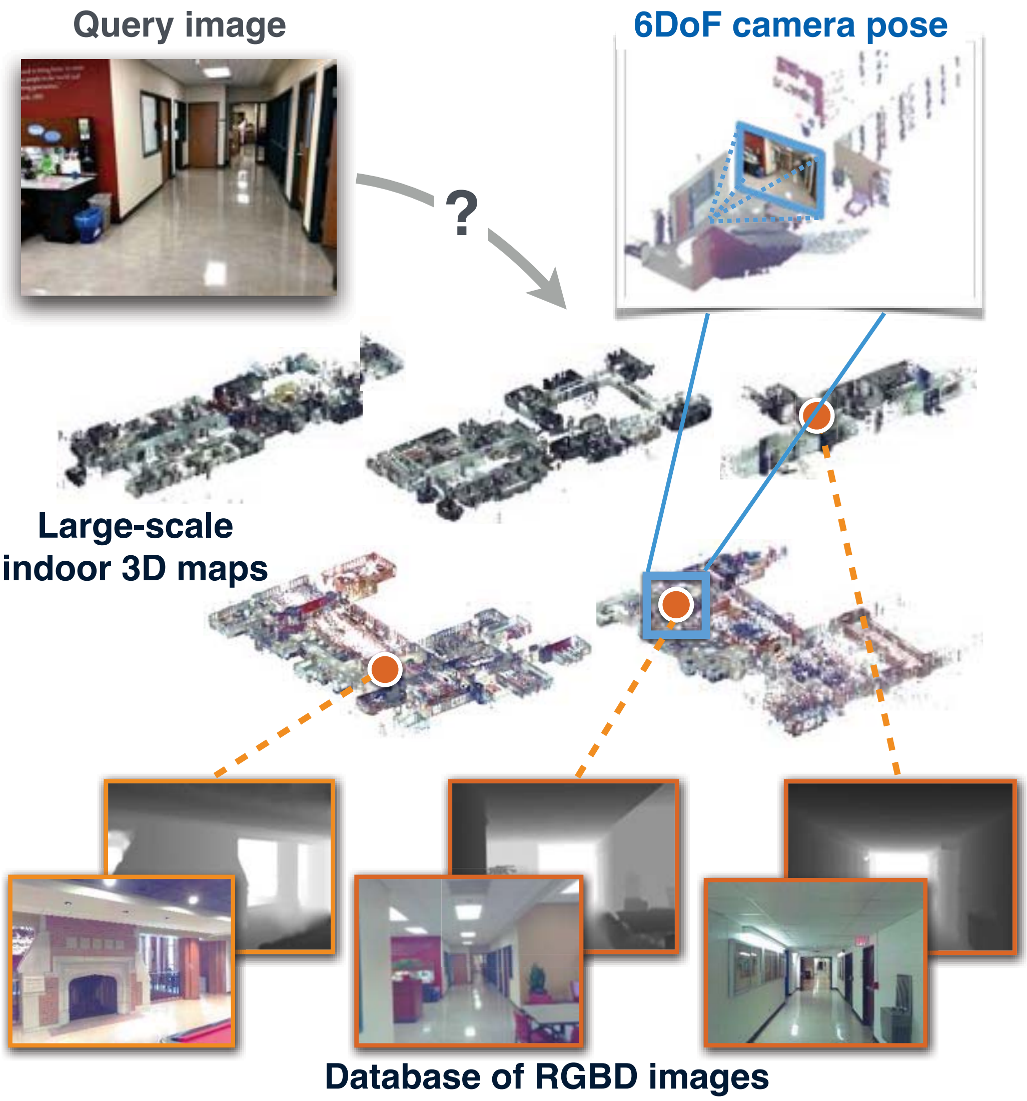


Figure 4.1: **Large-scale indoor visual localization.** Given a database of geometrically-registered RGBD images, we predict the 6DoF camera pose of a query RGB image by retrieving candidate database images, estimating candidate camera poses, and selecting the best matching camera pose. To address inherent difficulties in indoor visual localization, we introduce the "InLoc" approach that performs a sequence of progressively stricter verification steps ***based on dense information***.

a query and reference images.

2. Large parts of indoor scenes are ***textureless***, *e.g.*, walls, ceilings, floors, and windows, where sparse local features hardly exist. As a result, local feature-based localization methods often do not provide sufficient information for camera pose estimation [52] in indoor scenes.

3. To make matters worse, buildings are often ***highly symmetric*** with ***many repetitive elements***, both on large (similar corridors, rooms, etc.) and small (similar chairs, tables, doors, etc.) scale. While structural ambiguities also cause problems in urban environments, they often only occur in larger scenes [27, 37, 39].

4. The appearance of indoor scenes changes considerably over the course of a day due to the ***complex illumination conditions*** (indirect light through windows and active illumination from lamps).

5. Indoor scene structure is often ***highly dynamic over time*** as furniture and personal effects are moved through the environment. In contrast, the overall appearance of building facades does not change too much over time.

On the other hand, indoor environments often be represented by rich 3D point clouds composed of a set of RGBD images [19, 56, 57, 67, 68, 105, 150]. Obtained 3D database provides further more accurate and dense 3D structures of the scene, *i.e.*, pixel-wise depth values with centimeters-level accuracy, compared to the 3D database obtained via SfM [14–16, 18, 50], thus can potentially lead more accurate camera pose estimates for visual localization problem. Furthermore, the 3D database also includes additional properties in other modalities such as surface normal [68, 69] and semantic segmentation [68, 69, 105] for each 3D point, while predicting those modalities from a given image is a growing research area related to recent deep learning techniques [151–154]. Such additional information can also contribute to each component in visual localization, such as feature detection and description [41, 97], feature association [12, 155–160], image retrieval [41, 97, 161–164], and pose estimation [156, 160, 162, 165]. The benefits of using different modalities for visual localization in indoor environments are also illustrated in Fig. 4.2. However, exploiting dense 3D structures (and additional information in other modalities) to visual localization is still a non-trivial task, since existing methods relying on the 3D model mostly assume a rather sparse 3D point clouds obtained via SfM and often use local features extracted from images again sparsely.

Scenario. In this chapter, we propose a novel visual localization pipeline designed for a single image input captured in large-scale indoor environments. Fig. 4.1 illustrates our scenario of visual localization in an indoor environment. In contrast to the previous chapter that assumes image database in outdoor environments, we here assume the target scene is pre-captured as the

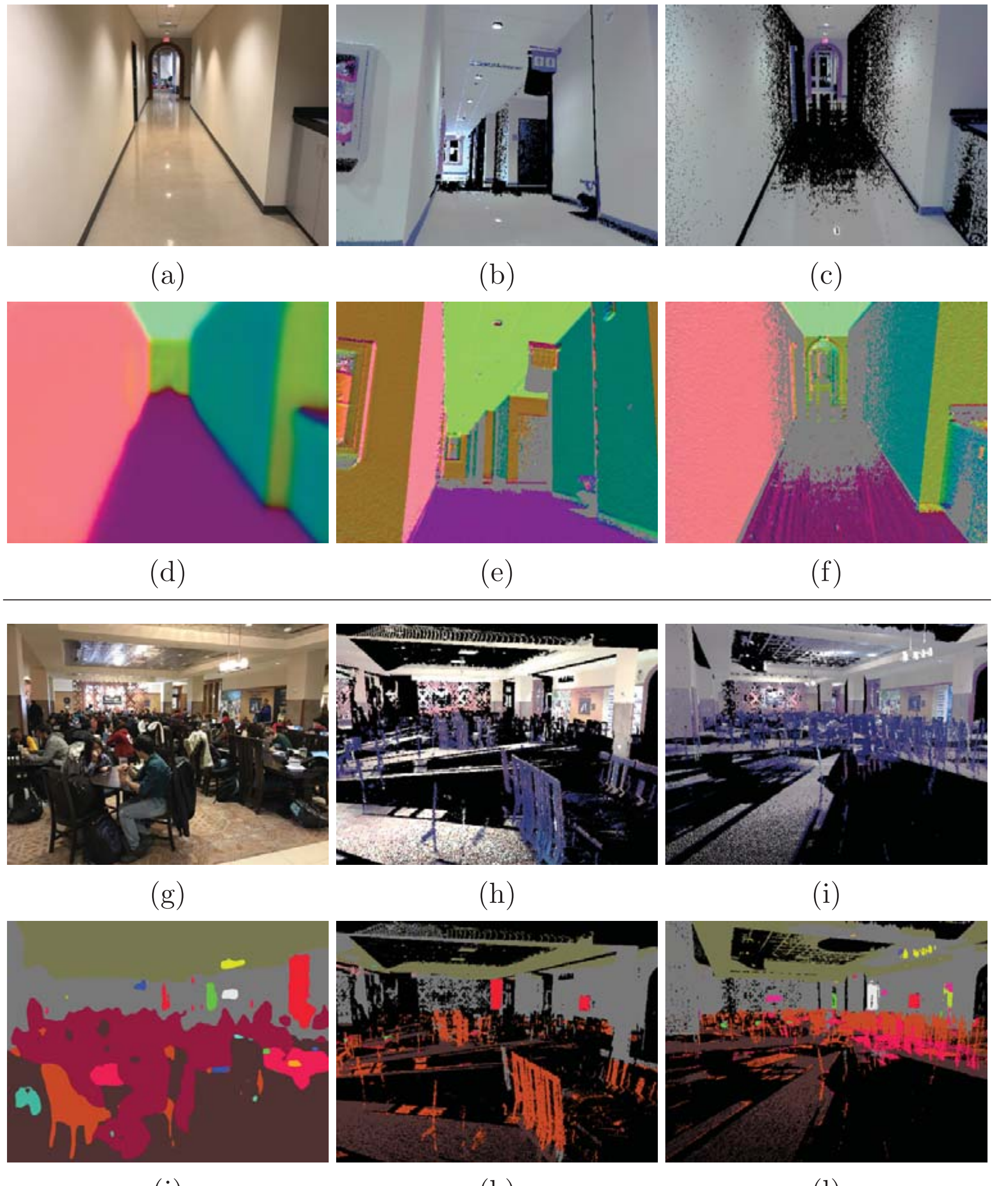


Figure 4.2: **Using further modalities for indoor visual localization.** Given a set of camera pose estimates for a query image (a, g), we seek to identify the most accurate estimate. (b, h) Due to severe occlusion and weak textures, a state-of-the-art method [22] fails to identify the correct camera pose. To overcome those difficulties, we use several modalities along with visual appearance: (top) surface normals and (bottom) semantics. (c, i) Our approach verifies the estimated pose by comparing the semantics and surface normals extracted from the query (d, j) and database (f, l).

3D database composed of a set of RGBD images registered to a known floor map[2]. While large-scale indoor scenes contain diverse circumstances with several inherent difficulties, applications for indoor situations such as AR and robot navigation often require much more accurate pose information than outdoor situations, *i.e.*, within few centimeters and few degrees. Therefore, we firstly aim to construct a general localization pipeline that provides accurate camera pose exploiting the rich database properties. Secondly, we investigate how the pipeline can be extended to address several severe conditions.

Motivated by the successes in large-scale outdoor environments (Chap. 3), we again employ progressive localization steps that first recognizes the rough location of the query and then estimates fine camera pose. In addition, we take advantages of "dense" properties of the database by using ***densely extracted features*** in each step of our localization pipeline. Proposed pipeline named **InLoc** starts with an image retrieval step, using a compact image representation [40] that scales to large scenes. Given a shortlist of potentially relevant database images, we apply two progressively more discriminative geometric verification steps: (i) We use ***dense matching*** of CNN descriptors that capture spatial configurations of higher-level structures (rather than *individual sparse local features*) to obtain the correspondences required for camera pose estimation. (ii) We then apply a novel ***pose verification step*** based on ***virtual view synthesis*** [106] that can accurately verify whether the view from estimated pose is consistent with the query image by dense pixel-level matching.

Contributions. The key novelty of our approach lies in carefully introducing dense feature extraction and matching in a sequence of progressively stricter verification steps. The proposed method significantly outperforms previous state-of-the-art methods in a large-scale indoor environment, showing an **absolute improvement of 17–20%** in the percent of correctly localized queries within a 0.25–0.5m error. The most effective part in InLoc pipeline is the novel pose verification stage. Based on virtual view synthesis using a highly accurate and dense 3D point cloud around the relevant location of a query (local 3D point cloud), the scheme achieves absolute 21% performance gains compared to previous stage. We also investigate the impact of other modalities employed in the pose verification step, which give several new insights of visual localization for indoor scenarios, and point potential future works.

Another contribution in this chapter is constructing a dataset for visual localization in a large-scale indoor environment. Historically, the datasets used to evaluate indoor visual localization were restricted to small, often room-scale, scenes [56, 57, 106]. Driven by the interest in semantic scene understanding [67, 68, 105] and enabled by scalable reconstruction techniques [166–168], large-scale indoor datasets covering multiple rooms or even whole build-

[2] Please note that 3D map for indoor environments are relatively easy to capture and update compared to that of outdoor scenes, thanks to the recent availability of new sensor devices such as ToF sensors and laser range scanners.

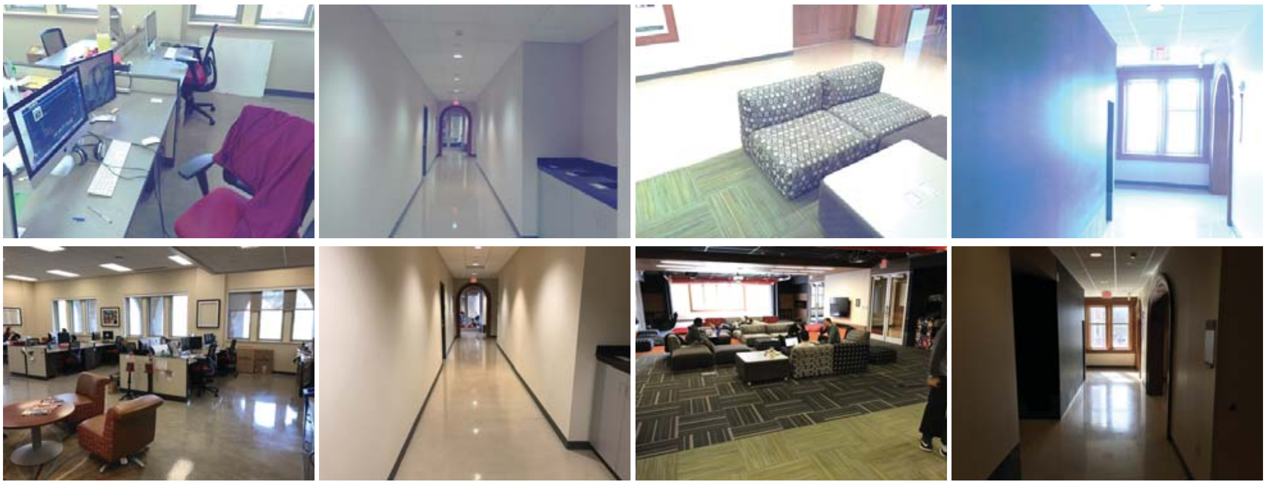


Figure 4.3: **Example images from our InLoc dataset**. (Top) Database images. (Bottom) Query images. The selected images show the challenges encountered in indoor environments: even small changes in viewpoint lead to large differences in appearance; large textureless surfaces (*e.g.*walls); self-repetitive structures (*e.g.*corridors); significant variation throughout the day due to different illumination sources (*e.g.*, active vs. indirect illumination).

ings are becoming available [19, 57, 67–70, 105, 150]. However, most of these datasets focus on reconstruction [19, 150] and semantic scene understanding [67–69, 105], and are not suitable for localization. In contrast to those existing indoor datasets, our new dataset (named as **InLoc dataset**) has two important properties. First, the dataset is large-scale, capturing two university buildings which consist of multiple rooms and floors. Second, the query images are acquired using a smartphone several months after the date of capture of the reference 3D model, and at a different time of the day. As a result, the query images and the reference 3D model often contain large changes in scene appearance due to the different layout of furniture, people, and different illumination, representing a realistic and challenging indoor localization scenario. We manually annotate 6DoF reference pose for each of queries as in Sec. 3.2, which enables us to validate the performance of localization in terms of camera pose accuracy.

This chapter is organized as follows; In Sec. 4.2, we provide details of our novel indoor dataset for visual localization; Sec. 4.3 describes each step of our novel InLoc localization pipeline that is designed for large-scale indoor environments; In addition, we try several modifications for our pose verification step exploiting additional information, geometric properties and semantics, namely (Sec. 4.4). Sec. 4.5 demonstrates the performance of proposed pipeline in our dataset. Each step of our pipeline designed by "dense" spirits gives progressive performance gains and outperforms existing methods at the end. We also tested variants of our pose verification step using other modalities, which gives new insights; We add summaries and conclude this chapter in Sec. 4.6.

Table 4.1: Statistics of the **InLoc dataset**.

	Number	Image size [pixel]	FoV [degree]
Query	356	4,032×3,024	65.57
Database	9,972	1,600×1,200	60

4.2 The InLoc dataset for indoor visual localization

This section provides details of our InLoc dataset for evaluating visual localization performance. Sec. 4.2.1 summarizes statistics and typical features of the images in InLoc dataset. We then introduce a detailed procedure of generating reference poses for query images in Sec. 4.2.2.

4.2.1 Overview

Our dataset is composed of a database of RGBD images [19] geometrically registered to floor maps, augmented with a separate set of RGB query images taken by hand-held devices to make it suitable for the task of indoor localization (Fig. 4.3). The provided query images are annotated with manually verified groundtruth 6DoF camera poses (reference poses) in the global coordinate system of the 3D map.

Database. The base indoor RGBD dataset [19] consists of 277 RGBD panoramic images obtained from scanning two buildings at the Washington University in St. Louis with a Faro 3D scanner. Each RGBD panorama has about 40M 3D points in color. The base images are divided into five scenes: DUC1, DUC2, CSE3, CSE4, and CSE5, representing five floors of the mentioned buildings, and are geometrically registered to a known floor plan [19]. The scenes are scanned sparsely on purpose, to cover a larger area with a small number of scans to reduce the required manual work, as well as due to the long operating times of the high-end scanner used. The area per scan varies between 23.5 and 185.8 m^2. This inherently leads to significant view changes between query and database images when compared with other existing datasets [57, 70, 106][3].

For creating an image database suitable for indoor visual localization evaluation, a set of perspective images is generated by following the best practices from outdoor visual localization [17, 20, 133]. We obtain 36 perspective RGBD images from each panorama by extracting standard perspective views (60° FoV) with a sampling stride of 30° in yaw and ±30° in pitch directions, resulting in about 10K perspective images in total (Tab. 4.1).

Query images. We captured 356 photos using a smart-phone camera (iPhone 7), distributed only across two floors, DUC1 and DUC2. The other three floors in the database are not

[3] For example, the dataset from [57] covers only a single floor, and the area covered per database image is less than 45 m^2.

represented in the query images, and play the role of confusers at search time, contributing to the building-scale localization scenario. Note that these query photos are taken at different times of the day, to capture the variety of occluders and layouts (*e.g.*, people, furniture) as well as illumination changes. Our dataset contains significant challenges, such as repetitive patterns (stairs, pillars), frequently appearing building structures (doors, windows), furniture changing position, people moving across the scene, and less textured and highly symmetric areas (walls, floors, corridors, classrooms, open spaces).

4.2.2 Reference pose generation

For all query photos, we estimate 6DoF reference camera poses with respect to the 3D map. Inspired by reference pose creation in an outdoor scene (discussed in Sec. 3.2), we manually select relevant 3D scan for each query at first. Since accurate 3D point cloud (comes from RGBD panoramic scan) is available, we compute 6DoF camera pose by obtaining 2D-3D correspondences (by feature matching and manual annotations) and solving PnP problem.

Each query camera reference pose is computed as follows:

1. *Selection of the visually most similar database images.* For each query, we manually select one panorama location which is visually most similar to the query image using the perspective images generated from the panorama.

2. *Automatic matching of query images to selected database images.* We match the query and perspective images by using affine covariant features [33] and nearest-neighbor search followed by Lowe's ratio test [32].

3. *Computing the query camera pose and visually verifying the reprojection.* All the panoramas (and perspective images) are already registered to the floor plan and have pixel-wise depth information. Therefore, we compute the query pose via Perspective-3-Points (P3P)-RANSAC [28], followed by nonlinear least-square optimization minimizing the reprojection error [138], using correspondences between query image points and scene 3D points obtained by feature matching. We evaluate the obtained poses visually by inspecting the reprojection of edges detected in the corresponding RGB panorama into the query image (see examples in Fig. 4.4).

4. *Manual matching of difficult queries to selected database images.* Pose estimation from automatic matches often gives inaccurate poses for difficult queries which are, *e.g.*, far from any database image. Hence, for queries with significant misalignment in reprojected edges, we manually annotate 5 to 20 correspondences between image pixels and 3D points and apply step (iii) on the manual matches.

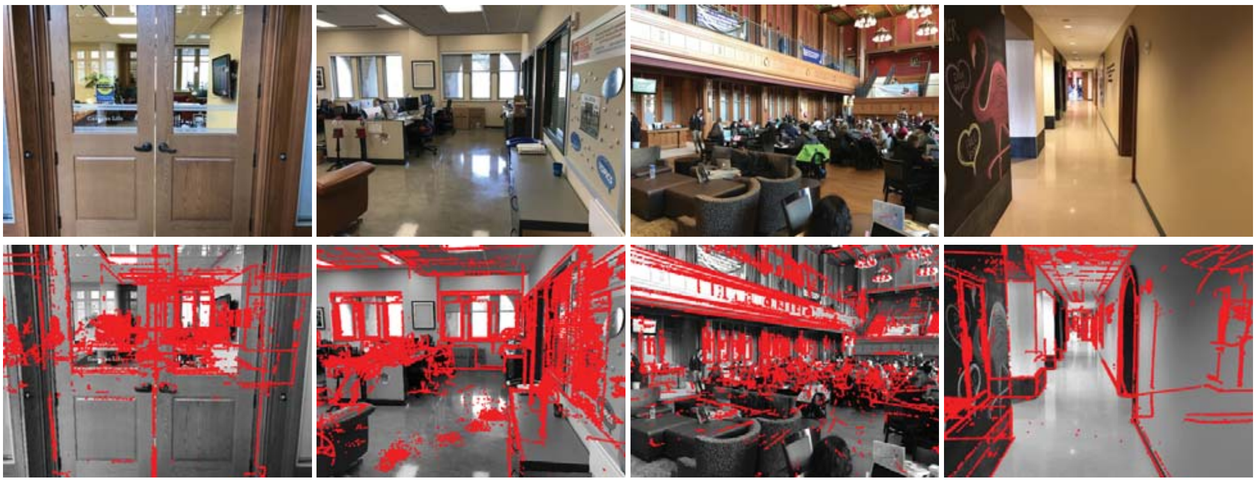


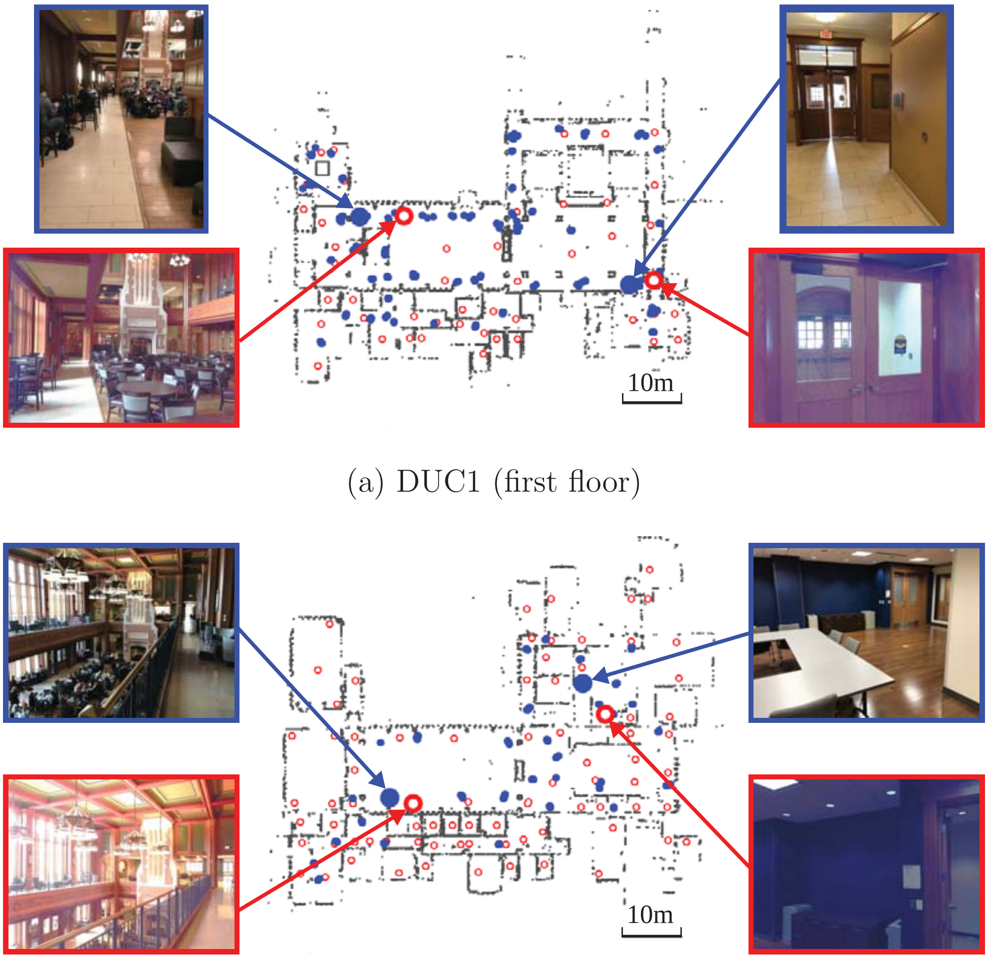
Figure 4.4: **Examples of query images and verified reference poses.** Each column show a query image on top and the same image with database edges projected onto it in the bottom.

5. *Quantitative and visual inspection.* For all estimated poses, we measure the median reprojection error at the Sobel edges detected on both query and database images. For each 3D point on an edge pixel in a database image, we measure the error by projecting edges into the query image. The reprojection error is defined as the distance between the database edges and the nearest edge pixel detected in the query image. After removing correspondences with gross errors (with distance over 20 pixels) due to, *e.g.*, occlusions, we manually inspect the reprojected edges in the query image that has under 5 pixels median reprojection error. Then we finally accept **329 reference poses** out of the 356 query images.

Fig. 4.5 shows the accepted queries and database images in our InLoc dataset. The query images are distributed across two floors (DUC1 and DUC2) that cover an area of ≈ 100,000 ft^2 (9,290 m^2) each [19], and are taken from significantly different poses than the database scans.

Merging the two floors. The two floors in the InLoc dataset, namely DUC1 and DUC2, share a common space in the form of a large room spanning two floors (cf. Fig. 4.5). Thus, a localization method that estimates the 6DoF pose of a query in this area can theoretically obtain an accurate query pose via a database image from either floor. However, the original dataset [19] provides each floor in a separate 3D model. We thus compute the alignment between DUC1 and DUC2 to obtain a single consistent 3D model for both floors, using the scanned 3D points.

For the 45 query images taken in the shared space, we first compute two reference poses registered in each of the two floors, DUC1 and DUC2. In other words, we get scanned 3D points



(a) DUC1 (first floor)

(b) DUC2 (second floor)

Figure 4.5: **Query reference positions in the InLoc dataset.** The 329 reference poses of query images (blue dots) are plotted on the 3D maps (grey dots) that are generated by panoramic 3D scans at 277 distinct positions (red circles).

with respect to the query camera coordinate system from both models. By projecting 3D points into the query image, we obtain 3D-3D correspondences between the DUC1 and DUC2 models. Using these correspondences, we compute the 7DoF similarity transform between DUC1 and DUC2 via LO-RANSAC [30, 169]. Registration errors of the merged 3D points are comparably small (4.8 cm on average), while the average error for scans on the same floor are 4.0 cm on average. Finally, we re-compute the reference poses of the queries in the common space using all the merged 3D points.

4.3 InLoc: indoor visual localization with dense matching and view synthesis

4.3.1 Concept

We propose a new method for large-scale indoor visual localization. We address the three main challenges of indoor environments:

(1) Lack of sparse local features. Indoor environments are full of large textureless areas, where sparse feature extraction methods detect very few features. As a result, local features are clustered in small, well-textured parts in the image, leading to potentially unstable configurations for pose estimation. To overcome this problem, we use *multi-scale dense CNN features* for both image description and feature matching. Our features are generic enough to be pre-trained beforehand on (outdoor) scenes, avoiding costly re-training, *e.g.*, as in [55, 59, 93], of the localization pipeline for each particular environment.

(2) Large image content changes. Indoor environments are cluttered with movable objects and contain complex 3D structures, which cause severe occlusions when viewed from a close distance. The most similar images obtained by retrieval may therefore be visually very different from a query image. We handle this problem by exploiting *dense feature matching to collect as much positive evidence as possible.* We employ image descriptors extracted from a CNN architecture that can match higher-level structures of the scene rather than relying on matching individual local features. In detail, our pose estimation step performs coarse-to-fine dense feature matching, followed by geometric verification and estimation of the camera pose using P3P-RANSAC.

(3) Self-similarity. Indoor environments are often very self-similar, due to highly symmetric building natures and standardized objects. Existing matching strategies count the positive evidence, *i.e.*, how much of the image (or how many inliers) have been matched, to decide whether two images match. This is, however, problematic as large textureless areas can be matched well, hence providing strong (incorrect) positive evidence. Instead, we propose to count also the *negative evidence*, *i.e.*, what portion of the image does not match, to decide whether two views are taken from the same location. We perform *explicit pose estimate verification based on view synthesis*, which compares the query image with a virtual view of the 3D model rendered from the estimated camera pose, thus taking both matching and non-matching pixels across the entire query image into account. As shown by our experiments, this approach is orthogonal to the choice of local descriptors: The proposed verification by view synthesis is consistently showing a significant improvement regardless of the choice of features used for estimating the pose.

The pipeline of InLoc has the following three steps. Given a query image, (1) we obtain a set of candidate images by finding the N best matching images from the reference image database registered to the map. (2) For these N retrieved candidate images, we compute the query poses using the associated 3D information that is stored together with the database images. (3) Finally, we re-rank the computed camera poses based on verification by view synthesis. The three steps are detailed next.

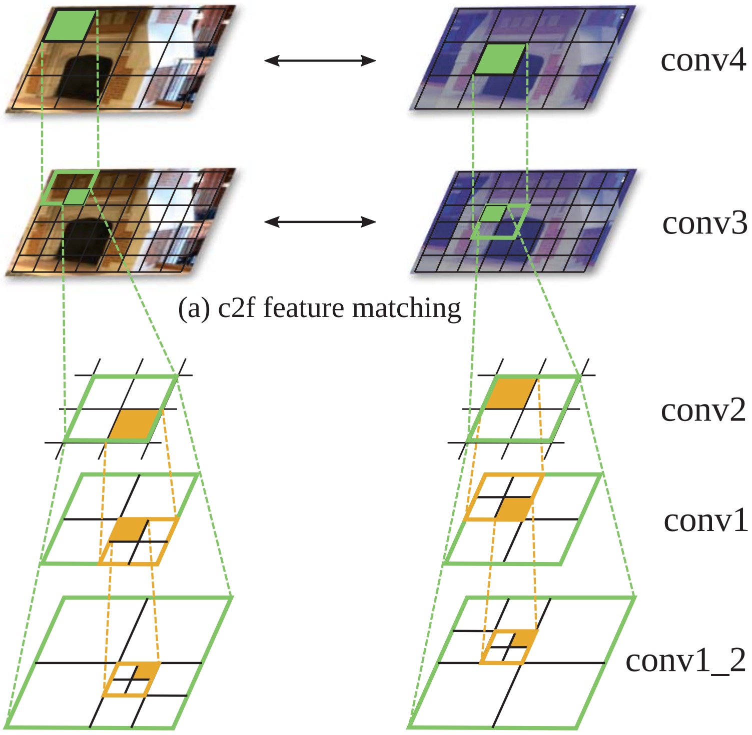
4.3.2 Candidate pose retrieval

As demonstrated by existing work [20, 40, 41, 136], aggregating feature descriptors computed densely on a regular grid mitigates issues such as a lack of repeatability of local features detected on textureless scenes, large-illumination changes, and a lack of discriminability of image description, dominated by features from repetitive structures (burstiness). As already mentioned in Sec. 4.1, these problems are also occurring in large-scale indoor localization, which motivates our choice of using an image descriptor based on dense feature aggregation. Both query and database images are described by NetVLAD [40] (but other variants could also be used), normalized L2 distances of the descriptors are computed, and the poses of the N best matching images from the database are chosen as candidate poses. In Sec. 4.5, we compare our approach with the state-of-the-art image representation using sparse feature and show benefits of our approach for indoor localization.

4.3.3 Pose estimation using dense matching

A severe problem in indoor localization is that standard geometric verification based on local feature detection [39, 42] does not work on textureless or self-repetitive scenes. A possible approach to lessen this problem is to perform dense matching, *e.g.*, matching densely extracted features [170, 171] or computing optical flow [172, 173], followed by RANSAC-based verification. Motivated by the improvements in candidate pose retrieval with dense feature aggregation (Sec. 4.3.2), we use features densely extracted on a regular grid for verifying and re-ranking the candidate images, as well as pose estimation (**DensePE**). Instead of adjusting the parameters of hand-crafted features [170, 172] (patch sizes, strides, scales) to match across images with significant viewpoint changes, we use an image representation extracted by a CNN architecture as densely extracted features. Compared to low-level hand-crafted features, CNN features extracted from deeper layers have a larger receptive field and cover higher-level information [174]. Using this type of information should have the benefit of being able to better handle large textureless image parts. However, the resulting features are also less well-localized in the image, due to the use of max-pooling layers in the CNN, which limits the camera pose accuracy that can be achieved.

To obtain spatially well-localized matches between images at a reasonable computational



(a) c2f feature matching

(b) Keypoint relocalization

Figure 4.6: **Dense CNN matching.** (a) In our coarse-to-fine (c2f) dense matching strategy we first perform exhaustive matching on a coarser layer (conv4), then search for correspondences in a lower layer (conv3), restricting the search area only to a local region around the coarse match. (b) Next, we use the keypoint relocalization scheme from [176] to refine the location of the matched features (shown by green squares) by relocalizing them in the earlier layers using the highest activation in their corresponding receptive fields (orange squares).

complexity, we use a hierarchical approach based on features extracted from multiple layers of a VGG-16 [175] network. Fig. 4.6 illustrates our coarse-to-fine (c2f) dense matching strategy. We first find initial correspondences using features from the conv4 layer by searching for mutual nearest neighbors between the images. Then we refine the correspondences using the conv3 layer, which has a higher spatial resolution compared to the conv4 layer. In particular, given a match between two conv4 features, we again perform mutual nearest neighbor matching for conv3 features. However, we restrict the search to the receptive fields of the matching conv4 features.

To further increase the positional accuracy of the features, we use a keypoint relocalization scheme [176] that hierarchically traces the feature receptive fields for each correspondence. It transfers a keypoint position from the current layer (*e.g.*, conv3 layer) to the lower layers (*e.g.*, conv2 layer) by taking the position with the highest activation in the receptive field, *i.e.*, the feature that has the largest L2-norm, as the new position. This continues up to conv1_2 layer, which has the same resolution as the input image. Finally, we obtain a set of geometrically consistent correspondences by fitting homographies via RANSAC [28].

As images in the database have depth values, and hence associated 3D points, the query camera pose can be estimated by finding pixel-to-pixel correspondences between the query and the matching database image followed by P3P-RANSAC [28], assuming a known focal length.

A detailed experimental evaluation (Sec. 4.5.2) and several qualitative examples in Sec. 4.5.7 demonstrate that our dense CNN matching obtains better matches and achieves better pose estimation in indoor environments when compared to matching standard local features, even for weakly textured areas. Notice that dense feature extraction and description requires no additional computation at query time as the intermediate convolutional layers are already computed when extracting the NetVLAD descriptor. As will also be demonstrated in Sec. 4.5.6, memory requirements of feature matching can be addressed by binarizing the convolutional features without loss in matching performance.

4.3.4 Pose verification with view synthesis

Retrieval via NetVLAD and subsequent pose estimation leads a set of candidate poses for the query image. Next, we need to select one of the estimated poses as the pose of the query image. In order to do so, we propose to collect both positive and negative evidence[4] to determine what in the observation (input image) *is* and *is not* matched to a known scene structure. This is achieved by harnessing the power of the high-resolution RGBD scans database that provides a dense and accurate observation of the 3D structure around the candidate pose. We use those structures to explicitly render a virtual view that shows how the scene looks like from the estimated query pose. It enables us to count which regions are and are not consistent between the query image and the underlying 3D structure, in a pixel-wise manner. Then we aggregate the similarities from the entire image that provides the certainty of the estimated camera pose, considering both positive and negative evidence.

Using colored 3D points associated with each retrieved candidate pose, *i.e.*, 3D points captured by an RGBD panoramic scan from which candidate database image originated, we generate the synthesized view by rendering 3D points with respect to the 6DoF query pose while taking care of self-occlusions. To compute the similarity of each region in the query to the 3D structure, we again use densely extracted features from the original and the synthesized query image. More precisely, we consider two approaches: The first one (**DensePV**) uses hand-crafted patch descriptors [32, 170] as the similarity metric. Given an original query image $\mathcal{Q}$ and synthetic one $\mathcal{Q}_{\mathcal{D}}$ associating the candidate $\mathcal{D}$, we compute patch descriptors $\mathbf{d}(x, y)$ for each bin (x, y) defined by a regular grid. To extract the features from synthesized view, we filled the blank pixels induced by missing 3D points by linear inter(/extra)-polation. Note that

[4] The impact of negative evidence in feature aggregation is demonstrated in [177].

this inpainting is done only for avoiding missing descriptors and we only evaluate the features on none-blank pixels. The consistency of each region is measured as the descriptor distance between descriptors extracted from the same region in both $\mathcal{Q}$ and $\mathcal{Q}_{\mathcal{D}}$.

$$S_{\mathrm{D}}(x, y, \mathcal{D}) = \|\mathbf{d}(x, y|\mathcal{Q}) - \mathbf{d}(x, y|\mathcal{Q}_{\mathcal{D}})\|^{-1} \tag{4.1}$$

We then compute the final score of pose candidate $\mathbf{DensePV}(\mathcal{Q}, \mathcal{Q}_{\mathcal{D}})$ as the median of the region similarities and choose the candidate $\hat{\mathcal{D}}$ with the highest score as the query pose.

$$\hat{\mathcal{D}}_{\mathbf{DensePV}} = \operatorname*{argmax}_{\mathcal{D}}(\mathbf{DensePV}(\mathcal{Q}, \mathcal{Q}_{\mathcal{D}})) = \operatorname*{argmax}_{\mathcal{D}}(\operatorname*{median}_{x,y}(S_{\mathrm{D}}(x, y, \mathcal{D}))) \tag{4.2}$$

The second approach (**DenseCNNPV**) selects the candidate in the same manner as in Eq. (4.2) but re-uses the dense CNN features $\mathbf{d}_{\mathrm{CNN}}$ from the image retrieval and the matching stages for pose estimation. We use the conv3 layer of VGG-16, which is also used in the pose estimation step. To compute a pixel-wise consistency map, we additionally extract CNN features from the synthesized image. The similarity S_{DC} is then computed as the cosine similarity of each corresponding feature.

$$S_{\mathrm{DC}}(x, y, \mathcal{D}) = \mathbf{d}_{\mathrm{CNN}}(x, y|\mathcal{Q})^{\mathrm{T}}\mathbf{d}_{\mathrm{CNN}}(x, y|\mathcal{Q}_{\mathcal{D}}), \tag{4.3}$$

$$\hat{\mathcal{D}}_{\mathbf{DenseCNNPV}} = \operatorname*{argmax}_{\mathcal{D}}(\mathbf{DenseCNNPV}(\mathcal{Q}, \mathcal{Q}_{\mathcal{D}})) = \operatorname*{argmax}_{\mathcal{D}}(\operatorname*{median}_{x,y}(S_{\mathrm{DC}}(x, y, \mathcal{D}))) \tag{4.4}$$

An ablation study in Sec. 4.5.2 shows that this newly proposed setting outperforms using hand-crafted features (DensePV).

4.4 Similarity metrics for pose verification using multiple modalities

Whereas DensePV and DenseCNNPV in Sec. 4.3.4 mainly rely on pure appearances of images, we are also interested in analyzing the benefits of using more information for our camera pose verification step. Such information extracted from input image can potentially maintain invariance to some changes in the images, *e.g.*, illumination and texture changes, and provide more guides to determine if the candidate is correct or not even in some severe conditions in indoor environments, *e.g.*, weakly textured scenes, occlusions caused by movable objects and people.

In this section, we propose multiple approaches for pose verification, *i.e.*, replacing **DensePV** in Eq. (4.2) by other metrics, based on the combination of appearance, scene geometry, and semantic information. Sec. 4.4.1 discusses how additional geometric information can be integrated into InLoc's pose verification stage. Similarly, Sec. 4.4.2 discusses how semantic information can be used for pose verification. Sec. 4.4.3 introduces an alternative way to integrate high-level scenes structures using a trainable CNN architecture.

4.4.1 Integrating scene structures

Eq. (4.2) measures the similarity in appearance between the original query image and its synthesized version. The original formulation in the InLoc pipeline has two drawbacks: 1) It only considers the 3D geometry seen from a single scan location corresponding to the retrieved database image $\mathcal{D}$. As the pose of the query image can substantially differ from the pose of the database image, this can lead to large regions in the synthesized image into which no 3D points are projected (*c.f.* Fig. 4.2 (i)). 2) Indoor scenes are often dominated by large untextured parts such as white walls. The image appearance of these regions remains constant even under strong viewpoint changes. As such, considering only image appearance in these regions does not provide sufficient information for pose selection. In the following, we propose strategies to address these problems.

Merging geometry through scan-graphs. To avoid large regions of missing pixels in the synthesized images, we use 3D data from multiple database RGB-D scans when re-rendering the query. We construct an *image-scan-graph* (*c.f.* Fig. 4.7) that describes which parts of the scene are related to each database image and are thus used for generating the synthetic query image. Given a retrieved database image $\mathcal{D}$, the graph enables us to re-render the query view using more 3D points than those visible in the panoramic RGB-D scan associated with $\mathcal{D}$[5]. To construct the graph, we first select the ten spatially closest RGB-D panoramic scans for each database image. We estimate the visibility of each 3D scan in the database image by projecting the 3D points into it while handling occlusions via depth and normal information. We establish a graph edge between the database image and a scan if more than 10% of the database image pixels share the 3D points originating from the scan.

Given a query image $\mathcal{Q}$, the retrieved database image $\mathcal{D}$, and the estimated camera pose obtained using **DensePE**, we can leverage the constructed scan-graph to render multiple synthetic query images, one for each scan connected to $\mathcal{D}$ in the graph. These views are then combined by taking depth and normal directions into account to handle occlusions.

Our approach assumes that the scans are dense and rather complete, and that different scans are registered accurately w.r.t. each other. These assumptions do not always hold in practice. Yet, our experiments show that using the scan-graph improves localization performance by reducing the number of invalid pixels in synthesized views compared to using individual scans (*c.f.* Sec. 4.5.8).

Measuring surface normal consistency. The problem of a lack of information in weakly textured regions can also be addressed by considering other complementary image modalities, such as surface normals (*c.f.* Fig. 4.2 (a-f)). When rendering the synthetic view, we can make use of the depth information in the RGB-D images to create a normal map with respect to a

[5] The InLoc dataset used in our experiments consists of multiple panoramic RGB-D scans, subdivided into multiple database images each.

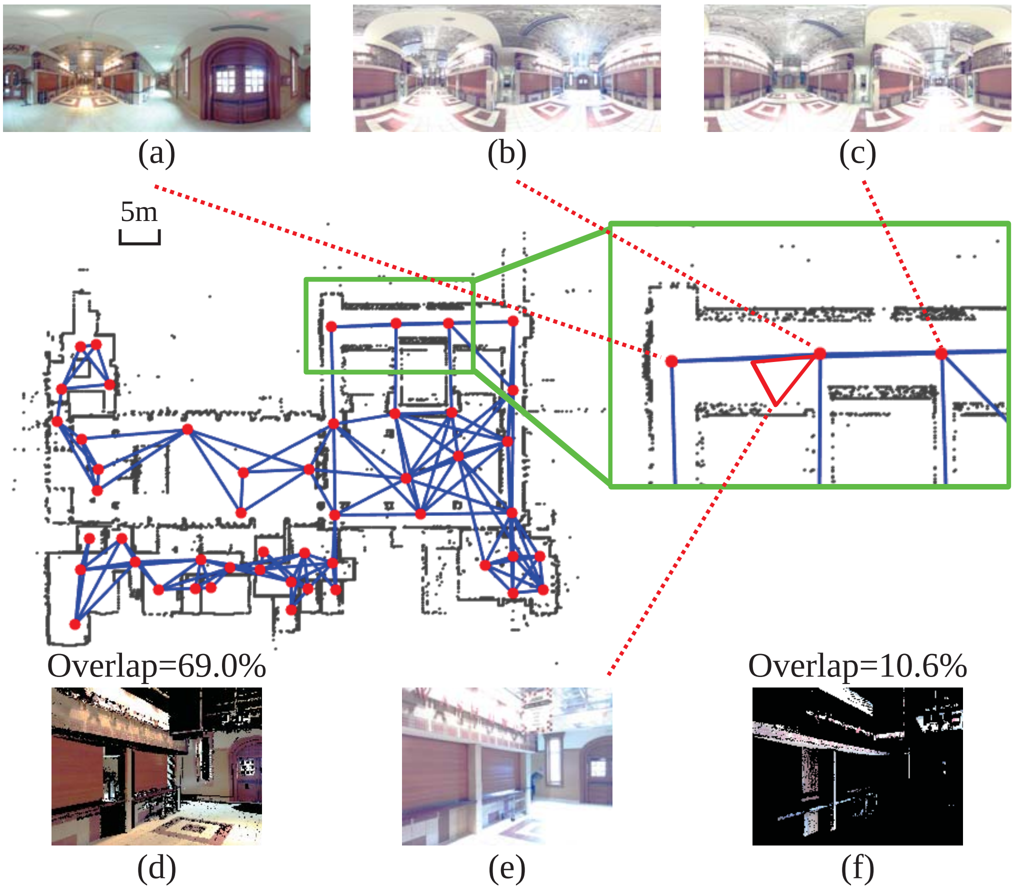


Figure 4.7: **Image-scan graph for the InLoc dataset.** For each perspective database image (e) which is cut out from the RGB-D panoramic scan (b), we compute the overlap with each adjacent scan (a, c) by projecting their 3D points into the perspective view (d, f). Red dots show where RGB-D panoramic scans (and corresponding perspective database images) are located. Blue lines indicate edges between panoramic scans and perspective database images, established based on visual overlap.

given pose. For each 3D point P that projects into a 2D point p in image space, the normal vector is computed by fitting a plane in a local 3D neighbourhood. This 3D neighborhood is defined as the set of 3D points that project within a 5×5 pixel patch around p. This results in a normal map $\mathcal{N}_\mathcal{D}$ as seen from the pose candidate $\mathcal{D}$, where each entry $\mathcal{N}_\mathcal{D}(x, y)$ corresponds to a unit-length surface normal direction. On the query image side, we use a neural network [151] to predict a surface normal map $\mathcal{N}_\mathcal{Q}$.

We define two verification approaches using surface normal consistency. Both are based on the cosine similarity S_N between normals estimated at pixel position (x, y):

$$S_\text{N}(x, y, \mathcal{D}) = \mathcal{N}_\mathcal{Q}(x, y)^\top \mathcal{N}_\mathcal{D}(x, y). \tag{4.5}$$

The first strategy, termed *dense normal verification* (**DenseNV**), mirrors DensePV but considers the normal similarities S_{N} instead of the descriptor similarities S_{D}.

$$\hat{\mathcal{D}}_{\mathbf{DenseNV}} = \underset{\mathcal{D}}{\operatorname{argmax}}(\mathbf{DenseNV}(\mathcal{Q}, \mathcal{Q}_{\mathcal{D}})) = \underset{\mathcal{D}}{\operatorname{argmax}}(\underset{x,y}{\operatorname{median}}(S_{\mathrm{N}}(x, y, \mathcal{D}))). \tag{4.6}$$

The surface normal similarity maps S_{N} can contain richer information than the descriptor similarity maps S_{D} in the case of untextured regions. Yet, the contrary will be the case for highly textured regions. Therefore, we propose a second strategy (**DensePNV**), which employs surface normal consistency as a weighting term for the descriptor similarity.

$$\hat{\mathcal{D}}_{\mathbf{DensePNV}} = \underset{\mathcal{D}}{\operatorname{argmax}}(\mathbf{DensePNV}(\mathcal{Q}, \mathcal{Q}_{\mathcal{D}})) = \underset{\mathcal{D}}{\operatorname{argmax}}(\underset{x,y}{\operatorname{median}}(w(x, y, \mathcal{D}) \cdot S_{\mathrm{D}}(x, y, \mathcal{D}))), \tag{4.7}$$

where the weighting term w shifts and normalizes the normal similarities as

$$w(x, y, \mathcal{D}) = \frac{1 + \max(0, S_{\mathrm{N}}(x, y, \mathcal{D}))}{2} \quad . \tag{4.8}$$

Through w, the normal similarities act as an attention mechanism on the descriptor similarities, focusing the attention on image regions where normals are consistent.

Implementation details. For the query images, for which no depth information is available, surface normals are estimated using [151]. The original implementation from [151] first crops the input image into a square shape and rescales it to 256 × 256 pixels. However, the cropping operation can decrease the field of view and thus remove potentially important information [178]. To preserve the field of view, we modified the network configuration to predict surface normals for rectangular images and scale each image such that its longer side is 256 pixels.

4.4.2 Integrating scene semantics

DensePV, DenseNV, and DensePNV implicitly assume that the scene is static, *i.e.*, that the synthesized query image should look identical to the real query photo. In practice, this assumption is often violated as scenes change over time. For example, posters on walls or bulletin boards might be changed or furniture might be moved around. Handling such changes requires a higher-level understanding of the scene, which we model via semantic scene understanding.

Projective Semantic Consistency (PSC).A standard approach to using scene understanding for pose verification is to measure semantic consistency [160, 162, 179]: These methods use a semantically labeled 3D point cloud, *e.g.*, obtained by projecting semantic labels extracted from RGB images onto a point cloud, and a semantic segmentation of the query image. The labeled 3D point cloud is projected into the query image via an estimated pose. Semantic consistency is then computed by counting the number of matching labels between the query and the synthetic image.

Ignoring transient objects. PSC works well in outdoor scenes, where there are relatively many classes and where points projecting into "empty" regions such as sky clearly indicate incorrect / inaccurate pose estimates. Yet, we will show in Sec. 4.5.8 that it does not work well in indoor scenes. This is due to the fact that there are no "empty" regions and that most pixels belong to walls, floors, or ceilings. Instead of enforcing semantic consistency everywhere, we use semantic information to determine where we expect geometric and appearance information to be unreliable.

We group the semantic classes into five "superclasses": *people*, *transient*, *stable*, *fixed*, and *outdoor*. The *transient* superclass includes easily-movable objects, *e.g.*, chairs, books, or trash cans. The *stable* superclass contains objects that are moved infrequently, *e.g.*, tables, couches, or wardrobes. The *fixed* superclass contains objects that are unlikely to move, *e.g.*, walls, floors, and ceilings. When computing DensePV, DenseNV, or DensePNV scores, we ignore pixels in the query image belonging to the *people* and *transient* superclasses. We refer to these approaches as **DensePV+S**, **DenseNV+S**, and **DensePNV+S**.

Implementation details. Semantics are extracted using the CSAIL Semantic Segmentation/Scene Parsing approach [154] based on a Pyramid Scene Parsing Network [152], trained on the ADE20K dataset [154] containing 150 classes. Details on the mapping of classes to superclasses are provided in [178].

4.4.3 Trainable pose verification

Motivated by the recent success of trainable methods for several computer vision tasks, we also present a trainable approach for pose verification (**TrainPV**), where we will train a pose verification scoring function from examples of correct and incorrect poses. We first describe the proposed CNN model, then how we obtained training data, and finally the loss used for training.

Network architecture for pose verification. Our network design follows an approach similar to that of DenseCNNPV, where given the original $\mathcal{Q}$ and a synthetic query image $\mathcal{Q}_\mathcal{D}$ we first extract dense feature descriptors $\mathbf{d}_{\text{CNN}}(x, y|\mathcal{Q})$ and $\mathbf{d}(x, y|\mathcal{Q}_\mathcal{D})$ using a pre-trained fully convolutional network[6]. Then, a descriptor similarity score map is computed by the cosine similarity as in Eq. (4.3). Instead taking a median as DenseCNNPV (Eq. (4.4)), we feed the similarity score-map $S_{\text{DC}}(x, y, \mathcal{D})$ to a *score regression CNN* that estimates the agreement between $\mathcal{Q}$ and $\mathcal{Q}_\mathcal{D}$, resulting in a scalar score. This score regression CNN is composed of several convolution layers followed by ReLU non-linearities and a final average pooling layer. The intuition is that the successive convolution layers can identify coherent similarity (and dissimilarity) patterns

[6] Whereas DenseCNNPV aim to re-use CNN features computed during image retrieval for efficiency, here we use more recent ResNet-18 architecture [180] pre-trained on ImageNet [181].

in the descriptor similarity score-map S_{D}. A final average pooling then aggregates the positive and negative evidence over the score map to accept or reject the candidate pose. Note that our architecture bears a resemblance to recent methods for image matching [182] and optical flow estimation [183]. Contrary to these methods, which estimate global geometric transformations or local displacement fields, our input images ($\mathcal{Q}$, $\mathcal{Q}_{\mathcal{D}}$) are already spatially aligned and we seek to measure their agreement.

Training data. In order to train the proposed network, we need appropriate annotated training data. For this, we use additional video sequences (3,442 frames) recorded for the InLoc benchmark (Sec. 4.2), which are separate from the actual test images. We created 6DoF camera poses for the new images via manual annotation and Structure-from-Motion (SfM) [178]. For each image, we generated pose candidates for training in two different ways.

The first approach randomly perturbs the ground-truth pose with 3D translations and rotations up to ±1m and ±20deg. We use the perturbed random poses to generate synthetic images by projecting the 3D point cloud from the InLoc database scan associated to that image.

The second approach uses the DensePE pipeline (Sec. 4.3.3) as a way of generating realistic estimated poses for the additional images. For this, we run the images through the localization pipeline, obtaining pose estimates and the corresponding database images. Then we run synthetic image rendering on these poses and use these images for training. Note that, contrary to the randomized approach where images are generated from the correct query-database image pair, here the synthetic images might be generated from unrelated pairs. This is because the localization pipeline might fail to generate "hard-negatives", *i.e.*, examples corresponding to other similar-looking but different locations.

In both cases, for each real and synthetic image pair, both the ground-truth (P_{GT}) and the estimated ($\tilde{P}$) poses are known. In order to generate a scalar score that can be used as a training signal, we compute the mean 2D reprojection error $r(\mathcal{Q}_{\mathcal{D}})$ of the 3D point cloud $\{X_i\}_{i=1}^{N}$ in image space:

$$r(\mathcal{Q}_{\mathcal{D}}) = \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{P}(X_i, P_{\mathrm{GT}}) - \mathcal{P}(X_i, \tilde{P})\| \ , \tag{4.9}$$

where $\mathcal{P}$ is the 3D-2D projection function.

Training loss. A suitable loss is needed in order to train the above network for pose verification. Given a query image $\mathcal{Q}$ and a set of candidate synthesized query images $\{\mathcal{Q}_{\mathcal{D}_i}\}_{i=1}^{N}$, we would like to re-rank the candidates in the order given by the average reprojection errors (*c.f.* Eq. (4.9)).

In order to do this, we assume that each synthetic image $\mathcal{Q}_{\mathcal{D}_i}$ has an associated discrete probability $p(\mathcal{Q}_{\mathcal{D}_i})$ of corresponding to the best matching pose for the query image $\mathcal{Q}$ among the N candidates. This probability should be inversely related to the reprojection error from

Eq. (4.9), *i.e.*, a pose with a high reprojection error has little probability of being the best match. Then, the scores $s_i = s(\mathcal{Q}, \mathcal{Q}_{\mathcal{D}_i})$ produced by our trained pose verification CNN can be used to model an estimate of this probability $\hat{p}(\mathcal{Q}_{\mathcal{D}_i})$ as

$$\hat{p}(\mathcal{Q}_{\mathcal{D}_i}) = \frac{\exp(s_i)}{\sum_{k=1}^{N} \exp(s_k)} \ . \tag{4.10}$$

To define the ground-truth probability distribution p, we make use of the reprojection error $r_i = r(\mathcal{Q}_{\mathcal{D}_i})$ from Eq. (4.9):

$$p(\mathcal{Q}_{\mathcal{D}_i}) = \frac{\exp(-\tilde{r}_i)}{\sum_{k=1}^{N} \exp(-\tilde{r}_k)} \ , \tag{4.11}$$

where $\tilde{r}_i = r_i / \min_k r_k$ is the relative reprojection error with respect to the minimum value within the considered candidates. The soft-max function is used to obtain a normalized probability distribution[7].

The training loss $\mathcal{L}$ is defined as the cross-entropy between the ground-truth and estimated distributions p and $\hat{p}$:

$$\mathcal{L} = -\sum_{i=1}^{N} p(\mathcal{Q}_{\mathcal{D}_i}) \log \hat{p}(\mathcal{Q}_{\mathcal{D}_i}) \ , \tag{4.12}$$

where the sum is over the N candidate poses.

Note that because the ground-truth score distribution p is fixed, minimizing the cross-entropy between p and $\hat{p}$ is equivalent to minimizing the Kullback-Leibler divergence between these two distributions. Thus, the minimum is achieved when $\hat{p}$ matches p exactly. Also note that, at the optimum, the ground-truth ranking between the candidate poses is respected, as desired.

Implementation details. The feature extraction network is composed of a fully convolutional ResNet-18 architecture (up to the `conv4-2` layer) [180], pretrained on ImageNet [181]. Its weights are kept fixed during training as the large number of parameters would lead to overfitting in our small-sized training sets. The score regression CNN is composed of four convolutional layers with 5 × 5 filters and a padding of 2, each followed by ReLU non-linearities. Each convolutional layer operates on 32 channels as input and output, except the first one, which takes the single channel descriptor similarity map S_{DC} as input, and the last one, which also outputs a single channel tensor. Finally, an average pooling layer is used to obtain the final score estimate $s(\mathcal{Q}, \mathcal{Q}_{\mathcal{D}})$. The score regression CNN is trained for 10 epochs using the PyTorch framework [184], with the Adam optimizer and a learning rate of 10^{-5}.

[7] Relative reprojection errors are used to prevent saturation of the soft-max function.

4.5 Experiments

In this section, we first provide several experimental results to demonstrate the performance of the whole InLoc pipeline (described in Sec. 4.3). Sec. 4.5.1 describes implementation details of our method and evaluation procotol for evaluation. Sec. 4.5.2 validates some design choices of each step in InLoc pipeline. Sec. 4.5.3 compares InLoc pipeline with state-of-the-art methods for indoor visual localization, using our InLoc dataset which presents a considerably large-scale environment compared to previous datasets. To confirm the relevance of the results, we also show the comparison using other datasets in Sec. 4.5.4. To discuss about the scalability of the proposed pipeline, we evaluate it on a expanded version of InLoc dataset (Sec. 4.5.5), and provide discussions about its computational cost (Sec. 4.5.6). We finally add some qualitative results in Sec. 4.5.7.

We next focus on our novel pose verification step and evaluate several modifications for it (discussed in Sec. 4.4) again using InLoc dataset.

4.5.1 Implementation details

We use the NetVLAD implementation provided by the authors and the pre-trained Pitts30K [40] VGG-16 [175] model. We also use this model to extract CNN features for pose estimation, by keeping the max-pooling outputs at the conv4 and conv3 layers and localizing keypoints via [176].

Using the pre-trained model, we compute $4,096$-dimensional NetVLAD descriptor vectors for both the query and the database images and pick the top-100 database images as the pose candidates. We then re-rank the candidates by dense CNN matching and estimate the 6DoF pose for each of the resulting top-10 images, as described in Sec. 4.3.3. Finally, we use the pose selected by the pose verification step using virtual view and dense features described in Sec. 4.3.4 as the pose of the query. We evaluate both of our two approaches, the one using hand-crafted descriptors (DensePV) and its variant using CNN features (DenseCNNPV). As the hand-crafted descriptor, we use the DenseSIFT [170] implementation in VLFeat [110]. For DenseCNNPV, we again use the conv3 features of the VGG-16 model trained for NetVLAD.

Evaluation metrics. We evaluate the localization accuracy as the consistency of the estimated poses with our reference poses. We measure positional and angular differences [141] in meters and degrees between the estimated poses and the manually verified reference poses.

4.5.2 Ablation study

In the first set of experiments, we evaluate our design choices: we first demonstrate the benefits of densely extracted features over sparsely extracted features. Next, we show how dense pose

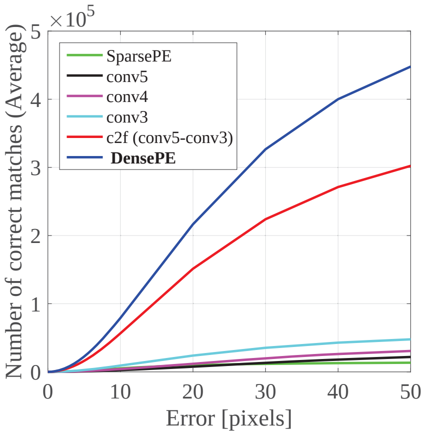


Figure 4.8: **Matching performance evaluation on the InLoc dataset.** The graph shows the impact of our coarse-to-fine matching using densely extracted features. Each curve shows the average number of matches for each query (y-axis) which are matched within a certain pixels error (x-axis) from the groundtruth.

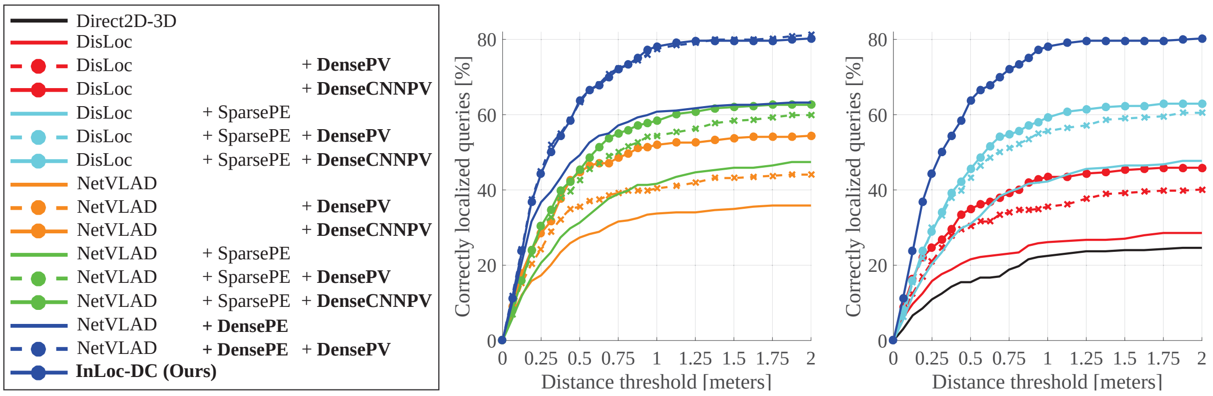
Table 4.2: **Ablation study on the InLoc dataset.** We compare several variants of our InLoc localization pipeline. Bold style denotes extensions from the conference paper [22] newly proposed in our journal [58]. The rightmost column shows the percentage of correctly localized queries within 1.0 meters and 10° errors of their reference poses.

PE Coarse Layer (*-conv3)	Reloc. scheme	Pose verification	Correctly loc. queries (%)
conv5	-	Inlier #	57.1
conv5	-	DensePV	72.0 (a)
conv5	-	**DenseCNNPV**	73.3
conv4	-	Inlier #	61.4
conv4	-	DensePV	73.9
conv4	-	**DenseCNNPV**	72.0
conv4	**yes**	Inlier #	60.8
conv4	**yes**	DensePV	77.5
conv4	**yes**	**DenseCNNPV**	**78.1** (b)

verification improves localization performance over inlier counting. Third, we validate the components newly proposed in our journal [58] compared to the conference version [22]. Finally, we show that binarizing the CNN features significantly reduces the memory footprint without significantly impacting localization performance.

Benefits of dense matching. As discussed in Sec. 4.3.3, we use densely extracted CNN features to robustly estimate camera poses even in weakly textured indoor environments, where sparse local features often do not yield sufficient matches. To show the benefits of using dense features, we perform two experiments.

In the first experiment, we compare dense CNN and sparse hand-crafted features [26, 33] (SparsePE), in terms of the number of correct matches generated, in indoor scenes. We create image pairs for feature matching by selecting the RGBD database image that has the largest overlap with a given query from the InLoc dataset. Feature matching is then performed between the two images. Fig. 4.8 shows the cumulative distributions over the reprojection errors of the feature matches, obtained by projection via the depth maps and known reference poses. The proposed **DensePE** approach, which consists of matching features first in the conv4 layer and then in the conv3 layer, results in a significant improvement compared to sparse feature matching (SparsePE). We also evaluate different variations of dense coarse-to-fine matching using the conv5 and conv3 layers (c2f (conv5-conv3)), as well as single-scale feature matching



(a) NetVLAD baselines (b) Other baselines

Figure 4.9: **Impact of different components.** The graphs show the impact of dense matching (**DensePE**) and dense pose verification (**DensePV** and **DenseCNNPV**) on pose estimation quality for (a) the pose candidates retrieved by NetVLAD and (b) state-of-the-art baselines. Plots show the fraction of correctly localized queries (y-axis) within a certain distance (x-axis) whose rotation error is at most 10°.

using the conv3, conv4, and conv5 layers, respectively. All features used are after max-pooling, and all keypoints are relocalized using the scheme of Widya et. al [176]. The single-scale dense features already result in more correspondences than SparsePE, but coarse-to-fine matching significantly increases the performance, with DensePE yielding the most correct matches.

The second experiment shows that the larger number of matches found by our DensePE approach translates to a better localization performance: Fig. 4.9(a) shows that our pose estimation with coarse-to-fine dense matching (NetVLAD [40]+**DensePE** "–") constantly improves the localization rate by about 15% when compared to the sparse local feature matching (NetVLAD [40]+SparsePE "–"). This supports our conjecture that dense feature matching is superior to sparse feature matching for often weakly textured indoor scenes. This effect is also clearly demonstrated in the qualitative results in Sec. 4.5.7 (cf. Fig. 4.11).

Benefits of pose verification with view synthesis. We apply our pose verification step using regular-gridded hand-crafted features (**DensePV**) and CNN features (**DenseCNNPV**) on query and rendered view to the top-10 pose estimates obtained by different baselines. Results are shown in Fig. 4.9 and demonstrate significant and consistent improvements obtained by our pose verification approach (compare "- × -" (**DensePV**) and "–●–" (**DenseCNNPV**) to "—" in Fig. 4.9). Improvements are most pronounced for the position accuracy within 1.0 meters (16% or more).

Validating design choices. Compared to the conference version [22], we changed some components of the InLoc pipeline: (i) coarse-to-fine matching from the conv4 layer to the conv3 layer instead of coarse-to-fine matching from the conv5 to conv3 layer, (ii) the use of

Table 4.3: **Comparison with state-of-the-art localization methods on the InLoc dataset.** The first three rows show the rate (%) of correctly localized queries within a given distance (m) and a 10° orientation error. The bottom two rows report the median position and orientation errors.

	Direct	MapNet	DisLoc	NetVLAD	NetVLAD	**InLoc**	**InLoc**	**InLoc-**
	2D-3D	[61]	[27]	[185]	[185]	[22]	+ [99]	**DC**
	[46]		+SpsPE	+SpsPE	**+DnsPE**	**(Ours)**		**(Ours)**
0.25m	10.9	0.0	20.4	20.7	36.8	38.0	42.2	**44.4**
0.50m	15.5	0.0	31.0	31.3	49.2	57.8	**65.3**	63.8
1.00m	22.5	0.0	42.2	41.6	60.8	72.0	77.8	**78.1**
Med.	4.99m,	11.24m,	1.04m,	1.08m,	0.51m,	0.38m,	0.31m,	**0.30m**,
error	33.95°	85.28°	3.93°	4.05°	2.29°	2.10°	1.74°	**1.73°**

the keypoint relocalization scheme of [176], and (iii) dense pose verification with CNN features (**DenseCNNPV**) instead of SIFT features (**DensePV**).

Tab. 4.2 compares the performance of our InLoc pipeline with the original and the newly proposed components. As it can be seen, conv4-to-conv3 instead of conv5-to-conv3 matching typically improves the percentage of query images that can be localized within the chosen error bounds. We further observe a significant boost in performance when using the keypoint relocalization scheme. This is due to the fact that more accurately localized keypoints result in increased accuracy of the estimated camera poses.

Compared to **DensePV**, **DenseCNNPV** has the advantage that it reuses the features extracted during the computation of the NetVLAD descriptor instead of requiring to extract additional SIFT features. As shown in Tab. 4.2, this approach performs similar to **DensePV**. This implies that the reduction in computational overhead does not come at the price of reduced performance. Both **DensePV** and **DenseCNNPV** outperform classical inlier counting by a large margin.

From Tab. 4.2, we can see that including all newly proposed modifications (row marked as (b)) leads to a gain of ≈ **6.0%** compared to the original InLoc version [22] (row marked as (a)). In the following, we will refer to these two versions as ***InLoc-DC*** (new) and ***InLoc*** (conference version).

4.5.3 Comparison with the state-of-the-art methods

Secondly we compare the proposed InLoc approach with several state-of-the-art localization methods in Tab. 4.3.

Direct 2D-3D matching [46, 49]. We first compare with a variation[8] of a state-of-the-art 3D structure-based image localization approach [46]. We compute affine covariant RootSIFT features for all the database images and associate them with 3D coordinates via the known scene geometry. Features extracted from a query image are then matched to the database 3D descriptors [118]. We select at most five database images receiving the largest numbers of matches and use all these matches together for pose estimation. Similar to [46], we did not apply Lowe's ratio test [32] as it lowered the performance. The 6DoF query pose is finally computed by P3P-LO-RANSAC [30]. As shown in Tab. 4.3 and Fig. 4.9 (b), our proposed approach (**InLoc** and **InLoc-DC**) outperforms Direct 2D-3D matching by a large margin (48.3% at the localization accuracy of 0.5m). We believe that this is because our large-scale indoor dataset involves many distractors and large viewpoint changes that present a major challenge for direct correspondences searching.

Disloc [27] + sparse pose estimation (SparsePE). We next evaluate a state-of-the-art image retrieval-based localization method. Disloc represents images using bag-of-visual-words with Hamming Embedding [82] while also taking local descriptor space density into account. We use a publicly available implementation [39] of Disloc with a 200K vocabulary trained on the database images of our InLoc dataset. The top-100 candidate images shortlisted by Disloc are re-ranked by spatial verification [42] using (sparse) affine covariant features [33]. The ratio test [32] was not applied here as it was removing too many features that needed to be retained in the indoor scenario. Using the inliers, the 6DoF query pose is computed with P3P-LO-RANSAC [30]. To allow a fair comparison, we use exactly the same features and P3P-LO-RANSAC for pose estimation as the Direct 2D-3D matching method described above. As shown in Tab. 4.3, Disloc [27]+SparsePE results in a 19.7% performance gain compared to Direct 2D-3D matching. This can be attributed to the image retrieval step that discounts the burst of repetitive features. We also evaluate localization using the original top-retrieved image (without re-ranking [42]) to compute the query pose ("—" in Fig. 4.9). This shows similar results to Direct 2D-3D matching, indicating that re-ranking based on the number of inlier matches largely improves localization if the shortlisting is reasonable. However, the results are still significantly worse compared to **InLoc**, and **InLoc-DC**.

NetVLAD [40] + SparsePE. We also evaluate a variation of the above image retrieval-based localization method. Here, the candidate shortlist is obtained by NetVLAD [40], which is then re-ranked using sparse features [42], followed by pose estimation using P3P-LO-RANSAC [30]. This is a strong baseline building on the state-of-the-art place recognition results from [40]. Interestingly, as shown in Tab. 4.3, there is no significant difference between NetVLAD+SparsePE and DisLoc+SparsePE, which is in line with results reported in outdoor settings [65]. Yet,

[8] Due to the sparse sampling of viewpoints in our indoor dataset, we cannot establish feature tracks between database images, preventing us from applying algorithms relying on co-visibility [46, 49–51].

Table 4.4: **Comparison on Matterport3D [69].** Numbers show the median positional (m) and angular (degrees) errors.

	Disloc [27]	NetVLAD [185]	NetVLAD [185]	**InLoc-DC**
	+SparsePE	+SparsePE	+DensePE	(Ours)
90 bldgs.	0.42, 4.58°	0.44, 4.70°	0.14, 1.81°	**0.13, 1.66°**

NetVLAD ("—") outperforms DisLoc (5.8% at the localization accuracy of 0.5m) before re-ranking via SparsePE (cf. Fig. 4.9) in our indoor setting (see also qualitative results in Sec. 4.5.7). Both methods, even though they are state-of-the-art in outdoor localization, still perform significantly worse than our proposed approach based on dense feature matching and view synthesis.

Learning based visual localization methods. We also compare on our dataset to the state-of-the-art learning based pose estimators, DSAC [59] (and its improved version [60] (DSAC++)), PoseNet [55], and MapNet [61]. These methods directly regress camera poses from images while learning a mapping from an image (patch) to a unique scene coordinate or camera pose. Despite our best efforts, none of them work well on our indoor dataset. Tab. 4.3 shows results of the best performing learning based method on our data – MapNet [61]. We have observed that learning based visual localization methods work well in scenes with unique appearance, but are not able to handle well ambiguities arising from repetitive (similar) structures in the same scene. Yet, such structures are very common in the indoor scenes in our dataset. In addition, sparsely distributed RGBD scans that are used to construct our database make the training difficult because the trained model has to generalize across large changes in viewpoint in order to estimate the query pose. We provide comparison with DSAC [59] and PoseNet [55] on much smaller datasets in Sec. 4.5.4.

Learning-based matching. The recent work of Rocco *et al.* [99] builds an end-to-end trainable network for generating reliable keypoint matches, based on local neighborhoods. These matches can be used as a part of our localization pipeline, resulting in comparable results (**InLoc**+ [99] in Tab. 4.3).

4.5.4 Evaluation on other datasets

We also evaluate InLoc on two existing indoor datasets [56, 69] to confirm the relevance of our results. The Matterport3D [69] dataset consists of RGBD scans of 90 buildings. Each RGBD scan contains 18 images that capture the scene around the scan position with known camera poses. We created a test set by randomly choosing 10% of the scan positions and selected their horizontal views. This resulted in 58,074 database images and a query set of 6,726 images. Results are shown in Tab. 4.4. Our approach (**InLoc-DC**) outperforms the baselines,

Table 4.5: **Evaluation on the 7 Scenes dataset [56, 104]**. We report the median position and orientation errors, as well as the percentage of queries localized within 5cm and 5° of the ground truth pose. "–" indicates that the corresponding number is not reported in the literature.

	PoseNet	RelocNet	ActiveSearch	DSAC	DSAC++	NetVLAD [185]	NetVLAD [185]
Scene	[55]	[62]	[49]	[59]	[60]	+SparsePE [42]	+**DensePE**
Chess	13cm, 4.48°, –	12cm, 4.14°, –	4cm, 1.96°, –	**2cm**, 1.2°, 94.6%	**2cm**, **0.5°**, **97.1%**	4cm, 1.83°, 66.6%	3cm, 1.04°, 84.0%
Fire	27cm, 11.3°, –	26cm, 10.4°, –	3cm, 1.53°, –	4cm, 1.5°, 74.3%	**2cm**, **0.9°**, **89.6%**	4cm, 1.55°, 70.4%	3cm, 1.07°, 88.5%
Heads	17cm, 13.0°, –	14cm, 10.5°, –	2cm, 1.45°, –	3cm, 2.7°, 71.7%	**1cm**, **0.8°**, 92.4%	2cm, 1.65°, 83.0%	2cm, 1.06°, **96.8%**
Office	19cm, 5.55°, –	18cm, 5.32°, –	9cm, 3.61°, –	4cm, 1.6°, 71.2%	**3cm**, **0.7°**, **86.6%**	5cm, 1.49°, 53.5%	**3cm**, 1.01°, 77.4%
Pumpkin	26cm, 4.75°, –	26cm, 4.17°, –	8cm, 3.10°, –	5cm, 2.0°, 53.6%	**4cm**, **1.1°**, **59.0%**	7cm, 1.87°, 31.2%	6cm, 1.53°, 44.1%
Red kitchen	23cm, 5.35°, –	23cm, 5.08°, –	7cm, 3.37°, –	5cm, 2.0°, 51.2%	**4cm**, **1.1°**, 66.6%	5cm, 1.61°, 54.8%	**4cm**, 1.25°, **68.0%**
Stairs	35cm, 12.4°, –	28cm, 7.53°, –	**3cm**, **2.22°**, –	117cm, 33.1°, 4.5%	9cm, 2.6°, **29.3%**	12cm, 3.41°, 13.8%	9cm, 2.25°, 24.5%

which is in line with results on the InLoc dataset. Interestingly, dense pose verification yields little improvement on this dataset. We attribute this to the Matterport3D dataset in general containing significantly more well-textured scenes. We also tested PoseNet [55] and DSAC [59] on a single (the largest) building. The test set is created in the same manner as above and contains 1,884 database images and 210 query images. Even in this much easier case, DSAC fails to converge. PoseNet produces large localization errors (24.8 meters and 80.0 degrees) in comparison with InLoc (0.26 meters and 2.78 degrees).

We also report results on the 7 Scenes dataset [56,104] which is, while relatively small, a standard benchmark for indoor localization. The dataset consists of geometrically-registered video frames representing seven scenes, together with associated depth images and camera poses. In this situation, each scene is much more textured than our InLoc dataset. Furthermore, the dataset captures the scenes much more densely, since it has the several sequential video frames distributed in rather smaller scenes than ours. Therefore, we observed that raw inlier counting after image retrieval (NetVLAD+**DensePE**) without a final dense pose verification step already provides a good ranking. Thus, we only compare NetVLAD+**DensePE** with the state-of-the-art methods. Tab. 4.5 shows localization performance for each method. CNN-based methods that directly obtain a 6DoF pose for an RGB image input [55, 62] show rather large positional errors. In contrast, Active-search [49], a state-of-the-art method using hand-crafted sparse features to obtain 2D-3D correspondences, provides more accurate localization, in line with our sparse feature baseline (NetVLAD+SparsePE). DSAC [59] obtains 2D-3D correspondences using a CNN architecture specifically trained on this dataset and its latest variant [60] gives the best results for five out of the seven scenes. Our method (NetVLAD+**DensePE**) also uses CNN features to obtain 2D-3D correspondences. In contrast to other methods [55, 59, 60, 62], our approach does not need any scene-specific training. We use an off-the-shelf NetVLAD CNN, trained on an outdoor dataset. Still, our approach performs comparably to [60].

Fine-tuning on indoor scenes. For image retrieval and matching we used NetVLAD trained on the Pittsburgh30k dataset, which only consists of outdoor images. Preliminary experiments

Table 4.6: **Results on the extended dataset,** as the percentage (%) of correctly localized queries within given distance (m) and 10° angular errors.

	NetVLAD [185]	NetVLAD [185] +SparsePE	NetVLAD [185] **+DensePE**	**InLoc-DC** (Ours)
0.25m	16.1	19.8	35.9	**42.6**
0.50m	25.8	31.0	47.4	**58.7**
1.00m	31.9	41.6	58.7	**75.1**

Table 4.7: **Computational time for each component of the proposed approach.** Average elapsed time for localizing one query in the InLoc dataset.

Process	Image retrieval	Feature extraction	Pose estimation	Pose verification
Time [sec]	∼0.07	∼1	136.3	6.08

to fine-tune NetVLAD on Matterport3D (indoor images) were not successful. We used only ≈ 60K images, selecting horizontal views, and excluding the ones in which camera was oriented towards the floor or the ceiling. Evaluating the models we trained using the NetVLAD pipeline [40] on the InLoc data, we observed a drop in performance, even for pure retrieval. This is very likely caused by lack of variability in the Matterport images, more precisely lack of occluders and no variation in illumination.

4.5.5 Measuring scalability

Our InLoc dataset covers multiple floors and multiple buildings. Yet, some applications might need to operate at an even larger scale, *e.g.*, at an international airport or a complete university campus. To measure the scalability of our approach to larger scenes, we create a larger version of our InLoc dataset by adding database images from the Matterport3D dataset [69], which also contains depth maps and camera poses. The resulting extended dataset has about 110K database images, covering 92 buildings with multiple rooms per building.

Tab. 4.6 shows localization results for NetVLAD [40]-based methods including our two proposed approaches (**DensePE** and **InLoc-DC** (**DensePE**+**DenseCNNPV**)) on the enlarged dataset. Compared to the results obtained on the smaller original InLoc dataset (Tab. 4.3), the loss in performance is rather small for all retrieval-based methods, even though the extended dataset contains 11 times more images. InLoc's performance drops by 5.1% at 0.5 meters, while the sparse feature baseline (NetVLAD+SparsePE) exhibits a rather smaller drop. This is because our dense CNN features capture higher-level structural information, whereas handcrafted sparse features capture lower-level information, *i.e.*, edges, corners. This causes more false matches for geometrically inconsistent image pairs, resulting in inaccurately estimated poses among the top-10 candidates selected via DensePE. **InLoc-DC** still shows the best performance, with a gain in pose accuracy of over 15% compared to NetVLAD+DensePE.

Also, the run time computational cost does not grow significantly when increasing the database size. This is because the initial step of obtaining the top 100 pose candidates is implemented using efficient retrieval. The follow-up stages are more expensive but are only applied to the shortlist of the top 100 (and later top 10) retrieved candidates. Detailed evaluation and discussion of the computational cost is given next.

4.5.6 Computational cost

Tab. 4.7 shows the average computational time for each step of our method (InLoc-DC). Please note that the the timing of the pose estimation and pose verification stages includes processing of the top retrieved 100 and 10 images. All numbers were obtained on an Intel Core i7-5930K CPU and GTX TITAN X GPU (used only for the dense feature extraction). We accelerate the pose estimation and pose verification steps by using 6 threads; further multi-threading can lead to further (linear) improvements. Reducing the computational time for large-scale visual localization is still an open problem. Recent CNN-based methods achieve precise localization in few milliseconds but only for smaller-scale environments[9]. A non-negligible bottleneck in our method is the dense CNN matching for pose estimation that requires over two minutes per input query (for processing the top 100 candidates), but it gives a large performance gain compared to sparse matching (*c.f.* section 4.5.2). This can be only partially mitigated by learned matchers [99, 144]. Another fairly expensive process is pose verification, which includes rendering virtual views from a high-quality RGBD scan. We observed most of the time is spent in the visibility check for each 3D point, which is currently implemented on CPU using an un-optimized Matlab script. Further significant speed-ups can be achieved by replacing this part of the pipeline using optimized GPU rendering.

Memory requirements are dominated by the size of the database to be stored. Storing 4096-dimensional NetVLAD descriptors for all InLoc database images requires only 163MB. However, storing also the intermediate features for the dense CNN matching requires additional 428GB. A binary representation of features (instead of floats) in the intermediate CNN layers can significantly reduce the needed space. We found that a simple binarization via the standard Hamming embedding approach [82] without dimensionality reduction can compresses the descriptors to 13.4GB with only a negligible performance loss (less than 1% at 0.5 meters) compared to the original descriptors (in line with results reported for object recognition [186]). The memory requirements of the approach can be further reduced by extracting the coarse to

[9] For example, [60] reported 200ms for a total processing time per image, on a Tesla K80 GPU and an Intel Xeon E5-2680 v3 CPU (6 cores).

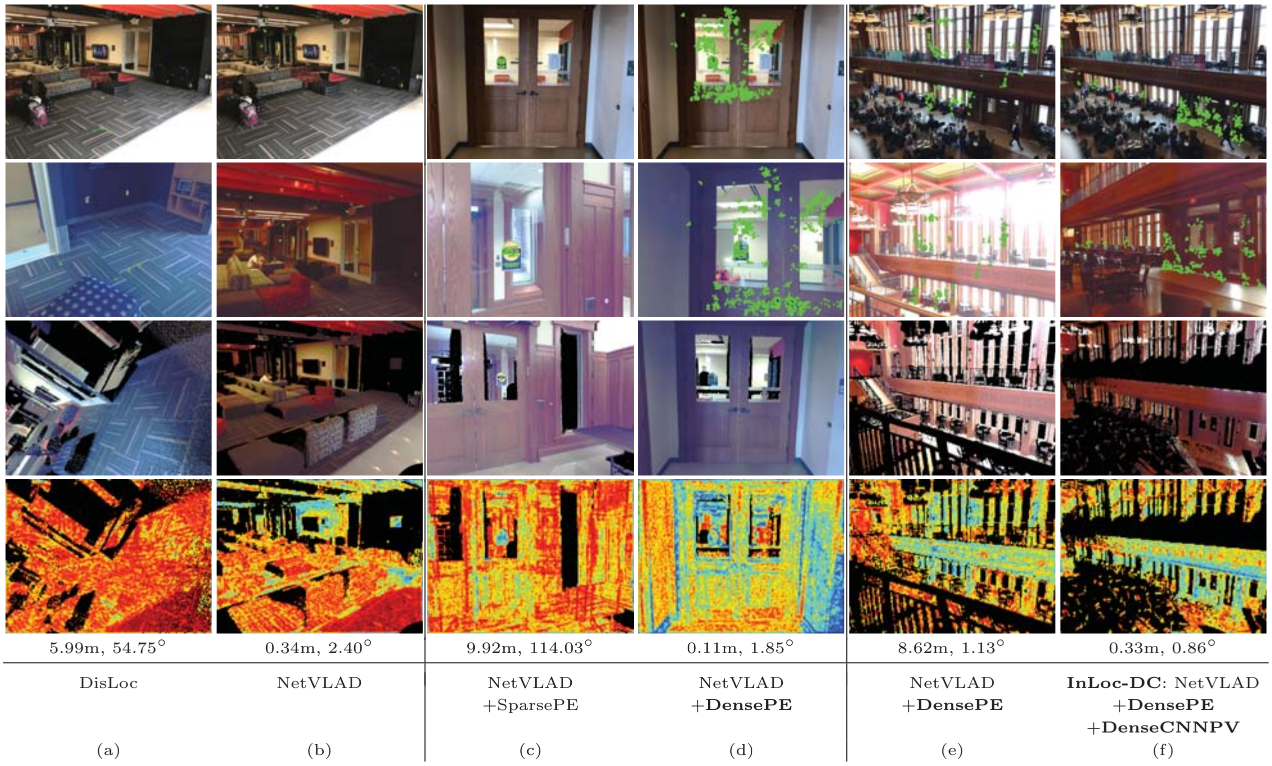


Figure 4.10: **Qualitative comparison of different localization methods (columns).** From top to bottom: query image, the best matching database image, synthesized view at the estimated pose (without inter/extra-polation), error map between the query image and the synthesized view, localization error (meters, degrees). Warm colors on the error map correspond to large errors. Green dots are the inlier matches obtained by P3P-LO-RANSAC.

fine CNN features of the database images on-line, at the expense of additional computing time (∼1 second per image, see Tab. 4.7.)

4.5.7 Qualitative results

We show qualitative localization results of the proposed InLoc approach and other baselines. In the following, we consider a query image as correctly localized if the error for the estimated pose is within 1 meter and 5° with respect to the reference pose.

Fig. 4.10 shows a component-wise comparison of our proposed approach with the baselines. We pick typical examples for each step of the proposed method (b,d,f) which uses dense CNN features for image retrieval, pose estimation, and pose verification. Our approaches adopting dense information successfully localize the queries, whereas the baselines (a,c,e) using sparse local features fail.

Next, we show several situations in which InLoc successfully localizes the query images while the baseline methods fail. Fig. 4.11 shows qualitative examples of the results obtained by

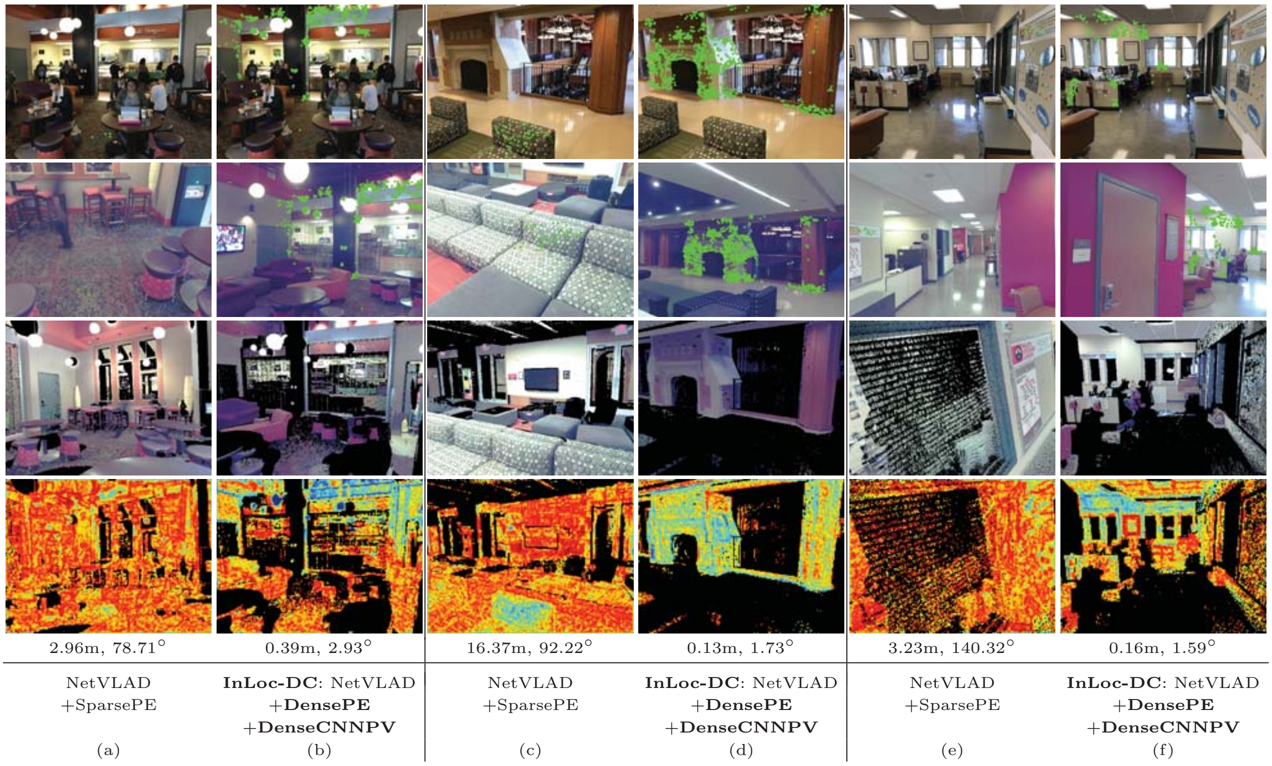
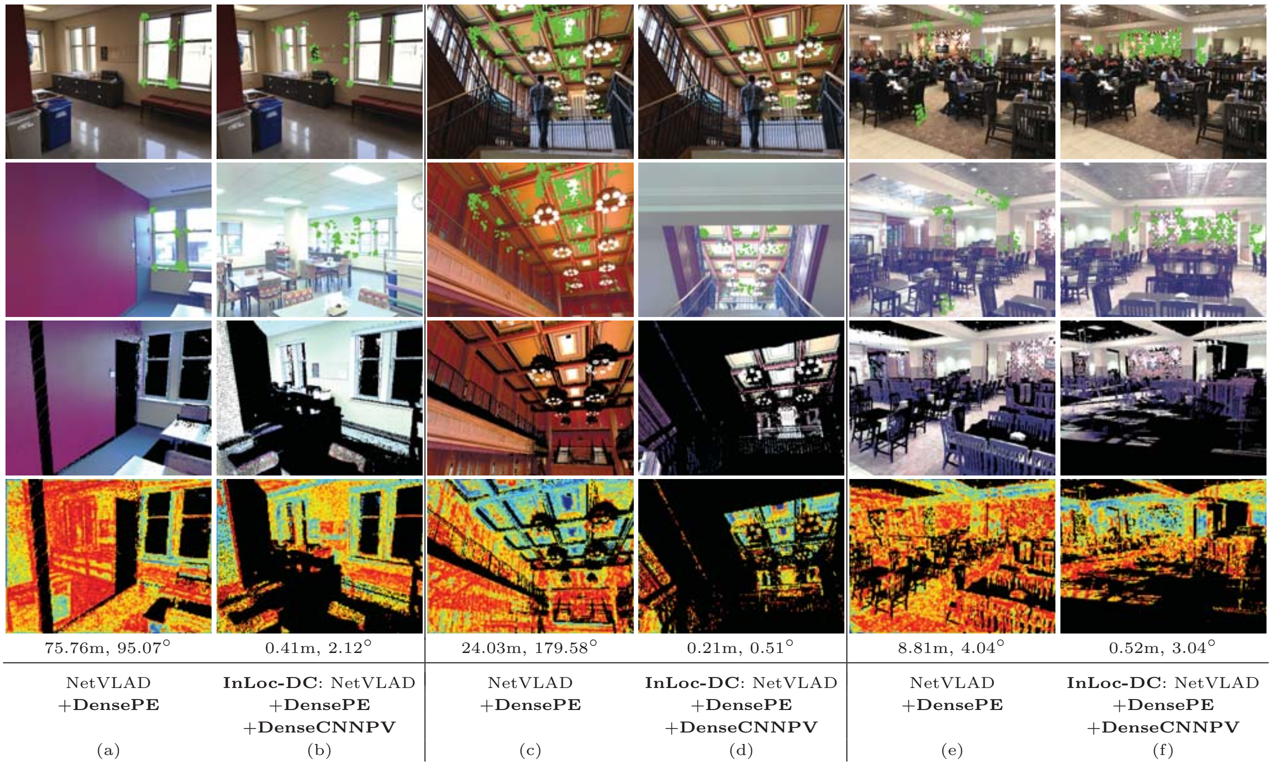

2.96m, 78.71°
0.39m, 2.93°
16.37m, 92.22°
0.13m, 1.73°
3.23m, 140.32°
0.16m, 1.59°
NetVLAD +SparsePE
(a)
InLoc-DC: NetVLAD +**DensePE** +**DenseCNNPV**
(b)
NetVLAD +SparsePE
(c)
InLoc-DC: NetVLAD +**DensePE** +**DenseCNNPV**
(d)
NetVLAD +SparsePE
(e)
InLoc-DC: NetVLAD +**DensePE** +**DenseCNNPV**
(f)

Figure 4.11: **Qualitative comparison between InLoc and NetVLAD+SparsePE.** See the caption of Fig. 4.10 for details.

NetVLAD+ SparsePE (a,c,e) versus our **InLoc-DC** (b,d,f). As shown in (a) and (c), sparse features are often detected on highly repetitive structures *e.g.*, textured surfaces (fabric pattern on the carpet and sofa). As shown in (a) for the baseline, matching features found on such objects can result in matches with unrelated parts of the scene, leading to incorrect camera pose estimates. The fact that sparse features are predominantly found in a few textured regions leads to problems in the largely untextured indoor scenes. This is shown in (e), where matches are found only in a small part of the query image, which leads to an unstable configuration for camera pose estimation. In contrast, dense matching leads well-distributed correspondences and thus more stable pose estimates, as can be seen in (b), (d), and (f). Our pose verification, **DenseCNNPV** (Sec. 4.3.4), allows us to identify incorrect poses, resulting from features found on repetitive structures, since most parts of the image rendered from a false pose are not consistent with the query image. Thus, InLoc approach is better suited to handle highly repetitive indoor scenes with rich feature correspondences.

The next set of qualitative results demonstrates the benefits of dense pose verification. Fig. 4.12 compares results obtained by **InLoc-DC** (b,d,f) with results obtained by the baseline NetVLAD+ **DensePE** (a,c,e). In this case, the baseline NetVLAD+**DensePE** uses our dense matching but selects the best pose based only on the number of inlier matches. For scenes dominated by symmetries and repetitive structures (a,c) or objects that frequently appear



NetVLAD +**DensePE** (a)

InLoc-DC: NetVLAD +**DensePE** +**DenseCNNPV** (b)

NetVLAD +**DensePE** (c)

InLoc-DC: NetVLAD +**DensePE** +**DenseCNNPV** (d)

NetVLAD +**DensePE** (e)

InLoc-DC: NetVLAD +**DensePE** +**DenseCNNPV** (f)

Figure 4.12: **Qualitative comparison between InLoc and NetVLAD+DensePE.** See the caption of Fig. 4.10 for details.

in the scene (a,e), there can be a large amount of geometrically consistent matches even for unrelated database images. This still holds true even if matches are obtained by dense features and are geometrically verified. Our dense pose verification strategy using synthesized images (b,d,f) effectively provides "negative" evidence in such situations. The error maps (bottom) clearly show that it detects (in)consistent areas between the query and its synthesized image.

4.5.8 Pose verification using geometric semantic information

Next we evaluate our several modifications of pose verification step exploiting additional geometric and semantic modalities (discussed in Sec. 4.4).

Baselines. To demonstrate the impact of each modality clearly, we test our pose verification for the candidate poses obtained by the most primitive version of DensePE [22] which estimates camera poses using conv5 and conv4 layer in the pre-trained VGG-16 [175] model (See details in Sec. 4.5.2). After re-ranking the image list according to the number of homography-inliers, we estimate pose candidates for the top 10 best matched images using a set of dense inlier matches and database depth information. The baseline localization performances when ranked by the number of inlier matches (DensePE) or by DensePV [22] (Sec. 4.3.4) are shown in the top or second row in Tab. 4.2.

Table 4.8: **The impact of using the scan-graph for pose verification evaluated on the InLoc dataset [22].** We report the percentage of queries localized within given positional and rotational error bounds.

	Error [meter, degree]			
Method	[0.25, 5]	[0.50, 5]	[1.00, 10]	[2.00, 10]
	w/o scan-graph			
DensePE [22]	35.0	46.2	57.1	61.1
DensePV [22]	38.9	55.6	69.9	74.2
PSC	30.4	44.4	55.9	58.4
DensePV+S	39.8	57.8	71.1	75.1
DenseNV	32.2	45.6	58.1	62.9
DenseNV+S	31.6	46.5	60.5	64.4
DensePNV	40.1	58.1	72.3	**76.6**
DensePNV+S	40.1	59.0	72.6	76.3
	w/ scan-graph			
DensePV	39.8	59.0	69.0	71.4
PSC	28.3	43.2	55.0	58.4
DensePV+S	**41.3**	**61.7**	71.4	74.2
DenseNV	34.3	50.5	62.9	66.6
DenseNV+S	35.9	51.4	64.4	68.4
DensePNV	40.4	60.5	**72.9**	75.4
DensePNV+S	41.0	60.5	72.3	75.1
TrainPV (random)	39.5	56.5	72.3	76.3
TrainPV (DPE)	39.5	56.8	72.3	76.3
Oracle (Upper-bound)	43.5	63.8	77.5	80.5

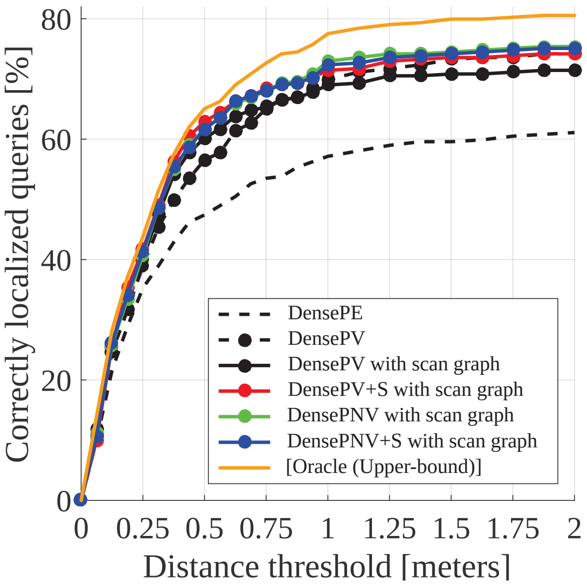
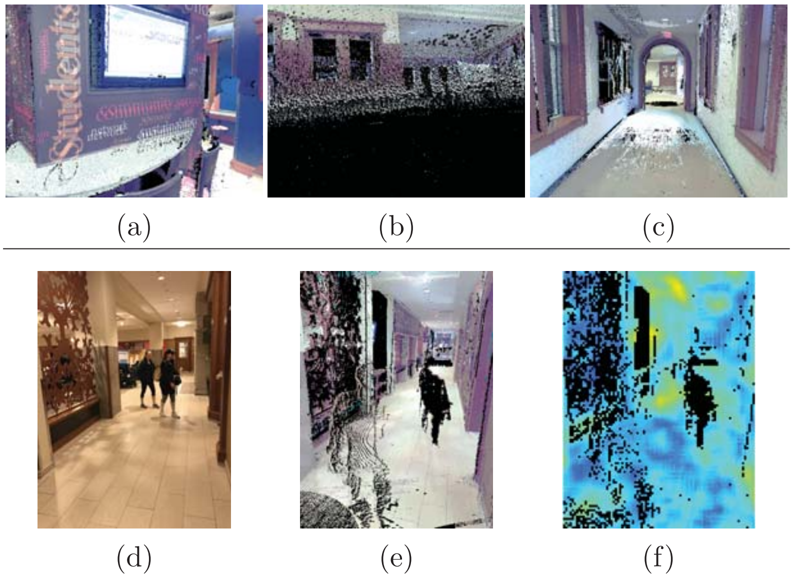


Figure 4.13: **The impact of geometric and semantic information on the pose verification stage.** Each curve shows the percentage of the queries localized within varying distance thresholds (x–axis) and a fixed rotational error of at most 10 degrees.

As a semantic baseline, we project the database 3D points, labeled via semantics [154] extracted on the database image, into the query and count the number of points with consistent labels (**PSC**).

The impact of using additional modalities. Tab. 4.8 and Fig. 4.13 compare the localization performance of the baseline pose verification methods against our novel variants proposed in Sec. 4.4. DenseNV and PSC perform worst, even compared to the baseline DensePE. This is not surprising as both completely ignore the visual appearance and instead focus on information that by itself is less discriminative (surface normals and semantic segmentation, respectively). On the other hand, combining geometric and / or semantic information with appearance information improves the localization performance compared to DensePV. This clearly validates our idea of using multiple modalities.

We observe the biggest improvement by using our scan-graph, which is not surprising as it reduces the number of invalid pixels and thus adds more information to the rendered images. DensePV+S using a scan-graph shows the best performance at higher accuracy levels. DensePNV using the scan-graph combines appearance and normal information and constantly shows more than a 5% performance gain compared to DensePV. Yet, DensePNV+S with the


(a) (b) (c)

(d) (e) (f)

Figure 4.14: **Typical failure cases of view synthesis using the scan-graph.** Top: Synthetic images obtained during DensePV with the scan-graph, affected by (a) misalignment of the 3D scans to the floor plan, (b) sparsity of the 3D scans, and (c) intensity changes. Bottom: A typical failure case of DensePV with the scan-graph: (d) query image, (e) re-rendered query, (f) error map computed with RootSIFT.

scan-graph shows less improvement compared to its single scan variant and even performs worse for larger error thresholds. This is partially due to inaccurate depths and camera poses of the database images (*c.f.* Fig. 4.14 (a–c)). There are also failures where a single scan already provides a rather complete view. Fig. 4.14 (d–f) shows such an example: due to weak textures, the rendering appears similar to the query image. Such failures cannot be resolved using the scan-graph.

Interestingly, simply combining all modalities does not necessarily lead to the best performance. To determine whether the modalities are simply not complementary or whether this is due to the way they are combined, we create an oracle. The oracle is computed from four of our proposed variants (DensePV [22], DensePV w/ scan-graph, DensePV+S w/ scan graph, and DensePNV w/ scan-graph): Each variant provides a top-ranked pose and the oracle, having access to the ground truth, simply selects the pose with the smallest error. As can be seen from Tab. 4.8 and Fig. 4.13, the oracle performs clearly better than any of our proposed variants. We also observed DenseNV+S provides better poses than the oracle (which does not use DenseNV+S) for about 9% the queries, which could lead to further improvements. This shows that the different modalities are indeed complementary. Therefore, we attribute the diminishing returns observed for DensePNV+S to the way we combine semantic and normal information. We assume that better results could be obtained with normals *and* semantics if one reasons about the consistency of image regions rather than on a pixel level (as is done by using the median).

Trainable pose verification. We next evaluate two trainable approaches (TrainPV), which are trained by randomly perturbed views (random) or by selecting views based on DensePE estimation (DPE) (*c.f.* Sec. 4.4.3). Even though both are trained using only appearance information, they still are able to use higher-level scene context as they use dense features extracted from a pre-trained fully convolutional network. Tab. 4.8 compares both TrainPV variants with the baselines and our hand-crafted approaches. Even though both variants use different training sets, they achieve nearly the same performance[10]. This indicates that the choice of training set is not critical in our setting. The results show that TrainPV outperforms the DensePV baseline, but not necessarily our hand-crafted variants based on multiple modalities. This result validates our idea of pose verification based on different sources of information. We also tried variants of TrainPV that use multiple modalities, but did not observe further improvements.

4.6 Summary

In this chapter, we firstly constructed a new indoor dataset for visual localization. Compared to existing datasets, our InLoc dataset (Sec. 4.2) is considerably large, composed of multiple buildings consisting of highly symmetric and repetitive rooms, weakly textured corridors, and movable objects. We also prepared a set of realistic and challenging query images captured by mobile phones. As a result, our dataset presents many challenging situations for indoor visual localization (*c.f.* Fig. 4.3). For further future works, we make our dataset and code to be publicly available[11].

We next presented a visual localization pipeline named InLoc (Sec. 4.3), which aims to estimate the 6DoF camera pose of a query image with respect to a large-scale indoor 3D map. To overcome the difficulties of indoor camera pose estimation, InLoc stores multiple pose candidates for the query and re-rank them through the progressively stricter verification steps, inspired by our outdoor visual localization pipeline described in Chap. 3. At the same time, assuming a high-quality 3D database (composed of database images and associating pixel-wise depth information), we employed densely extracted features for pose estimation and ranking processes, to adopt those accurate and dense 3D structural information into localization as much as possible. This concept is particularly reflected in our novel pose verification scheme that select the best matched camera pose from candidates based on pixel-wise image similarity measurement. For each of pose candidates, we synthesized a query view using database 3D point clouds around the estimated pose (relying on the density of the 3D points), and compare

[10] We are considering a discrete re-ranking problem on a few candidate poses per query. As such, it is not surprising to have very similar results.

[11] http://www.ok.sc.e.titech.ac.jp/INLOC

it with the original image. This process serves a pixel-wise consistency check between 2D query image and 3D structures, resulting in a considerable performance gain compared to existing scoring schemes based on rather sparse observations, *e.g.*, counting inlier matches. This fact is clearly demonstrated through experiments in Sec. 4.5 (*c.f.* Fig. 4.9, Tab. 4.3, Tab. 4.4, and Fig. 4.11). Also, as discussed in Sec. 4.5.5 and Sec. 4.5.6, InLoc pipeline naturally scales to larger-scale scenarios since it is based on an efficient image retrieval.

We next focused on improving performance of InLoc by using better metric to score the pose candidates (Sec. 4.4). Besides visible (color) information used in original InLoc (DensePV formulated by Eq. (4.2)), we also motivated to use geometric and semantic information, which are commonly used to understand the scene. Integrated to our pose verfication process, these additional information give improvements with nice margin (shown in Tab. 4.8 and Fig. 4.13). The experimental results also suggest that multiple modalities provide better metrics complementary to each, thus can potentially achieve the best performance when be combined properly (*c.f.* "Oracle" performances in Tab. 4.8). Yet, a simple combination of all modalities does not achieve this. These observations highlight potential future works for this area.

Another room for improvement is the computational timing, which will be a crucial issue for practical applications. InLoc requires additional computational time for processing dense features together with an enormous amount of 3D points (Tab. 4.7). This can potentially be addressed by integrating recent CNN-based methods for matching [99, 144] and pose estimation [59, 64] to large-scale environments, while optimizing implementations.

Chapter 5

Conclusion

In this thesis, we investigated visual localization (and its key technologies) for large-scale environments.

In Chap. 2, we first discussed about robust feature matching between images, which is an essential technology for each step of visual localization, including image ranking, camera pose estimation, and database construction (SfM). We mainly focused on feature deformations in the image induced by: (a) distorted projection model (Sec. 2.2) and (b) large viewpoint changes (Sec. 2.3). While such deformations actually are inevitable for efficient database construction and camera pose estimation, they can be a critical sources of miss matches. We addressed both issues by simulating all possible deformations on the image, as the pre-processing for standard feature matching algorithms. We generate a set of synthetic images approximating the possible deformations from the original image and extracts local features from them. In Sec. 2.3, we additionally provide simple metric learning scheme that achieves more robust and efficient feature matching. We confirmed the performances of our feature matching algorithms using both synthetic and real images datasets. We also demonstrated our feature matching approach combined with SfM pipeline can provide rich 3D models robustly, which can potentially be used as the 3D database for visual localization.

In Chap. 3, we proposed a general localization pipeline that ensures both scalability to larger database and accuracy of the estimated pose. The key feature of our pipeline is to integrate both 2D image-based approaches and 3D structure-based approaches into the progressive location (pose) determination scheme (Sec. 3.4). As illustrated in Fig. 1.7, We firstly perform image retrieval to find coarse location candidates of the query and estimates 6DoF camera pose using relevant database images. Our pose estimation based on SfM pipeline can provide an accurate 6DoF camera pose, while only requires 2D image database. Furthermore, this scheme can be adapted to any kind of image retrieval-based methods. To compare our pipeline with both 2D-based and 3D-based methods, we prepared a novel dataset for a large-scale urban scenario (Sec. 3.2). For each input image originally provided by [17], we newly provided the reference pose that has been manually annotated to represent 6DoF camera pose of the query and can be used for computing errors of the estimated pose. Using several evaluation metrics (Sec. 3.5), we confirmed our localization pipeline can provide accurate camera pose of the query using only 2D images, and shows a comparable computational time with state-of-the-art method.

In Chap. 4, we generalized our pipeline to indoor scenarios, which additionally assume a high-quality 3D scan for each local area. In Sec. 4.3.3, we adapted these accurate and dense 3D information to camera pose estimation step by exploiting feature matching for densely extracted features. In addition, we proposed novel pose verification process (Sec. 4.3.4) that effectively use dense 3D point cloud to validate (score) the estimated poses and select the best matching pose from candidates. Assuming a set of 6DoF pose candidates for a query, it renders each local 3D model around the candidate with respect to the camera pose and evaluate the similarity of the synthetic image to original one. Using rather dense information than standard inlier counting for sparse matches, our pose verification can achieve more robust pose estimation in some challenging scenes in indoor environments, *e.g.*, weakly textured and repetititve scenes. To address the lack of dataset for indoor visual localization, we again created a dataset (Sec. 4.2), consists of RGBD database images and query images taken in far different conditions from database images, in terms of viewpoints, capturing devices, and seasons. We also provided 6DoF reference poses for queries that are manually annotated and verified. Using this dataset, we showed that our visual localization pipeline again achieves state-of-the-art performance in indoor large-scale environments.

Altogether, we investigated each key component of visual localization pipeline including robust feature matching, and proposed a general pipeline that provides an accurate camera pose estimates even using only 2D image database. Since our pipeline is based on efficient location recognition via image retrieval followed by 3D reconstruction for local area around the query, it ensures computational efficiency and scalability to much larger scenes. Also, the proposed pipeline can be generalized to both outdoor and indoor scenarios, which assume 2D image database and additional rich 3D information for each local region. In addition, we provided two new datasets depicting large-scale visual localization scenarios in outdoor and indoor environments. Since these datasets contain reliable camera poses for queries that are essential for fairly evaluating both 2D- and 3D-based methods. We believe our datasets will be beneficial for future researches in this area.

Bibliography

[1] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele, “Real-time image-based 6-dof localization in large-scale environments,” in *Proc. CVPR*, 2012.

[2] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt, “Scalable 6-dof localization on mobile devices,” in *Proc. ECCV*, 2014.

[3] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart, “Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization,” in *Proc. RSS*, 2015.

[4] P. Bahl and V. N. Padmanabhan, “Radar: An in-building rf-based user location and tracking system,” in *Proc. INFOCOM*, 2000.

[5] Y.-C. Chen, J.-R. Chiang, H.-h. Chu, P. Huang, and A. W. Tsui, “Sensor-assisted wi-fi indoor location system for adapting to environmental dynamics,” in *Proc. ACM MSWiM*, 2005.

[6] B. F. D. Hähnel and D. Fox, “Gaussian processes for signal strength-based location estimation,” in *Proc. RSS*, 2006.

[7] S. Zickler and M. Veloso, “Rss-based relative localization and tethering for moving robots in unknown environments,” in *Proc. Intl. Conf. on Robotics and Automation*, 2010.

[8] J. Biswas and M. Veloso, “Wifi localization and navigation for autonomous indoor mobile robots,” in *Proc. Intl. Conf. on Robotics and Automation*, 2010.

[9] M. B. Alatise and G. P. Hancke, “Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended kalman filter,” *Sensors*, vol. 17, no. 10, p. 2164, 2017.

[10] H. Yan, Q. Shan, and Y. Furukawa, “RIDI: Robust IMU double integration,” in *Proc. ECCV*, 2018.

[11] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[12] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," in *Proc. CVPR*, 2013.

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[14] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *Proc. ACM SIGGRAPH*, 2006.

[15] N. Snavely, S. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *IJCV*, vol. 80, no. 2, pp. 189–210, 2008.

[16] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching," in *Proc. ECCV*, 2010.

[17] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys *et al.*, "City-scale landmark identification on mobile devices," in *Proc. CVPR*, 2011.

[18] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building rome in a day," *Comm. ACM*, vol. 54, no. 10, pp. 105–112, 2011.

[19] E. Wijmans and Y. Furukawa, "Exploiting 2D floorplan for building-scale panorama RGBD alignment," in *Proc. CVPR*, 2017.

[20] A. Torii, R. Arandjelovi̇c, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," *IEEE PAMI*, vol. 40, no. 2, pp. 257–271, 2018.

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

[22] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, "InLoc: Indoor visual localization with dense matching and view synthesis," in *Proc. CVPR*, 2018.

[23] C. Wu, S. Agarwal, B. Curless, and S. Seitz, "Multicore bundle adjustment," in *Proc. CVPR*, 2011.

[24] C. Wu, "Towards Linear-time Incremental Structure From Motion," in *Proc. 3DV*, 2013.

[25] J. L. Schönberger and J.-M. Frahm, "Structure-From-Motion Revisited," in *Proc. CVPR*, 2016.

[26] R. Arandjelović and A. Zisserman, “Three things everyone should know to improve object retrieval,” in *Proc. CVPR*, 2012.

[27] ——, “Dislocation: Scalable descriptor distinctiveness for location recognition,” in *Proc. ACCV*, 2014.

[28] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Comm. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[29] O. Chum, J. Matas, and J. Kittler, “Locally optimized ransac,” *Pattern recognition*, pp. 236–243, 2003.

[30] K. Lebeda, J. Matas, and O. Chum, “Fixing the locally optimized ransac–full experimental evaluation,” in *Proc. BMVC.*, 2012.

[31] T. Lindeberg, “Feature detection with automatic scale selection,” *IJCV*, vol. 30, no. 2, pp. 79–116, 1998.

[32] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[33] K. Mikolajczyk and C. Schmid, “Scale & affine invariant interest point detectors,” *IJCV*, vol. 60, no. 1, pp. 63–86, 2004.

[34] C. Wu, “SiftGPU: A GPU implementation of scale invariant feature transform (SIFT),” http://cs.unc.edu/~ccwu/siftgpu, 2007.

[35] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in *Proc. ECCV*, 2006.

[36] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, H. Chen, R. Vedantham, R. Grzeszczuk, and B. Girod, “Residual enhanced visual vectors for on-device image matching,” in *Proc. ASILOMAR*, 2011.

[37] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, “Visual place recognition with repetitive structures,” in *Proc. CVPR*, 2013.

[38] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla, “24/7 place recognition by view synthesis,” in *Proc. CVPR*, 2015.

[39] T. Sattler, M. Havlena, K. Schindler, and M. Pollefeys, “Large-scale location recognition and the geometric burstiness problem,” in *Proc. CVPR*, 2016.

[40] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," *IEEE PAMI*, vol. 40, no. 6, pp. 1437–1451, 2018.

[41] H. J. Kim, E. Dunn, and J.-M. Frahm, "Learned contextual feature reweighting for image geo-localization," in *Proc. CVPR*, 2017.

[42] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. CVPR*, 2007.

[43] J. L. Schönberger, T. Price, T. Sattler, J.-M. Frahm, and M. Pollefeys, "A Vote-and-Verify Strategy for Fast Spatial Verification in Image Retrieval," in *Proc. ACCV*, 2016.

[44] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE PAMI*, vol. 34, no. 9, pp. 1704–1716, 2012.

[45] R. Arandjelovic and A. Zisserman, "All about vlad," in *Proc. CVPR*, 2013.

[46] T. Sattler, M. Havlena, F. Radenovic, K. Schindler, and M. Pollefeys, "Hyperpoints and fine vocabularies for large-scale location recognition," in *Proc. ICCV*, 2015.

[47] L. Quan and Z. Lan, "Linear n-point camera pose determination," *IEEE PAMI*, vol. 21, no. 8, pp. 774–780, 1999.

[48] S. Cao and N. Snavely, "Minimal scene descriptions from structure from motion models," in *Proc. CVPR*, 2014.

[49] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE PAMI*, vol. 39, no. 9, pp. 1744–1756, 2017.

[50] Y. Li, N. Snavely, D. P. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3d point clouds," in *Proc. ECCV*, 2012.

[51] S. Choudhary and P. Narayanan, "Visibility probability structure from sfm datasets and applications," in *Proc. ECCV*, 2012.

[52] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof, "From structure-from-motion point clouds to fast location recognition," in *Proc. CVPR*, 2009.

[53] F. Camposeco, T. Sattler, A. Cohen, A. Geiger, and M. Pollefeys, "Toroidal constraints for two-point localization under high outlier ratios," in *Proc. CVPR*, 2017.

[54] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler, "Hybrid Camera Pose Estimation," in *Proc. CVPR*, 2018.

[55] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *Proc. CVPR*, 2017.

[56] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *Proc. CVPR*, 2013.

[57] X. Sun, Y. Xie, P. Luo, and L. Wang, "A Dataset for Benchmarking Image-based Localization," in *Proc. CVPR*, 2017.

[58] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, "InLoc: Indoor visual localization with dense matching and view synthesis," *IEEE PAMI*, 2018.

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

[60] E. Brachmann and C. Rother, "Learning less is more-6d camera localization via 3d surface regression," in *Proc. CVPR*, 2018.

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

[62] V. Balntas, S. Li, and V. Prisacariu, "RelocNet: Continuous Metric Learning Relocalisation using Neural Nets," in *Proc. ECCV*, 2018.

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

[64] E. Brachmann and C. Rother, "Expert sample consensus applied to camera relocalization," in *Proc. ICCV*, 2019.

[65] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla, "Are large-scale 3D models really necessary for accurate visual localization?" in *Proc. CVPR*, 2017.

[66] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. ECCV*, 2012.

[67] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using sfm and object labels," in *Proc. ICCV*, 2013.

[68] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, “3D Semantic Parsing of Large-Scale Indoor Spaces,” in *Proc. CVPR*, 2016.

[69] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, “Matterport3D: Learning from RGB-D Data in Indoor Environments,” in *Proc. 3DV*, 2017.

[70] S. Wang, S. Fidler, and R. Urtasun, “Lost shopping! monocular localization in large indoor spaces,” in *Proc. ICCV*, 2015.

[71] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother, “Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image,” in *Proc. CVPR*, 2016.

[72] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary robust independent elementary features,” in *Proc. ECCV*, 2010.

[73] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: an efficient alternative to sift or surf,” in *Proc. ICCV*, 2011.

[74] J.-M. Morel and G. Yu, “Asift: A new framework for fully affine invariant image comparison,” *SIAM journal on imaging sciences*, vol. 2, no. 2, pp. 438–469, 2009.

[75] D. Mishkin, J. Matas, and M. Perdoch, “MODS: Fast and robust method for two-view matching,” *CVIU*, vol. 141, pp. 81–93, 2015.

[76] Y. Ke and R. Sukthankar, “PCA-SIFT: A more distinctive representation for local image descriptors,” in *Proc. CVPR*, 2004.

[77] J. Philbin, M. Isard, J. Sivic, and A. Zisserman, “Descriptor learning for efficient retrieval,” in *Proc. ECCV*, 2010.

[78] K. Mikolajczyk and J. Matas, “Improving descriptors for fast tree matching by optimal linear projection,” in *Proc. ICCV*, 2007.

[79] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, “Learning a mahalanobis metric from equivalence constraints,” *J. Machine Learning Research*, vol. 6, no. 6, pp. 937–965, 2005.

[80] S. Karaoglu, I. Everts, J. C. van Gemert, and T. Gevers, “Per-patch metric learning for robust image matching,” in *Intl. Conf. Image Proc.*, 2015.

[81] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, “Total recall: Automatic query expansion with a generative feature model for object retrieval,” in *Proc. ICCV*, 2007.

[82] H. Jegou, M. Douze, and C. Schmid, “Hamming embedding and weak geometric consistency for large scale image search,” in *Proc. ECCV*, 2008.

[83] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” in *Proc. CVPR*, 2006.

[84] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *Proc. ICCV*, 2003.

[85] J. C. Van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek, “Visual word ambiguity,” *IEEE PAMI*, vol. 32, no. 7, pp. 1271–1283, 2010.

[86] O. Chum, A. Mikulik, M. Perdoch, and J. Matas, “Total recall ii: Query expansion revisited,” in *Proc. CVPR*, 2011.

[87] H. Jégou, M. Douze, and C. Schmid, “On the burstiness of visual elements,” in *Proc. CVPR*, 2009.

[88] P. Gronat, J. Sivic, G. Obozinski, and P. Tomas, “Learning and calibrating per-location classifiers for visual place recognition,” *IJCV*, vol. 118, no. 3, pp. 319–336, 2016.

[89] T.-Y. Lin, A. RoyChowdhury, and S. Maji, “Bilinear cnn models for fine-grained visual recognition,” in *Proc. ICCV*, 2015.

[90] T. Weyand, I. Kostrikov, and J. Philbin, “Planet-photo geolocation with convolutional neural networks,” in *Proc. ECCV*, 2016.

[91] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt, “Image Retrieval for Image-Based Localization Revisited,” in *Proc. BMVC.*, 2012.

[92] L. Liu, H. Li, and Y. Dai, “Efficient Global 2D-3D Matching for Camera Localization in a Large-Scale 3D Map,” in *Proc. ICCV*, 2017.

[93] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers, “Image-Based Localization Using LSTMs for Structured Feature Correlation,” in *Proc. ICCV*, 2017.

[94] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys, “Quad-networks: unsupervised learning to rank for interest point detection,” in *Proc. CVPR*, 2017.

[95] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned Invariant Feature Transform," in *Proc. ECCV*, 2016.

[96] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," in *NIPS*, 2017.

[97] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, "Semantic Visual Localization," in *Proc. CVPR*, 2018.

[98] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," in *Proc. ICCV*, 2017.

[99] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic, "Neighbourhood consensus networks," in *NIPS*, 2018.

[100] I. Melekhov, A. Tiulpin, T. Sattler, M. Pollefeys, E. Rahtu, and J. Kannala, "DGC-Net: Dense geometric correspondence network," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[101] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Deepmatching: Hierarchical deformable dense matching," *IJCV*, vol. 120, no. 3, pp. 300–323, 2016.

[102] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *Proc. Intl. Conf. on Robotics and Automation*, 2014.

[103] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena, "Contextually guided semantic labeling and search for three-dimensional point clouds," *Intl. J. of Robotics Research*, vol. 32, no. 1, pp. 19–34, 2013.

[104] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi, "Real-time RGB-D camera relocalization," in *Proc. ISMAR*, 2013.

[105] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes," in *Proc. CVPR*, 2017.

[106] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to Navigate the Energy Landscape," in *Proc. 3DV*, 2016.

[107] J. Son, S. Kim, and K. Sohn, "Fast affine-invariant image matching based on global bhattacharyya measure with adaptive tree," in *Intl. Conf. Image Proc.*, 2015.

[108] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J. Frahm, "USAC: A universal framework for random sample consensus," *IEEE PAMI*, vol. 35, no. 8, pp. 2022–2038, 2013.

[109] J. Cruz-Mota, I. Bogdanova, B. Paquier, M. Bierlaire, and J.-P. Thiran, “Scale invariant feature transform on the sphere: Theory and applications,” *IJCV*, vol. 98, no. 2, pp. 217–241, 2012.

[110] A. Vedaldi and B. Fulkerson, “VLFeat: An open and portable library of computer vision algorithms,” http://www.vlfeat.org/, 2008.

[111] A. Torii, M. Havlena, and T. Pajdla, “From google street view to 3D city models,” in *OMNIVIS*, 2009.

[112] T. Sato, T. Pajdla, and N. Yokoya, “Epipolar geometry estimation for wide-baseline omnidirectional street view images,” in *Proc. ICCV Workshop*, 2011.

[113] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

[114] Z. Wang, B. Fan, and F. Wu, “Affine subspace representation for feature description,” in *Proc. ECCV*, 2014.

[115] K. Mikolajczyk and C. Schmid, “An affine invariant interest point detector,” in *Proc. ECCV*, 2002.

[116] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua, “LDAHash: Improved matching with smaller descriptors,” *IEEE PAMI*, vol. 34, no. 1, pp. 66–78, 2012.

[117] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen, “On benchmarking camera calibration and multi-view stereo for high resolution imagery,” in *Proc. CVPR*, 2008.

[118] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *Proc. VISSAPP*, 2009.

[119] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, “Visual place recognition: A survey,” *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 1–19, 2016.

[120] J. L. Schönberger, F. Radenović, O. Chum, and J.-M. Frahm, “From Single Image Query to Detailed 3D Reconstruction,” in *Proc. CVPR*, 2015.

[121] D. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher, “Discrete-continuous optimization for large-scale structure from motion,” in *Proc. CVPR*, 2011.

[122] T. Weyand and B. Leibe, “Discovering Details and Scene Structure with Hierarchical Iconoid Shift,” in *Proc. ICCV*, 2013.

[123] S. Gammeter, T. Quack, and L. Van Gool, "I Know What You Did Last Summer: Object-Level Auto-Annotation of Holiday Snaps," in *Proc. ICCV*, 2009.

[124] A. Torii, Y. Dong, M. Okutomi, J. Sivic, and T. Pajdla, "Efficient localization of panoramic images using tiled image descriptors," *IPSJ Trans. Computer Vision and Applications*, vol. 6, pp. 58–62, 01 2014.

[125] A. Torii, J. Sivic, M. Okutomi, and T. Pajdla, "Visual place recognition with repetitive structures," *IEEE PAMI*, vol. 37, no. 11, pp. 2346–2359, 2015.

[126] B. Zeisl, T. Sattler, and M. Pollefeys, "Camera Pose Voting for Large-Scale Image-Based Localization," in *Proc. ICCV*, 2015.

[127] D. Sibbing, T. Sattler, B. Leibe, and L. Kobbelt, "SIFT-Realistic Rendering," in *Proc. 3DV*, 2013.

[128] M. Aubry, B. C. Russell, and J. Sivic, "Painting-to-3d model alignment via discriminative visual elements," *ACM Trans. Graphics*, vol. 33, no. 2, p. 14, 2014.

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

[130] R. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle, "Review and analysis of solutions of the three point perspective pose estimation problem," *IJCV*, vol. 13, no. 3, pp. 331–356, 1994.

[131] S. Cao and N. Snavely, "Graph-Based Discriminative Learning for Location Recognition," in *Proc. CVPR*, 2013.

[132] W. Zhang and J. Kosecka, "Image based localization in urban environments," in *Proc. 3DPVT*, 2006.

[133] A. R. Zamir and M. Shah, "Accurate image localization based on google maps street view," in *Proc. ECCV*, 2010.

[134] E. Zheng and C. Wu, "Structure From Motion Using Structure-Less Resection," in *Proc. ICCV*, 2015.

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

[136] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla, "Benchmarking 6DOF outdoor visual localization in changing conditions," in *Proc. CVPR*, 2018.

[137] A. R. Zamir and M. Shah, “Image Geo-Localization Based on MultipleNearest Neighbor Feature Matching Using Generalized Graphs,” *IEEE PAMI*, vol. 36, no. 8, pp. 1546–1558, 2014.

[138] S. Agarwal, K. Mierle, and Others, “Ceres solver,” http://ceres-solver.org.

[139] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson, “City-Scale Localization for Cameras with Known Vertical Direction,” *IEEE PAMI*, vol. 39, no. 7, pp. 1455–1461, 2017.

[140] A. Mikulík, F. Radenović, O. Chum, and J. Matas, “Efficient image detail mining,” in *Proc. ACCV*, 2014.

[141] R. Hartley, J. Trumpf, Y. Dai, and H. Li, “Rotation Averaging,” *IJCV*, vol. 103, no. 3, pp. 267–305, 2013.

[142] H. Jégou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE PAMI*, vol. 33, no. 1, pp. 117–128, 2011.

[143] T. Duff, K. Kohn, A. Leykin, and T. Pajdla, “Plmp - point-line minimal problems in complete multi-view visibility,” in *Proc. ICCV*, 2019.

[144] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, “D2-Net: A Trainable CNN for Joint Detection and Description of Local Features,” in *Proc. CVPR*, 2019.

[145] A. Debski, W. Grajewski, W. Zaborowski, and W. Turek, “Open-source localization device for indoor mobile robots,” *Procedia Computer Science*, vol. 76, pp. 139–146, 2015.

[146] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, “Real-time detection and tracking for augmented reality on mobile phones,” *Visualization and Computer Graphics*, vol. 16, no. 3, pp. 355–368, 2010.

[147] R. O. Castle, G. Klein, and D. W. Murray, “Video-rate localization in multiple maps for wearable augmented reality,” in *ISWC*, 2008.

[148] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 Year, 1000km: The Oxford RobotCar Dataset,” *IJRR*, vol. 36, no. 1, pp. 3–15, 2017.

[149] T. Schmidt, R. Newcombe, and D. Fox, “Self-Supervised Visual Descriptor Learning for Dense Correspondence,” *IEEE RA-L*, vol. 2, no. 2, pp. 420–427, 2017.

[150] J. Xiao and Y. Furukawa, “Reconstructing the world’s museums,” *IJCV*, vol. 110, no. 3, pp. 243–258, 2014.

[151] A. R. Zamir, A. Sax, , W. B. Shen, L. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proc. CVPR*, 2018.

[152] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, 2017.

[153] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Proc. ECCV*, 2014.

[154] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ADE20K dataset," *IJCV*, vol. 127, no. 3, pp. 302–321, 2019.

[155] N. Kobyshev, H. Riemenschneider, and L. V. Gool, "Matching Features Correctly through Semantic Understanding," in *Proc. 3DV*, 2014.

[156] E. Stenborg, C. Toft, and L. Hammarstrand, "Long-term Visual Localization using Semantically Segmented Images," in *Proc. Intl. Conf. on Robotics and Automation*, 2018.

[157] F. Yu, J. Xiao, and T. A. Funkhouser, "Semantic alignment of LiDAR data at city scale," in *Proc. CVPR*, 2015.

[158] M. Schreiber, C. Knöppel, and U. Franke, "LaneLoc: Lane marking based localization using highly accurate maps," in *Proc. IV*, 2013.

[159] N. Atanasov, M. Zhu, K. Daniilidis, and G. J. Pappas, "Localization from semantic observations via the matrix permanent," *Intl. J. of Robotics Research*, vol. 35, no. 1-3, pp. 73–99, 2016.

[160] C. Toft, E. Stenborg, L. Hammarstrand, L. Brynte, M. Pollefeys, T. Sattler, and F. Kahl, "Semantic Match Consistency for Long-Term Visual Localization," in *Proc. ECCV*, 2018.

[161] R. Arandjelović and A. Zisserman, "Visual vocabulary with a semantic twist," in *Proc. ACCV*, 2014.

[162] C. Toft, C. Olsson, and F. Kahl, "Long-term 3D Localization and Pose from Semantic Labellings," in *Proc. ICCV Workshop*, 2017.

[163] G. Singh and J. Košecká, "Semantically Guided Geo-location and Modeling in Urban Environments," in *Large-Scale Visual Geo-Localization*, 2016.

[164] X. Yu, S. Chaturvedi, C. Feng, Y. Taguchi, T.-Y. Lee, C. Fernandes, and S. Ramalingam, "VLASE: Vehicle Localization by Aggregating Semantic Edges," in *Proc. IEEE/RSJ Conf. on Intelligent Robots and Systems*, 2018.

[165] N. Radwan, A. Valada, and W. Burgard, "VLocNet++: Deep multitask learning for semantic visual localization and odometry," *IEEE RA-L*, vol. 3, no. 4, pp. 4407–4414, 2018.

[166] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D Reconstruction at Scale using Voxel Hashing," *ACM TOG*, vol. 32, no. 6, p. 169, 2013.

[167] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proc. ISMAR*, 2011.

[168] M. Halber and T. Funkhouser, "Fine-To-Coarse Global Registration of RGB-D Scans," in *Proc. CVPR*, 2017.

[169] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE PAMI*, no. 4, pp. 376–380, 1991.

[170] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman, "Sift flow: Dense correspondence across different scenes," in *Proc. ECCV*, 2008.

[171] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *IEEE PAMI*, vol. 32, no. 5, pp. 815–830, 2010.

[172] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. IJCAI*, 1981.

[173] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. CVPR*, 2017.

[174] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Proc. ECCV*, 2014.

[175] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.

[176] A. R. Widya, A. Torii, and M. Okutomi, "Structure from motion using dense cnn features with keypoint relocalization," *IPSJ Transactions on Computer Vision and Applications*, vol. 10, no. 1, p. 6, 2018.

[177] H. Jégou and O. Chum, "Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening," in *Proc. ECCV*, 2012.

[178] H. Taira, I. Rocco, J. Sedlar, M. Okutomi, J. Sivic, T. Pajdla, T. Sattler, and A. Torii, "Is This The Right Place? Geometric-Semantic Pose Verification for Indoor Visual Localization," *arXiv preprint arXiv:1908.04598*, 2019.

[179] A. Cohen, T. Sattler, and M. Pollefeys, "Merging the Unmatchable: Stitching Visually Disconnected SfM Models," in *Proc. ICCV*, 2015.

[180] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[181] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009.

[182] I. Rocco, R. Arandjelović, and J. Sivic, "Convolutional neural network architecture for geometric matching," in *Proc. CVPR*, 2017.

[183] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proc. ICCV*, 2015.

[184] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[185] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. CVPR*, 2016.

[186] P. Agrawal, R. B. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Proc. ECCV*, 2014.