TUTORIAL

Visual Place Recognition

A Tutorial

By Stefan Schubert, Peer Neubert, Sourav Garg, Michael Milford, and Tobias Fischer

Localization is an essential capability for mobile robots, enabling them to build a comprehensive representation of their environment and interact with the environment effectively toward a goal. A rapidly growing field of research in this area is visual place recognition (VPR), which is the ability to recognize previously seen places in the world based solely on images.

INTRODUCTION

The volume of published research on VPR has shown significant and continuous growth over the years, from two articles with "visual place recognition" and seven articles with "place recognition" in the title in 2006, to 65 and 163 articles, respectively, in 2022 (source: Google Scholar with query *allintitle: "title"*). A number of survey and benchmarking articles have discussed the challenges, open questions, and achievements in the field of VPR [1], [2], [3], [4].

This present work is the first tutorial article on VPR. It unifies the terminology of VPR and complements prior research in two important directions.

1) It provides a systematic introduction for newcomers to the field, covering topics such as the formulation of the VPR problem, a generic algorithmic pipeline, an evaluation methodology for VPR approaches, and the major challenges for VPR and how they may be addressed.
2) As a contribution to researchers acquainted with the VPR problem, it examines the intricacies of different VPR problem types regarding input (database or query set), data processing (online or offline), and output (one or multiple matches per query image). The tutorial also discusses the subtleties behind the evaluation of VPR algorithms, e.g., the evaluation of a VPR system that has to find *all* matching database images per query as opposed to just a *single* match.

Practical code examples in Python illustrate to prospective practitioners and researchers how VPR is implemented and evaluated. The corresponding source code is available online, along with a list of other open source implementations from the literature: `https://github.com/stschubert/VPR_Tutorial`. The link also provides a Jupyter notebook written in Python that guides users through a basic VPR pipeline. It allows users to experiment with additional image descriptors, benchmark datasets, and evaluation metrics.

The following section, "The Basics of VPR," provides a basic introduction to the VPR problem. The "The VPR Problem and Its Details as Reflected in This Tutorial" section then outlines the structure of the remainder of this tutorial.

THE BASICS OF VPR

VPR involves matching one or multiple image sets to determine which images show the same places in the world. These image sets are typically recorded with a mobile device, such as

Digital Object Identifier 10.1109/MRA.2023.3310859
Date of current version: 22 September 2023

a cell phone or an augmented reality/virtual reality headset, or with a camera mounted on a variety of platforms, such as a robot, uncrewed aerial vehicle, car, bus, train, bicycle, or boat.

Essentially, VPR is an image retrieval problem where the context is to recognize previously seen places. This context provides additional information and structure beyond a general image retrieval setup. Many VPR methods exploit the context to match images of the same places in a wide range of environments, including those with significant appearance and viewpoint differences. For example, one piece of additional information that is often exploited is that *consecutive* images taken by a camera mounted on a car will depict spatially close places in the world.

Figure 1 provides an overview of the typical steps and components of a basic VPR pipeline. Given a reference set composed of database images $\mathbf{I}_i \in DB$ of known places and one or multiple query images $\mathbf{I}_j \in Q$, the goal is to find matches between these two sets, i.e., those instances where image j from the query set shows the same place as image i from the database. To find these matches, it is essential to compute one or multiple descriptors $\mathbf{d}$ for each image—these descriptors should be similar for images showing the same place and dissimilar for different places. A descriptor is typically represented as a numerical vector (e.g., 128-D or 4,096-D). Conceptually, we can think of a matrix $\mathbf{S}$ of all pairwise descriptor similarities s_{ij} between the database and query images as the basis for deciding which images should be matched. In practice, we must carefully choose the algorithms used to compute and compare the image descriptors $\mathbf{d}$, taking into account the specific challenges and context of the VPR problem at hand. The remainder of this tutorial will provide more detail on these aspects and discuss the VPR problem from a broader theoretical and practical perspective.

THE VPR PROBLEM AND ITS DETAILS AS REFLECTED IN THIS TUTORIAL

The "History, Relevance, and Related Areas" section of this tutorial will outline the relevance and history of the VPR problem as well as its relation to other areas, particularly its importance for topological simultaneous localization and mapping (SLAM), where the database *DB* corresponds to the set of previously visited places in the map. In fact, one of the original drivers for VPR research was the generation of loop closures for SLAM systems, that is, recognizing a place when revisiting it (e.g., in a loop) and tying the current observation with that already in the map (i.e., closure) [6]. One of the earliest examples of such a topological SLAM system is FAB-MAP [7], also referred to as *appearance-only SLAM,* where loop closure generation is based on appearance only (i.e., images) and thus different from 3D/metric SLAM systems such as ORB-SLAM [8], where the map and the visual landmarks are expressed in 3D.

The definition of a *place* is an integral aspect of VPR. In this tutorial, we follow the definition that two images must have some visual overlap, i.e., shared image content like the same buildings, to be considered as "taken at the same place" [2]. This definition allows one to subsequently estimate the camera transformation between matched images for tasks like visual localization, mapping, or SLAM—indeed, the required amount of visual overlap depends on the specific application. We note that an alternative definition used in particular by some researchers [9] is that two places are matching purely based on their position, without taking the orientation and, in turn, the visual overlap, into account. The "VPR Problem Categories and Use Cases" section will present different applications for VPR and discuss the various subtypes of VPR problems that arise from variations in the available input data, the required data processing, and the requested output.

VPR algorithms are often tailored to the particular properties of an application. The "A Generic Pipeline for VPR" section will provide details on a *generic VPR pipeline* that serves as a common basis for diverse practical settings and their unique characteristics. From this section onward, this tutorial includes practical code examples in Python.

It is important to note that not all VPR algorithms address the same VPR problem, e.g., regarding the requested number of image matches per query. This is particularly critical when it comes to evaluating and comparing the performance of

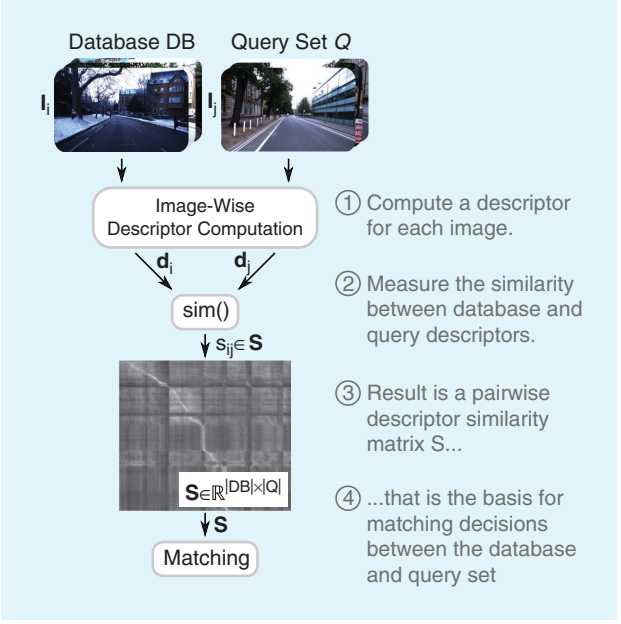


FIGURE 1. This figure illustrates the key steps and components of VPR as outlined in the "Introduction" section. Historically, the matching decisions in step 4 were used for loop closure in simultaneous localization and mapping (SLAM); the "History, Relevance, and Related Areas" section provides an overview of the history and relevance of the VPR problem. While this figure illustrates a common use case where incoming imagery in the query set is compared to a database, the "VPR Problem Categories and Use Cases" section distinguishes different VPR problem categories based on this pipeline and also relates them to VPR use cases. The details of each shown computational step will be discussed in the "A Generic Pipeline for VPR" section, followed by details on the evaluation of VPR pipelines in the "Evaluation of the Performance" section. (Source: Photos from [5].)

different VPR algorithms. The "Evaluation of the Performance" section explains and discusses the evaluation pipelines that consider various datasets, ground truth subtleties, and different performance metrics.

The properties of the underlying data have a significant impact on the difficulty of the resulting VPR problem and the suitability of a particular algorithm. The "Challenges and Common Ways of Addressing Them" section will discuss challenges such as severe appearance changes due to varying illumination or weather conditions, large viewpoint changes between images of the same place, and perceptual aliasing, i.e., the challenge that images taken at two distinct places can appear remarkably similar. This section will also present common ways of addressing these challenges to improve robustness, performance, runtime, and memory efficiency. These approaches include methodological extensions of the general purpose pipeline that partially build upon a robotic context (e.g., with image sets recorded as videos along trajectories) where VPR differs from pure image retrieval. This often allows the exploitation of additional knowledge and information such as spatiotemporal sequences (i.e., consecutive images in the database *DB* and query *Q* are also neighboring in the world) or intra-set similarities (i.e., similarities *within DB* or *Q*).

HISTORY, RELEVANCE, AND RELATED AREAS

VPR research can be traced back to advances in visual SLAM, visual geolocalization, and image retrieval applied to images of places [10]. In the robotics literature, VPR has historically been called *loop closure detection* and was mainly used for this purpose for visual SLAM [10]. VPR gained more prominence in the field as the earlier metric SLAM methods based on global and local bundle adjustment techniques could handle only limited-size environments, thus paving the way for topological SLAM techniques based on bag-of-words approaches, such as FAB-MAP [7]. In addition to its relevance within SLAM pipelines, VPR also remains a crucial component of localization-only pipelines where the map is available a priori.

Early VPR research primarily focused on place recognition under constant or slightly varying environmental conditions. Addressing appearance changes due to more severe condition changes, such as day-night cycles or seasonal shifts, emerged in the late 2000s. These methods relied, for example, on local feature matching [11] or on continuously updating appearance-based maps [12]. Since then, research on VPR under challenging conditions has steadily increased—for example, tackling the challenging day-night shift [13]. Recent works make heavy use of datasets with condition changes that have appeared since 2012 [1], [3]. In 2014, the use of deep learning for VPR [14] emerged as a way to handle challenging data and has since proven effective in changing environments [15]. In addition to images and image descriptors, VPR research has also explored the use of additional information, such as sequences, intra-set similarities, weak GPS signals, or odometry, to improve performance [16].

In terms of the relationship between VPR and other fields, we recommend the following tutorials. Durrant-Whyte and Bailey [10] provide an overview of probabilistic SLAM and include a section on loop closure detection, although a lot of progress has been made in VPR as this tutorial was published more than 15 years ago. Tsintotas et al. [6] specifically investigate the loop closure problem in SLAM. Scaramuzza and Fraundorfer [17] discuss visual odometry, which involves estimating the ego-motion of an agent based on visual input. Visual odometry is thus complementary to VPR and can be combined with VPR to detect loop closures when building a SLAM system. It is important to note, however, that the scope of this tutorial is limited to providing an accessible introduction to VPR and its core concepts. Aspects such as the integration of VPR methods into a complete SLAM or relocalization system are beyond the scope of this tutorial and would require discussing many additional aspects, such as batch optimization, which are not directly related to VPR.

Beyond loop closure detection, VPR is necessary if global position sensors such as global navigation satellite systems (GNSS) like GPS, Galileo, or BeiDou are not available or are inaccurate. In urban environments, buildings or other structures can lead to "urban canyons" that block line-of-sight satellite signals, causing occlusions that prevent a GNSS receiver from obtaining accurate position information. In addition to occlusions, reflections of GNSS signals off buildings and other structures, so-called non-line-of-sight signals, can further hinder the accuracy of GNSS. This issue is not limited to urban environments as similar occlusions and reflections can occur in natural environments, such as in valleys or canyons. Similarly, indoor environments and caves also hinder GNSS due to the absorption or reflection of satellite signals by walls.

Alternatively, VPR can serve as a redundant component in autonomous systems for fault tolerance and general GNSS outages, such as satellite service disruptions, degradation, or position/time anomalies. It is worth noting that all GNSS systems can potentially be hacked or blocked for nonmilitary use by a central authority. Other systems may not be equipped with a GNSS receiver due to cost or security concerns. In the case of robotic extraterrestrial missions, installing a GNSS system may be too expensive or time consuming.

VPR PROBLEM CATEGORIES AND USE CASES

In the localization and mapping literature, VPR has been used in different ways depending on three key attributes of its formulation: the *input*, which deals with how the reference and query images are made available (i.e., single session versus multisession); *data processing*, which defines the mode of operation (i.e., online versus batch); and *output*, which determines the kind of expected output (i.e., single best match versus multimatch). The following section, "VPR Problem Categories," explains these problem categories in more detail. The "VPR Use Cases" section then presents different VPR use cases using these categories. Table 1 summarizes these use cases along with their required input and data processing. Note that there might be exceptions and deviations from these categories, such as [18], which uses multiple disjoint sequences as reference. However, we believe that the

proposed taxonomy serves as a good starting point for future research to organize the various VPR use cases.

VPR PROBLEM CATEGORIES

We distinguish three main dimensions along which VPR problems can vary, creating different VPR problem categories or subtypes.

1) *Input—single-session VPR versus multisession VPR:* Are there two separate input sets, one for the database DB and one for the query Q, or is it a single set that is compared to itself? Single-session VPR is the matching of images within a single set of images so that the query set Q equals the database DB (i.e., $Q = DB$). A practical consideration in this case is the suppression of matches with recently acquired images—while full SLAM systems typically rely on a motion model for such suppression, standalone VPR systems often use heuristics. In contrast, multisession VPR is the matching of the two disjoint image sets (i.e., $DB \cap Q = \emptyset$), which were recorded at different times (e.g., summer and winter) or by different platforms (e.g., mobile robot and cell phone).
2) *Data processing—online VPR versus batch VPR:* Are the images available and processed individually, one after the other, or are they all available in a single batch from the beginning? Online VPR has to deal with a growing set Q (i.e., $Q \neq$ const) (where const means *constant*) and a set DB that is either given (i.e., $DB =$ const) or also growing (i.e., $DB \neq$ const). In contrast, batch VPR can build upon the full sets Q (i.e., $Q =$ const) and DB (i.e., $DB =$ const). Growing image sets in the case of online VPR limit the number of viable methods. For example, approaches like descriptor standardization [19] based on the statistics of all image descriptors or similarity matrix decomposition [20] cannot be used without modifications. This is further discussed in the "Challenges and Common Ways of Addressing Them" section. Note that for single-session VPR, the difference between a growing or constant query set pertains to whether VPR is being performed online or in batch mode.
3) *Output—single-best-match VPR versus multimatch VPR:* Is the intended output for a query image a single image from the database that shows the same place, or do we request all images of this place? Single-best-match VPR returns only the best matching database image $I_i^* \in DB$ per query image $I_j \in Q$. In contrast, the aim of multimatch VPR is to find all matching database images for each query image. In practice, the difference between single-best-match VPR and multimatch VPR often boils down to finding either the maximum similarity between a query and all database images or all similarities above a certain threshold, as shown in the "Output: Matching Decisions" section. Identifying all matching images is often more challenging than finding only one correct match as it requires an explicit decision for each database image whether it shows the same place as the query image or not [1].

Let us illustrate these problem categories with the example of determining the rough pose [*x, y, heading, floor*] of a cell phone in a building, e.g., to guide people to desired places. To achieve this, we first need to map the building before the people can use their cell phones to localize in the building. For this first step of mapping the building, we could use a manually controlled mobile robot equipped with a camera to collect a query set Q^{mapping} of images together with some additional sensor data like odometry. Given these images of all places, we can run a mapping algorithm that processes all images and other data to obtain a metric map of the building, which associates all images in Q^{mapping} with metric poses. Part of this mapping is a *single-session batch multimatch VPR* for loop closure detection that compares the whole image set Q^{mapping} (batch VPR) to itself (single-session VPR) to find *all* loop closures for each image (multimatch VPR). Here, batch processing the whole set Q^{mapping} allows the application of computationally expensive but accurate algorithms.

After mapping (potentially years later), the second step is the actual localization of a cell phone using its camera stream. When localizing, we treat the robot's mapping query set Q^{mapping} as database DB^{loc} and compare it to query images Q^{loc} from a cell phone's camera. To determine the location of a cell phone, a *multisession online single-best-match VPR* can be used that compares the stream of query images Q^{loc} to the fixed database DB^{loc} (multisession VPR) online (online VPR) to find the best matching database image (single-best-match VPR) with its corresponding pose information.

TABLE 1. Combinations of the VPR *input* and *data processing* categories (cf. the "VPR Problem Categories" section) with corresponding use cases (cf. the "VPR Use Cases" section). In single-session VPR, there is a single input set that is compared to itself. This is, for example, the case in online SLAM, where this set grows while the robot is exploring its environment (online VPR), and in mapping, where prerecorded data are processed in a batch manner. In the case where the input consists of two sets, the database *DB* and the query set Q (multisession VPR), one can again distinguish the case that data need to be processed online as they are being collected (online VPR) or in a batch as in multisession mapping. In multisession online VPR, *DB* can be either growing (as in multirobot mapping) or fixed (as in visual (re-)localization).

	DATA PROCESSING		
INPUT	**ONLINE VPR**		**BATCH VPR**
SINGLE-SESSION VPR	Online SLAM		Mapping
MULTISESSION VPR	***DB* grows**	***DB* const**	Multisession Mapping
	Multirobot mapping	Visual (re-) localization	

In summary, VPR can be used for a variety of different use cases, as discussed in more detail in the following "VPR Use Cases" section and shown in Table 1. Here, each use case typically requires a certain combination of the *input* (single-session/multisession VPR) and *data processing* (online/batch VPR) VPR categories. The choice of the *output* category (single-best-match/multimatch VPR) also depends on the use case and, in particular, on the algorithm that is used after VPR. For example, for pure (re-)localization [21], one may need only a single best match so that the choice of the output category depends mainly on the use case, while for graph-based SLAM, the required output category also depends on the post-processing after VPR, as explained in the following. In graph-based SLAM [10], each node encodes the pose of an image in Q. The corresponding edges of connected nodes represent the transformation between them. An edge can be established either between temporally consecutive nodes (using the odometry) or between nodes that were identified as loop closures by VPR. Here, single-best-match VPR could be used to match and fuse two nodes that correspond to the same place to represent each place always by only one node. Alternatively, multimatch VPR could be used to create multiple edges between all existing nodes of the same place. This is particularly helpful if we cannot guarantee that there is a single node for each place in the graph or if we perform a batch optimization of the poses using a robust optimization approach that can benefit from the additional information provided by multiple matches while handling potential outlier matchings.

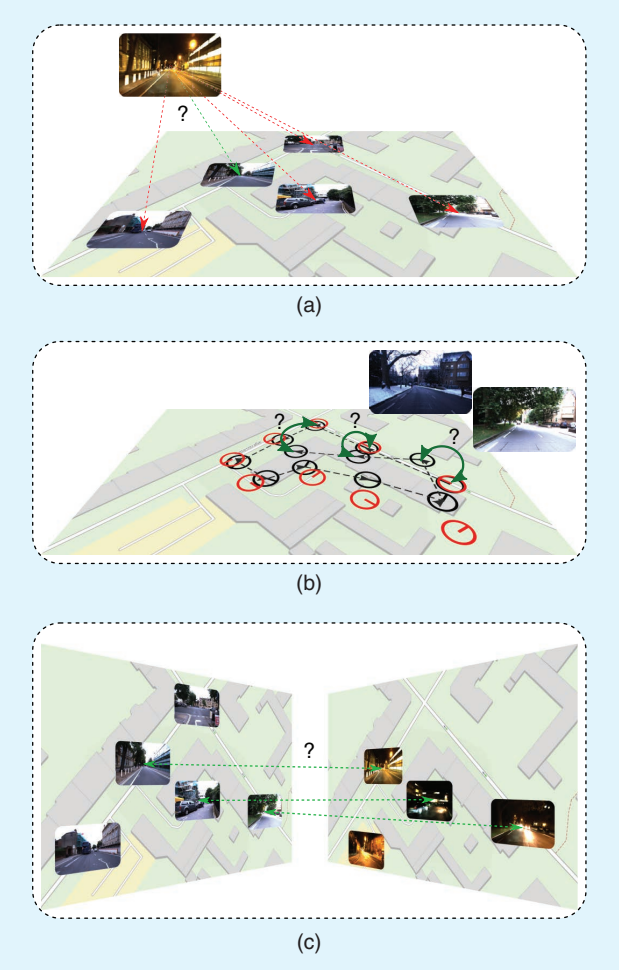


FIGURE 2. An overview of VPR use cases. (a) VPR is used to select a small set of candidate images from the database for visual (re-) localization, and the images are processed with computationally expensive 6-DoF pose estimation methods. (b) VPR can be used to detect loop closures in a SLAM pipeline (green arrows) or to relocalize after mislocalization or if the robot was moved (kidnapped robot). (c) In multisession mapping, VPR can again be used for loop closure detection but this time in a batch manner where all images are known in advance. In multirobot mapping, VPR is used to merge maps by detecting places that have been visited by multiple robots. (a) Visual (re-)localization. (b) Online SLAM and mapping. (c) Multirobot mapping and multisession mapping. (Source: Map data from OpenStreetMap.)

VPR USE CASES

VPR is a key component in a variety of robotic applications, including autonomous driving, agricultural robotics, and robotic parcel delivery as well as in the creation of a metaverse. Some common tasks that VPR is used for include the following:

1) *Candidate selection for 6-degree of freedom (6-DoF) visual localization [21]:* 6-DoF visual localization (also termed *city-scale/natural geolocalization*) involves estimating the 6-DoF (6D) pose (position and orientation) of a camera in a particular environment. *Multisession online VPR with fixed DB* is used to select candidates $\mathbf{I}_i \in DB$ that have the highest similarity to the current query images $\mathbf{I}_j \in Q$ [cf. Figure 2(a)]. These candidates can then be used for a computationally intensive 6D pose estimation using local image descriptors and more complex algorithms, which would be infeasible for the complete *DB* set.
2) *Loop closure detection and relocalization for online SLAM [10]:* Online SLAM is used to estimate the current pose of a camera while creating a map of the environment at the same time. *Single-session online VPR* is used for loop closure detection (i.e., the recognition of previously visited places), as shown in Figure 2(b), to compensate for accumulated errors in odometry data and create a globally consistent map. It is also used for relocalization in the event of mislocalization or if the camera/robot was moved by hand (known as the *kidnapped robot problem*).
3) *Loop closure detection for mapping [10]:* Mapping (also full SLAM or offline SLAM) involves estimating the entire path at once to generate a map. This allows for the use of *single-session batch VPR* for loop closure detection [cf. Figure 2(b)], which is based on slower but more robust algorithms that run on powerful hardware.
4) *Loop closure detection for multisession mapping [22]:* Multisession mapping combines the results of multiple SLAM missions performed repeatedly over time in the same environment. *Multisession batch VPR* is used to find shared places between the individual maps of all missions for map merging [cf. Figure 2(c)]. Alternatively, *multisession online VPR with a given DB* can be used to

detect previously mapped areas (potentially for loop closure) and include unseen areas of the map in real time.

5) *Detection of shared places for multirobot mapping [23]:* Multirobot mapping (also termed *decentralized SLAM*) involves the distributed mapping of an environment using multiple robots. Here, *multisession online VPR with a growing DB* is used to find shared places between the individual maps of each robot for subsequent map merging, as shown in Figure 2(c).

In summary, this section provided an overview of the different problem categories and corresponding subtypes of VPR and discussed common use cases where VPR is applied.

A GENERIC PIPELINE FOR VPR

This section outlines a generic pipeline for VPR. The steps involved in this pipeline are shown in Figure 1. The inputs to the pipeline are two sets of images: *DB* and *Q* (these may be the same for single-session VPR, as explained in the "VPR Problem Categories and Use Cases" section). The pipeline produces matching decisions, meaning that for each query image $\mathbf{I}_j \in Q$, one or more database images $\mathbf{I}_i \in DB$ can be associated. The pipeline includes these intermediate steps and components: 1) computing image-wise descriptors, 2) pairwise comparing of descriptors, 3) creating a descriptor similarity matrix $\mathbf{S}$, and 4) making matching decisions. In the following sections, we will discuss each of these elements in more detail. Extensions to this generic pipeline that can be used to improve performance and robustness against various challenges are presented in the "Challenges and Common Ways of Addressing Them" section.

INPUTS: THE DATABASE AND QUERY IMAGE SETS

To recap, two sets of images serve as the input in a VPR pipeline: the database set *DB* and a set of current images in the query set *Q*. The *DB* set, which is also called the *reference set*, represents a map of known places and is often recorded under ideal conditions (e.g., sunny) or by a different platform than *Q* (e.g., a second robot). The query set *Q*, on the other hand, is the "live view" recorded by a different platform than the *DB* or after the *DB*—potentially days, months, or even years later. Both sets will have a geographical overlap and share some or all seen places.

There are different VPR problem categories: using just a query set *Q* (single-session VPR) or using both the *DB* and *Q* sets (multisession VPR). Also, the image sets can either be specified before processing (batch VPR) or grow during an online run (online VPR).

"Code Snippet 1" provides example code for loading a dataset with both image sets *Q* and *DB* as well as the ground truth matrices $\mathbf{GT}$ and $\mathbf{GT}^{\text{soft}}$. Briefly, $\mathbf{GT}$ is a logical matrix that indicates whether corresponding images show the same or different places, while $\mathbf{GT}^{\text{soft}}$ is a dilated version of $\mathbf{GT}$ that accounts for image pairs with small visual overlap, avoiding penalization for matches in such cases. We detail these matrices in the "The Ground Truth" section.

IMAGE-WISE DESCRIPTOR COMPUTATION

This section describes the process of computing image descriptors, which are abstractions of images that extract features from raw pixels to be more robust against changes in appearance and viewpoint (step 1 in Figure 1; see also "Code Snippet 2"). The tutorial covers two primary types of image descriptors.

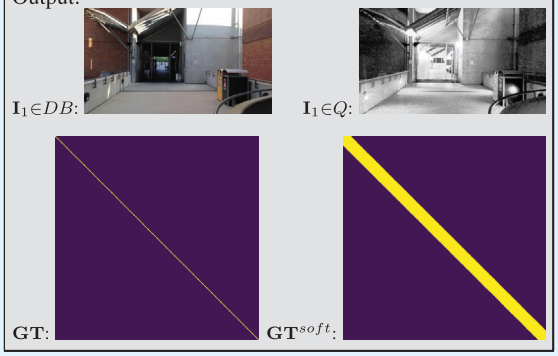
1) *Holistic descriptors:* Holistic descriptors (also called *global descriptors*) represent an image $\mathbf{I}_i \in DB, Q$ with a single vector $\mathbf{d}_i \in \mathbb{R}^d$ (cf. "Code Snippet 2"). This allows for efficient pairwise descriptor comparisons with low runtimes. Note that when an exhaustive k-nearest neighbor search is used to obtain the nearest neighbors for a candidate selection of similar database descriptors, the execution time scales linearly with both the descriptor dimension and the number of images contained in the database.
2) *Local descriptors:* Local descriptors encode an image $\mathbf{I}_i$ with a set $D_i = \{\mathbf{d}_k \,|\, k = 1, \ldots, K\}$ of vectors $\mathbf{d}_k \in \mathbb{R}^d$ at K regions of interest. They often provide better performance than holistic descriptors but require computationally expensive methods for local feature matching, like a left-right check (also termed *mutual matching*), a homography estimation, a computation of the epipolar constraint, or deep learning matching techniques. Therefore, local

CODE SNIPPET 1

In this example, the inputs to a visual place recognition (VPR) algorithm are two disjoint image sets: the database DB and query Q. We load the GardensPoint Walking dataset [24] and ground truth information about correspondences. This ground truth serves only for later evaluation and will neither be available nor required when deploying the algorithm.

```
# load dataset GardensPoint Walking [24] with two
# image sets DB and Q and ground truth
from datasets.load_dataset \
    import GardensPointDataset
dataset = GardensPointDataset()
DB, Q, GT, GT^soft = dataset.load()
```

Output:



CODE SNIPPET 2

The main sources of information about image correspondences are image descriptors. Since holistic image descriptors allow for efficient pairwise descriptor comparisons, we compute a holistic HDC-DELF [28] descriptor for each image (step 1 in Figure 1).

```
# compute holistic HDC-DELF descriptors [28]
from feature_extraction. \
    feature_extractor_holistic import HDCDELF
feature_extractor = HDCDELF()
D^DB = feature_extractor.compute_features(DB)
D^Q = feature_extractor.compute_features(Q)
```

descriptors are typically used in a hierarchical pipeline, where first the holistic descriptors are used to retrieve the top-K matches, which are then reranked using local descriptor matching.

The abstraction of the VPR pipeline in terms of holistic and local descriptors serves as the foundation for many localization, mapping, and SLAM solutions. Alternative approaches include place classification [25], regional descriptors [26], and incremental bags of binary words [27]. Furthermore, in the "Challenges and Common Ways of Addressing Them" section, we list the common shortcomings of this VPR pipeline and ways to address them.

To convert a set of local descriptors from a single image into a holistic descriptor, one can use *local feature aggregation* methods like bag of visual words (BoVW), Vector of Locally Aggregated Descriptors (VLAD), or hyperdimensional computing (HDC) [28]. In a hierarchical pipeline, this allows a local descriptor to be used for both candidate selection (after aggregation) and verification (with the raw local descriptors).

As the descriptor computation is one of the first steps in a pipeline for VPR, it has a significant impact on the performance of subsequent steps and the overall performance of the VPR system. The algorithm used to obtain the descriptors determines how well the descriptors are suited for a specific environment, the degree of viewpoint change, or the type of environmental condition change. For example, convolutional neural network (CNN)-based holistic descriptors like AlexNet-conv3 [15] perform well in situations with low or negligible viewpoint changes but perform poorly with large viewpoint changes. On the other hand, VLAD-based descriptors like NetVLAD [29] tend to perform better in settings with large viewpoint changes.

Additionally, the specific training data of deep learned descriptors affect the performance in different environments. For example, some descriptors may perform better in urban environments, while others may be more effective in natural environments or in specific geographic regions such as Western cities [30].

DESCRIPTOR SIMILARITY BETWEEN TWO IMAGES

To compare the image descriptors of two images, a measure of similarity or distance must be calculated (see step 2 of Figure 1 and "Code Snippet 3"). This process compares the descriptors $\mathbf{d}_i$ and $\mathbf{d}_j$ (holistic) or D_i and D_j (local) of images i and j. Note that similarity s_{ij} and distance dist_{ij} can be related through inversely proportional functions such as

$$s_{ij} = -\text{dist}_{ij} \tag{1}$$

or the reciprocal

$$s_{ij} = \frac{1}{\text{dist}_{ij}}. \tag{2}$$

Holistic descriptors can be compared more efficiently than local descriptors as they require only simple and computationally efficient metrics like the cosine similarity

$$s_{ij} = \frac{\mathbf{d}_i^T \cdot \mathbf{d}_j}{\|\mathbf{d}_i\| \cdot \|\mathbf{d}_j\|} \tag{3}$$

or the negative Euclidean distance

CODE SNIPPET 3

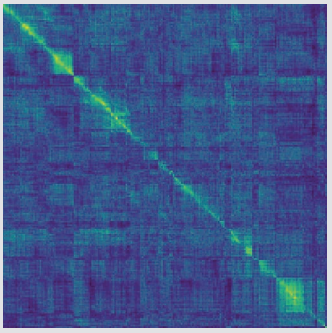
To compare database and query descriptors to obtain the descriptor similarities $\mathbf{S}$ (steps 2 and 3 in Figure 1), we use their cosine similarity (computed by the inner product of the normalized descriptor vectors). Although we might not want to compute the full similarity matrix $\mathbf{S}$ of all possible pairs in a large-scale practical application, it can be useful for visual inspection purposes.

```
# compare all descriptors using cosine similarity
# normalize descriptors
D^DB = D^DB /
      np.linalg.norm(D^DB, axis=1, keepdims=True)

D^Q = D^Q /
      np.linalg.norm(D^Q, axis=1, keepdims=True)
# dot product between descriptors using matrix
# multiplication
S = np.matmul(D^DB, D^Q.transpose())
```

Output:

S:



$$s_{ij} = -\left\| \mathbf{d}_i - \mathbf{d}_j \right\|. \quad (4)$$

In contrast, comparing local descriptors requires more complex and computationally expensive algorithmic approaches, as previously mentioned in the "Image-Wise Descriptor Computation" section.

THE PAIRWISE SIMILARITY MATRIX S

The pairwise descriptor similarity matrix **S** is a key component of VPR. As shown in step 3 of Figure 1, **S** contains all calculated similarities s_{ij} between the descriptors of images in the database and query sets. In single-session VPR, **S** has dimensions $|Q| \times |Q|$, while in multisession VPR, **S** has dimensions $|DB| \times |Q|$. Depending on the approach used, **S** may be dense (if all descriptors are compared) or sparse (if only a subset of descriptors is compared using the approximate nearest neighbor search or sequence-based comparison strategies).

The overall appearance of **S** is influenced by the camera's trajectories during the acquisition of *Q* and *DB*, as illustrated in Figure 3. The pattern of high similarities within **S** can have a significant impact on the performance of the VPR pipeline and may enable or hinder the use of certain algorithmic steps for performance improvements. The following relations between camera trajectories and the appearance of **S** can be observed (cf. Figure 3 for corresponding examples in a map):

1) *General:* If the images in *DB* and *Q* are recorded at arbitrary positions without a specific order, there are no discernible patterns in **S**. This is typical for general visual localization and global geolocalization.
2) *Sequence:* If the images in *DB* and *Q* are recorded along trajectories as spatiotemporal sequences (i.e., consecutive images are also neighbors in the world), continuous lines of high similarities may be observed in **S**. This setup is typical for many robotic tasks, including online SLAM, mapping, and multirobot/multisession mapping (see the "VPR Problem Categories and Use Cases" section). In this setup, sequence-based methods can be used for performance improvements (cf. the "Challenges and Common Ways of Addressing Them" section). The camera's trajectories can affect **S** in the following ways:
   a) *Speed:* If the camera moves at the same speed in the same locations in *DB* and *Q*, lines of high similarities with 45° slope will be observed. Otherwise, the slope will vary.

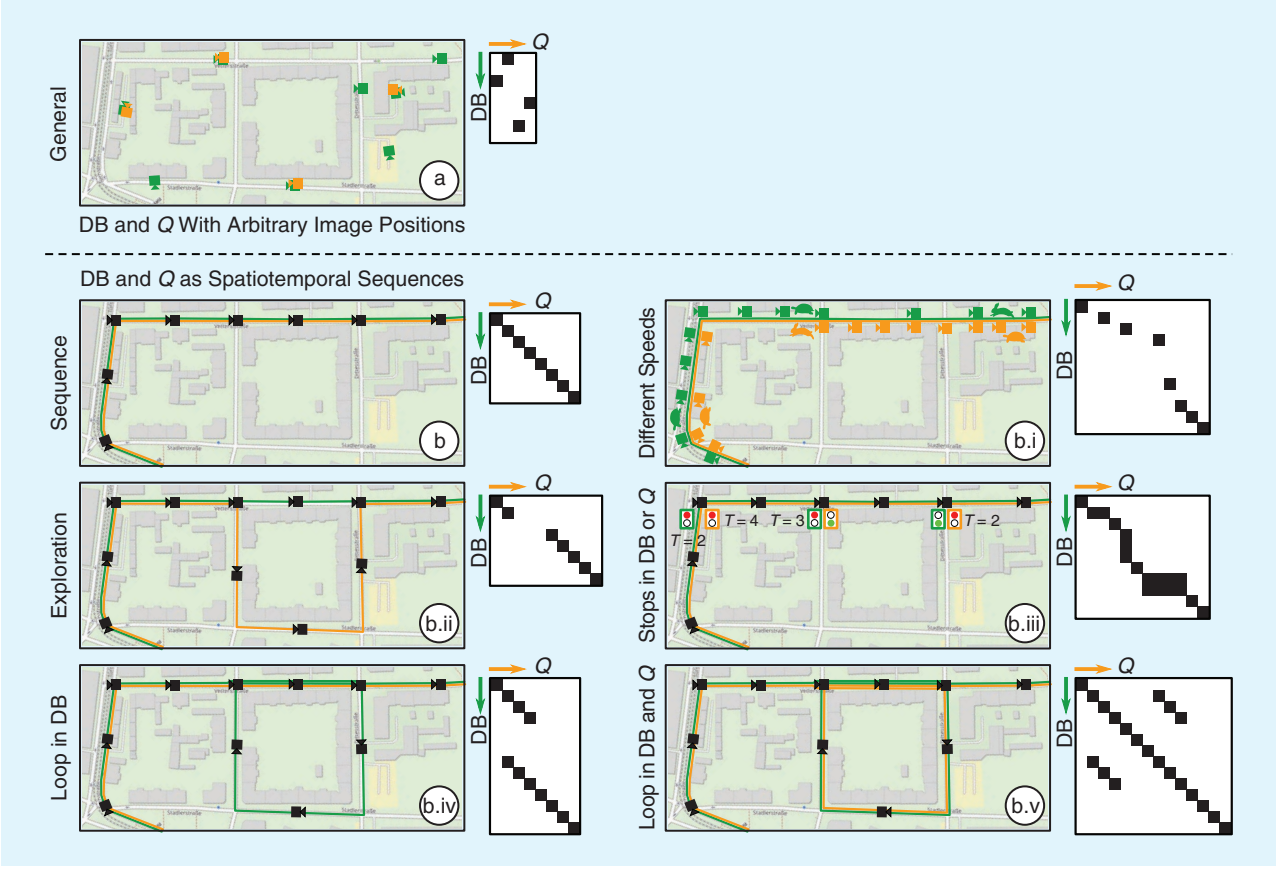


FIGURE 3. The relation between the similarity matrix **S** and the trajectory during the database and query run. The green and orange cameras depict images in *DB* and *Q*, respectively. The green and orange lines indicate that images were recorded as video along a trajectory (also called the *spatiotemporal sequence*). In (b.i), a rabbit or turtle indicates fast or slow speeds when traversing the route, and similarly, in (b.iii), traffic lights indicate stops in *Q* ($T = 2$), *DB* ($T = 3$), or both ($T = \{2,4\}$) for *T* time steps.

b) *Exploration:* If a place shown in a query image Q is not present in *DB*, the line of high similarities will be discontinuous.
c) *Stops:* If the camera stops temporarily (zero velocity) during either the database run or the query run, it will result in multiple consecutive matches in the other set.
   - *Stops in DB:* Stops in the database run will result in a vertical line (within the same column) of high similarities in $\mathbf{S}$.
   - *Stops in Q:* Stops in the query run will result in a horizontal line (within the same row) of high similarities in $\mathbf{S}$.
   - *Stops in DB and Q:* If the camera stops in both the database run and query run at the same place, a block of high similarities will be observed in $\mathbf{S}$.
d) *Loops in DB:* Loops in *DB* can result in multiple matching database images for a single query image in Q. Unlike stops, the multiple matching images due to a loop are not consecutive in their image set.
e) *Loops in DB and Q:* Loops in *DB* and Q can result in additional matching query images for a single database image in *DB*. This results in a more complex structure of high similarities in $\mathbf{S}$.

CODE SNIPPET 4

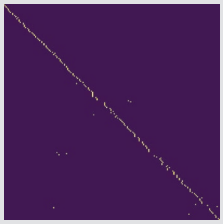
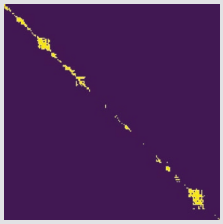
The output of a VPR pipeline is typically a set of discrete matchings, i.e., pairs of query and database images. To obtain matchings for a query image from the similarity matrix (step 4 in Figure 1), we can either find the single best matching database image ($\mathbf{M}_1$) or try to find all images in the database that show the same place as the query image ($\mathbf{M}_2$).

```
# match images based on S
from matching import matching

# best matching per query in S for
# single-best-match VPR
M1 = matching.best_match_per_query(S)

# find matches with S>=thresh using an auto-tuned
# threshold for multi-match VPR
M2 = matching.thresholding(S, thresh='auto')
```

Output:

$\mathbf{M}_1$:  $\mathbf{M}_2$: 

Examples for correct and wrong matches from $\mathbf{M}_2$:

OUTPUT: MATCHING DECISIONS

The output of a VPR system is a set of matching decisions $m_{ij} \in \mathbf{M}$ (step 4 in Figure 1 and "Code Snippet 4") with $\mathbf{M} \in \mathbb{B}^{|Q|\times|Q|}$ (single-session VPR) or $\mathbf{M} \in \mathbb{B}^{|DB|\times|Q|}$ (multisession VPR) that indicate whether the ith database/query image and the jth query image show the same place ($m_{ij} = \text{true}$) or different places ($m_{ij} = \text{false}$). Existing techniques for matching range from choosing the best match per query or a simple thresholding of the pairwise descriptor similarities $s_{ij} \in \mathbf{S}$ to a geometric verification with a comparison of the spatial (using e.g., the epipolar constraint) or semantic constellation of the scene. For example in "Code Snippet 4," M_1 is computed by selecting the best matching database image per query image, i.e., the maximum similarity s_{ij} per column in $\mathbf{S} \in \mathbb{R}^{|DB|\times|Q|}$ (single-best-match VPR). Another example is the computation of M_2 in "Code Snippet 4," where a similarity threshold θ is applied to $\mathbf{S}$; if $s_{ij} \geq \theta$, the ith and jth images are assumed to show the same place (multimatch VPR). The next section is concerned with the performance evaluation of these outputs.

EVALUATION OF THE PERFORMANCE

This section is concerned with the evaluation of the matching decisions $\mathbf{M}$ or the pairwise similarities $\mathbf{S}$, which allows the comparison of different VPR methods. This requires datasets, corresponding ground truth, and performance metrics. In the following, we outline these components for evaluation and discuss their properties and potential pitfalls.

DATASETS

For VPR, a dataset is composed of one or multiple image sets that have to be matched to find shared places. For example, the popular Nordland dataset [31] provides four image sets, one for each season, i.e., spring, summer, fall, and winter. These can be arbitrarily combined for VPR, but a typical choice might be to use *summer* as *DB* and *spring*, *fall*, or *winter* as Q.

Existing datasets vary in the type of environment as well as in the type and degree of appearance and viewpoint change. The *type of environment* includes indoor environments, urban environments, suburban environments, and natural environments like countryside, forests, or lakes. *Appearance changes* occur due to dynamic objects like pedestrians; time of day with lighting changes and moving shadows or day versus night; weather that is sunny, cloudy, overcast, rainy, foggy, or snowy; seasons like spring, summer, fall, and winter or dry and wet season; elapsed time with roadworks; construction sites or new and demolished

buildings up to modern versus historical imagery; or catastrophic scenarios, e.g., after an earthquake. *Viewpoint changes* between images of the same place range from nearly pixel-aligned to the left-to-right side of the walkway, left-to-right side of the street, panoramic-aligned images to a single image, panoramic-aligned images to panoramic-aligned images, bikeway to street, aerial to ground, or inside to outdoor. For a comprehensive overview of existing datasets, please refer to [1] and [3].

THE GROUND TRUTH

Ground truth data tell us which image pairs in a dataset show the same places and which show different places. These data are necessary for evaluating the results of a place recognition method. Either the ground truth is directly given as a set of tuples indicating which images in the database *DB* and the query set *Q* belong to the same places, or it is provided via GNSS coordinates or poses using their maximum allowed distances. Alternatively, some datasets are sampled so that images with the same index in each image set show the same place.

DEFINITION OF THE GROUND TRUTH

To evaluate a VPR result, the definition of a logical ground truth matrix $\mathbf{GT}$ is required. This matrix has the same dimensions as $\mathbf{S}$ and $\mathbf{M}$, i.e., $\mathbf{GT} \in \mathbb{B}^{|Q|\times|Q|}$ or $\mathbf{GT} \in \mathbb{B}^{|DB|\times|Q|}$. The elements $gt_{ij} \in \mathbf{GT}$ define whether the ith image in Q or *DB* and the jth image in Q show the same place ($gt_{ij} = \text{true}$) or different places ($gt_{ij} = \text{false}$). Their values are set using the ground truth matches from the dataset.

An additional way of evaluating VPR performance that is used by some researchers is the soft ground truth matrix $\mathbf{GT}^{\text{soft}}$. The soft ground truth matrix addresses the problem that we do not expect a VPR method to match images with a very small visual overlap, i.e., $gt_{ij} = \text{false}$, as illustrated in Figure 4. However, if a method indeed matches these images with a small overlap, we avoid penalization by setting $gt_{ij}^{\text{soft}} = \text{true}$. Image pairs without any visual overlap are also labeled $gt_{ij}^{\text{soft}} = \text{false}$. Therefore, $\mathbf{GT}^{\text{soft}}$ is a dilated version of $\mathbf{GT}$, i.e., it contains all true values contained in $\mathbf{GT}$ as well as additional true values for image pairs with small visual overlap. Image pairs must be matched if

$$\mathbf{GT} = \text{true}\,. \tag{5}$$

Image pairs can but do not need to be necessarily matched if

$$\neg\mathbf{GT} \wedge \mathbf{GT}^{\text{soft}} = \text{true}\,. \tag{6}$$

Note that we use $\neg$ to denote the logical negation operator. These are usually ignored during evaluation. Image pairs must not be matched if

$$\mathbf{GT}^{\text{soft}} = \text{false}. \tag{7}$$

How $\mathbf{GT}$ and $\mathbf{GT}^{\text{soft}}$ are actually used for evaluation is presented in the following.

METRICS

This section presents established metrics to evaluate a VPR method, including precision and recall, the precision-recall curve, the area under the precision-recall curve (AUPRC), maximum recall at 100% precision, and recall@K [32]. All metrics are implemented in the associated code repository (see the "Introduction" section). These metrics are based on

- the pairwise descriptor similarities $s_{ij} \in \mathbf{S}$ (cf. the "The Pairwise Similarity Matrix $\mathbf{S}$" section) or the image matches $m_{ij} \in \mathbf{M}$ (cf. the "Output: Matching Decisions" section)
- with corresponding ground truth $gt_{ij} \in \mathbf{GT}$ (and $gt_{ij}^{soft} \in \mathbf{GT}^{\text{soft}}$ in case the soft ground truth is used).

For single-best-match VPR, the evaluation considers only the best matching image pair per query with the highest similarity s_{i^*j}

$$i^* = \underset{i}{\operatorname{argmax}}\, s_{ij}\,. \tag{8}$$

PRECISION AND RECALL

Precision (P) and recall (R) are important metrics in the information retrieval domain. In the context of VPR, *precision* (P) represents the ratio of correctly matched images of the same places to the total number of matched images with

$$P = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}}\,. \tag{9}$$

TP represents true positives, and FP represents false positives. *Recall* (R) expresses the ratio of the correctly matched images of the same places to the total number of ground truth positives (GTPs)

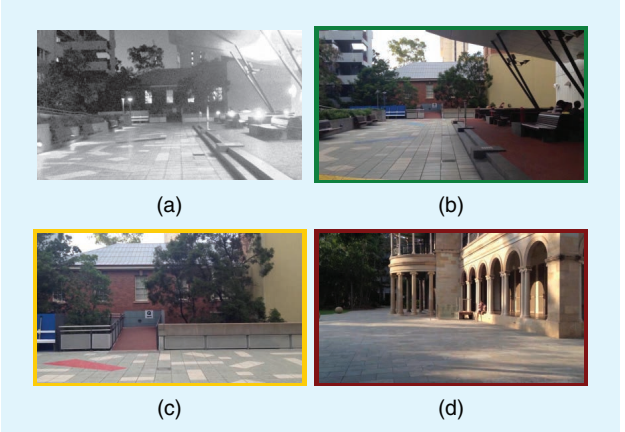


FIGURE 4. (a)–(d) The relation between (a) the visual overlap of a query image and (b)–(d) the reference images, and their corresponding ground truth values gt_{ij} and gt_{ij}^{soft}. Since (c) and (a) have only a small visual overlap, we do not expect a VPR method to match both images and set $gt_{ca} = \text{false}$. However, we also avoid penalization in case the VPR method indeed matches both images by setting $gt_{ca}^{\text{soft}} = \text{true}$. (a) Query image. (b) Same location, high visual overlap, $gt_{ba} = \text{true}$, $gt_{ba}^{\text{soft}} = \text{true}$. (c) Same location, small visual overlap, $gt_{ca} = \text{false}$, $gt_{ca}^{\text{soft}} = \text{true}$. (d) Different location, no visual overlap, $gt_{da} = \text{false}$, $gt_{da}^{\text{soft}} = \text{false}$.

$$R = \frac{\#\mathrm{TP}}{\#\mathrm{GTP}}. \tag{10}$$

In the case of single-best-match VPR, the number of GTPs refers to the total number of query images for which a ground truth match exists, i.e.,

$$\#\mathrm{GTP} = \sum_{\forall j} \begin{cases} 1, & \text{if } \exists i: gt_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

whereas in the case of multimatch VPR, the number of GTPs refers to the number of actually matching image pairs, i.e.,

$$\#\mathrm{GTP} = \sum_{\forall i,j} \begin{cases} 1, & \text{if } gt_{ij} \\ 0, & \text{otherwise} \end{cases}. \tag{12}$$

#TP and #FP are the number of correctly matched and wrongly matched image pairs. More specifically, TPs are actual matching image pairs that were classified as matches

$$\#\mathrm{TP} = \sum_{\forall i,j} \begin{cases} 1, & \text{if } gt_{ij} \wedge m_{ij} \\ 0, & \text{otherwise} \end{cases}. \tag{13}$$

For single-best-match VPR, only i^* from (8) is evaluated in (13) for each query image. The same is true for the following (14).

FPs are nonmatching image pairs that were incorrectly classified as matches

$$\#\mathrm{FP} = \sum_{\forall i,j} \begin{cases} 1, & \text{if } \neg gt_{ij} \wedge m_{ij} \\ 0, & \text{otherwise} \end{cases}. \tag{14}$$

Note that when using the soft ground truth, image pairs with $\neg gt_{ij} \wedge gt_{ij}^{\mathrm{soft}} = \text{true}$ [cf. (6)] are ignored during the computation of #TP and #FP. While *false negatives* (FNs) are indirectly involved in the calculation of recall R, *true negatives* (TNs) are usually not evaluated due to the typically imbalanced classification problem of VPR with $\#\mathrm{TN} \gg \#\mathrm{TP}, \#\mathrm{FP}, \#\mathrm{FN}$.

PRECISION-RECALL CURVE

Precision-recall curves can be used to avoid actual matching decisions, which are often made after VPR using a computationally expensive verification algorithm. The idea is to make matching decisions with $\mathbf{M} = \mathbf{S} \geq \theta_k$ over a range of thresholds $\boldsymbol{\theta} = \{\min(\mathbf{S}), \ldots, \max(\mathbf{S})\}$. For instance, the number of TPs $\#\mathrm{TP}_k$ for one specific $\theta_k \in \boldsymbol{\theta}$ is then computed with

$$\#\mathrm{TP}_k = \sum_{\forall i,j} \begin{cases} 1, & \text{if } gt_{ij} \wedge (s_{ij} \geq \theta_k) \\ 0, & \text{otherwise} \end{cases}. \tag{15}$$

Following (9) and (10), this leads to two vectors of precision and recall values, $P(\theta_k)$ and $R(\theta_k)$, which in combination formulate the *precision-recall curve*. The full pipeline for the computation of the precision-recall curve is depicted in Figure 5, and the code is provided in "Code Snippet 5."

AUPRC

The AUPRC (also termed *average precision*) can be used to compress a precision-recall curve into a single number, as shown in "Code Snippet 6." In Figure 5, the AUPRC is visualized as the green area under the precision-recall curve.

MAXIMUM RECALL AT 100% PRECISION

The maximum recall at 100% precision (short *R@100P*) represents the maximum recall where $P = 1$ (100%), i.e., the maximum recall without FPs [cf. (14)]. In the past, this metric was important to evaluate VPR methods for loop closure detection in SLAM. Keeping the precision at $P = 1$ avoids the wrong loop closures and, consequently, mapping errors. However, since the advent of robust graph optimization techniques for SLAM [33], the avoidance of wrong loop closures has become less relevant. With robust graph optimization, it is more important to find enough correct loop closures (*TP*) than to avoid wrong loop closures (FP). Therefore, using multimatch VPR to identify all loop closures should be preferred over tuning the R@100P for such applications.

If the precision never reaches $P = 1$, the maximum recall at 100% precision is undefined. Therefore, the maximum recall at 99% or 95% precision has been used alternatively.

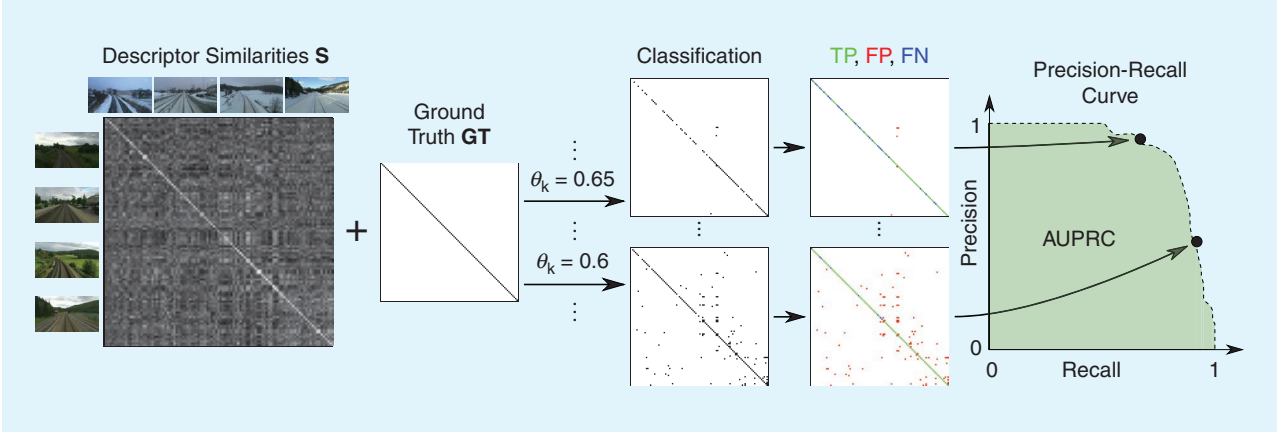


FIGURE 5. The evaluation pipeline for multimatch VPR, including the precision-recall curve and the AUPRC. Given the similarity matrix **S** and ground truth **GT** and $\mathbf{GT}^{\mathrm{soft}}$ (cf. the "The Ground Truth" section), a range of thresholds $\theta_k \in \{\min(\mathbf{S}), \ldots, \max(\mathbf{S})\}$ is applied with $\mathbf{S} \geq \theta_k$ to obtain binary matching decisions $m_{ij} \in \mathbf{M}_k$ for each θ_k. In combination with the ground truth, these can be labeled as either TPs, FPs, FNs, or TNs and then converted into a precision-recall curve and the AUPRC.

RECALL@K

The recall@K (also termed *top-K success rate*) is an often used metric for the evaluation of image classifiers. For place recognition, it is defined as follows. For each query image, given the K database images with the K highest similarities s_{ij}, the recall@K measures the rate of query images with at least one actually matching database image. That means this metric requires at least one matching image in the database for each query image, which corresponds to a typical localization scenario without exploration. For mapping with newly visited places, the metric is not defined. In such a scenario, an implementation of recall@K could simply ignore all query images without a matching database image—however, this workaround would not evaluate the (in)ability of a method to handle exploration during the query run, i.e., new places that are not part of the database set.

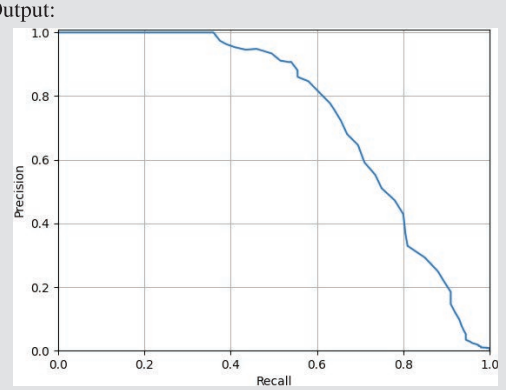
The recall@K is particularly suited for visual localization tasks, where the K most similar database or query images are retrieved for a subsequent geometric verification. Note that for VPR, in the context of localization without exploration (i.e., all query images have at least one matching reference image), the recall@1 and the precision at 100% recall are identical.

CODE SNIPPET 5

To evaluate the quality of a similarity matrix $\mathbf{S}$, we can apply a series of decreasing thresholds $\boldsymbol{\theta}$ to match more and more image pairs. Combined with ground truth information, each threshold leads to a different set of true positives (TPs), false positives (FPs), true negatives (TNs), and false negatives (FNs), which then provides one point on the precision-recall curve. In this example, we create the precision-recall curve for multi-match VPR.

```
# precision-recall curve performance evaluation
# for multi-match VPR
from evaluation.metrics import createPR
P, R = createPR(S, GT, GT^soft, matching='multi')
```

Output:



CODE SNIPPET 6

Finally, to summarize the place recognition quality in a single number, we can use the area under the precision-recall curve (AUPRC).

```
# evaluate the performance using area under curve
import numpy as np
AUPRC = np.trapz(P, R)
```

Output:
AUPRC: 0.742

MEAN, BEST-CASE, AND WORST-CASE PERFORMANCE

To get a comprehensive understanding of how well a VPR method performs in different environments, with different types of appearance and viewpoint changes, it is best practice to evaluate it using multiple datasets. The aforementioned metrics measure the performance on each single dataset. One can get a more condensed view of the overall performance by considering the mean, best-case, and worst-case performance. The mean performance allows for a quick comparison with other evaluated methods. The best-case performance shows the maximum achievable performance and reveals the potential strengths of an approach; if the best-case performance of a method is higher than that of the compared methods, this method is well suited for the conditions under which the best-case performance was achieved. The worst-case performance reveals the weaknesses of a method and its sensitivity to certain conditions or trajectories (cf. Figure 3). For example, if the worst-case performance of a method is lower than the worst-case performance of the compared methods, it indicates that this method is less robust and struggles with the specific property of at least one of the evaluated datasets.

We would like to note that there are various other metrics for evaluating VPR methods, including those that take computational time into account. We refer interested readers to [32] for a comprehensive overview, which also includes examples of performance evaluations for the same algorithm across multiple metrics.

CHALLENGES AND COMMON WAYS OF ADDRESSING THEM

The previous sections introduced a generic pipeline for VPR and how to evaluate such a pipeline. In this section, we go beyond this basic pipeline and list typical challenges that researchers face in the field of VPR and the ways that prior work has addressed them.

SCALABILITY

A major challenge in VPR is how to scale up the system to handle large numbers of images in the database or query set. As discussed in the "Image-Wise Descriptor Computation" section, holistic image descriptors allow for fast retrieval. To reduce the computational effort for descriptor comparison,

dimensionality reduction techniques like random projections or principal component analysis have been proposed.

However, the computation time for recognizing places is typically still proportional to the number of images in the database *DB* or query set *Q*. To further improve efficiency, an approximate nearest neighbor search (e.g., a combination of KD-tree and product quantization as with DELF [34]) can be employed instead of a linear search of all database descriptors, which leads to a sublinear time complexity. Additionally, incorporating coarse position data from weak GPS signals can increase efficiency as it reduces the search space [35].

Finally, to compensate for the reduced accuracy of holistic descriptors, hierarchical place recognition can be employed. This approach reranks the top-*K* retrieved matches from holistic descriptors through geometric verification with local image descriptors [7].

APPEARANCE VARIATIONS

When a robot revisits a place, its current image observation often experiences significant variations in appearance (as discussed in the "Datasets" section), which can negatively affect the performance. To reduce the discrepancy between the query observation and the observation stored in the database, techniques such as illumination invariant images [36], shadow removal [37], appearance change prediction [38], linear regression [39], and deep learning-based methods using generative adversarial networks (GANs) [40] can be used to convert all images into a reference condition [19]. Such techniques require that the correspondence between each image and its actual condition is provided by human supervision or a condition classifier (e.g., database: summer, query: winter).

To avoid such condition-specific approaches that are trained or designed only for specific conditions (e.g., night-to-day GAN: night and day; shadow removal: different times of day), a condition-wise descriptor standardization can be used to significantly improve performance over a wide range of conditions [19]. This standardization normalizes each dimension of the descriptors from one condition to zero mean and unit standard deviation (e.g., once for the database in summer, once for the query set in winter). Furthermore, if appearance variations occur not only *across* the query and database traverses (e.g., database: sunny; query: rainy) but also *within* a traverse (e.g., database: sunny→cloudy→overcast→rainy; query: sunny), descriptors can be clustered and then standardized per cluster. Besides addressing individual appearance challenges as mentioned previously, a common trend in recent research has been to train deep architectures on large-scale diverse datasets [30] to achieve global [41] and local [42] descriptors that are robust to appearance variations. Alternatively, one can combine the strengths of multiple descriptors by simply concatenating them (which sums up their dimensionalities) or combining them using techniques such as hyperdimensional computing (which limits the dimensionality) [28].

VIEWPOINT VARIATIONS

A robot may revisit a place from a different viewpoint. For drones, this change could be due to a varying 6-DoF pose, and for an on-road vehicle, it could be due to changes in lanes and direction of travel. In addition to recognizing a local feature or region from different viewpoints, one also needs to deal with the often limited visual overlap between an image pair captured from different viewpoints. The problem of viewpoint variations becomes even more challenging when simultaneously affected by appearance variations that widen the scope for perceptual aliasing (the problem of distinct places looking very similar, as discussed in the "Introduction" section and detailed in, e.g., [2]). A popular solution to deal with viewpoint variations is to learn holistic descriptors by aggregating local features in a permutation-invariant manner, that is, independent of their pixel locations, as in NetVLAD [29].

IMPROVING PERFORMANCE

In addition to the approaches mentioned earlier, there are several ways to improve VPR performance by using task-specific knowledge. *Sequence-based methods* leverage sequences in the database and query set, leading to continuous lines of high similarities in the similarity matrix $\mathbf{S}$ (cf. the "The Pairwise Similarity Matrix $\mathbf{S}$" section and Figure 3). We can divide these methods into two categories. *Similarity-based sequence methods* use the similarities $s_{ij} \in \mathbf{S}$ to find linear segments of high similarities, e.g., SeqSLAM [13], or continuous lines of high similarities with potentially varying slopes, e.g., based on a flow network. One can also use available odometry information to find sequences with varying slopes. On the other hand, a sequence of holistic image descriptors can be combined into a single vector. A *sequence descriptor* defined for a place thus accounts for the visual information observed in the preceding image frames [43], [44]. These sequence descriptors can be compared between the database and the query sets to obtain place match hypotheses.

Besides leveraging descriptor similarities $\mathbf{S}$ *between* the database and query sets, *intra-database and intra-query similarities* $\mathbf{S}^{DB}$ *and* $\mathbf{S}^{Q}$, i.e., descriptor similarities *within* the database and query sets, can be used to improve performance. For example, in [16], the intra-set similarities $\mathbf{S}^{DB}$ and $\mathbf{S}^{Q}$ are used in combination with $\mathbf{S}$ and sequence information to formulate a factor graph that can be optimized to refine the similarities in $\mathbf{S}$. In this graph, the intra-set similarities are used to connect images within the database or query sets that are likely to show the same or different places due to a high or low intra-set similarity. For example, let us suppose that the lth query image has high similarities s_{il} and s_{jl} to the ith and jth database images. Let us further suppose that the similarity s_{kl} to the kth database image is low, although the ith, jth, and kth database images have high intra-database similarities s_{ij}^{DB}, s_{ik}^{DB}, and s_{jk}^{DB}. The graph optimization then detects that the similarity s_{kl} between the kth database image and the lth query image is also likely to be high.

Methods such as experience maps [45] and co-occurrence maps [46] can be used in cases where the robot *frequently revisits the same places*. These "memory-based" methods continually observe each place and create a descriptor every time the appearance changes. During a comparison of a new query descriptor with this aggregated "map," only one descriptor of a similar condition needs to be matched to recognize the place, reducing the need for condition-invariant descriptors. Several approaches go beyond these memory-based techniques by modeling spatiotemporal dynamics to forecast feature persistence [47], expected outdoor conditions [48], or map occupancy [49].

In the case of robot localization with known places (i.e., each visited query place is guaranteed to be in the database and no exploration beyond this mapped area is performed), VPR can benefit from *place-specific classifiers*, which can improve accuracy with reduced map storage or retrieval time [50]. A similar approach is to train a deep learning-based *place classifier* that directly outputs a place label for a given image or to create environment-specific descriptors. Another direction is to exploit known place types for place type matching to limit the number of potential matches between the database and query set [15]. For example, instead of searching through all database images, if the query image was taken in a forest, such semantic categorization constrains the database images to only those that were also taken in a forest.

CONCLUSION

VPR is a well-established problem that has found widespread interest and use in both computer vision and robotics. In this tutorial, we have described the VPR task, including its various problem categories and subtypes, their typical use cases, and how it is typically implemented and evaluated. Additionally, we discussed a number of methods that can be used to address common challenges in VPR.

There are a number of open challenges, such as system integration, enriched reference maps, view synthesis, and the design of a "one-fits-all" solution, that still need to be tackled by the community. While we do not discuss these challenges in this tutorial, we refer the interested reader to [1] and [2].

ACKNOWLEDGEMENT

Stefan Schubert acknowledges support from the German Federal Ministry for Economic Affairs and Climate Action. Sourav Garg, Michael Milford, and Tobias Fischer acknowledge support by the QUT Centre for Robotics, funding from the ARC Laureate Fellowship FL210100156 to Michael Milford, and a grant from Intel Labs to Tobias Fischer and Michael Milford. The authors would like to thank Dr. Mark Zolotas, Dr. Alejandro Fontan, and Somayeh Hussaini for valuable insights on drafts of the article.

AUTHORS

Stefan Schubert, Department of Electrical Engineering and Information Technology, Chemnitz University of Technology, 09126 Chemnitz, Germany. E-mail: stefan.schubert@etit.tu-chemnitz.de.

Peer Neubert, Computational Visualistics, University of Koblenz, 56070 Koblenz, Germany. E-mail: neubert@uni-koblenz.de.

Sourav Garg, Australian Institute for Machine Learning, University of Adelaide, Adelaide 5000, Australia. E-mail: sourav.garg@adelaide.edu.au.

Michael Milford, QUT Centre for Robotics, School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane 4000, Australia. E-mail: michael.milford@qut.edu.au.

Tobias Fischer, QUT Centre for Robotics, School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane 4000, Australia. E-mail: tobias.fischer@qut.edu.au.

REFERENCES

[1] S. Schubert and P. Neubert, "What makes visual place recognition easy or hard?" 2021, *arXiv:2106.12671*.

[2] S. Garg, T. Fischer, and M. Milford, "Where is your place, visual place recognition?" in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2021, pp. 4416–4425.

[3] C. Masone and B. Caputo, "A survey on deep visual place recognition," *IEEE Access*, vol. 9, pp. 19,516–19,547, Jan. 2021, doi: 10.1109/ACCESS.2021.3054937.

[4] S. Lowry et al., "Visual place recognition: A survey," *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 1–19, Feb. 2016, doi: 10.1109/TRO.2015.2496823.

[5] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *Int. J. Robot. Res.*, vol. 36, no. 1, pp. 3–15, 2017, doi: 10.1177/0278364916679498.

[6] K. A. Tsintotas, L. Bampis, and A. Gasteratos, "The revisiting problem in simultaneous localization and mapping: A survey on visual loop closure detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 19,929–19,953, Nov. 2022, doi: 10.1109/TITS.2022.3175656.

[7] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, 2008, doi: 10.1177/0278364908090961.

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.

[9] P. Yin et al., "General place recognition survey: Towards the real-world autonomy age," 2022, *arXiv:2209.04497*.

[10] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006, doi: 10.1109/MRA.2006.1638022.

[11] C. Valgren and A. J. Lilienthal, "SIFT, SURF and seasons: Appearance-based long-term localization in outdoor environments," *Robot. Auton. Syst.*, vol. 58, no. 2, pp. 149–156, Feb. 2010, doi: 10.1016/j.robot.2009.09.010.

[12] F. Dayoub and T. Duckett, "An adaptive appearance-based map for long-term topological localization of mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2008, pp. 3364–3369, doi: 10.1109/IROS.2008.4650701.

[13] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2012, pp. 1643–1649, doi: 10.1109/ICRA.2012.6224623.

[14] Z. Chen, O. Lam, A. Jacobson, and M. Milford, "Convolutional neural network-based place recognition," in *Proc. Australas. Conf. Robot. Automat. (ACRA)*, 2014, pp.1–8.

[15] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2015, pp. 4297–4304, doi: 10.1109/IROS.2015.7353986.

[16] S. Schubert, P. Neubert, and P. Protzel, "Fast and memory efficient graph optimization via ICM for visual place recognition," in *Proc. Robot., Sci. Syst. (RSS)*, 2021, pp. 1–11.

[17] D. Scaramuzza and F. Fraundorfer, "Visual odometry [Tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011, doi: 10.1109/MRA.2011.943233.

[18] O. Vysotska and C. Stachniss, "Effective visual place recognition using multi-sequence maps," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1730–1736, Apr. 2019, doi: 10.1109/LRA.2019.2897160.

[19] S. Schubert, P. Neubert, and P. Protzel, "Unsupervised learning methods for visual place recognition in discretely and continuously changing environments,"

in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2020, pp. 4372–4378, doi: 10.1109/ICRA40945.2020.9197044.

[20] K. Ho and P. Newman, "Detecting loop closure with scene sequences," *Int. J. Comput. Vision*, vol. 74, no. 3, pp. 261–286, Sep. 2007, doi: 10.1007/s11263-006-0020-1.

[21] T. Sattler et al., "Benchmarking 6DOF outdoor visual localization in changing conditions," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2018, pp. 8601–8610.

[22] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard, "6-DoF multi-session visual SLAM using anchor nodes," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, 2011, pp. 69–76.

[23] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual SLAM," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2018, pp. 2466–2473, doi: 10.1109/ICRA.2018.8461155.

[24] A. Glover, "Day and night, left and right," 2014, doi: 10.5281/zenodo.4590133.

[25] T. Weyand, I. Kostrikov, and J. Philbin, "PlaNet – Photo geolocation with convolutional neural networks," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, 2016, pp. 37–55, doi: 10.1007/978-3-319-46484-8_3.

[26] X. Zhang, L. Wang, and Y. Su, "Visual place recognition: A survey from deep learning perspective," *Pattern Recognit.*, vol. 113, May 2021, Art. no. 107760, doi: 10.1016/j.patcog.2020.107760.

[27] E. Garcia-Fidalgo and A. Ortiz, "iBoW-LCD: An appearance-based loop-closure detection approach using incremental bags of binary words," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3051–3057, Oct. 2018, doi: 10.1109/LRA.2018.2849609.

[28] P. Neubert and S. Schubert, "Hyperdimensional computing as a framework for systematic aggregation of image descriptors," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2021, pp. 16,938–16,947.

[29] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2016, pp. 15,297–15,307.

[30] F. Warburg, S. Hauberg, M. Lopez-Antequera, P. Gargallo, Y. Kuang, and J. Civera, "Mapillary street-level sequences: A dataset for lifelong place recognition," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2020, pp. 2626–2635.

[31] N. Sünderhauf, P. Neubert, and P. Protzel, "Are we there yet? Challenging SeqSLAM on a 3000 km journey across all four seasons," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA) Workshop Long-Term Autonomy*, 2013, pp. 1–3.

[32] M. Zaffar et al., "VPR-Bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change," *Int. J. Comput. Vision*, vol. 129, pp. 2136–2174, Jul. 2021, doi: 10.1007/s11263-021-01469-5.

[33] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, 2012, pp. 1879–1884, doi: 10.1109/IROS.2012.6385590.

[34] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2017, pp. 3456–3465.

[35] O. Vysotska, T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Efficient and effective matching of image sequences under substantial appearance changes exploiting GPS priors," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2015, pp. 2774–2779, doi: 10.1109/ICRA.2015.7139576.

[36] W. Maddern, A. Stewart, C. McManus, B. Upcroft, W. Churchill, and P. Newman, "Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles," in *Proc. Int. Conf. Robot. Automat. (ICRA), Workshop Vis. Place Recognit. Changing Environ.*, 2014, pp. 1–8.

[37] C. McManus, W. Churchill, W. Maddern, A. D. Stewart, and P. Newman, "Shady dealings: Robust, long-term visual localisation using illumination invariance," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2014, pp. 901–906, doi: 10.1109/ICRA.2014.6906961.

[38] P. Neubert, N. Sünderhauf, and P. Protzel, "Superpixel-based appearance change prediction for long-term navigation across seasons," *Robot. Auton. Syst.*, vol. 69, pp. 15–27, Jul. 2015, doi: 10.1016/j.robot.2014.08.005.

[39] S. Lowry and M. J. Milford, "Supervised and unsupervised linear learning techniques for visual place recognition in changing environments," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 600–613, Jun. 2016, doi: 10.1109/TRO.2016.2545711.

[40] H. Porav, W. Maddern, and P. Newman, "Adversarial training for adverse conditions: Robust metric localisation using appearance transfer," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2018, pp. 1011–1018, doi: 10.1109/ICRA.2018.8462894.

[41] G. Berton, C. Masone, and B. Caputo, "Rethinking visual geo-localization for large-scale applications," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2022, pp. 4878–4888.

[42] B. Cao, A. Araujo, and J. Sim, "Unifying deep local and global features for image search," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, 2020, pp. 726–743, doi: 10.1007/978-3-030-58565-5_43.

[43] S. Garg and M. Milford, "SeqNet: Learning descriptors for sequence-based hierarchical place recognition," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4305–4312, Jul. 2021, doi: 10.1109/LRA.2021.3067633.

[44] R. Mereu, G. Trivigno, G. Berton, C. Masone, and B. Caputo, "Learning sequential descriptors for sequence-based visual place recognition," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10,383–10,390, Oct. 2022, doi: 10.1109/LRA.2022.3194310.

[45] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *Int. J. Robot. Res.*, vol. 32, no. 14, pp. 1645–1661, 2013, doi: 10.1177/0278364913499193.

[46] E. Johns and G. Yang, "Feature co-occurrence maps: Appearance-based localisation throughout the day," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2013, pp. 3212–3218, doi: 10.1109/ICRA.2013.6631024.

[47] D. M. Rosen, J. Mason, and J. J. Leonard, "Towards lifelong feature-based mapping in semi-static environments," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA),* 2016, pp. 1063–1070, doi: 10.1109/ICRA.2016.7487237.

[48] C. Linegar, W. Churchill, and P. Newman, "Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA),* 2015, pp. 90–97, doi: 10.1109/ICRA.2015.7138985.

[49] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "FreMEn: Frequency map enhancement for long-term mobile robot autonomy in changing environments," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 964–977, Aug. 2017, doi: 10.1109/TRO.2017.2665664.

[50] C. McManus, B. Upcroft, and P. Newman, "Learning place-dependant features for long-term vision-based localisation," *Auton. Robots*, vol. 39, pp. 363–387, Oct. 2015, doi: 10.1007/s10514-015-9463-y.