Improving Visual Place Recognition in Changing Environments

Bruno Ferrarini

A thesis submitted for the degree of

Doctor of Philosophy in Computer Science

School of Computer Science and Electronic Engineering

University of Essex

July 2023

Abstract

For many years, the research community has been highly interested in autonomous robotics and its various applications, from healthcare to manufacturing, transportation to construction, and more. An autonomous robot's key challenge is the ability to determine its location. A fundamental research topic in localization is Visual Place Recognition (VPR), a task of detecting a previously visited location through visual input alone. One specific challenge in VPR is dealing with a place's appearance variation across different visits, which can occur due to viewpoint and environmental changes such as illumination, weather, and seasonal variations. While appearance changes already make VPR challenging, a further difficulty is posed by the resource constraints of many robots employed in real-world applications that limit the usability of learning-based techniques, which enable state-of-the-art performance but are computationally expensive.

This thesis aims to combine the need for accurate place recognition in changing environments with low resource usage. The work presented here explores different approaches, from local image feature descriptors to Binary Neural Networks (BNN), to improve the computational and energy efficiency of VPR. The best BNN-based VPR descriptor obtained runs up to one order of magnitude faster than many CNN-based and hand-crafted approaches while maintaining comparable performance and expending a small amount of energy to process an image. Specifically, the proposed BNN can process an image 7 to 14 times faster than AlexNet, spending 13% of the power at most when deployed on a low-end ARM platform. The results in this manuscript are presented using a new performance metric and an evaluation framework designed explicitly for VPR applications aiming at the two-fold purpose of providing meaningful insights into VPR performance and making results easily comparable across the chapters.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this or any other university. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others except as specified in the text and Acknowledgements.

Bruno Ferrarini

July 2023

To my family.

Acknowledgements

My deepest gratitude goes to my supervisors, Prof. Klaus McDonald-Maier and Prof. Shoaib Ehsan, and my collaborator Prof. Michael Milford, who have provided me invaluable guidance and feedback throughout my research. Their expertise and support have led to the successful completion of this thesis. I am grateful for the enlightening and productive conversations I have had with Prof. Shoaib Ehsan on Visual Place Recognition (VPR), which have greatly helped me navigate this exciting field. I also extend my thanks to my colleagues and friends, Maria and Sania Waheed, Bruno Arcanjo, and Mihnea-Alexandru Tomita, for the valuable time spent working together. Last but not least, I want to express my appreciation to Guido Mondelli and Marcello Boiardi at MyWay s.r.l. for allowing flexible working hours, which enabled me to complete this research while working full-time.

Contents

List of Figures

List of Tables

Abbreviations

ARM	Advanced RISC Machines
AUC	Area Under Curve
BNN	Binary Neural Network
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DOF	Degree of Freedom
EP	Extended Precision
FN	False Negative
FP	False Positive
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
GPU	Graphical Processing Unit
HB-DS	Half-Binary Depthwise Separable (module)
LCD	Loop Closure Detection
MAC	Multiply-Accumulate operations
PN	Predicted Negative
PP	Predicted Positive

PR, P-R	Precision-Recall
PR-Curve	Precision-Recall Curve
RPI4	Raspberry Pi 4
TN	True Negative
TP	True Positive
UAV	Unmanned aerial vehicle
VPR	Visual Place Recognition

Chapter 1

Introduction

The development of autonomous robots has been a major focus of the robotics community for many years, motivated by the potential to improve efficiency, reduce costs, and enhance safety in various industries, including manufacturing, transportation, agriculture, and healthcare. A major challenge in robotics is enabling robots to move around and navigate autonomously in the working space. This is critical for a wide range of applications, such as self-driving cars, aerial surveillance, exploration missions, delivery services, and cleaning robots. Given its significance, autonomous navigation has become a highly researched area in the field of robotics over the past decades, attracting attention from both the research community and industry. For example, the 2023 *International Conference on Robotics and Automation* (ICRA 2023) has several workshops dedicated to autonomous navigation methods[1]. Meanwhile, the 2022 edition of *Intelligent Robots and Systems* (IROS 2022) hosted the *13th Workshop on Planning, Perception, and Navigation for Intelligent Vehicles*[2]. On the industry front, there have been several salient develop-

[1]https://www.icra2023.org/programme/workshops-tutorials
[2]https://project.inria.fr/ppniv22/

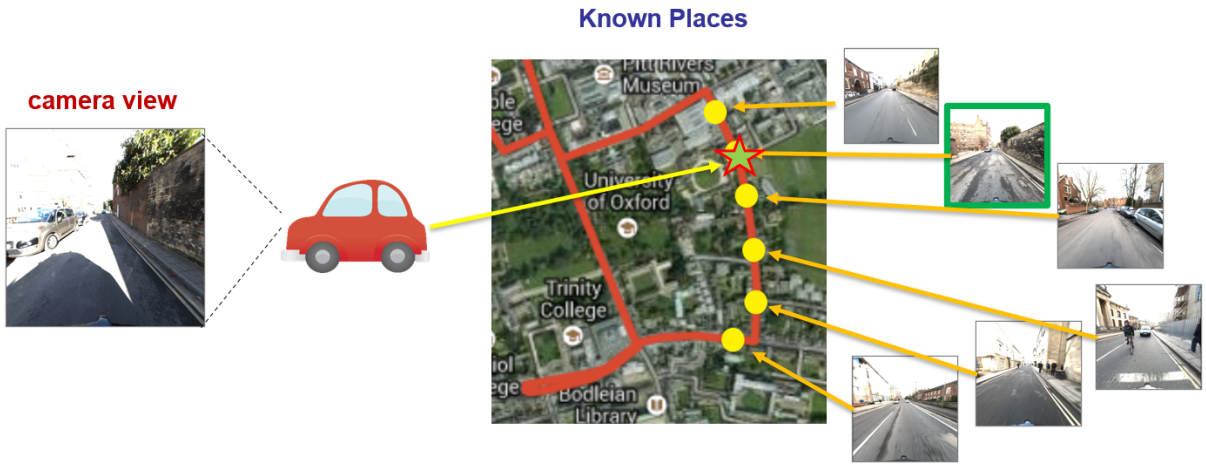


Fig. 1.1. Localization based on Visual Place Recognition. Place images from [1].

ments in the field. Tesla launched its Full Self-Driving Autopilot beta program in the last quarter of 2022[3], Amazon started a small-scale program to deliver goods with unmanned aerial vehicles in mid-2022 in California[4], and Waymo by Google[5] powers taxi services in Phoenix, Arizona and San Francisco, CA.

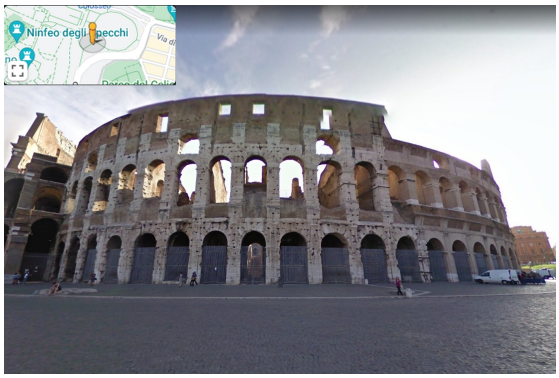
Self-navigating robots need to know their position to plan and control their movement toward a goal. When GPS (Global Positioning System) is unavailable, or odometry estimates drift due to accumulated errors over time, a robot can still determine its position through Loop Closure Detection (LCD) [13]. LCD involves detecting when a robot has returned to a previously visited location. There are various methods for performing LCD, each utilizing different types of sensory information and algorithms. Within visual-based navigation, the core of LCD is Visual Place Recognition (VPR), consisting of matching one or more frames from the robot's onboard camera with a collection of tagged images that depict different locations in the workspace, as exemplified in Fig. 1.1.

[3]https://www.tesla.com/en_eu/support/autopilot
[4]https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries
[5]https://waymo.com/waymo-one/

Localization based on visual information has drawn the attention of the research community, becoming one of the most researched topics in recent years: a Google Scholar search for "Visual Place Recognition" returns thousands of results for 2022 alone. One of the reasons for this widespread interest is the abundance of rich environmental information that can be obtained from cameras, which are now readily available on a variety of robots, including those with strict payload limitations, such as small drones [14, 15]. Although extensively investigated, VPR remains a challenging task due to factors such as the dynamic nature of the environment, the flexibility of a robot's movements, and the limited resources available on hardware platforms, especially those designed to be cost-effective or to save battery life. The following section is dedicated to the main VPR challenges, focusing mainly on those addressed by this thesis work.

1.1 VPR Problem Statement

As part of the localization process, VPR is a fundamental task to enable a robot to navigate autonomously. Although VPR is a well-defined problem of deciding if a place has already been visited [11], it is necessary to make some assumptions related to the domain of an application. In particular, it is important to define what a place is in that domain. For example, if two images show two points of interest in Rome, are they showing Rome city or two different places in Rome? Or, if a picture shows a close-up of the Colosseum and another has it far in the background, are they showing the same location or two parts of Rome that are too distant to be considered the same place? To avoid such ambiguity, this thesis adopts a common approach in the literature that Garg and Fisher have recently formalized [16]. Instead of basing VPR on an explicit definition of place,



(a) North-East view.

(b) South-West view.

Fig. 1.2. Two images captured in opposite directions from the same geographical location. Images from Google Maps [18].

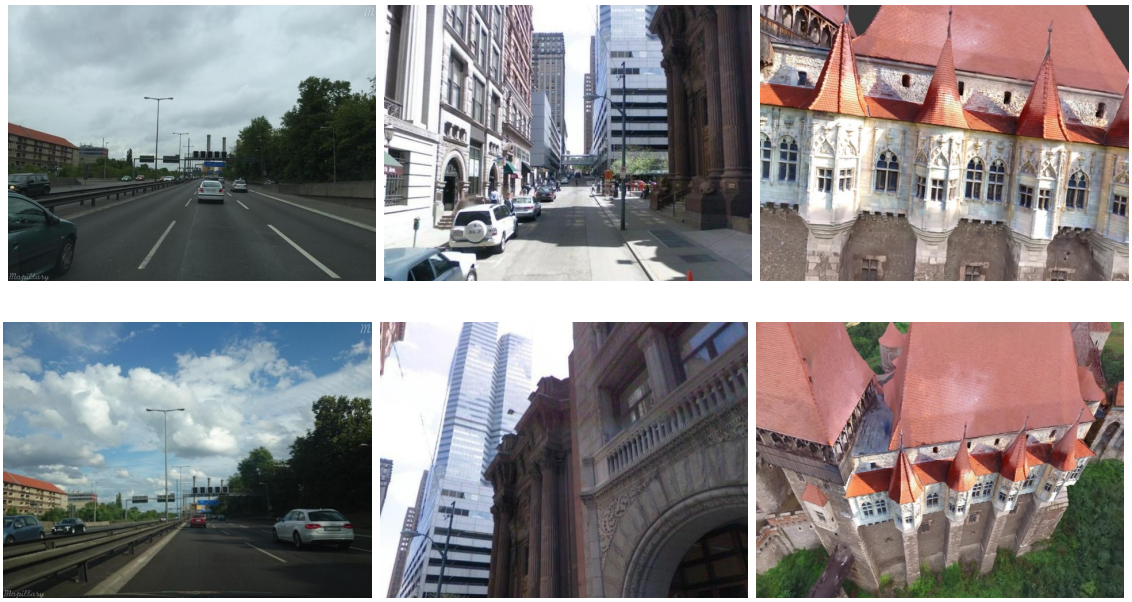
VPR is defined as comparing images of the same physical location with a sufficient degree of field-of-view overlap to enable matching. This implies that capturing two images from the same location but in opposite directions, as shown in Fig. 1.2, does not guarantee the successful place matching. This approach does not reduce the generality of the VPR formulation as a place can be represented by multiple images on the map to enable loop closure from different directions [17].

1.2 Challenges and Open Problems

This section summarizes the most critical challenges in Visual Place Recognition ranging from place appearance changes to the problem of efficient resource utilization, which is crucial to enable VPR on edge devices and low-end hardware platforms.

1.2.1 Dynamic Place Appearance in Visual Place Recognition

The core problem of Visual Place Recognition is to match images showing the workplace's locations. Apart from those related to general image retrieval tasks, place matching has



(a) Lateral shift. **(b)** 3-axis. **(c)** 6-DOF.

Fig. 1.3. Viewpoint variations can result from a lateral shift (a), 3D motions in 3-axis (b) , and 6-DOF movements (c). Images from [2, 3, 4].

specific problems to be addressed to perform VPR effectively. A place can appear differently across successive traversals of the environment for several reasons. A robot might re-enter a location from another direction, looking at the scene from a different point of view, the environmental conditions may vary between traversals, or the scene itself may mutate over time with new or missing elements. Appearance changes can be separated into two broad categories: viewpoint and conditional changes, depending on the causes. As illustrated in Fig. 1.3, changes in the viewpoint may result from a lateral shift, an orientation change in the 3-axis of the camera, or a combination of movement in both the 3-axis and spatial directions, resulting in a 6-DOF (Degree-Of-Freedom) viewpoint change, which is typical of aerial platforms. Conditional appearance changes are due to environmental causes. Fig. 1.4 highlights the influence of environmental factors on a place's appearance. The cyclical alternation between day and night and the use of ar-

(a) Snowy winter.

(b) Snowy at night.

(c) Rainy weather.

(d) Sunny in summer.

Fig. 1.4. A place from SPED dataset [5] in various times.

Fig. 1.5. Two examples of perceptual aliasing from [6, 7].

tificial lights cause variations in illumination. Additionally, dynamic elements entering or leaving a scene, such as vehicles, can cause appearance changes by obstructing or exposing certain parts. In the long term, fluctuations in weather and seasonal cycles may also impact VPR, as well as modifications to the physical environment caused by either natural events (e.g., tree growth) or human activities (e.g., new buildings).

As opposed to the problem of matching different images of the same place, perceptual aliasing [11] is the problem of distinguishing between different sites looking similar. Perceptual aliasing is more common in indoor environments with repetitive elements such as doors, corridors, and furniture but also in natural settings such as grass and fields. Fig. 1.5 presents examples of indoor and outdoor places that look similar.

The experiments in this thesis aim to provide general results and do not address specific viewing conditions or environment types, except in Chapter 4, which mainly considers 6-DOF viewpoint changes. Therefore, the benchmark data for the experiments are selected among well-established VPR datasets to include several viewpoints and en-

vironmental changes.

1.2.2 Resource Utilization

The research community has spent considerable effort to achieve high VPR accuracy, paying less attention to the resource-demanding to execute a technique. With the diffusion of small robots supplied by batteries in long-term operations, the resource amount demanded by a navigation system acquired importance [19, 20, 21]. Improving the efficiency of VPR means reducing its computational complexity, memory footprint, and energy usage at runtime, as well as shrinking the space required to store the VPR module itself and the environmental knowledge (i.e., map) and the amount of data exchanged between cooperating robots. One of the primary goals of this thesis is to reduce the computational and energy costs of VPR in changing environments, dedicating two chapters to this topic, the 5^{th} and 6^{th}.

1.2.3 Performance Evaluation

Decades of active research have led to the development of numerous techniques and approaches to solve the VPR problem. Determining the most suitable state-of-the-art method for a specific application is not trivial. Each work assesses its newly proposed technique in its own application context and experimental setup making cross-paper comparison unreliable [22]. This consideration raises the need for updated comparisons based on uniform evaluation criteria as new VPR approaches are proposed. A second problem identified by this thesis's author relates to reducing VPR resource usage, which is one of the targets of this research. Improving efficiency often results in reduced per-

formance, creating a need for an analysis methodology to identify the actual performance difference between VPR methods to evaluate efficiency-performance trade-offs appropriately. Chapter 3 addresses these challenges by proposing an evaluation framework and a performance metric explicitly designed to assess VPR.

1.3 Thesis Contributions

The most significant contributions made during this research work are outlined below.

1. The first contribution of this thesis is an evaluation method designed explicitly for VPR applications. It is based on a new performance metric, Extended Precision (EP), computed from a Precision-Recall Curve to overcome the limitation of the well-established R_{P100} in assessing the lower spectrum of VPR performance. This is because R_{P100} can only be computed for those VPR techniques performing well enough to reach Precision = 1. EP uses a Precision-Recall Curve's features differently than R_{P100} to be defined regardless of the VPR performance level. The proposed evaluation method is developed around EP and proposes two kinds of analysis. One assesses the VPR performance level and consistency across an operating environment. The second is a comparison methodology that identifies statistically significant performance differences using a McNemar test variant [23]. This test is used to confirm the reliability of a comparison outcome in determining the best VPR technique.

2. The second significant contribution of this thesis revolves around solving the VPR problem efficiently on the fronts of computation, power usage and memory footprint to enable VPR on low-end hardware platforms and small robots. The research

started by considering the use of handcrafted local feature descriptors. The results obtained showed that local feature descriptors are prone to conditional changes. At the same time, the approaches based on Convolutional Neural Networks (CNN) [24] perform better in such operating conditions but at the cost of heavier resource usage at runtime [4]. This evidence suggested searching for more efficient learning-based approaches than CNN. Binary Neural Networks (BNN) [25] are investigated as a possible alternative to CNN. BNNs use 1-bit per parameter and bit-wise arithmetics to achieve high computational efficiency and compact model sizes. Typically, a BNN has a model size 90% smaller than a CNN with the same structure and executes around an order of magnitude faster using only a fraction of the energy consumption. This thesis first investigates the suitability of BNNs for VPR. Then it proposes a compact BNN called FloppyNet, which runs seven times faster than a standard AlexNet [26], reaching similar performance under conditional changes.

3. The final contribution of this research addresses the computational bottleneck in the first convolutional layer that generally affects BNNs, including the one proposed in this thesis. BNNs work better if they take high-precision inputs instead 1-bit inputs. Consequently, the first convolution cannot use bit-wise arithmetic resulting in one of the most computationally expensive layers of a BNN. The solution proposed by the author is an in-place replacement for the first convolution that drastically reduces the number of non-binary computations while keeping the same performance level. It was possible to halve the energy usage and processing times of FloppyNet with no VPR performance reduction.

1.4 Thesis Structure

The reminder of this manuscript is organized into six chapters as follows.

Chapter 2 presents a literature review of the existing research on Visual Place Recognition and Binary Neural Networks. This chapter is organized into a detailed survey on image representation and matching, followed by a description of metrics and benchmark datasets used to evaluate VPR. It concludes with a survey on BNNs and the compute engines used for their deployment on embedded platforms.

Chapter 3 presents a new evaluation framework designed explicitly for VPR based on a new metric called Extended Precision (EP). The proposed framework offers two types of evaluations. The first is designed to assess the performance and consistency of a VPR descriptor over an entire traversal of the working space. The second has the purpose of identifying the performance differences between VPR methods. The comparison approach is based on a McNemar's test variant [23] to provide statistically significant outcomes. In order to demonstrate the utility of the proposed framework, results for several state-of-the-art VPR techniques are presented for various appearance changes using five different benchmark datasets.

Chapter 4 investigates the local feature descriptors' [27] suitability for VPR under extreme viewpoints and environmental changes as a more efficient alternative to learned-based descriptors, which achieve the highest VPR performance at the cost of a high resource demand at runtime. Chapter 4 starts with analyzing the trade-off between VPR accuracy and place matching time for several well-established local feature descriptors. Then, results are presented to demonstrate that local features exhibit the same VPR accuracy as CNN-based approaches on 6-DOF viewpoint changes while running faster

but are outperformed in dynamic environments by a significant margin.

Chapter 5 attempts to conjugate convolutional neural networks' performance in changing environments with the resource limitations of low-end hardware platforms. The author investigates the suitability of BNNs for addressing VPR in changing environments and then proposes FloppyNet, a compact global image descriptor of the same kind. It has lower but comparable performance to deeper and regular Convolutional Neural Networks while spending considerably less energy and time to process an image.

Chapter 6 addresses a computational bottleneck concerning the first convolution of BNNs. A BNN needs full-precision inputs to avoid the noticeable performance drop caused by binary inputs. Consequently, the first convolution is not entirely binary and results in one of the slower stages of a BNN. Especially in compact networks like FloppyNet, the processing time spent on the first layer is a large share of the total. This chapter proposes an in-place replacement module for the first convolutional layer to greatly reduce the number of floating-point operations computed in the first layer of a BNN. This module, called HB-DS, is then used on FloppyNet to obtain a BNN twice as fast without any accuracy loss.

Finally, **Chapter 7** provides the author's final remarks and suggests how to develop further the research presented in this thesis.

1.5 List of Publications

Following publications were made during the course of this PhD:

1. B. Ferrarini, M. Milford, K. D. McDonald-Maier and S. Ehsan, "Highly-Efficient Binary Neural Networks for Visual Place Recognition," 2022 IEEE/RSJ International

Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 5493-5500, doi: 10.1109/IROS47612.2022.9981978.

2. B. Ferrarini, M. J. Milford, K. D. McDonald-Maier and S. Ehsan, "Binary Neural Networks for Memory-Efficient and Effective Visual Place Recognition in Changing Environments," in IEEE Transactions on Robotics, vol. 38, no. 4, pp. 2617-2631, Aug. 2022, doi: 10.1109/TRO.2022.3148908.

3. B. Ferrarini, M. Waheed, S. Waheed, S. Ehsan, M. J. Milford and K. D. McDonald-Maier, "Exploring Performance Bounds of Visual Place Recognition Using Extended Precision," in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 1688-1695, April 2020, doi: 10.1109/LRA.2020.2969197.

4. B. Ferrarini, M. Waheed, S. Waheed, S. Ehsan, M. Milford and K. D. McDonald-Maier, "Visual Place Recognition for Aerial Robotics: Exploring Accuracy Computation Trade-off for Local Image Descriptors," 2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2019, pp. 103-108, doi: 10.1109/AHS.2019.00011.

5. B. Ferrarini, S. Ehsan, A. Bartoli, A. Leonardis., K. D. McDonald-Maier K.D. (2019) "Assessing Capsule Networks with Biased Data". In: Felsberg M., Forssén PE., Sintorn IM., Unger J. (eds) Image Analysis. SCIA 2019. Lecture Notes in Computer Science, vol 11482. Springer, Cham. doi: 10.1007/978-3-030-20205-7_8.

6. M. A. Tomiţă, M. Zaffar, B. Ferrarini, M. Milford, K. Mcdonald-Maier and S. Ehsan, "Sequence-Based Filtering for Visual Route-Based Navigation: Analysing the Benefits, Trade-offs and Design Choices," in IEEE Access, 2022, doi: 10.1109/ACCESS.2022.3196389.

7. B. Arcanjo, B. Ferrarini, M. Milford, K. D. McDonald-Maier and S. Ehsan, "An Efficient and Scalable Collection of Fly-Inspired Voting Units for Visual Place Recognition in Changing Environments," in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 2527-2534, April 2022, doi: 10.1109/LRA.2022.3140827.

Chapter 2

Literature Review

This chapter presents a survey of the existing literature on the primary topics covered by this thesis. The first one is Visual Place Recognition (VPR), a very active research topic with countless publications in the literature. The most relevant methods in the field are presented, breaking them down into two broad categories: local and global VPR descriptors, which are further separated into handcrafted descriptors and deep-learning-based techniques. The survey continues with the evaluation of VPR, covering the principal metrics, methodologies, and datasets to frame one of the contributions of this work, which concerns a new evaluation approach for VPR. The last part of this chapter focuses on Binary Neural Networks (BNN), a class of neural networks extensively used in this work to improve the computational and energy efficiency of VPR. This survey provides an in-depth overview of BNNs, including design principles, training methodology, and compute engines for binary arithmetic.

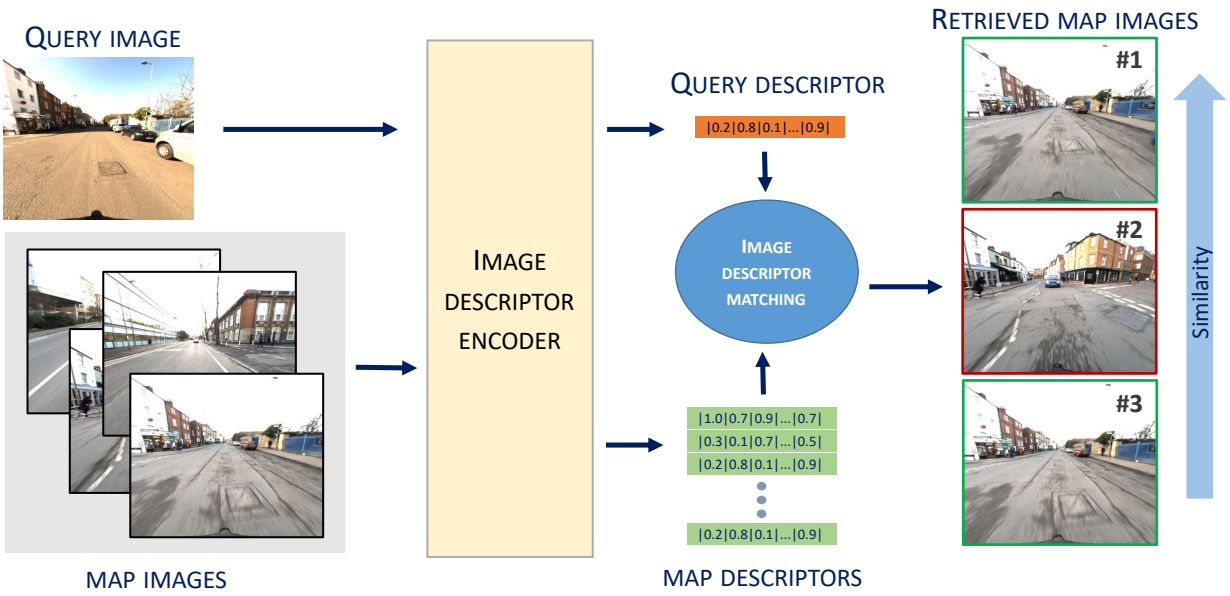
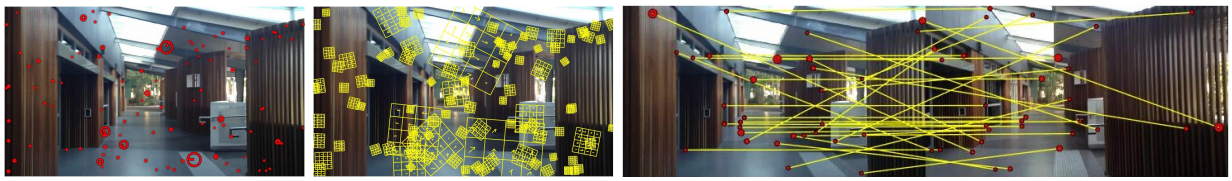


Fig. 2.1. A typical Visual Place Recognition pipeline based on image retrieval. Images from [1].

2.1 Visual Place Recognition

Visual Place Recognition (VPR) enables a robot to determine its location by searching the place images in the workspace map for the closest match to the camera's view. VPR is commonly cast as an image retrieval task based on image matching [28, 29, 11], as depicted in Fig. 2.1. The prior knowledge of an environment is represented by a collection of tagged images, each corresponding to a single place. This manuscript refers to this collection of place images as the *reference dataset* or *map*. The image a robot captures with the onboard camera is used as a *query* to search the map for the most similar images. The place shown in the best-matching image with the query is regarded as the current robot's location.

A place's appearance typically varies depending on the time and perspective in which it is captured. Therefore, images are not compared directly but are processed to compute a *descriptor*, which is an image representation invariant to those changes. The



(a) Local features. **(b)** Local descriptors. **(c)** Pairwise descriptor matching.

Fig. 2.2. Features extraction and pairwise descriptor matching. Images from [8].

query descriptor is compared against map descriptors to find the most similar images, which are then returned as query results. Image representations are often separated into handcrafted and deep-learning approaches to emphasize the changes in the field after the introduction of CNNs [11, 28, 30]. However, the author of this thesis prefers to draw attention to the type of image representation instead of the method used to obtain it, as recent works have returned to handcrafted techniques for their higher computational efficiency [4, 31]. In contrast, several learned approaches are inspired by "traditional" algorithmic pipelines [32, 29, 33]. Therefore, this section presents a selection of approaches to compute a descriptor suitable for VPR, dividing them into the two broad categories of local feature image representations and global image representations, which are subsequently separated into several narrower categories, including, among others, handcrafted and deep-learning approaches. This section also covers VPR evaluation methodologies and benchmark datasets.

2.1.1 Local Feature Representations

Building an image representation based on local image features requires two steps. The first analyzes an image to localize distinctive patterns such as edges and corners. The second computes a representation of the neighborhood of those keypoints' locations.

Figs. 2.2a and 2.2b show an example of this process for SIFT [34]. The resulting overall image representation is a collection of local descriptors that can be either pairwise compared to match images (Fig. 2.2c) or further processed to obtain more robust or compact image representations [35, 36]. The sections below provide an overview of several approaches, including handcrafted and deep learning-based ones.

Handcrafted Local feature Descriptors

As mentioned above, the first step to take to build an image representation is analyzing an image to search for specific patterns: the local image features. For example, Harris-Affine and Hessian-Laplace detectors search for corners [37], Edge-Based Region operates an edge-based region detection [38], and SIFT [34] detector stage identifies blob regions from the Difference-of-Gaussian (DoG) function's local response [39]. Then, a patch around every detected feature is processed to compute a representation suitable for matching (e.g., a vector or real numbers) using a local descriptor such as SURF [40] and ORB[41]. Once the salient locations of an image are identified, a representation of their neighborhood needs to be computed. Other than the detector stage mentioned above, SIFT has a descriptor stage. It employs Histogram-of-Oriented-Gradients (HOG) [42, 43] to compute a descriptor of keypoints' neighborhoods. SIFT is used for VPR in [44, 45]. SURF [40] uses Haar-wavelet [46] to obtain a relatively short descriptor of 64 elements. Because of its descriptor length, SURF is more computationally efficient than SIFT, which produces a longer vector of 128 elements. SURF is used in a variety of VPR approaches such as [47, 48]. While SIFT and SURF are robust to rotation and scaling, BRIEF [49] is not invariant to any geometrical change but is very efficient and deals reasonably well with dynamic environmental conditions, as shown by Churchill

and Newman in their long-term navigation system [50]. BRIEF uses the intensity of randomly sampled pixels to build local image descriptors. BRISK [51] features vector is computed similarly to BRIEF using pixel intensity but along concentric circular patterns. This sampling strategy allows the use of distant and close pixel pairs to assign the intensity gradient with an orientation. BRISK is used in UAV[1] localization for its efficiency [4, 52]. ORB combines FAST detector [53] with a rotation-invariant evolution of BRISK. As confirmed later in Section 4.3.1, ORB offers a good trade-off between matching accuracy and computational efficiency. Mur-Artal *et al.* used it for ORB-SLAM [54, 55] and Wu *et al.* to identify Regions-of-Interest (ROIs) to adapt the feature extraction to illumination changes in indoor environments [56]. ORB also found application in space exploration because its computational efficiency [57, 31]. AKAZE [58] has been employed in indoor localization based on landmark detection [59], exhibiting better place recognition performance than SIFT. CoHOG [60] is a recent image descriptor proposed as a training-less and computationally efficient alternative to CNN-based techniques to address VPR. It uses image entropy to detect regions of interest that are subsequently assigned with a HOG descriptor. CoHOG is designed to achieve lateral viewpoint tolerance by region matching.

Deep Learning Approaches

The main goals of a descriptor are invariance to appearance changes and distinctiveness to deal with perceptual aliasing. The fundamental idea of most of the proposed approaches in the literature is training a CNN to enforce it to output the same descriptor for different patches corresponding to the same image's region and, conversely, different

[1] Unmanned Aerial Vehicles.

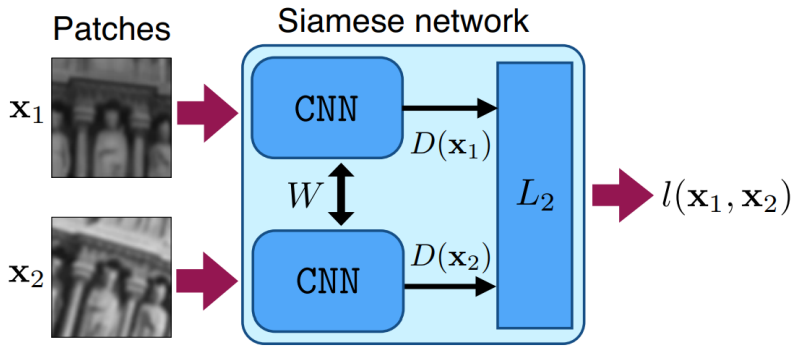


Fig. 2.3. A Siamese network to train a local image descriptor. Image taken from [9].

descriptors for non-matching patches. This goal is obtained by training multiple CNN instances while sharing their weights. The most relevant example of this approach is the Siamese architecture [61], which can be extended to multiple branches [33]. [62] proposes a two-branch CNN to extract the features from patch pairs that are subsequently fed into a binary classifier that determines whether the patches match. Simo-Serra *et al.* [9] use the Siamese network illustrated in Fig. 2.3 to train a model that produces a vectorized local descriptor. The shared weights between the two CNNs are updated in the backward pass, enforcing $D(x_1) == D(x_2)$ for matching patches and $D(x_1) \neq D(x_2)$ for non-matching patches. The main advantage of training a patch descriptor on a distance-based loss function is that it can directly replace standard handcrafted descriptors in existing systems. Triplet Siamese Networks are employed in [63] to train a vectorized descriptor as in [9]. The main advantage of a triplet network over a standard Siamese network is that the loss function allows training distinctive descriptors without mining *hard samples* through multiple optimization iterations of the training data, as needed in [9]. Learned Invariant Feature Transform (LIFT) [33] consists of a detector and descriptor stages trained end-to-end on SIFT features. End-to-end training improves patch representation as the detection and descriptors tasks are optimized together as

they were a single task to reach the local minimum of a single loss function. Like LIFT, SuperPoints [32] is a complete detector-descriptor pipeline. It simultaneously detects corners in an image and computes the corresponding local descriptors. SuperPoints consists of an encoder followed by two decoders: one to find keypoints and the other to compute the corresponding descriptors. Unlike LIFT and other methods mentioned thus far, SuperPoints operates on the entire input image simultaneously. The primary advantage is a more effective feature extraction at different scales that is limited by the patch size in patch-based methods.

Features Processing and Aggregation

Typically, an image representation includes hundreds to thousands of local descriptors, depending on the image size and content [64]. Without further post-processing, image matching requires pairwise comparisons between local descriptors (Fig. 2.2c). While this approach guarantees an excellent tolerance to geometrical variation as keypoints are tracked across images, on the other hand, it does not scale well to search large reference datasets, as often needed by VPR applications. A more efficient approach for large-scale image matching is comparing images indirectly through their statistics. Bag-of-Words (BoW) [35, 36] quantizes the feature space in clusters using a *codebook of words*, a set of centroids learned with a clustering algorithm (e.g., k-means [65]). All the local descriptors are then assigned to the closest cluster to form a histogram of frequencies, which is used as an image representation. BoW descriptors are compared through a single normalized inner product operation (i.e., cosine similarity [66]). BoW is used to aggregate convolutional features in several works [67, 48, 2, 68]. Vector of Locally Aggregated Descriptor (VLAD) [69] is based on a similar idea of clustering the feature

(a) **(b)**

Fig. 2.4. (a) Salient locations used by local descriptor-based representations; (b) the block pattern of GIST [10], a global descriptor. Images taken from [11].

space employing a dictionary. However, instead of using frequencies to build a histogram, it accumulates the differences between clusters and their assigned feature descriptors to form a residue vector. The VLAD descriptor is considerably longer than a typical BoW representation, but it is more distinctive [70]. Several VPR apparoaches employ VLAD [71, 72, 31], including this thesis work (Chapter 4). Triangular embeddings with democratic aggregation (T-Embedding) [73] exhibits good performance in place retrieval [74]. Unlike VLAD, it only accumulates residues into the closest centroid to a local descriptor to reduce false positives. As mentioned above, these image representations are convenient for image retrieval from large datasets because they can be efficiently compared with vectorial operands, whereas local descriptors require pair-wise matching. Moreover, they inherit some of the invariance properties of the local descriptors they are built from [75].

2.1.2 Global Image Representations

As opposed to local features-based techniques that target salient locations of an image, global representation methods follow a pre-defined pattern or divide an image into pre-defined blocks and process them regardless of their content, as exemplified in Fig. 2.4.

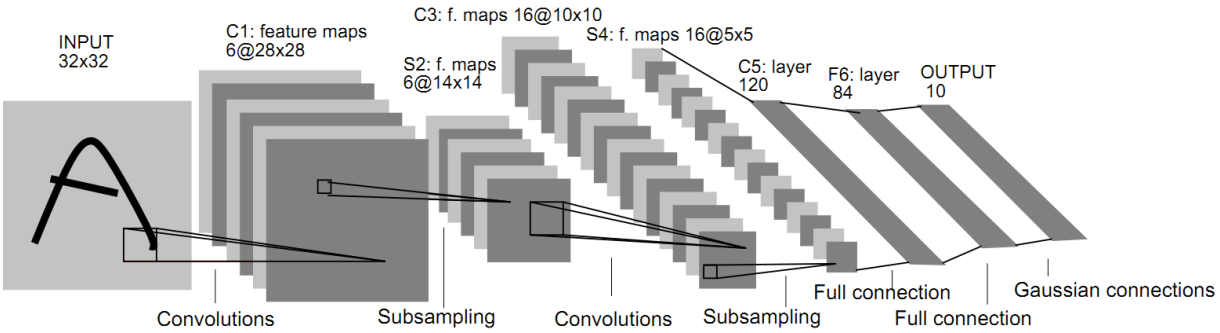


Fig. 2.5. LaNet CNN architecture from [12].

Handcrafted Approaches

Gist [10] is an example of a global image descriptor that is used for matching place images in [76, 77, 78] and [79]. Gist extracts global features from an image using a set of Gabor filters [80] at different orientations and frequencies. The results are then averaged to obtain an image representation as a vector. Another global descriptor is Histogram-of-Oriented Gradients (HOG) [42, 43]. It calculates the gradient of all image pixels and uses the results to create a histogram, with each bar representing the gradient angles and carrying the summation of gradient magnitudes. McManus *et al.* used HOG for VPR in [81]. BRIEF-Gist is proposed to address the loop closure detection problem [82]. The BRIEF-Gist algorithm downsamples an image to 60×60 pixels and then computes a BRIEF descriptor around the image center. Handcrafted global descriptors are generally more computationally efficient than local features-based approaches [28]. Therefore they are used as a comparison baseline to demonstrate the computational efficiency of the VRP approaches proposed in this work in Chapters 5 and 6.

Deep Learning Approaches

Convolutional Neural Networks (CNN) are neural networks designed to exploit local correlation and repetitive patterns in images. Since the pioneering work of Yann LeCun *et al.* in 1998 [12] and the pivotal AlexNet in 2012 [26], this type of network has grown in popularity for many vision tasks, including VPR. A typical CNN is shown in Fig. 2.5, where the shaded planes are the feature maps computed by CNN's layers, and the convolution and pooling kernels are represented with squares.

CNN-based methods achieve high performance in various environmental conditions [21] and under viewpoint variations [83]. A pre-trained CNN for a different task can be used off-the-shelf for generating an image descriptor in place of handcrafted image descriptors. The earliest attempts [84, 85, 86] to build an effective descriptor for place recognition used fully-connected features from a CNN pre-trained on ImageNet [87], a generic dataset for image classification. However, better image-matching performance is obtainable if the fully-connected layers are fine-tuned specifically for image matching using a triplet architecture, as shown in [88]. A fully-connected representation is akin to a global image descriptor embedding the whole information of an image. Conversely, a convolutional layer produces a feature tensor of $H \times W \times C$, where every vector of C elements in the $H \times W$ maps describes a region in the input image. Such a tensor can be vectorized and used as a whole image descriptor for image matching, working reasonably well even if computed by a CNN originally trained for classification on generic datasets. Hou *et al.* [89] showed that the features extracted from *conv3* layer of AlexNet are robust to conditional variations, while those from *pool5* work better for viewpoint changes. Bai *et al.* [90] used those layers' features to improve the matching performance of SeqSLAM [91] under viewpoint changes. AMOSNet and HybridNet [5] are variants of AlexNet

Fig. 2.6. Convolutional features in shifted images (left). Significant shifts cause fails in element-wise comparison between feature maps (right). Images from [5].

trained on Specific PlacEs Dataset (SPED) [5] in order to compute more specific image representations for VPR. PlaceNet [92] is based on the same idea but uses VGG-16 [93], which is trained on a large dataset, called Places365, organized in 365 place categories (classes). CALC [94] is a lightweight CNN proposed for efficiently addressing the loop closure detection problem. The peculiarity of CALC is that it is trained in an unsupervised manner using an autoencoder to recreate a HOG descriptor from geometrically distorted place images. NetVLAD [71] was published in 2016, but it still can be considered among the best-performing global descriptors for place recognition even nowadays. It consists of two stages, trained end-to-end. The first is a VGG-16 network that extracts the features from an image, followed by an aggregation layer implementing a VLAD-like embedding [69]. The aggregation has more trainable parameters than standard VLAD providing more flexibility and suitability to work with convolutional features.

Convolutional Features Pooling and Embedding

As mentioned above, a convolution produces a tensor of $H \times W \times C$ elements where a $1 \times 1 \times C$ vector describes a rectangular region in the input image. An element in the image that produces a response peak in the feature map appears in different locations of shifted images. Hence, comparing two vectorized convolutional representations results in mismatches between shifted images beyond a CNN's tolerance, as exemplified in

Fig. 2.6, and under viewpoint changes in general. This problem is addressed by processing convolutional features with a pooling or embedding stage to build a more robust image representation to viewpoint changes. This approach appears similar to local features extraction in the two steps of identifying regions of interest and then computing the corresponding descriptors. However, there is a significant difference: the regions of interest are identified in a convolutional layer's output and not in the input image, allowing VPR to take advantage of the representative power of learned features in coping with appearance changes [95, 90].

Regional Maximum Activations of Convolutions (R-MAC) [96] uses a pooling scheme directly on one of the deeper convolutional feature maps. Regions of increasing size are identified to include the entire feature map following a similar pattern as spatial pyramid pooling [97]. Each region is max-pooled to obtain a region-specific descriptor. The resulting vectors are combined and L_2-normalized to form the whole image representation. Cross-Region-Bow [2] identifies regions of interest (ROIs) in the feature map of a specific convolutional layer (e.g., *conv5*). An ROI is the neighborhood of a local maximum similar to those shown in Fig. 2.6. The features underlying those ROIs in the preceding layer (e.g., *conv4*) are pooled to form an ROI local descriptor. Then, an image representation is computed from those ROI descriptors using BoW. RegionVLAD [72] follows the same idea as Cross-Region-Bow. The only differences are the pre-trained network, PlaceNet [92] instead of VGG-M [93], and the post-processing phase using VLAD instead of BoW. Hustler *et al.* proposed a region-based descriptor for place recognition aiming at a robust viewpoint and scale invariance based on NetVLAD. Patch-NetVLAD [98] first extracts a set of matching candidate image regions using NetVLAD, then computes a new patch-level representation to perform matching. To this end, the Patch-NetVLAD pipeline is

similar to a traditional local feature representation but entirely based on deep learning.

2.1.3 Place Classification

As mentioned above, the most common formulation of the VPR is an image retrieval problem. The main advantage of this approach is that there are no constraints on the number of locations on the map. Also, training or setting up the VPR module for a specific environment is no longer necessary if a technique has enough generalization skills. However, when an application domain has a fixed number of places, approaching VPR as a classification task might be convenient. Due to the relative simplicity of the application domain, the network does not need a large number of parameters, and the data required for training is a few compared to typical VPR datasets such as Places365 [92] or Pittsbourg250K [3], which include hundreds of thousands of images.

FlyNet [99] is a recent VPR technique working as a classifier on place sequences. It uses a module inspired by the olfactory neural system of the drosophila melanogaster to obtain a compact binary image representation fed into a single layer of neurons serving as a place classifier. Flynet is tested on various setups with 1000 places that have undergone conditional changes exhibiting reasonably good performance that can be further improved to compete with larger and deeper models when extended with a CANN stage [100]. Like FlyNet, Drosonet [101] is a drosophila-inspired network, which is quantized to 8-bit to improve memory efficiency. Exploiting the low memory requirement of Drosonet, the authors merged multiple units using a voting system to deal with extreme conditional changes. Hussaini *et al.* [102] proposed a place classifier based on Spiking Neural Networks (SNNs) [103], a neural network type carrying a single bit over a sequence of spikes to encode information to emulate the synaptic transmission of bio-

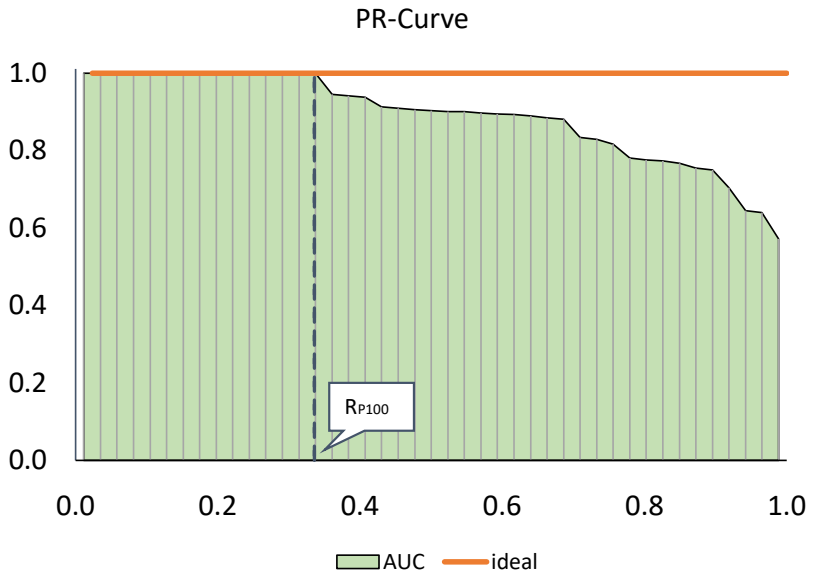


Fig. 2.7. A PR-Curve example highlighting two derived metrics used to evaluate VPR performance: AUC and R_{P100}.

logical neural systems. The classifier proposed in [102] is expected to run efficiently on neuromorphic hardware [104], which has an architecture designed to emulate the information propagation model of biological neural systems used by SNNs.

2.1.4 Visual Place Recognition Evaluation

Because of the many techniques proposed in the literature, measuring VPR performance has become a relevant task for the robotics community in recent years [105, 106, 28]. This section presents several metrics and tools used to evaluate VPR and motivates the new one proposed later in Chapter 3.

Fig. 2.7 shows a Precision-Recall Curve (PR-Curve). PR-Curves are suitable for unbalanced data [107, 108], which is the case of VPR, where the positive matches for a query in the reference dataset are a few compared to the negative matches. Hence, PR-Curves are often used to evaluate VPR [11]. A curve is plotted from a set of Precision-Recall

pairs at different cut-offs of the similarity matching threshold [109] or varying a parameter that affects the skill of a VPR technique [91]. Precision and Recall are computed as:

$$\textit{Precision} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}}, \tag{2.1}$$

$$\textit{Recall} = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}, \tag{2.2}$$

where True-Positives (TP) are the places that a VPR matched correctly, False-Positives (FP) are those erroneously matched, and False-Negatives (FN) are real positive matching places discarded by the VPR technique.

From a PR-Curve can be derived several performance measurements. Area-Under-Curve (AUC) is the area underlying a PR-Curve [107], which is often used as a performance metric for VPR [4, 72, 94, 101]. AUC is a suitable criterion for applications requiring high Precision and Recall. A higher AUC value indicates better performance, with an AUC of 1 representing the best. Average Precision (AP) is suitable when Recall is less critical than Precision in retrieval tasks. Indeed, AP is computed similarly to AUC but considers only the Precision values corresponding to a TP in the retrieved sequence of images:

$$\textit{AP} = \frac{1}{N_T} \sum_{i=1}^{N_R} P(i) r(i). \tag{2.3}$$

N_R is the number of points in the curve, which are highlighted by vertical lines in Fig. 2.7. $P(i)$ is the precision value at i, which is counted depending on the relevance function, $r(i)$, whose value is 1 in correspondence with a TP, 0 otherwise. N_T indicates the number of true matches in the dataset. AP varies from 0 to the maximum value of 1, which indicates the highest performance level. The mean of the Average Precision (mAP) on

multiple query images is a standard metric to indicate the overall VPR performance on a dataset [74, 73, 69]. If an application is particularly susceptible to False-Positives such as loop closure detection, a suitable metric is the highest recall at 100% Precision [90, 110, 85], denoted as R_{P100} in Fig. 2.7. $F1$ is the harmonic mean of Precision and Recall. Its maximum along the PR-Curve is used as an evaluation criterion [22, 111]:

$$F1_{Max} = \max_{\tau} \left(\frac{2P_\tau R_\tau}{P_\tau + R_\tau} \right), \tag{2.4}$$

where τ is the variable parameter to plot the curve. $F1_{Max}$ is also helpful in optimizing the Precision-Recall trade-off as a function of τ [112].

Recall@k (also termed RecallRate@k) measures how many relevant images are returned in the *top-k* results against how many relevant images exist in the entire dataset. It is considered a reliable metric to evaluate place retrieval [29, 3].

$$\textit{Recall@k} = \frac{\textit{TP}_k}{N_T} = \frac{\textit{TP}_k}{\textit{TP}_k + \textit{FN}_k}, \tag{2.5}$$

where N_T is the number of true matches in the dataset, $\textit{TP}_k$ is the count of TPs in the *top-k* retrieved images, and $\textit{FN}_k$ are the real positives discarded. Recall@k is suitable when the correct retrieved images do not have to be necessarily those with the highest similarity but need only be within the *top-k*. This is the case of a navigation system using a post-processing step to filter False-Positives and re-ranking the retrieved images to improve localization [96]. The highest possible value for this metric is 1, which occurs when $\textit{FN}_k = 0$. A second metric that is computed from the *top-k* is Precision@k (also termed PrecisionRate@k). It concerns the relevant images withing the k considered retrieved

TABLE 2.1: WELL-ESTABLISHED VPR BENCHMARK DATASETS.

Dataset	Appearance Variation		Environment
	Viewpoint	Conditional	
GardenPoints	Lateral Shift	Night-Day	Mixed
Old City	6-DOF	None	Urban
Lagout	6-DOF	None	Synthetic Urban
Corvin	6-DOF	None	Synthetic Urban
SPED-Test	None	Night-Day Weather / Seasonal Dynamic Elements	Urban Natural
RobotCar Cross-Seasons	Lateral Shift	Illumination Weather	Urban
Nordland	None	Seasonal	Synthetic Urban
Berlin Datasets	Lateral Shift 6-DOF	Illumination Dynamic Elements	Urban
Campus Loop	Lateral Shift	Seasonal	Mixed
Living Room	Lateral Shift	Illumination	Indoor
17 Places	Lateral Shift	Illumination	Indoor
Essex 3IN1	Lateral Shift	None	Mixed
Pittsburgh250K	6-DOF	Illumination	Urban
Alderley Day and Night	Lateral Shift	Illumination Weather	Urban

images:

$$\textit{Precision@}k = \frac{TP_k}{k} \ . \tag{2.6}$$

Precision@k is employed in retrieval [113]. Jointly with Recall@k, it can be used to plot a PR-Curve for a single query image by varying k. This plotting method is used in Chapter 3, which presents a VPR evaluation method based on Extended Precision (EP), a new evaluation metric proposed as a replacement for R_{P100}, which is not defined if Precision is below 1 for all Recall values. On the contrary, EP can be computed for every PR-Curve, enabling the measurement of the lower performance spectrum of a VPR technique.

Fig. 2.8. VPR dataset exemplary places.

2.1.5 Benchmark Datasets

VPR benchmark data incorporate challenges a robot typically encounters in real-world applications. A dataset should include at least two loops of the same environment to show each place in different viewing conditions to test a VPR method to detect known places regardless of their appearance. One loop is used as a reference, while the other is for testing, simulating the current view frame of a robot's camera. Table 2.1 presents a selection of VPR benchmark datasets publicly available to the research community, while Fig. 2.8 shows some exemplar images. The remainder of this section describes those benchmark datasets.

The GardenPoints dataset [8] includes three loops recorded walking through the Queensland University of Technology, Australia. Two loops are captured during the day on the opposite side of the walking path and have a lateral shift viewpoint. The third one is captured on the right side at night to assess tolerance to illumination changes.

Lagout and Corvin are synthetic datasets having multiple loops captured at different height angles to obtain several magnitudes of 6-DOF viewpoint variations. Lagout has

three loops at 0°, 30°, and 45° while Corvin has only two loops recorded at 30° and 45°. They were introduced in [4] to simulate unmanned aerial vehicle navigation along with the Old City dataset, which consists of two long loops recorded in Zurich, Switzerland.

The Specific Places Dataset (SPED) [5] consists of images of various places in the world obtained from CCVT cameras over extended recordings. This dataset is suitable to simulate long-term robot runs as it includes weather, day-night, and seasonal changes other than dynamic elements such as cars and pedestrians. SPED dataset was introduced to train CNNs for VPR. The images excluded from the training process were organized to create the SPED-Test by the same author [114] to serve as a benchmark dataset.

The Cross-Seasons dataset [1] is a subset of the Oxford RobotCar dataset [115]. It consists of two sequences of sunny and dusk images recorded onboard a car driving in an urban environment. This dataset has illumination variations, mild lateral viewpoint shifts, and dynamic elements such as pedestrians, cars, and shadows.

Nordland [116] has four traversals captured along a rail track in Norway in every season. The video frames have synchronized to build frame correspondences for every place. This dataset does not have any viewpoint change. It is suitable to assess VPR under extreme appearance changes (e.g., winter vs. summer loops) and perceptual aliasing (e.g., spring vs. summer loops).

Berlin is a collection of three urban datasets [2]. It has been built using the image-sharing platform Mapillary[2] to create challenging viewpoint variations between cluttered images by dynamic elements such as vehicles and people.

Campus Loop [94] has two sequences recorded inside a campus during the summer and winter. Besides the changes yielded by seasons, it presents a lateral shift between

[2]https://www.mapillary.com

images. Campus Loop is a small-scale dataset with only 100 outdoor and indoor places.

The 17-Places dataset [7] includes several indoor scenes, such as hallways, bedrooms, and labs. Another indoor dataset is Living Room [117]. It consists of high-resolution images captured from a home service robot on an exploration task. The peculiarity of this dataset is the close-to-ground perceptive of the onboard camera rather than the usual first-person viewpoint. 17-Places and Living room datasets exhibit viewpoint and illumination changes.

Essex 3IN1 [6] is a mixed dataset of indoor, outdoor, and natural scenes from the University of Essex Campus, UK. Similarly to Living Room datasets, it is obtained by simulating a robot exploring an environment rather than recording a video through a smooth or linear pathway. Such a building approach resulted in confusing images characterized by close-ups of cluttered images with a few disincentive elements suitable for image matching.

Pittsburgh [3] is a large dataset of 250K images generated from Google Street View[3] panoramas of Pittsburgh, Pennsylvania. It includes 24 perspective images for each place captured in 12 pitches and two yews directions. It is suitable to build image pairs at various 6-DOF variation intensities.

Alderley Day and Night [91] has two driving sequences in an urban environment. One is captured in the daytime, while the other is during the night in rainy weather. This dataset poses a severe challenge for VPR because it combines lateral shifts with extreme conditional changes.

[3]https://www.google.com/streetview/

2.1.6 Runtime Benchmarking

Runtime benchmarks of BNNs and CNNs are relevant to the results presented in later Chapters 5 and 6. This section provides an overview of the existing criteria to assess deep neural models.

The dominant evaluation metrics are inference time (or inference latency), power consumption, and memory usage. TANGO [118] employs those metrics to assess CNN models deployed on several hardware platforms, including NVIDIA GPUs for workstation and mobile usage and an FPGA[4] platform. The importance of energy usage is emphasized by Palit *et al.* [119], who presents EVA-DNN, a model to estimate the energy usage of deep neural networks. Results are presented for several well-established networks to validate the proposed model. DNNTune [120] uses inference time and energy consumption to tune both CNNs and quantized networks for several application setups. Howard *et al.* [121] proposed a class of efficient networks for mobile applications using *classification accuracy to parameters* as a tuning criterion to train models for various scenarios. Bianco *et al.* [122] employed the ratio between accuracy and the number of parameters to measure the efficiency of a model in using memory to accomplish its task. This thesis develops this idea around binary networks. BNNs' memory footprint is usually evaluated only relatively to full-precision counterparts disjointly from their performance [123, 124]. Chapter 5 extends the accuracy-parameter trade-off analysis by Bianco to binary networks proposing a metric to assess the memory allocation efficiency of BNNs.

[4]Field-Programmable Gate Array

2.2 Efficient VPR: from Design Solutions to Binary Neural Networks

The increasing utilization of visual-based applications on small robots is pressing toward increasing the efficiency of deep neural networks. This problem is also relevant for VPR as it is often carried out onboard mobile robots where the computational power is often limited for practical reasons such as battery saving or payload limitations (e.g., UAV) [4, 52]. As discussed above and shown later in Chapter 4, CNNs effectively address VPR in dynamic environments. However, they include many parameters (i.e., weights) that result in large model sizes and heavy computational efforts. There are several approaches for reducing the resource requirements of CNNs. Some address the network architecture to decrease the number of parameters and operations required to complete an inference; others compress models post-training by pruning redundant connections or by quantizing a model up to the extreme of binarization, 1 bit per parameter. Binary Neural Networks (BNNs) are a class of compact and efficient deep neural networks that reached maturity a few years ago and are now supported by several machine learning frameworks. This section presents a selection of methodologies to improve the efficiency of neural networks with a focus on BNNs.

2.2.1 Design-Based Approaches

Various solutions have been proposed to make neural networks more efficient while aiming at the minimum impacts on performance. One immediate solution is reducing the number of parameters using shallow networks. A shallow network uses only one hidden layer and can virtually approximate any function [125, 126]. However, in practice, shal-

low networks need more computation than deep neural networks to achieve comparable performance in complex visual tasks [127]. Dauphin *et al.* [128] trained a shallow network on SIFT features to cope with the ImageNet challenge [129], reporting difficulties in training as the number of parameters increased. Another option is to rethink a deep network's architecture. SqueezeNet [130] uses 1×1 convolutional kernels instead of the more expensive 3×3 commonly used in CNNs [93, 131]. A further approach is the depthwise separable factorization [132, 133] to split a convolution into two more lightweight stages: a depthwise convolution and a pointwise convolution, which uses efficient 1×1 kernels. Depthwise separable factorization is discussed more deeply later in Chapter 6, which shows how to combine it with binarization to improve the computational efficiency of BNNs.

2.2.2 Post-Training Processing

A trained model can be made more compact and computationally efficient by targeting redundant and non-informative parameters. Weight decay [134] was one of the earliest attempts at network pruning proposed back in 1989. Optimal Brain Damage [135] and Optimal Brain Surgeon [136] decrease the number of connections using the Hessian of the loss function. Han *et al.*, [137] showed how to reduce the number of parameters by one order of magnitude in several state-of-the-art networks by redundant pruning weights. [138] targets hidden neuron activations that are replaced with a more efficient approximation function. The idea of approximating operations on parameters with lighter functions is also exploited in [139]. A matrix decomposition is applied to weight tensors to find an optimal and lightweight approximation to speed up the inference. Combining pruning with weight quantization and data compression enables significant

model size reductions. Han *et al.* [140] proposed a pipeline to combine these three techniques to achieve a model size compression rate of $49\times$ and a computational speed-up of $4\times$ for large networks such as VGG-16 [93] without any significant accuracy loss.

2.2.3 Parameter Quantization and Binarization

All the techniques above have in common a relatively complex process that is only sometimes straightforward to implement. In the author's opinion, it has been an obstacle to their diffusion and implementation in standard software frameworks and libraries. In contrast, parameter quantization appears simpler to apply to a pre-trained model. Nowadays, several machine learning frameworks, such as Pytorch[5] [141] and Tensorflow[6] [142], offer a standard procedure to quantize a model post-training. The purpose of quantization is to reduce the number of bits representing weights and activations[7]. Quantization can be pushed to binarization, 1-bit precision, to enable maximum compression and computational speed. A full-precision network uses 32/64-bit weights, floating-point multiplications, and summations to compute an output. A quantized network uses fewer bits and works with fixed point calculation, which is expected to be more computationally efficient if properly implemented and supported by the hardware. It is worth mentioning that modern GPUs and CPUs are highly optimized for floating-point operations, so the multiply-accumulate operations (MACs) necessary for neural networks are computed in a single instruction. The same support is not available for fixed-point and bitwise operations that require multiple instructions to compute a MAC [143, 144]. This topic is fundamental for exploiting the full potential of quantized and

[5]https://pytorch.org/docs/stable/quantization.html
[6]https://www.tensorflow.org/lite/guide
[7]Activations are the output of the previous layers that are taken as inputs by a neuron.

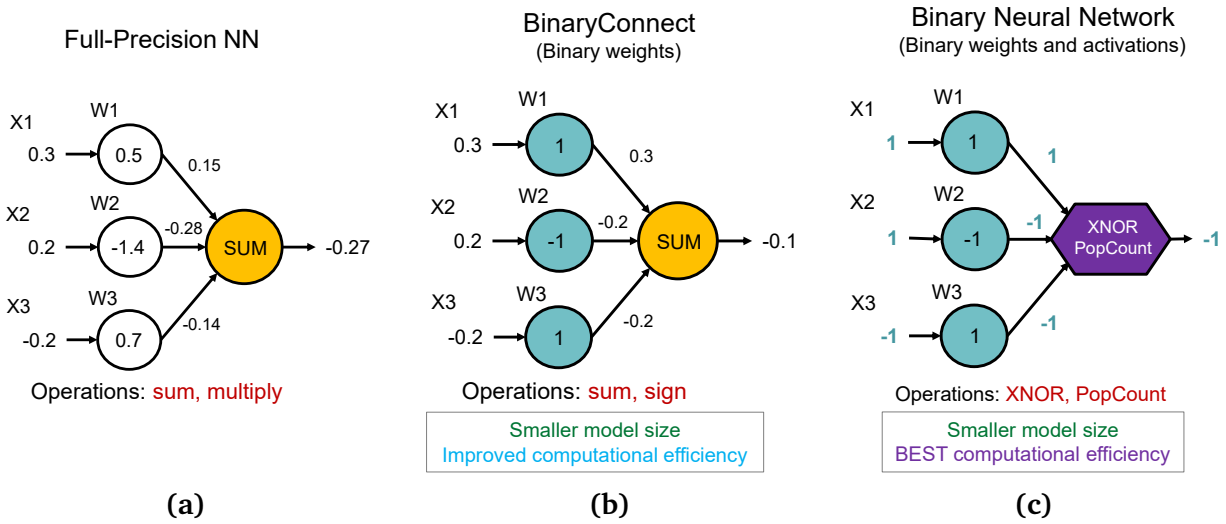


Fig. 2.9. Computation in a standard neural network (a), in BinaryConnect (b), and in a Binary Neural Network (c).

binarized networks. It will be more discussed later in Section 2.2.4 and Chapter 5.

Quantization can be applied after or during the training phase. While post-training quantization with 8 or more bits works reasonably well in practical cases [145], binarization dramatically affects a model's performance [146]. In contrast, Binary-aware training enables low-precision models with acceptable classification accuracy [147]. Although training binary models from scratch was attempted decades ago [148], only recently have gradient-based techniques become applicable. Courbariaux *et al.* trained a binary-weight network through back-propagation, BinaryConnect [149], using Straight-Through-Estimator (STE) [150]. The key idea of STE is to keep real-valued weights in memory, binarize them only in the forward pass to compute neurons' activation, and update them during back-propagation as in a standard neural network. However, BinaryConnect retains full-precision neurons' activations. This eliminates the need for floating-point multiplications in convolutions, but summations (accumulation) cannot be avoided. Later the same authors applied STE to train a fully binary network [25].

With both weights and activations binarized, convolutions can use bit-wise arithmetic to boost the computation speed-up by one order of magnitude [151]. Fig. 2.9 compares a neuron computation in a standard network, BinaryConnect, and a Binary Neural Network. The main perk of BinaryConnect is the model size reduction compared to a full-precision network: 1 bit per weight instead of 32/64 bits. The computational speed is also improved as the faster *sign* operation can replace multiplication if the binary weights are encoded as $\{1, -1\}$. BNNs achieve an even higher speed by replacing multiplications and summations with XNOR and Popcount on multiple operands into a single register [25]. BNNs implementation and computation are extensively described later in Chapter 5, which presents a binary network optimized for VPR.

Afterward, several additions to the field were proposed to improve BNNs. In XNOR-Net [123], the convolutional blocks are rearranged to increase classification accuracy. Batch-Normalization (BatchNorm) [152], a fundamental layer for BNNs [153, 154], is usually placed between the activation function and the successive convolution. XNOR-Net's authors observed that binarized feature maps do not retain any information on the activation's magnitude, causing the pooling layer not to be able to reliably select the most significant features to feed the successive layer of the network. Based on this observation, XNOR-Net is designed with BatchNorm and binary activation preceding convolution so that pooling occurs before binarization. DoReFa-Net [155] is a variable precision network that exploits bitwise operations to efficiently compute the dot product between a layer's weights and the inputs to speed up training. DoReFa-Net is trainable as a binary or at different quantization levels. The work presented in [156] points out that the commonly used norm-based regularizers for CNNs, such as L_2, are unsuitable for binary networks as they tend to keep the weight values low, resulting in many zeroes once they

are binarized. Hence, the authors proposed a weight regularizer specific to BNNs, which considers the encoding of the weights to avoid pushing their value flat to 0. [156] also improves the compactness of BNNs by using a learned scale layer to help the binary-wise training of the fully connected layers, which are often kept in full-precision to improve the accuracy in classification tasks [155]. Accurate-Binary-Convolutional-Net (ABC-Net) [157] trains multiple sets of binary weights using different bases that are combined to approximate full-precision weights and activations using coefficients that are also learned. While ABC-Net significantly narrows the performance gap with full-precision networks, it has a more complex architecture than regular BNNs, potentially resulting in slightly lower computational and memory efficiency. Before this thesis, BNNs were used only for different tasks than VPR. This work aims to contribute to the field by proposing a class of highly compact binary networks to solve the VPR problem effectively in changing environments. The implementation details and the results obtained are given in later Chapter 5.

Finally, although the research presented in this manuscript focuses on BNNs, several significant works proposing quantized networks are worth mentioning. Ternary networks [158, 159] use three values to encode weights: $\{-1, 0, 1\}$. They are considered a suitable alternative to BNNs when higher accuracy is necessary. Although they exhibit a significant memory reduction and simpler arithmetic than regular CNNs, ternary networks require 2 bits to store parameters without outperforming state-of-the-art BNNs by a wide margin [147]. Esser *et al.* [160] proposed learning the quantization thresholds to shorten the accuracy gap with full-precision classifiers on ImageNet [87]. The proposed learning approach applies to 2, 3, 4, and 8-bit quantized networks but not to binary ones. This approach is further developed by Bhalgat *et al.* [161], achieving higher accuracy on

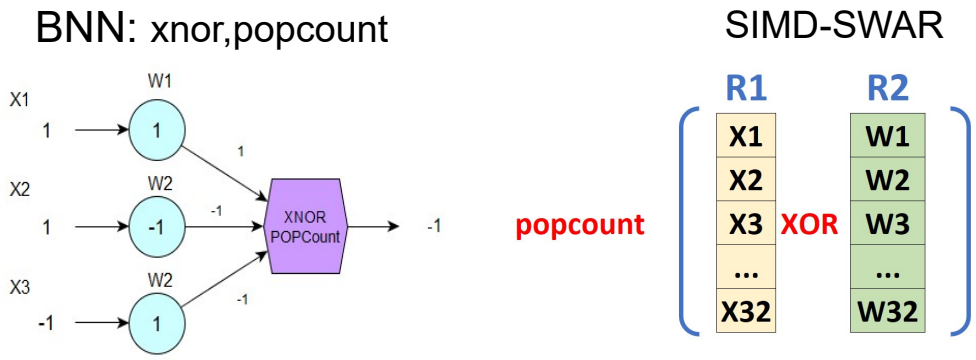


Fig. 2.10. Single Instruction, Multiple Data (SIMD) Within A Register (SWAR)

ImageNet.

2.2.4 Compute Engines for Binary Neural Networks

Binarization reduces a model's size dramatically and enables efficient convolution computation. However, BNNs can only express their full potential with an inference engine suitable to compute bitwise operations. Hardware architectures implement binary primitives, such as XNOR and pop-count, differently. Therefore, inference libraries and compute engines typically target one or a few hardware platforms to guarantee that the deployed models can run efficiently. This section presents a selection of the most relevant libraries and tools for deploying BNNs.

Courbariaux *et al.* [25] released a CUDA[8] [162, 143] kernel-based that speeds up convolutions by seven times using SIMD (single instruction, multiple data) [163] within a register (SWAR) [151] technique (Fig. 2.10). DaBNN [164] is a stand-alone library to deploy BNNs on ARM platforms. Binary convolutions are computed by combining im2col (Image to Column) transformation and GEMM (GEneral Matrix Multiplication). DaBNN uses ad hoc implementation written in ARM assembly that enables $8-10\times$ speed-

[8]Compute Unified Device Architecture

up compared to a full-precision implementation. BMXNet [165] extends MXNet [166], providing a complete ecosystem to train and deploy BNNs. Like DaBNN, it computes binary convolutions by combining im2col with GEMM. BMXNet utilizes standard C++ to implement binary operations reaching a $13\times$ speed-up compared to floating-point convolutions written with the CBLAS library [167]. Riptide [168] is another end-to-end framework to train and deploy BNNs, focusing on integrating binary convolutions with those layers that cannot be binarized, such as BatchNorm [152] and activation functions [169]. Models are trained with Tensorflow [170] and compiled for deployment with TVM [171]. A binary model compiled with Riptide is $4\times$ to $12\times$ faster than its full-precision counterpart. Likewise, Larq Compute Engine (LCE) [172] aims to integrate efficiently binary convolutions with a model's full-precision components. LCE is part of the Larq project [173] to train and deploy BNNs. It uses a binary kernel highly optimized for ARM platforms to speed-up convolution by 8.5 to 18.5 times. The Larq framework is used in this thesis work to train all the binary models and to perform benchmarks on real hardware. Finally, FINN [174] is an experimental tool targeting FPGAs compatible with PYNQ [175]. FINN does not include a training framework but can compile PyTorch [141] models optimized with Bravitas [176].

2.3 Summary

The literature survey presented in this chapter provides an overview of the areas concerning the research presented in this thesis. It is organized into two main parts. Section 2.1 covers the main research domain of VPR, describing several approaches to image matching and VPR evaluation methodology. The second part is centred around the problem of

solving the VPR problem efficiently.

The VPR survey introduces local features (Section 2.1.1) and CNN-based descriptors (Section 2.1.2). Those two kinds of approaches are compared in addressing VPR under 6-DOF viewpoint and conditional changes in Chapter 4. Relevant VPR performance evaluation metrics are detailed in Section 2.1.4 to provide a background for the new metric and methodology to assess VPR performance described in the next chapter. The resource utilization at runtime is covered in Section 2.1.6 to provide the necessary background to support the memory allocation metric presented in Chapter 5. As stated in Section 1.3, one of the most significant contributions of this thesis aims to improve VPR efficiency to enable it on low-end hardware platforms. Therefore, the second part of the survey (Section 2.2) describes the principal approaches to address the efficiency problem in neural networks, focusing on Binary Neural Networks (BNNs), which are extensively used in this research. Specifically, Chapters 5 and 6 present lightweight BNNs trained to address VPR in changing environments whose runtime requirements are an order of magnitude smaller than regular CNNs. The last section of this survey, Section 2.2.4, is dedicated to BNNs deployment on embedded platforms to provide the necessary background to interpret the analysis of the computational and energy usage of the proposed BNNs for VPR applications.

Chapter 3

A Generic Evaluation Framework for Visual Place Recognition[1]

Chapter 1 highlighted the importance of having a standard methodology to assess and compare different VPR techniques so that it is possible establishing the most suitable one for a given application or identify the actual performance difference between two different approaches. The evaluation framework presented in this chapter is built around Extended Precision (EP), a new performance metric, and has the two-fold goal of addressing the problem of evaluating VPR reliably and providing a uniform way to present the results across the entire manuscript. In summary, the proposed framework consists of several steps. In the preliminary one, the framework assigns each place of an environment with an EP score. Then, these scores are further elaborated in two successive stages. The first is an assessment over an entire traversal of the working space finding the maximum and minimum guarantee performance level to determine the consistency

[1]This work is published in IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 1688-1695, April 2020, and presented at ICRA-2020 in Paris, France, with tile of: "Exploring Performance Bounds of Visual Place Recognition Using Extended Precision". DOI: 10.1109/LRA.2020.2969197.

of a VPR technique in a working environment. The second identifies statistically relevant performance differences between VPR methods. The utilization of this evaluation methodology is demonstrated by presenting results for several state-of-the-art techniques operating under various imaging conditions from five well-established benchmark datasets.

3.1 The Need for VPR Evaluation

Chapter 1 presents VPR as a fundamental but highly challenging task within autonomous robotics. It has been subject to significant advancements in recent times regarding existing algorithms and new approaches [177, 178, 16, 11]. With the significant addition of VPR techniques, an inquiry that rises in importance is the evaluation of performance differences between these methods. Indeed, each approach is usually assessed by its application contexts and experimental setup, making cross-paper comparison difficult [22]. Moreover, these assessments are mostly based on the sole PR-Curves or PR-Curve-derived metrics to express the overall performance of a technique without providing further analysis and insight into VPR performance [4, 90, 74, 179, 180]. Ehsan *et al.* [23] present a performance comparison made for evaluating the limitations of image feature detectors utilizing repeatability measures [181]. Nevertheless, it highlights the importance of in-depth analysis and statistically reliable comparisons between VPR techniques.

This chapter presents a new performance metric denoted as Extended Precision (EP) and an evaluation framework that aims to tackle the potentially overlooked features in previous VPR performance comparisons. EP is obtained by combining several features of a PR-Curve into a scalar value, which is used within the evaluation framework to

measure VPR performance and carry out statistical tests. The evaluation framework has a preliminary step consisting of a place-by-place VPR technique assessment where an EP score is assigned to each query image of the current traversal. Then, these scores are further elaborated in two successive steps. The first explores the upper and lower performance bounds of VPR techniques across an environment to assess the accuracy and consistency of the image matching across the entire traversal. The second identifies statistically relevant performance differences between VPR methods utilizing a variant of McNemar's test [23]. The proposed framework is demonstrated through the evaluation of several VPR techniques across different datasets, each presenting different types of environmental changes.

The reminder of this chapter is organized as follows. Section 3.2 provides an introduction to PR-Curves. Section 3.3 introduces Extended Precision and the evaluation framework. The experimental results are presented and discussed in Section 3.4. Finally, a summary of this chapter is given in Section 3.5.

3.2 Precision-Recall Curves: an Introduction

The evaluation framework presented here utilizes PR-Curves (Precision-Recall Curves) [108] in the preliminary step of assigning each query image with an EP score. This section introduces this important tool to evaluate VPR before describing the methodology proposed in this chapter.

A visual place recognition module operates on highly skewed data where the positive matches for a query image are rare compared to negative matches. PR-Curves are preferable with imbalanced data [107, 108], so they are frequently used to evaluate VPR [11].

TP	FP	PP (TP + FP)
FN	TN	PN (FN + TN)
RP (TP + FN)	RN (FP + TN)	N

Fig. 3.1. A binary contingency table.

VPR is a dichotomous binary matching problem where a query image (e.g. a robot's camera frame) is compared with reference images showing the places of the map. Let be N the number of those images. A VPR system regards every reference image as a positive or negative match to the query separating N into two subsets of Predicted Positives (PP) and Predicted Negatives (PN) matches. However, those predictions might be inaccurate. When compared with Real Positives (RP) and Real Negatives (RN) correspondences in the ground truth of the query, they generate four mutually exclusive cases that can be organized in a binary contingency table, as shown in Fig. 3.1. Those cases are:

- True Positives (TP): Real Positives are predicted as positives;
- False Positives (FP): Real Negatives are erroneously predicted as positives;
- True Negatives (TN): Real Negatives are predicted as negatives;
- False Negatives (FN): Real Positives are erroneously predicted as negatives.

A binary contingency table comprehensively describes the results of image matching. The four cases mentioned above are related to PP, PN, RP and RN. PP is the set of reference images the VPR system believes to be correct correspondences to the query. As indicated in the first row in Fig. 3.1, the PP set includes both TP and FP. Following the same reasoning, PN is the sum of FN and TN. The columns of a contingency table are related to the ground truth, namely to real positive and negative matches to the query. An RP

can be either a TP, a real positive match correctly predicted by the VPR module, or an FN, a real positive match erroneously classified as a non-matching image. Therefore, RP is given by the sum of TP and FN. Similarly, RN $=$ FP $+$ TN. A perfect VPR technique correctly labels all the N reference images, obtaining FN $=$ FP $= 0$, which means that PP $=$ RP and PN $=$ RN.

Precision (P) and Recall (R) are computed from the binary contingency table. Precision is the ratio between the correct matches and the total of the Predicted Positive matches. Recall is the ratio between the correct matches and the total of Real Positive matches. Formally:

$$P = \frac{TP}{PP} = \frac{TP}{TP + FP} \,, \tag{3.1}$$

$$R = \frac{TP}{RP} = \frac{TP}{TP + FN} \,. \tag{3.2}$$

Precision and Recall are often in tension: improving one worsens the other. Privileging one of those two metrics depends on the application context. For example, an FN might have disastrous consequences for medical images and document retrieval for diagnostic purposes. Hence, several FPs might be tolerated to retrieve as many relevant results as possible, aiming at the highest Recall [108]. On the contrary, for VPR and LCD in general, Precision is more critical than Recall, as a single FP might cause the localization system fails [90]. A PR-Curve shows the relationship between Precision and Recall, helping to find the best trade-off for an application and evaluating the VPR performance. A perfectly skilled VPR technique always correctly matches places resulting in a constant PR-Curve having Precision equal to 1 for all Recall values. The P-R pairs necessary to plot a PR-Curve are obtained in several ways, depending on the application or characteristics of the VPR technique to assess. Some examples are varying an algorithm's parameters

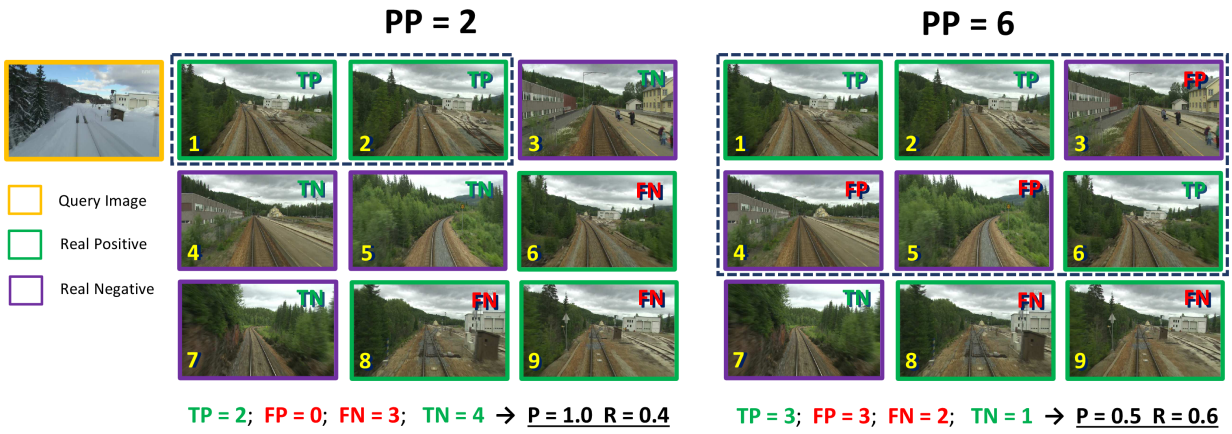


Fig. 3.2. Precision and Recall computed for two cut-offs of the query results: PP $= 2$ and PP $= 6$.

[91], the score threshold to call a positive match [28], or the cut-off number of the top retrieved images by similarity[2] with the query [109]. The latter is the approach used in the proposed evaluation framework, as it makes it possible to plot a PR-Curve for every query image. In detail, let q be a query image to perform VPR (e.g., robot's camera frame). The image base to search for know places is the map, I_M, including N images. Each place is associated with one or more images, that is, the number of real positives matches (RP) in I_M for the query image. For q, the VPR technique returns a cut-off of the top PP images from I_M by similarity to the query. The number of retrieved images classified as positive matches, PP, is used a variable parameter from 1 to RP. This process allows the construction of multiple binary contingency tables (Fig. 3.1) and the related P-R pairs to plot a PR-Curve, with each PP value representing a point on the curve. Fig. 3.2 show an example of this process for two cut-offs: 2 and 6 images. For PP $= 2$, P is at the maximum but only two out of five RPs are retrieved for the query image. Increasing PP to 6, a third Real Positive is included increasing R from 0.4 to 0.6. However, some FPs are also added, reducing P from 1.0 to 0.5. It is relevant noting that if the top-1 image

[2]The meaning of *similarity* between images is covered in Section 3.3.3. It concerns the nature of a VPR technique and the comparison method between image representations.

(the one retrieved for PP $= 1$) is a false positive, P cannot reach 1 for any recall value. As previously noted, the main advantage of this approach is that a PR-Curve is computed for each query image with the two-fold purpose of enabling place-by-place assessment and further characterization of a VPR method's performance across the entire environment, as proposed in Sections 3.3.3 and 3.3.4.

3.3 Evaluation Framework

The proposed evaluation framework consists of a preliminary phase of assessing a VPR technique in every query image of the current traversal. The resulting data is further processed in two successive analysis steps. One determines the upper and lower performance bounds of a VPR method allowing for determining the performance consistency across an operating environment. The second phase is designed to compare VPR methods. Comments [182] suggest that many evaluation approaches emphasize beating the latest benchmark numbers without considering whether the improvement of the vision system over the others is statistically significant. This consideration can be extended to VPR evaluation, where most methods have confined themselves to some particular test conditions to demonstrate their superiority over other competing techniques. Driven by these motivations, the second phase of the evaluation framework uses a variant of McNemar's test [23] to determine whether the performance differences are statistically significant or occurred by chance because of random factors. The proposed evaluation framework uses the new EP metric, which is motivated and detailed first, as in the below section.

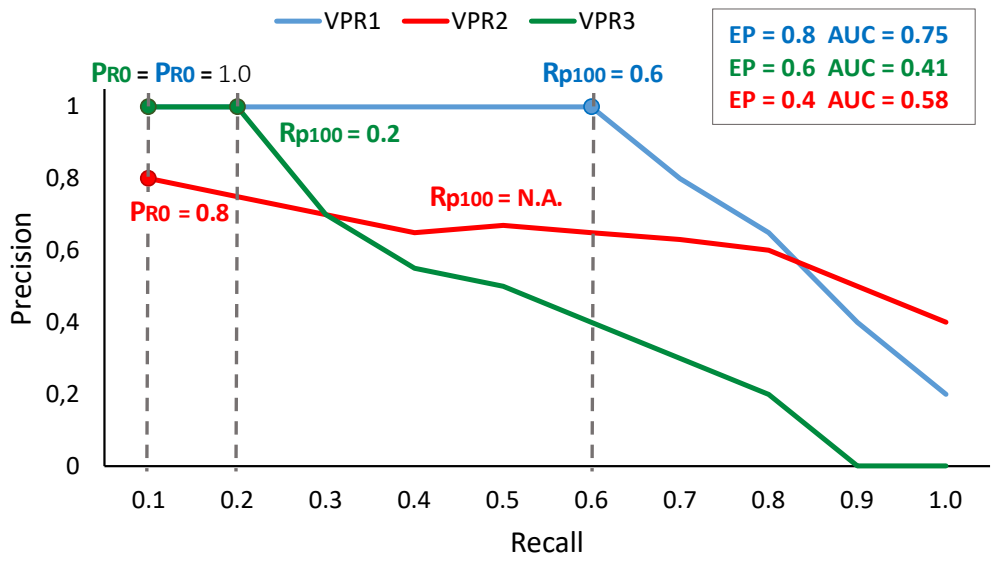


Fig. 3.3. An example of comparison among three hypothetical VPR techniques.

3.3.1 Extended Precision

As noted in Section 2.1.4, AUC [107] is often used to evaluate VPR [4, 72, 94, 101]. It expresses VPR performance with a value between 0 and 1. However, AUC does not retain any information regarding the features of the original PR-Curve, including whether Precision reaches or not 1 at any Recall value. R_{P100} [90] is also an important performance metric; it corresponds to the Recall value until Precision keeps the value of 1, which is the highest Recall value that can be reached without any FP (Section 3.2). As a single FP may cause severe failures for many robotic applications [90, 183], R_{P100} is considered a good performance metric and is widely employed for VPR evaluation [90, 110, 85]. However, R_{P100} cannot determine the lower performance bounds of a VRP method in every case, as it cannot be computed for those PR-Curves that never reach $\mathrm{P} = 1$. To circumvent this problem, the author introduces Extended Precision:

$$\mathrm{EP} = \frac{P_{R0} + R_{P100}}{2}\,, \tag{3.3}$$

where P_{R0} denotes the precision at the minimum Recall value (R_0) and the factor '2' in the denominator is to have EP $\in [0, 1]$. When $P_{R0} < 1$, R_{P100} is set to 0, and EP depends only on the Precision at minimum Recall (R_0), while for $P_{R0} = 1$, EP is greater than 0.5 and assumes the same meaning as R_{P100}.Fig. 3.3 shows three examples of PR-Curves to illustrate the use of EP and highlight the differences with AUC. VPR1 and VPR2 are similar curves having $P_{R0} = 1$ and differing only in R_{P100}, which is larger for VPR1. In this case, both EP and AUC identify VPR1 as the best. VPR3 represents a different case. It has $P_{R0} < 1$ and R_{P100} is not defined, indicating false positives in the retrieved images (Section 3.2). When VPR3 is compared against VPR2 using EP, the latter is identified as the best, scoring 0.6 against 0.4. Conversely, accordingly to the AUC metric, VPR3 outperforms VPR2. This difference arises from the strongest penalization given by EP to false positives compared to AUC. As noted in Section 3.2, if the top-1 retrieved image is a false positive, P_{R0} cannot reach 1, capping EP to 0.5 . Such a penalization is not present in AUC, which can reach relatively large values regardless of false positives in the top ranking of retrieved mages. The differences between EP and AUC are further discussed in Section 3.4.3 when utilized within the proposed evaluation framework.

3.3.2 Computing EP Scores

The preliminary phase of collecting EP scores across the working space is detailed in this section. The process requires two sets of images: a reference dataset (I_M) representing the previously visited places and a query dataset (I_Q) representing the current traversal of the environment. Algorithm 1 shows the procedure of computing an EP score set (E_v) for a VPR technique (v) on a benchmark dataset (I_M,I_Q). The inner loop (lines from 5 to 9) computes the similarity scores between image descriptors. The framework does

Algorithm 1 EP scores computation for a VPR technique.

Input: v # VPR method to evaluate
Input: I_M # reference dataset
Input: I_Q # query dataset image
Input: GT # Ground Truth for q
Output: E_v # The set of EP scores for v

$E_v := [\,]$
for $\forall q_j \in I_Q$ **do**
  $D_{q_j} \leftarrow v(q_j)$ # query image descriptor
  $S_{q_j} := [\,]$ # similarity scores set
  for $\forall r_i \in I_M$ **do**
    $D_{r_i} \leftarrow v(r_i)$ # reference image descriptor
    $s_{ji} \leftarrow \texttt{similarity}(D_{q_j}, D_{r_i})$ # e.g., cosine similarity, Eq. 3.4
    $S_{q_j} \leftarrow \texttt{append}(s_{r_i})$
  end for
  $\widehat{I_M} \leftarrow \texttt{sort\_descending}(S_{q_j}, I_M)$ # sort I_M by similarity
  $PR \leftarrow \texttt{COMPUTE\_PR\_CURVE}(\widehat{I_M}, GT)$
  $EP_j \leftarrow \texttt{compute\_EP}(PR)$ # Extended Precision, Eq. 3.3
  $E_v \leftarrow \texttt{append}(EP_j)$
end for
return E_v

function $\texttt{COMPUTE\_PR\_CURVE}(\widehat{I_M}, GT)$
  $PR := [\,]$ # the PR-Curve's points
  $RP \leftarrow |GT|$ # Number or Real Positives for q
  for $i \in 1 : RP$ **do**
    $PP \leftarrow \texttt{get\_top}(\widehat{I_M}, i)$ # top i ref. images by similarity
    $TP, FP, FN \leftarrow \texttt{conf\_matrix}(PP, GT)$
    $P \leftarrow \texttt{precision}(TP, FP)$ # Precision, Eq. 3.1
    $R \leftarrow \texttt{recall}(TP, FN)$ # Recall, Eq. 3.2
    $PR \leftarrow \texttt{append}([P, R])$
  end for
  return PR
end function

not have any requirements about similarity, except it must be sortable to build the image raking required by the PR-Curve construction method detailed in Section 3.2. A well-established similarity score is the cosine product between image descriptors [184, 2], and it will be used throughout this chapter's experiments unless specified differently. The cosine similarity is computed as follows:

$$s_{ji} = \frac{D_{q_j} D_{r_i}}{||D_{q_j}|| \, ||D_{r_i}||} , \tag{3.4}$$

where the vectors D_{q_j} and D_{r_i} are the image descriptors computed by v for the query and reference image, respectively. Similarity scores are computed between q_j and all $r_i \in I_M$. The resulting set of values is used to sort the map images, I_M, by similarity with the query in decreasing order to form the set $\widehat{I_M}$ (line 10). Then, $\widehat{I_M}$ is used jointly with the ground truth (GT) information to compute the PR-Curve for q_j (line 11) . The loop is repeated for all $q_j \in I_Q$ so that E_v includes a score for all query images:

$$E_v = \{EP_1, EP_2, \ldots\ldots EP_n\}. \tag{3.5}$$

E_v has a performance measurements for every place of the environment enabling further analysis to characterize or summarize a VPR method performance.

3.3.3 VPR Performance Bounds and Overall Performance Measurement

The upper and lower performance bound for v on the dataset I_Q correspond to the highest and lowest EP values in E_v, respectively:

$$EP_{Max} = max(E_v) \ , \tag{3.6}$$

$$EP_{min} = min(E_v) \ . \tag{3.7}$$

EP_{min} is indicative of then minimum performance level of v in the operating environment [185].

A single FP might severely impact robotic applications, especially when VPR is used directly for loop-close detection [90] or the localization module relies on a single image to work. To this end, the share of place successfully recognized with a single image retrieved is a relevant measurement to provide along with performance bounds. Per Eq. 3.3 and PR-Curve building process, $EP \geq 0.5$ indicates that the top-1 retrieved image is a TP. Then, the share of places successfully recognized with a single image retrieved is as follows:

$$S_{P100} = \frac{|\{q_j \in I_Q | EP \geq 0.5\}|}{|I_Q|} \ . \tag{3.8}$$

This evaluation framework uses S_{P100} as a global performance metric on a dataset.

3.3.4 Identification of Statistically Significant Performance Differences

While the previous routine is designed to assess the performance level and consistency of a VPR system, this one is focused on comparing different VPR techniques. In particular, this step determines if the performance differences between VPR techniques are statistically significant or due to random data artifacts. Following the approach described in [23], the proposed evaluation method interprets the process of testing VPR against a sequence of query images as a series of success/failure trails on the same dataset. Under this assumption, the resulting distribution follows a binomial model, and the comparison between the two algorithms (v and w) can be addressed using the McNemar test [186, 187] with continuity correction [188]:

$$X^2 = \frac{(|N_{sf} - N_{fs}| - 1)^2}{N_{sf} + N_{fs}}, \tag{3.9}$$

where N_{sf} denotes the number of trials where the algorithm v succeeded, and w failed; N_{fs} denotes the number of trials where v failed and w succeeded. X^2 is distributed, to a good approximation, as chi-squared with one degree of freedom, χ_1^2. Under the assumption of $N_{sf} + N_{fs} \geq 10$, McNemar's test distribution results from the squared samples of a normal distribution [189]. Therefore, Eq. 3.9 can be rewritten as:

$$Z = \frac{|N_{sf} - N_{fs}| - 1}{\sqrt{N_{sf} + N_{fs}}}, \tag{3.10}$$

where the confidence interval associated with Z can be determined using tables [190]. In particular, 95% significance level corresponds to $Z = 1.96$ in a two-tails table. Therefore,

if the Z value is larger than 1.96, one can say the comparison result occurred by chance as a consequence of random factors or artifacts in data only one in twenty ($p = 0.05$). McNemar's test cannot be used to compare more than two VPR methods simultaneously, so a series of independent pairwise tests are necessary to compare multiple methods.

McNemar's test requires determining when a VPR method fails or succeeds. In the proposed framework, success occurs when EP is greater than a threshold t; otherwise, it is a failure. EP is characterized by two intervals: 0 to 0.5, where the value depends only on P_{R0}, and 0.5 to 1, where EP mimics the behavior of R_{P100}. This characteristic of EP allows for comparing VPR methods from different perspectives using multiple thresholds. For thresholds below 0.5, the outcome depends only on P_{R0}. Therefore, the VPR method retrieving the lower number of FPs before the first TP is best. Conversely, with thresholds from 0.5 and above, successes and failures are determined by R_{P100}, namely by the length of the TP sequence before the first FP occurrence in the retrieved images.

Let v and w be two methods to compare. First, the EP scores for both of them are computed as described above in Section 3.3.2. Then a threshold set T is defined:

$$T = \{t_1, t_2, \ldots, t_p\}, \quad \forall i = 1, 2, \ldots, p \tag{3.11}$$

For a pair of VPR methods to compare, a set of Z scores is computed using Eq. 3.10 for all $t_i \in T$.

$$Z_{vw} = \{Z_1, Z_2, \ldots, Z_p\} \tag{3.12}$$

where Z_i denotes the value of Z obtained with the i^{th} threshold in T. Although there is no specific selection criterion for T, a good practice is to choose the threshold values to capture the entire spectrum of variations of the performance metric [23]. A good setup

Fig. 3.4. A sample of the datasets used for the experiment. The reference images are in the top row, and the query images are in the bottom row.

is with nine evenly spaced thresholds between 0.1 and 0.9, as suggested in Section 3.4.

3.4 Evaluation Framework Demonstration

The author demonstrates the use of the proposed evaluation framework by comparing several state-of-the-art VPR techniques: AMOSNet [5], HybridNet [5], R-MAC [96], NetVLAD [71], and Cross-Region-Bow [2]. The experiments were conducted using the VPR methods with the default configuration as shared by their authors. A trained model on SPED dataset [5] is available for AMOSNet and HybridNet at [191]. The second-last fully-connected layer of the network computes the image descriptors. R-MAC implementation is available at [192]. For a fair comparison, the geometric verification module has been deactivated for the tests. The MATLAB source of NetVLAD is available from [193]. The results presented in this section are obtained using the VGG-16 [93] model trained with the Pittsburgh 250K dataset [194] using a dictionary of 64 words. Cross-Region-Bow is also available as a MATLAB implementation [195]. For the experiments, the VGG-16 model pre-trained on the ImageNet dataset [87] has been utilized with a

TABLE 3.1: BENCHMARK DATASET. APPEARANCE VARIATIONS AND GROUND TRUTH TOLERANCE.

Dataset	Appearance Variation		Reference Images	Query Images	Ground Truth
	Viewpoint	Condition			
Berlin Halen. Strasse	moderate lateral shift	Dynamic Elements	154	66	25m by GPS
Lagout 0-15	Mild 6-DOF	None	330	324	by authors
Corvin 0-45	Wide 6-DOF	None	1179	1298	by authors
GardenPoints (Day-Night Right)	Mild to moderate lateral shift	Night-Day	200	200	± 2 frames
Norland (Summer-Winter)	None	Seasons	1622	1622	± 5 frames

BoW dictionary of 10K words. is a region-based approach that does not produce a global image descriptor suitable for being used with cosine similarity (Eq. 3.4). Thus, the experiments use its built-in similarity score instead.

In real-world applications, the look of a place captured by a robot's camera varies between different traversals due to environmental changes or because the robot enters a place from another direction. VPR methods are assessed under different appearance variations to provide comprehensive results using the five datasets shown in Fig. 3.4 and summarized in Table 3.1. Berlin Halensee Strasse [2] includes two traversals of an urban environment. This dataset exhibits mild to moderate viewpoint variations and includes dynamic elements such as cars and pedestrians. The ground truth is obtained using GPS coordinates to build place-level correspondence using a maximum distance of 25 meters as a criterion. The image set berlin-halenseestrasse-1 is used as a reference and berlin-halenseestrasse-2 as a query traversal. Lagout and Corvin [4] are synthetic datasets consisting of several flights around buildings. Lagout loops at 0° and 15° are used as reference and query datasets to test VPR techniques under moderate viewpoint changes. Similarly, Corvin's loops captured at ground level and 45° are used to assess

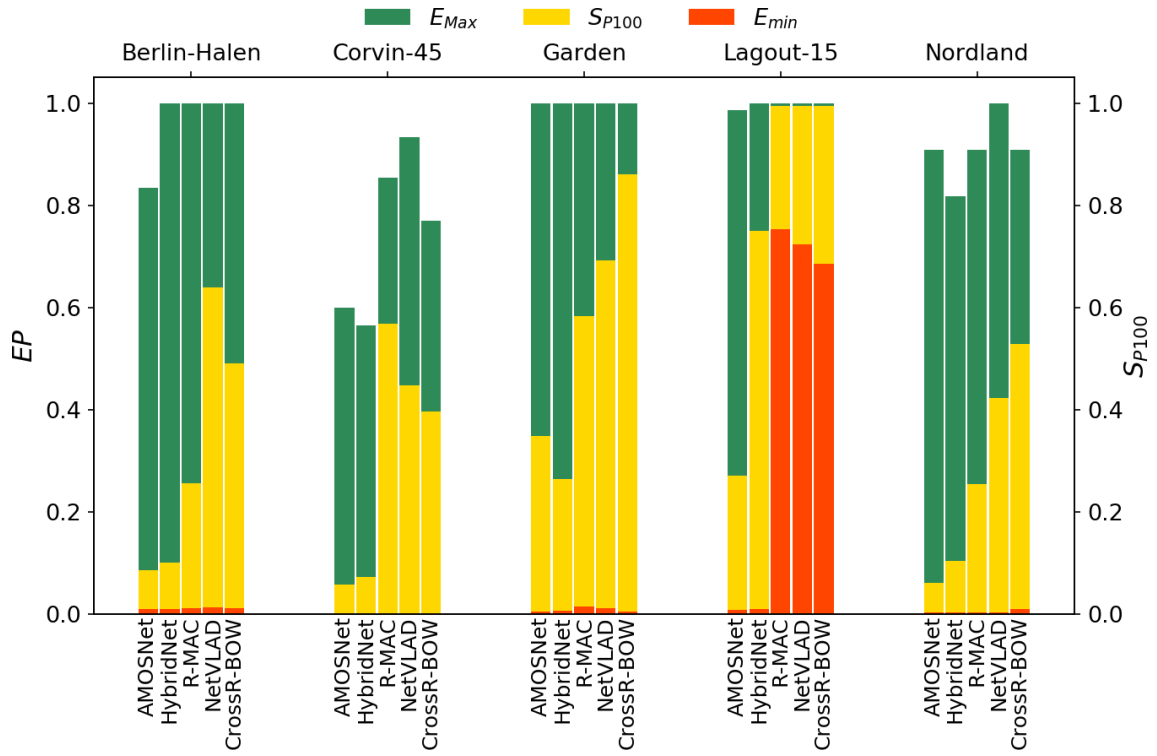


Fig. 3.5. Upper and lower performance bounds and S_{P100} for the assessed VPR methods.

VPR methods under significant 6-DOF viewpoint changes. The ground truth data for Lagout and Corvin are made available by their authors [196]. GardensPoints Dataset [8] consists of three traversals of the Queensland University of Technology (QUT). Two occurred during the daylight by walking on the two opposite sides of the walking path and the third during the night on the right side. The results are presented for illumination changes. Thus the right-day and right-night traversals are used as reference and query datasets, respectively. The footages are synchronized, so the ground truth is obtained by frame correspondences. For the test, a match will be considered correct if the query and the retrieved images are within a 5-frame range, which is considered a reasonable tolerance [110]. Hence, a map image must fall between $i - 2$ and $i + 2$ to have a TP, where i is the query image index. Nordland Dataset [116] is built from footages

for all four seasons along a railroad in Norway. It shows extreme seasonal changes, especially between summer and winter journeys, which are used as reference and query datasets to obtain the results presented here. Similarly to GardenPoints, the videos are synchronized, but the train speed is considerably faster than a human walk. Therefore, the ground-truth uses a larger tolerance of 11 frames, as suggested in [110].

3.4.1 VPR Performance Analysis

Fig. 3.5 presents the bounds and S_{P100} for the assessed VPR methods. Green bars indicate the upper-performance bounds EP_{Max}, while the lower-performance bounds EP_{min}, are represented with red bars. The values of S_{P100} are indicated in yellow and are read on the right-side y-axis.

Regarding EP_{Max}, the considered VPR methods exhibit similar performance on Berlin, GardenPoints, and Lagout with EP values equal to or close to 1. Thus, all the VPR techniques reach an excellent performance peak with dynamic objects and illumination but moderate under wide 6-DOF viewpoint changes. The prominent viewpoint variations of Corvin pose a complex challenge. None of the tested methods can reach EP $= 1$. In every Corvin's location, the assessed methods can only recover some of the Real Positives for a query image with at least one or more FPs in the result set. Nordland is also challenging as the experiments used the summer and winter traversals that exhibit prominent variations in appearance.

S_{P100} indicates the place fraction where a VPR technique successfully retrieves a TP in the top-1. From the perspective of S_{P100}, the differences between VPR methods are more significant than looking at their bounds. NetVLAD is the best approach in the urban environment of Berlin-Halensee. It is not surprising as the NetVLAD model is optimized

Fig. 3.6. Pairwise comparisons between the VPR methods considered. A sign convention is used to present the results: a positive value of Z indicates that the first method of the pair outperforms the second one, whereas a negative Z score has the opposite meaning.

for urban scenes. Cross-Region-Bow is the most reliable VPR method for illumination and seasonal changes. It scores the highest S_{P100} of 0.88 and 0.53 on GardenPoints and Nordland, respectively. Considering that Cross-Region-Bow uses a pre-trained network on ImageNet which is not prominent in any specific image transformation, its good performance is likely due region-based pooling stage. Corvin is confirmed to be the most challenging dataset. The only technique that can hit at least $S_{P100} = 0.5$ is R-MAC result-

ing in the most effective VPR method on this dataset.

The lowest performance bound is close to zero in most of the tested scenarios. Some places are extremely challenging for most of the tested techniques. The only exceptions are R-MAC, NetVLAD, and Cross-Region-Bow, whose lower bounds are constantly above 0.5 on Lagout. As $EP_{min} \geq 0.5$ requires $P_{R0} = 1$, one can conclude that the top-1 match retrieved by these three VPR methods is a TP in every Logout's location.

3.4.2 McNemar's Test Interpretation

The considered VPR methods are pairwise compared as described in Section 3.3.4 to confirm if the performance differences highlighted by the previous analysis are reliable and statistically significant. Fig. 3.6 shows the Z scores for every possible VPR method pair. A color code is used to represent the Z values for all combinations of threshold and dataset. Under the assumption of $N_{sf} + N_{fs} \geq 10$, Z is always positive. Hence, a sign convention is used to indicate which VPR method obtains better performance. A positive Z score means that the first technique of the pair is better than the second, that is, $N_{sf} > N_{fs}$. A negative value of Z indicates that the second VPR method of the pair outperforms the first one ($N_{sf} < N_{fs}$). Z is set to 0 when $N_{sf} = N_{fs}$ or $N_{sf} + N_{fs} < 10$. Is it relevant mentioning that $|Z|$ is indicative of the performance gap between two VPR methods as it is computed from $|N_{sf} - N_{fs}|$. However, it expresses a relative measure without giving any indication of the absolute VPR performance level that has to be assessed using the three metrics introduced in Section 3.3.3. The threshold set T used for the experiments includes 9 values ($p = 9$) equally spaced between 0.1 to 0.9. It enables a comparison along the entire spectrum of EP.

NetVLAD and Cross-Region-Bow outperform the other approaches on Berlin-Helensee,

TABLE 3.2: SOME EXAMPLES OF COMPARISONS WITH $|Z| < 1.96$.

NetVALD Vs Cross-Region-Bow	$\|Z\|$**@0.5**	**Confidence**
GardenPoints	1.20	77.0%
HybridNet vs AmosNet	$\|Z\|$**@0.5**	**Confidence**
Berlin-Helenstrasse	0.25	19.8%
Corvin-45	1.50	86.6%
GardenPoints	1.10	72.9%

GradenPoints, and Nordland datasets, as confirmed by their large positive Z values (Figs. 3.6b to 3.6d and Figs. 3.6f to 3.6h). They have similar performance on Lagout, while NetVLAD is generally better than Cross-Region-Bow on Corvin and Berlin-Halenstrasse at every threshold and worse on Nordland up to 0.5 (Fig. 3.6a). HybridNet outperforms or achieves comparable performance as AMOSNet in most test scenarios (Fig. 3.6u). These results are coherent with the performance analysis by their authors [5]. On Corvin, R-MAC presents large positive Z values for thresholds between 0.1 and 0.5 against every other VPR technique (third row in Fig. 3.6). At larger thresholds, Z decreases and becomes negative against NetVLAD starting from 0.7 (Fig. 3.6i). Thus, R-MAC outperforms the other approaches when the evaluation is carried out by observing low EP values, which P_{R0} mostly influences. As the threshold increases, the number of successes is determined by the contribution of R_{P100}. In such an evaluation perspective, R-MAC is outperformed by NetVLAD, which demonstrates to be capable of retrieving longer sequences of TPs on Corvin. McNemar's test results confirms and supports the bounds analysis for Corvin presented in the previous section: R-MAC has the best S_{P100} while NetVLAD reaches higher EP_{Max} (Fig. 3.5).

A particularly relevant threshold value is 0.5, which is used to compute S_{P100} (Eq. 3.8). Hence, the corresponding Z score might confirm or not the statistical significance

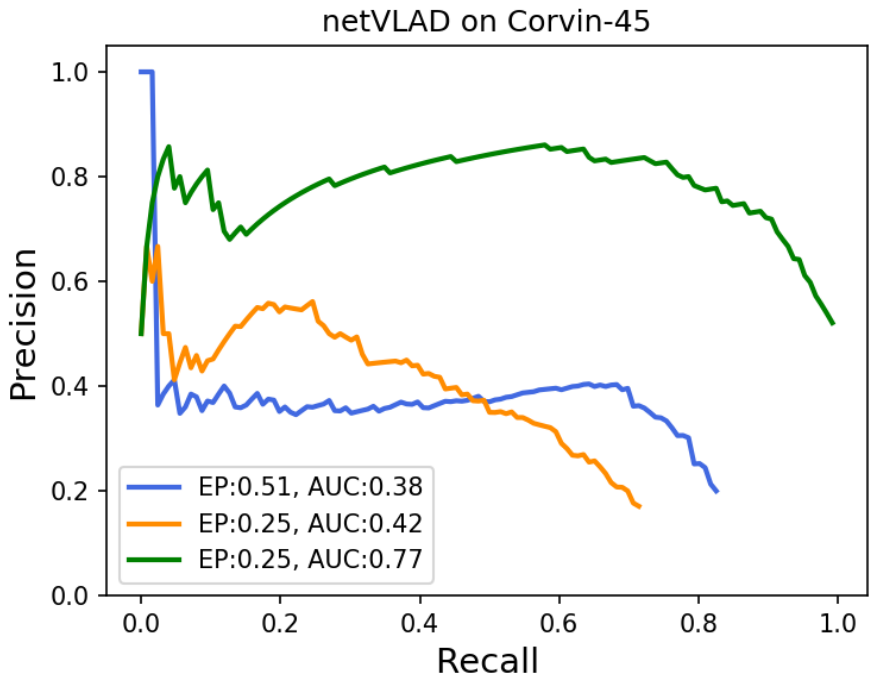
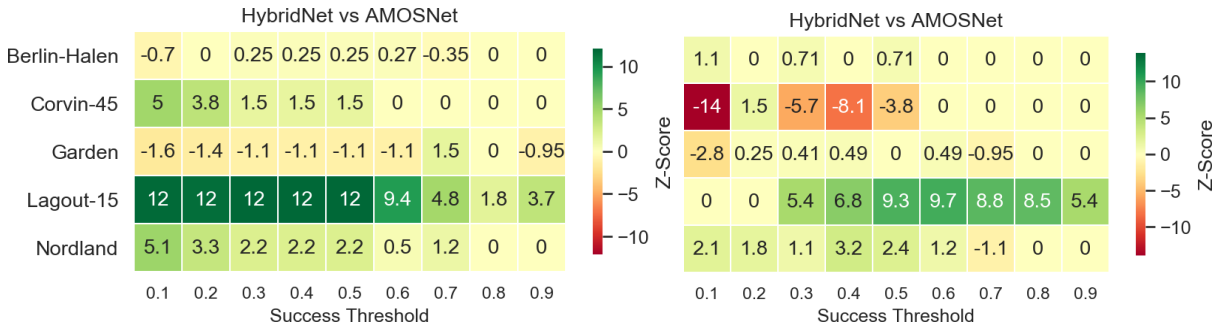


Fig. 3.7. A comparison between three PR-Curve for netVLAD on an image in Corvin dataset with their respective EP and AUC values.

of the S_{P100} scores presented in Fig. 3.5. As detailed above, a $|Z|$ value larger than 1.96 confirms that a comparison outcome is reliable with a confidence interval of 95%. It is not the case for the comparisons reported in Table 3.2, which shows some of tests scoring a small Z value and the corresponding confidence intervals. A small $|Z|$ means that the two methods scored similar successes and failures on a dataset (Eq. 3.4.2). An insufficiently large $|Z|$ yields the rejection of the hypothesis, which is a method is superior to the other. In this thesis, two methods with similar S_{P100} and $|Z| < 1.96$ are considered having *comparable* VPR performance.



(a) Extended Precision (EP). **(b)** Area Under Curve (AUC).

Fig. 3.8. McNemar's test using EP (left) and AUC (right) to compare HybridNet and AMOSNet.

3.4.3 Is AUC a suitable alternative to EP?

AUC can be used as an alternative to EP to measure VPR performance. However, the author considers AUC less appropriate than Extended Precision for use in the proposed evaluation framework. The most important reason is that AUC does not penalize top-ranked FPs in the query results. Indeed, AUC might be significantly incremented by long sequences of TPs regardless of their position in the retrieved image ranking. As opposed to this, the P_{R0} component of Extended Precision penalizes top-ranked false positives capping EP below the value of 0.5. In other words, a large AUC does not guarantee that the top retrieved images are correct matches. For example, the blue curve in Fig. 3.7 has AUC $= 0.38$ and EP $= 0.51$. The green curve has a larger AUC (0.77) and a smaller EP (0.25). As explained in Section 3.3.3, $P_{R0} = 1$ guarantee that the top-1 result is a TP, which is important for VPR and LCD in general. Therefore the blue curve should be considered better than the green one regardless of the smaller AUC.

Finally, AUC is more difficult to interpret than EP. Except for 0 and 1, the value of AUC is not related to any specific condition or PR-Curve feature. For this reason, McNemar's test based on AUC is harder to understand. Fig 3.8 shows the test results for a pair of VPR methods using EP and AUC. The high negative score of HybridNet against

AMOSNet at 0.5 on Corvin means that HybridNet's AUC cannot reach the threshold as often as AMOSNet. However, a clear interpretation of this outcome is hard to give as there is not any particular key value associated with AUC. Conversely, the positive Z value at 0.5 for the EP-based test indicates that the top-1 retrieved image by HybridNet on Corvin is more often a TP than for AMOSNet.

3.5 Summary

The evaluation method proposed in this chapter is based on a new performance metric, Extended Precision (EP), to measure the VPR performance. The evaluation process consists of three steps. Firstly an EP score is assigned to every environment's place. Then, the subsequent analysis assesses the VPR performance consistency across the environment and compares VPR methods, ensuring the outcome is statistically significant. EP summarizes a PR-Curve by combining two of its most relevant features, P_{R0} and R_{P100}, into a scalar measurement of VPR performance. EP addresses several shortcomings of AUC, which would produce less significant and unintelligible results if used within the proposed analysis method.

The remainder of this thesis utilizes the metrics and methodologies proposed in this chapter to ensure that the results can be easily compared across the entire manuscript. To this end, Section 3.4 provided a demonstration that serves two purposes: firstly, it shows how to utilize the evaluation frameworks effectively, and secondly, it offers valuable case studies on state-of-the-art VPR techniques that can be used as a reference in the following chapters.

Chapter 4

Exploring Accuracy-Computation Trade-off of Local Image Descriptors for Aerial Robotics[1]

As noted in Section 1.2, accuracy is undoubtedly fundamental for VPR-based applications. However, low resource usage is also relevant when a robot cannot rely on powerful hardware or need to save battery life. This chapter presents the first attempt of this thesis to address VPR efficiently using Unmanned Aerial Vehicle (UAVs) as an application context. UAVs are a category of mobile robots particularly susceptible to the problem of performing VPR efficiently. While UAVs face one of the most challenging VPR conditions, 6-DOF viewpoint changes, they are affected by payload restrictions limiting the battery weight and, consequently, the hardware computational power. For this reason,

[1]This chapter extends that paper published and presented at NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Colchester, UK, 2019, pp. 103-108, with tile of: "Visual Place Recognition for Aerial Robotics: Exploring Accuracy-Computation Trade-off for Local Image Descriptors". DOI: 10.1109/AHS.2019.00011.

computation-intensive VPR methods, such as those based on CNNs, are unsuitable for UAVs. A viable approach is using local image descriptors, as these can be computed relatively efficiently without needing any special hardware, such as a GPU. However, the choice of a local feature descriptor is not trivial and calls for a detailed investigation, as there is a trade-off between VPR accuracy and computational effort. In this chapter, the author examines several state-of-the-art local feature descriptors' performance and computational efficiency for VPR applications utilizing standard aerial and appearance change datasets. The results presented in this chapter show that a trade-off between accuracy and computational effort is inevitable. Also, experiments confirm that local feature descriptors are substantially more efficient than CNN-based techniques but lack accuracy when coping with environmental changes. Therefore, while local feature descriptors are good candidates for resource-constrained hardware, they have a limited range of applicability in dynamic environments compared to CNN-based methods.

4.1 Background and VPR Challenges in Aerial Robotics

UAVs have been receiving great attention recently as they have a wide variety of industrial applications, such as aerial imaging, communication, and surveying [4, 197, 198]. As part of autonomous navigation, place recognition is critical for UAV localization [199]. UAVs can often rely on GPS to track their position. However, there are several situations where GPS is unavailable or degraded, such as underground operations [200], space exploration [57], and close-ground flying next to tall buildings [201]. Tracking the position with internal sensors enables localization in GPS denied environment. However, the estimation tends to drift over time because of accumulated errors requiring

re-localization at a certain point [201]. VPR is one the most suitable way to address loop closure detection onboard a UAV. This is motivated by the availability, cost, size, and weight of modern cameras, which make them suitable for small-payload aerial vehicles. However, VPR is particularly challenging for small UAVs due to the extreme viewpoint changes they experience and the limited computational power onboard [4]. Some state-of-the-art VPR methods are only robust to small viewpoint changes [91, 202], or their computation-intensive nature makes their use prohibitive for UAVs [21]. Some recent works [31, 57, 52, 4] proposed VPR pipelines based on local image features as they can be extracted efficiently on resource-constrained hardware. As noted in [52], the choice of the local feature descriptor implies a trade-off between the computation efficiency and the accuracy of the visual place recognition module. This consideration calls for a detailed investigation into the accuracy-computation trade-off for local feature descriptors specifically for VPR applications.

The study presented in this chapter explores the accuracy-computation trade-off of several state-of-the-art local feature descriptors in the challenging operating scenario of UAVs using standard ground-aerial image datasets exhibiting mild to extreme 6-DOF viewpoint changes. VPR performance is assessed using the framework presented in Chapter 3, and the image processing time is taken as a computational complexity measure. Results are presented for several well-established local descriptors in VPR applications: SIFT [34], SURF [40], BRISK [51], AKAZE [58], and ORB [41]. The experiments also confirm that local feature descriptors perform similarly to CNN-based techniques in dealing with 6-DOF viewpoint changes while running considerably faster. However, they are outperformed under conditional changes, in particular those induced by illumination [8] and seasonal [116] variations.

The reminder of this chapter is organized into the following sections. Section 4.2 describes the method and criteria used for performance evaluation. The experimental results are discussed in Section 4.3. Conclusions and next research steps are given in Section 4.5.

4.1.1 Local Feature Descriptors: an Overview

This section introduces the local feature descriptors used for the experiments conducted in this chapter, expanding the overview provided in Section 2.1.1.

SIFT [34] detector assigns a keypoint with a location, scale, and orientation. The descriptor algorithm takes an oriented window around the feature location of the 16×16 neighborhood in a scale pyramid, which is divided into 16 sub-block of 4×4 size. To form the feature descriptor, an 8-bin HOG [42, 43] is created for each sub-block for 128 values. The resulting image representation has a feature descriptor per detected feature. SIFT descriptor is invariant to both scale and rotation and is agnostic to the feature detection method to localize keypoints.

SURF [40] assigns the orientation to a local feature by analyzing the Haar-wavelet response in a circular area around its location. The radius is $6 \times$ *scale* to render SURF scale-invariant. The orientation is the direction where the wavelet response is the strongest. The descriptor is computed for an oriented square area of $20 \times$ *scale*. The square region is divided into 4×4 squares. The Haar-wavelet response is computed along horizontal (x) and vertical (y) directions in a 5×5 grid. Every sub-region is then described by a vector including four elements: the algebraic sum and the sum of the module of wavelet response along x and y axis. The result is a vector of 64 elements to describe a feature, one half then SIFT.

BRISK [51] is a local binary descriptor. While the features vector is computed similarly to BRIEF [49] using pixel intensity, the sampling strategy differs. BRISK uses concentric circular patterns instead of random sampling. This approach allows selecting distant to close pixel pairs that are used to assign an orientation to the descriptor. The process consists in computing the gradient between pairs at multiple sampling pattern orientations. The primary orientation of BRISK corresponds to the direction where the sum of the gradients is maximum.

ORB [41] includes a feature detector stage that is a scale-invariant version of FAST [53], a stand-alone corner detector. The descriptor stage uses BRISK with a rotation matrix based on the keypoint's orientation to achieve rotation invariance.

AKAZE [58] evolves from KAZE [203]. As its predecessor, AKAZE comes with a detector based on the determinant of the Hessian Matrix [37] and achieves scale invariance by searching a nonlinear scale-space constructed by the Fast Explicit Diffusion (FED) algorithm [203]. The descriptor follows the same idea as BRIEF but works on the average intensity of areas instead of single pixels for better robustness to appearance changes. Finally, rotation invariance is obtained by finding the dominant orientation in a circular neighborhood around each detected feature.

4.2 Experimental Setup

The proposed approach evaluates local image feature descriptors addressing VPR as an image retrieval task, as done in the previous chapter. Local image feature descriptors are used to build a map from images of previously visited places. During the localization phase, the map is searched to retrieve the reference images that match a query image

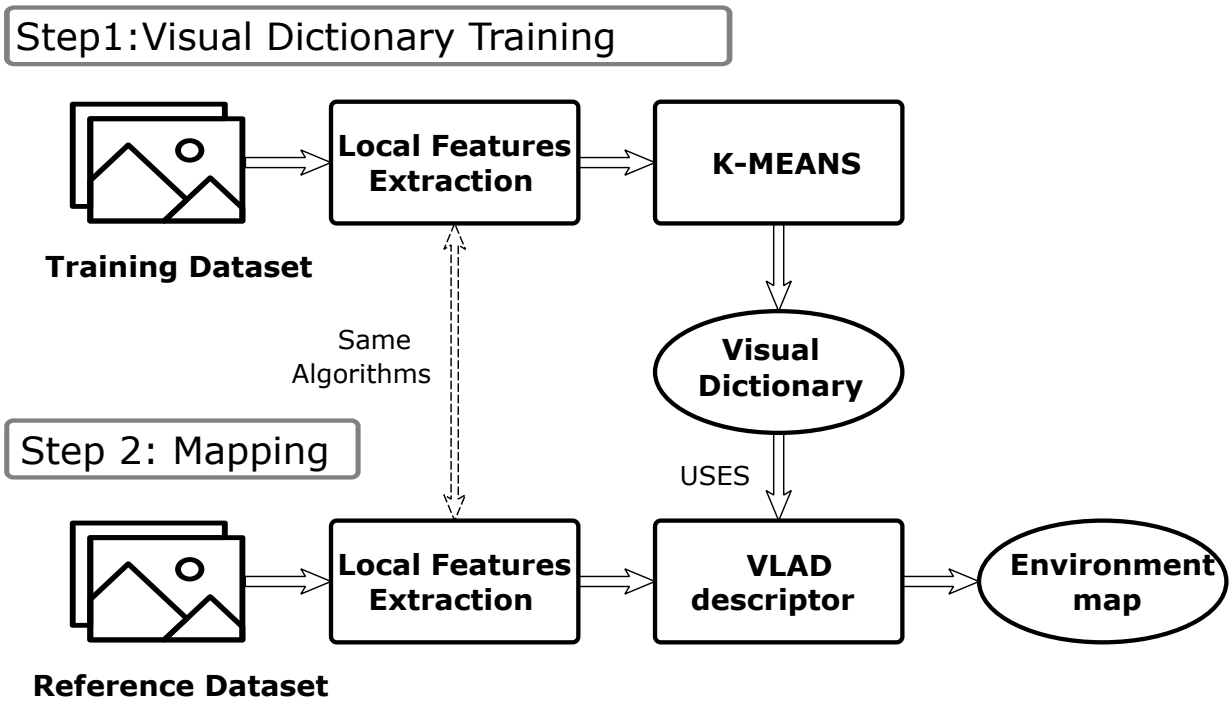


Fig. 4.1. Map images processing diagram.

(i.e., a frame captured by the onboard camera). Localization succeeds if the query frame is correctly matched with a reference image in the map. The VPR algorithm and the evaluation criteria used to assess local image feature descriptors are detailed below.

4.2.1 Mapping

A map represents the knowledge of a robot about the environment. The map is built offline from a set of images showing environment locations denoted as *reference dataset*, I_M, as illustrated by the diagram in Fig. 4.1. For each image in I_M, the set of local image feature descriptors is extracted by the assessed local descriptor (e.g., SIFT) and combined into a global image descriptor using VLAD [69, 70]. The resulting VLAD descriptors are finally organized in a ball-tree structure [204] to make the localization faster. VLAD requires a visual dictionary, V_k, which is computed using k-means clustering [65] on

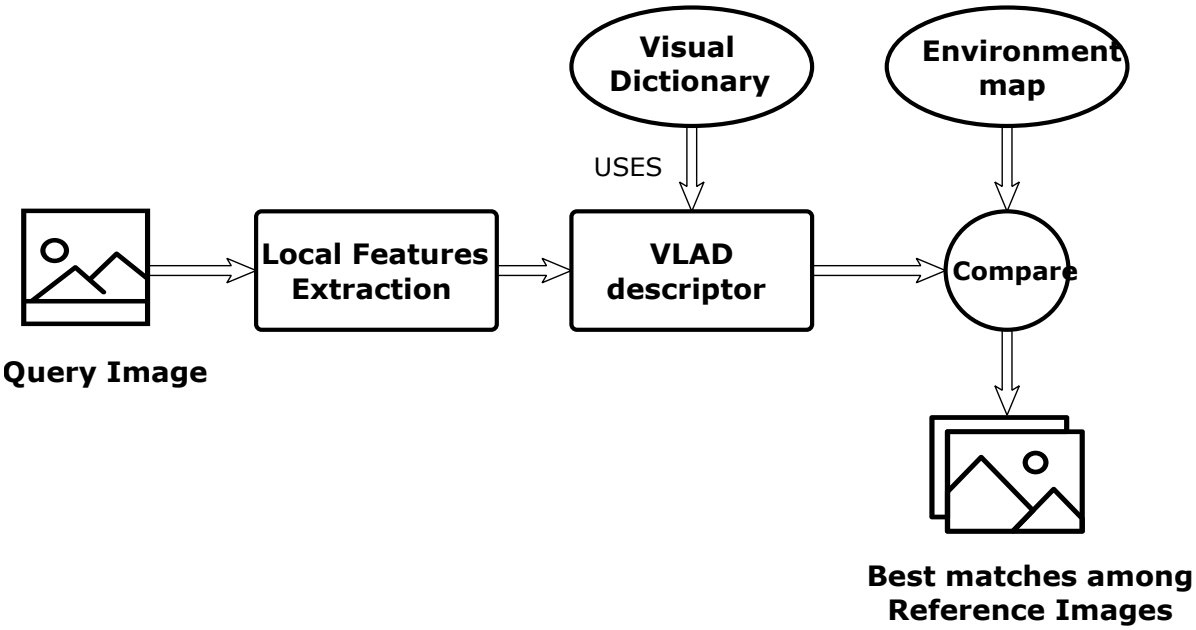


Fig. 4.2. Image retrieval diagram for a query image.

the features extracted from a *training dataset*, I_R, using the same assessed local image descriptor.

4.2.2 Localization

The image retrieval process is illustrated in Figure 4.2. A VLAD descriptor is computed for the query image and compared with the VLAD descriptors in the map using cosine (Eq. 3.4). As described in Section 3.3, the reference images are ranked for their similarity with the input image, and the resulting sequence is returned for the query image. A higher rank means a higher similarity between a reference image and the query image. If the localization succeeds, the image ranked at the top shows the same place as the query image.

Fig. 4.3. A place from each dataset as it appears in different loops.

4.2.3 Evaluation Method

VPR performance is evaluated using EP bounds (Eqs. 3.6 and 3.7) and S_{P100} (Eq. 3.8). The metrics for computational efficiency are the required time to process an image to build a VLAD descriptor (encoding time) and the matching time.

4.2.4 Benchmark Datasets

The 6-DOF benchmark datasets are Lagout, Old City, and Corvin [4]. GardenPoints [8] and Norland [116] are used for testing conditional changes. These datasets are introduced in Section 2.1.5. Here are provided additional details pertinent to the experiments,

TABLE 4.1: APPEARANCE VARIATIONS AND GROUND TRUTH OF THE BENCHMARK DATASETS.

Dataset	Appearance Variation		Reference Images	Query Images	Ground Truth
	Viewpoint	Condition			
Lagout 0-15	Mild 6-DOF	None	330	324	by authors
Lagout 0-30	Moderate 6-DOF	None	330	397	by authors
Lagout 0-45	Wide 6-DOF	None	330	395	by authors
Corvin 0-30	Moderate 6-DOF	None	1179	1440	by authors
Corvin 0-45	Wide 6-DOF	None	1179	1298	by authors
Old City	Extreme 6-DOF	None	5408	5641*	by authors
GardenPoints (Day-Night Right)	Mild to moderate lateral shift	Night-Day	200	200	± 2 frames
Norland (Summer-Winter)	None	Seasons	1622	1622	± 5 frames

(*) ONLY A SUBSET OF 200 IMAGES IS USED FOR OLD CITY TO KEEP THE EXPERIMENTS WITHIN A REASONABLE DURATION.

such as the indication of which loops are used and how many images they include.

Lagout is a synthetic dataset consisting of four aerial footages captured at different angles: $0°$, $15°$, $30°$, and $45°$. Corvin is similar to Lagout and includes three aerial loops around the Corvin castle at $0°$, $30°$, and $45°$. Old City consists of two long loops captured in an urban environment. Fig. 4.3 provides samples of those datasets. The reference dataset used for mapping is the loop at $0°$ for Lagout and Corvin and Old City-1 for Old City. The ground truth is provided by their authors [196]. GardenPoints (right-night to right-day) [8] and Norland (winter to summer) [116] datasets are used to compare local image feature descriptors with CNN-based descriptors under conditional changes. The characteristics of these datasets are summarized in Table 4.1, including the number of images, which influences the retrieval time. Please note that only a subset of 200 random images is used for Old City to keep the experiments within a reasonable duration. This exact Old City query subset is also employed for the other experiments presented in this

Fig. 4.4. Six images from VASE-JBL dataset.

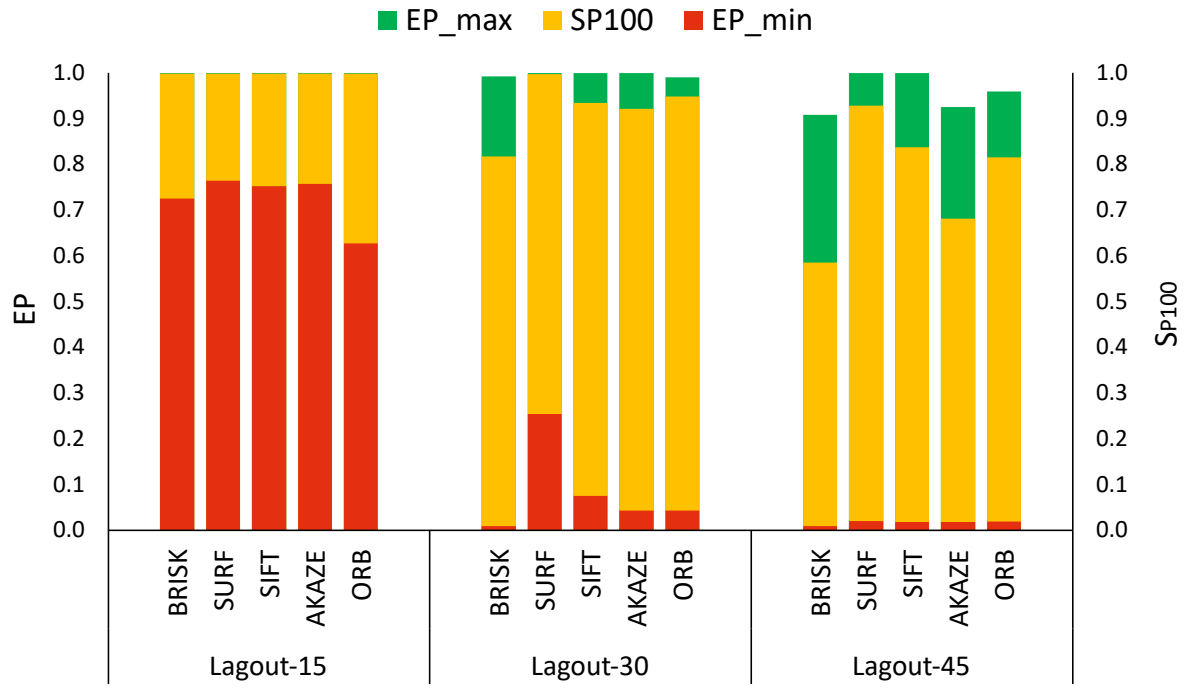
thesis unless otherwise specified.

4.2.5 Training Data

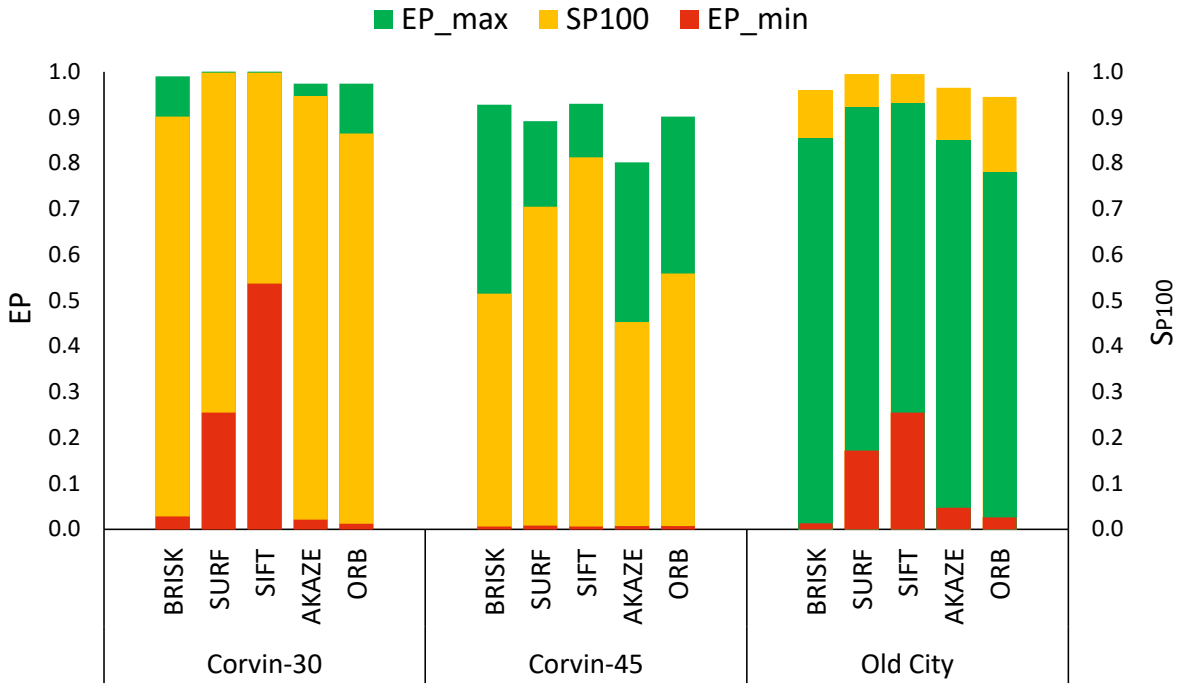
The training data are chosen to obtain a generic visual dictionary (V_k) that works reasonably well for every benchmark dataset. Further analysis and insights into training data selection are provided in Section 4.4. The results presented in the next section are obtained with V_k trained on VASE-JBL [205], a dataset including 539 images showing a wide variety of outdoor and indoor scenes captured in real-world environments [206]. VASE-JBL samples are provided in Fig. 4.4.

4.3 Experimental Results Discussion

The descriptors included in the tests are SIFT [34], SURF [40], BRISK [51], AKAZE [207] and ORB [41]. Each descriptor has been used with its native local feature detector stage. The VPR pipelines used for the tests has been implemented on top of the source code available at [208]. The local descriptors implementations are from OpenCV (3.4.2.17)



(a) Results for all Lagout loops.



(b) Results for Corvin and Old City datasets.

Fig. 4.5. EP bounds and S_{P100} for ORB, BRISK, SIFT, SURF and AKAZE.

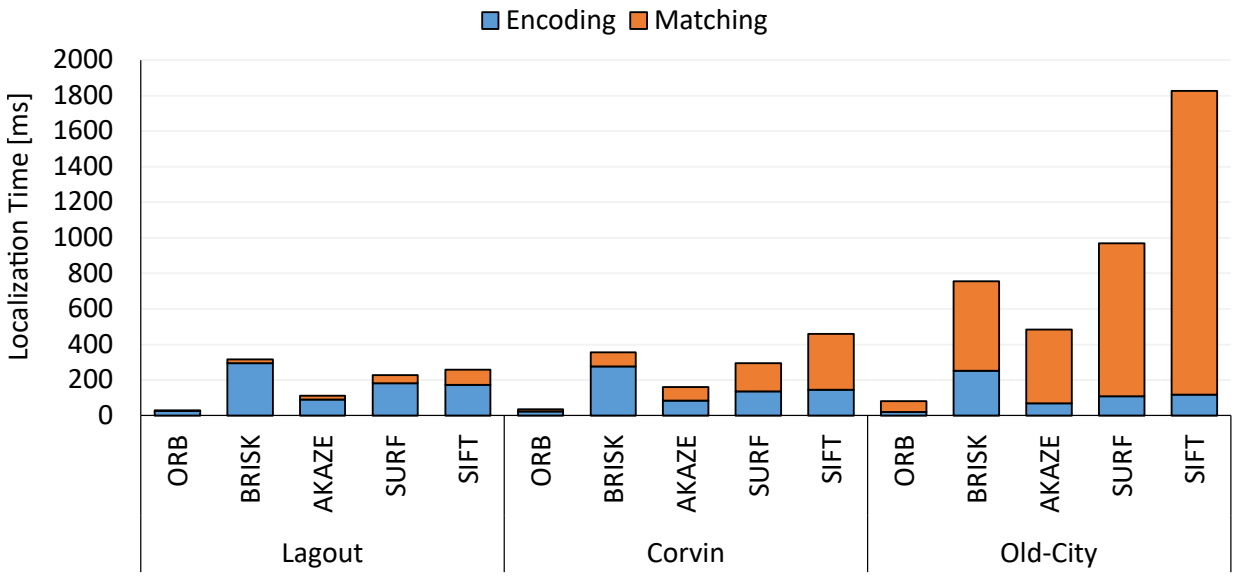
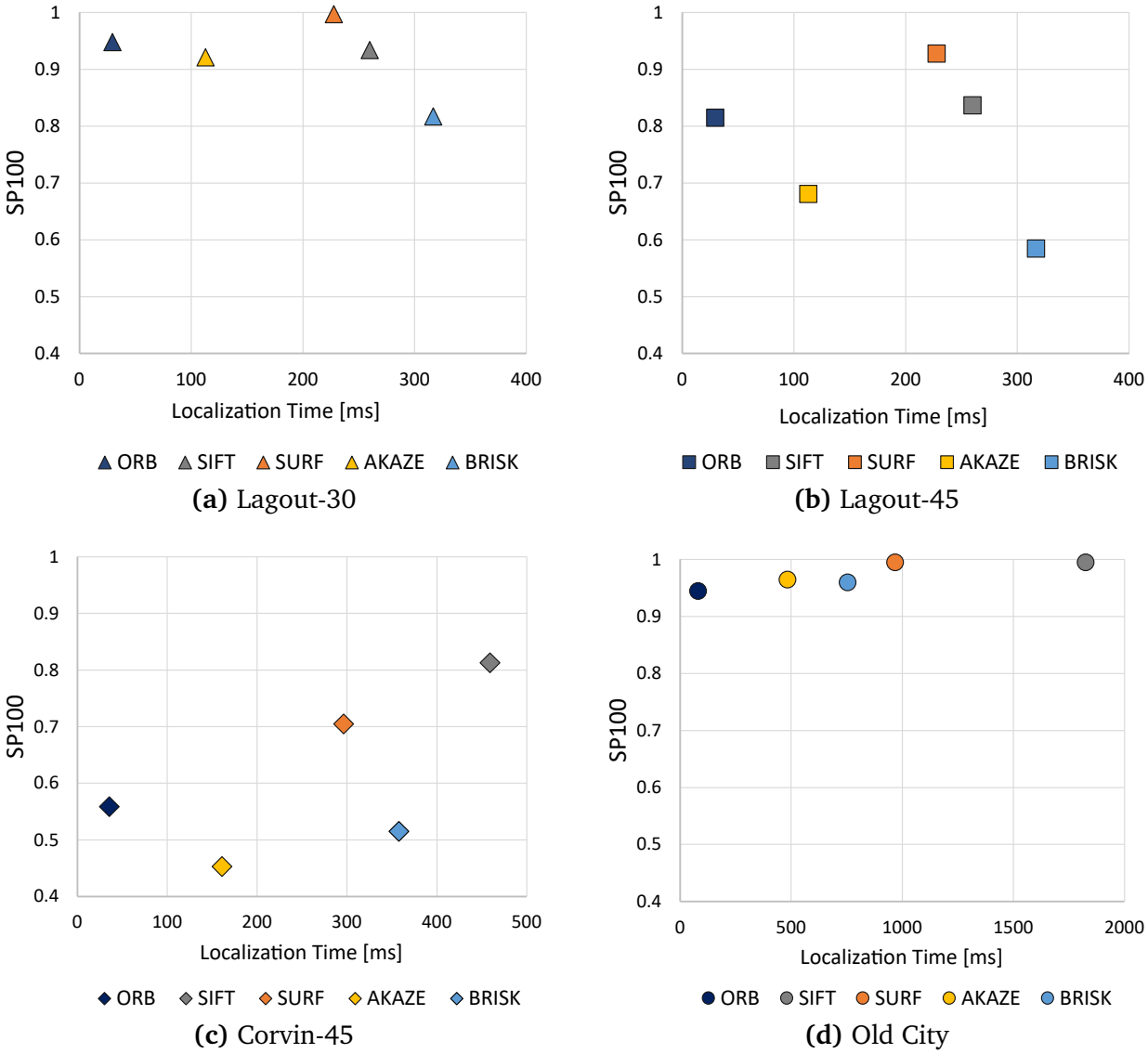


Fig. 4.6. The time required for localization in Lagout, Corvin and Old City datasets.

[209] and are used with the default parameters suggested by their respective authors. The size of the visual dictionary, V_k, has a significant impact on VPR performance. A grid search yielded 2048 words for SURF and SIFT, 1024 for BRISK and AKAZE, and 256 words for ORB.

4.3.1 Accuracy and Computation Time

Figs. 4.5a and 4.5b show the EP bounds and S_{P100} for the considered local image descriptors. VPR exhibits the highest performance when using SURF features in Lagout. SIFT has a wider lower bound than SURF on Corvin-30 and Old City and slightly higher S_{P100} on Corvin-45. However, VPR performance is only one of the evaluation criteria being considered in this study. The complete picture includes the data from Fig. 4.6 showing the localization time of each descriptor measured for a single thread execution on an Intel i7-7700K CPU. The encoding time is similar for SURF, SIFT, and AKAZE. BRISK is the slowest, which is quite unexpected as ORB, the fastest, uses a similar algorithm to



(a) Lagout-30 **(b)** Lagout-45

(c) Corvin-45 **(d)** Old City

Fig. 4.7. VPR performance versus total localization time (encoding and matching).

compute the descriptor. This discrepancy is likely due to BRISK implementation flaws or sub-optimal settings of its detector stage. Overall, the localization takes longer for Old City because of its map's higher number of images. There are 5408 images to search in Old City-1 while only 1179 and 330 images in Corvin-00 and Lagout-00, respectively. SIFT and SURF yield the best VPR performance. However, the latter is considerably more

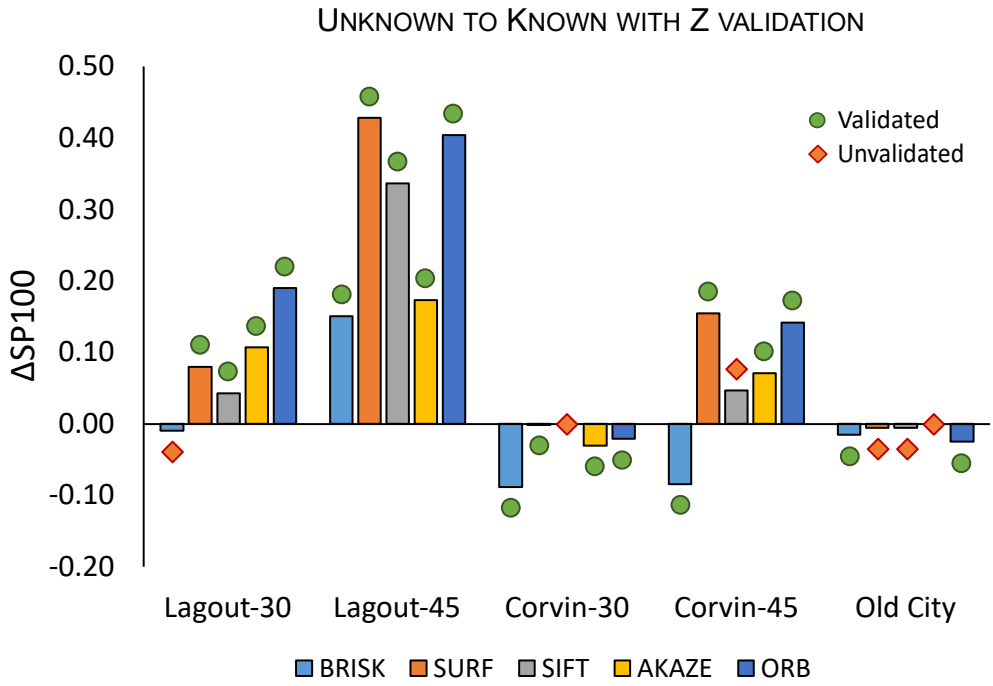


Fig. 4.8. S_{P100} difference between V_k trained with environment-unrelated (Unknown) and related (Known) data with Z-test validation.

efficient and, therefore, a better choice. In particular, this gap is vast when the localization occurs in Old City, where SURF is about $800\,ms$ faster than SIFT. The significant difference in matching time is mainly due to the SIFT descriptor, which is twice as long as SURF's (Section 4.1.1). Fig. 4.7 offers an overview of VPR performance to localization time trade-off, suggesting further considerations about ORB. Although SURF-based VPR can be considered a good trade-off, it still requires about 1 second to complete the localization in Old City using a workstation CPU. For UAVs, whose computation power is generally lower, ORB can be a better option. Indeed, although ORB enables less accurate VPR than SIFT and SURF, it can complete localization about 10 times faster.

TABLE 4.2: CORRELATION COEFFICIENTS BETWEEN TRAINING AND MAP DATASETS.

Training Dataset	Map Dataset	ORB	BRISK	SIFT	SURF	AKAZE
Lagout-15	Lagout-00	0.151	0.216	0.308	0.223	0.215
Lagout-30	Lagout-00	0.154	0.221	0.312	0.217	0.218
Lagout-45	Lagout-00	0.153	0.220	0.313	0.216	0.216
VASE-JBL	Lagout-00	0.148	0.220	0.310	0.222	0.215
Corvin-30	Corvin-00	0.153	0.216	0.313	0.220	0.210
Corvin-45	Corvin-00	0.149	0.219	0.310	0.218	0.213
VASE-JBL	Corvin-00	0.155	0.219	0.312	0.219	0.214
Old City-2	Old City-1	0.152	0.222	0.310	0.22	0.214
VASE-JBL	Old City-1	0.152	0.222	0.307	0.222	0.216

4.3.2 Considerations on Training Data

The results presented above are for a scenario where the VPR module is agnostic to the operating environment. In particular, the images used to train the visual dictionary were from VASE-JBL, which is unrelated to Lagout, Corvin, and Old City datasets. This section examines the use of environment-related data to train the visual dictionary for VLAD to provide the VPR system with some prior knowledge of the operating environment. More precisely, the dictionary to operate in an environment is obtained from the map: V_k is trained on Lagout-00 for Lagout, Corvin-00 for Corvin, and Old City-1 for Old City. Fig. 4.8 shows S_{P100} difference between using environment-unrelated (Unknown) data and map images (Known) to train V_k. The markers above the bars indicate McNemar's test result for the unknown and known pairs (Section 3.4.2). Green circles indicate a confidence interval equal to or above 95%, allowing ΔS_{P100} to determine the best training approach, unknown or known. Red diamonds are for narrower confidence intervals indicating that the two approaches are regarded as having comparable performance. For

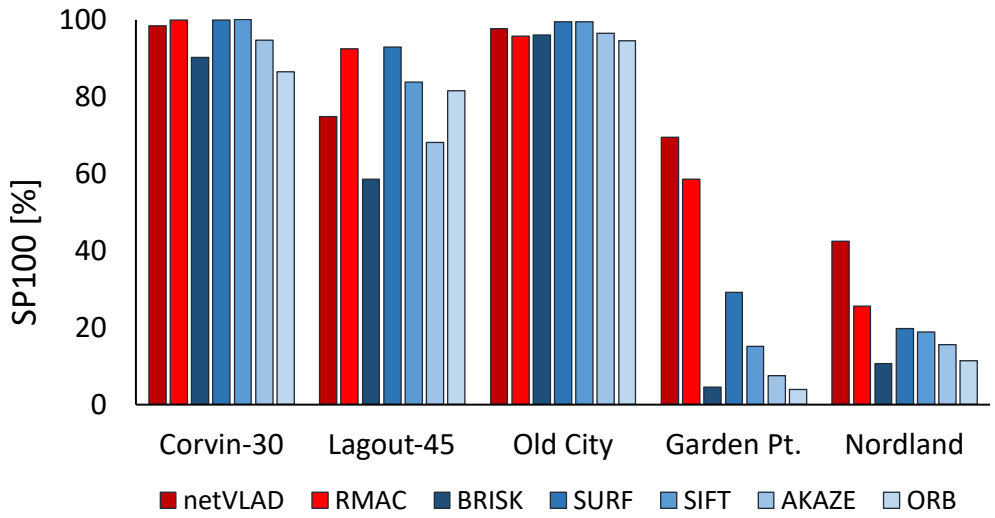


Fig. 4.9. S_{P100} comparison between CNN-based and local descriptor-based VPR methods.

example, SURF-based VPR has higher performance on Lagout-45 when trained on VASE-JBL ($\Delta S_{P100} = 0.43$). This gap is statistically significant, as indicated by the green circle mark above the corresponding bar. Conversely, SIFT on Corvin-45 has $\Delta S_{P100} = 0.04$ with $|Z| = 0.12$, which is below the threshold of 1.96. Then, SIFT-unknown and SIFT-known are regarded as comparable, accordingly to the considerations made in Section 3.4.2.

Training V_k on the map images generally results in worse VPR performance than using VASE-JBL. Although the loops appear to be correlated when looking at the images as a whole, the same does not happen for the corresponding feature sets. Table 4.2 shows the correlation coefficients from every pair of training and mapping datasets used for the experiments. The difference between the correlation coefficients of VASE-JBL and the other training datasets is negligible, explaining why there is no improvement by using prior knowledge for local features-based VPR. From these results, the author concludes that the feature space partitioning determined by the training data to build the visual dictionary is the dominant factor influencing the VPR performance.

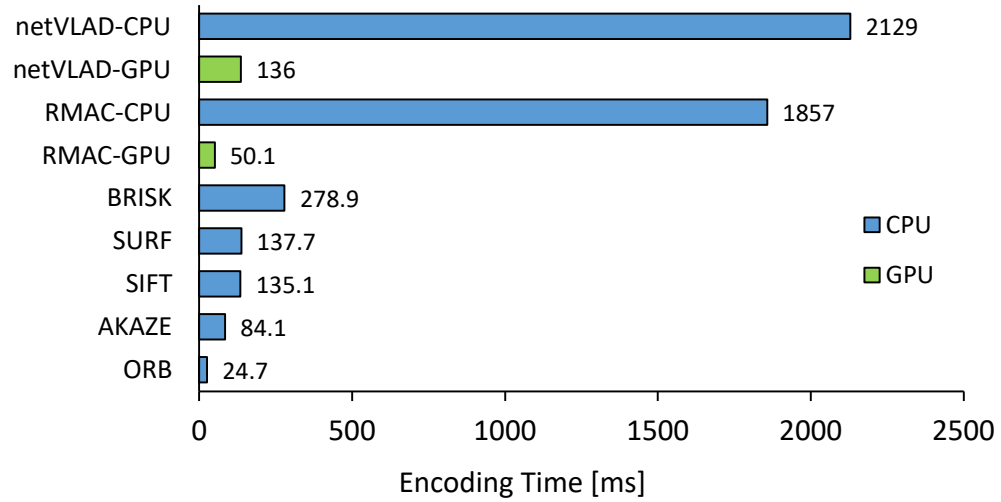


Fig. 4.10. Encoding on Corvin-30 measured for an Intel i7-7700K CPU and Nvidia GTX-1080 GPU.

4.4 Can Local Feature Descriptors replace CNNs in VPR Applications?

Figs. 4.9 and 4.10 show S_{P100} and the encoding time for local feature descriptors compared to CNN-based methods. The comparison includes the CPU and GPU encoding time for RMAC [96] and NetVLAD [71]. While the GPU encoding time of RMAC and NetVLAD are comparable to local descriptors, on a CPU they are considerably slower. Moreover, local descriptors are competitive with CNNs under mild to extreme 6-DOF viewpoint variations, as shown in Fig. 4.9. On the other hand, CNN-based methods are better on conditional changes outperforming by a significant margin local descriptors on GardenPoints and Nordland. Therefore, local feature descriptors may not meet accuracy requirements in changing environments.

4.5 Summary and Next Steps Toward Addressing Changing Environments

This chapter proposes a comparison of several state-of-the-art local feature descriptors for VPR under mild to extreme viewpoint changes in small UAVs using ground-aerial image datasets. VPR accuracy is crucial for loop-closure-based localization, but it is not the only property to be considered for mobile robotics based on resource-constraint hardware. As UAVs are very agile vehicles, they need to re-localize quickly, but, at the same time, they are equipped with resource-constraint hardware to meet payload limitations and extend battery life. Driven by this consideration, the evaluation of local feature descriptors is based on the VPR performance and localization time to determine the descriptor with the best trade-off for the potential use with low-end hardware. The results show that SURF and SIFT descriptors reach the highest VPR performance at the cost of a long localization time. ORB can be a better option for UAVs as it allows much faster localization while keeping a reasonable accuracy with most of the datasets considered for the experiments.

In Section 4.4, local feature descriptors have been compared with CNN-based methods exhibiting higher efficiency but lower performance under conditional change. The results indicate that local descriptors are the best option to operate in the absence of conditional changes. Conversely, dynamic environments require the invariance provided by learned image descriptors. This consideration raises the need for a learning-based approach suitable for running smoothly on low-end hardware without needing a GPU to deal with conditional changes. The next chapter investigates this problem and proposes the use of Binary Neural Networks [25] as a more efficient alternative to CNNs for

changing environments.

Chapter 5

Binary Neural Networks for Efficient and Effective Visual Place Recognition in Changing Environments[1]

The previous chapter showed how conventional hand-crafted methods based on local features fail under extreme environmental appearance changes. Conversely, those based on Convolutional Neural Networks (CNNs) achieve significantly better performance but result in heavy runtime processes and model sizes that demand a large amount of memory. This chapter focuses on reducing the runtime requirements of the convolutions to render image processing more efficient and suitable for resource-constraint hardware to operate in changing environments. This chapter discusses two contributions introduced in Section 1.3. The first contribution confirms the effectiveness of Binary Neural Networks (BNNs) in addressing VPR in dynamic environments with conditional changes.

[1]This work is published in IEEE Transactions on Robotics, vol. 38, no. 4, pp. 2617-2631, Aug. 2022, and presented at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022. DOI: 10.1109/TRO.2022.3148908.

The second and foremost contribution is a multi-step approach that reduces the precision of Convolutional Neural Network (CNN) parameters, reduces network depth, and uses fewer neurons in the classifier stage to train a new class of highly compact models. These models significantly reduce the memory requirements and computational effort while maintaining good VPR performance coping with conditional changes. To the author's knowledge, this is the first attempt at using Binary Neural Networks to solve the visual place recognition problem effectively under changing conditions and with significantly reduced resource requirements. The proposed binary neural network, FloppyNet, achieves comparable VPR performance when considered against its full-precision and deeper counterparts with up to a 99% smaller model size and increasing the inference speed by seven times.

5.1 Addressing Changing Environments Efficiently

The importance of VPR for autonomous navigation has been remarked on multiple times in the previous chapters. The VPR module enables a robot to re-localize when the position tracking fails or drifts due to accumulated errors. However, changes in appearance due to seasons, weather, and illumination render VPR challenging for mobile robots. Section 4.4 demonstrated that conventional hand-crafted techniques for VPR fail under extreme environmental changes. At the same time, those based on deep CNNs achieve higher performance but require considerably longer processing times. Considering that VPR is executed onboard mobile robots usually equipped with resource-constrained hardware, such demanding techniques may be inapplicable [4, 52]. Increasing the efficiency of VPR by saving memory and reducing the computational effort to run a model without

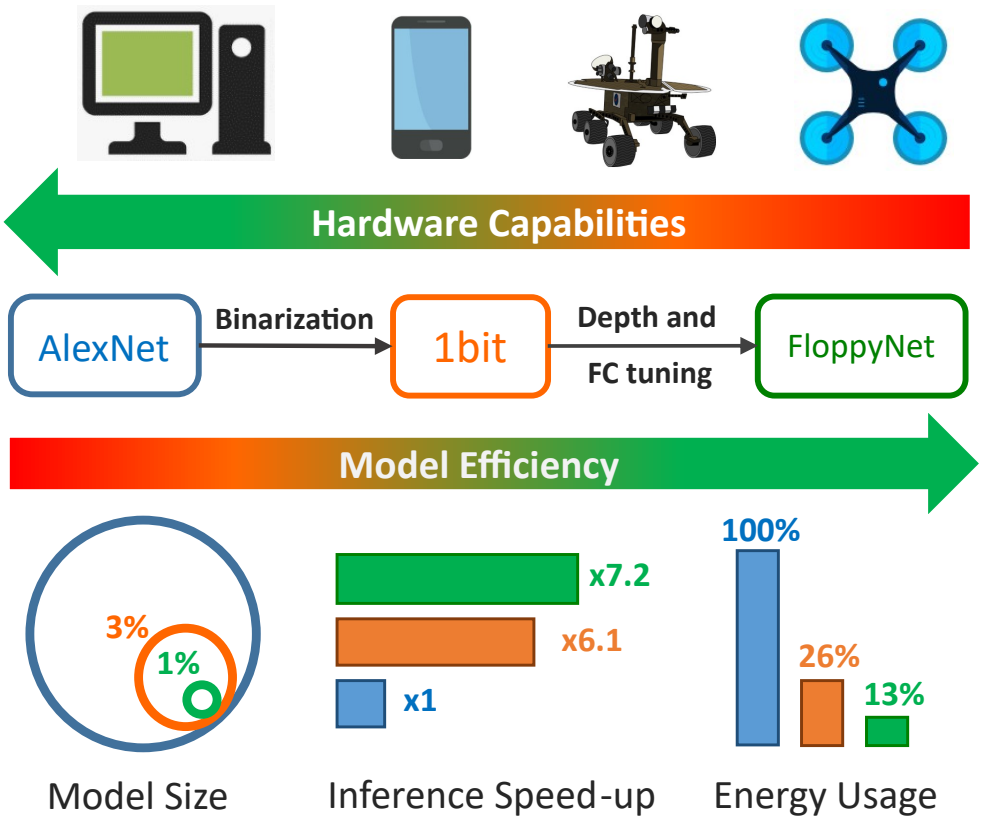


Fig. 5.1. FloppyNet is a compact and efficient BNN derived from AlexNet to enable VPR on edge devices and robots with severe hardware constraints.

sacrificing performance is paramount for a resource-constrained robot. Higher efficiency enables VPR on cheap hardware and frees resources for additional functionalities to improve a robot's navigation system. However, reducing resource demand while keeping VPR performance at a reasonable level is difficult. To tackle this challenge, the author proposes the multi-step approach summarized in Fig. 5.1 that combines Binary Neural Networks [25, 151] and depth reduction to obtain very compact models that drastically decrease the memory requirements and improve computational efficiency. The subsequent VPR performance loss is mostly countered by training the model using a classifier stage with a reduced number of full-precision neurons.

BNNs are a class of networks characterized by a single-bit precision for both weights

and activations instead of the 32 or 64 bits used by conventional deep neural networks. So far, BNNs have been employed and highly optimized for classification tasks only, where they exhibit lower yet comparable accuracy to their full-precision counterparts [147, 153]. However, classification and VPR are different problems, as highlighted in Section 2.1. The first aims to find the best fit among categories, while VPR consists of matching different images of the same scene. The work presented in this thesis proposes a generic BNN network architecture specifically trained on place images to solve the VPR problem effectively under environmental changes and with significantly reduced memory requirements and computational effort. The best network obtained[2], called FloppyNet, achieves close VPR performance to its full-precision and deeper counterpart, AlexNet [26], while using 99% less memory and running seven times faster. With a model size of 154 Kilobytes, FloppyNet can be stored on an old $5\frac{1}{4}$-inch floppy disk!

The reminder of the chapter in organized as follows. Section 5.2 describes the proposed multistep approach while introducing training and design principles of BNNs. Section 5.3 presents FloppyNet. The evaluation criteria are detailed in Section 5.4 along with the benchmark data used to obtain the results. The successive five sections are for presenting experimental results. Section 5.5 offers a comprehensive analysis of binary layers demonstrating the suitability of BNNs for VPR. Sections 5.6 and 5.7 discuss the VPR performance of the proposed BNN. Section 5.8 presents computation and energy benchmarks, and Section 5.9 compares FloppyNet with hand-crafted image descriptors on a low-end hardware platform. Finally, Section 5.10 draws conclusions and indicates how to develop further the research presented here.

[2]https://github.com/bferrarini/FloppyNet_TRO

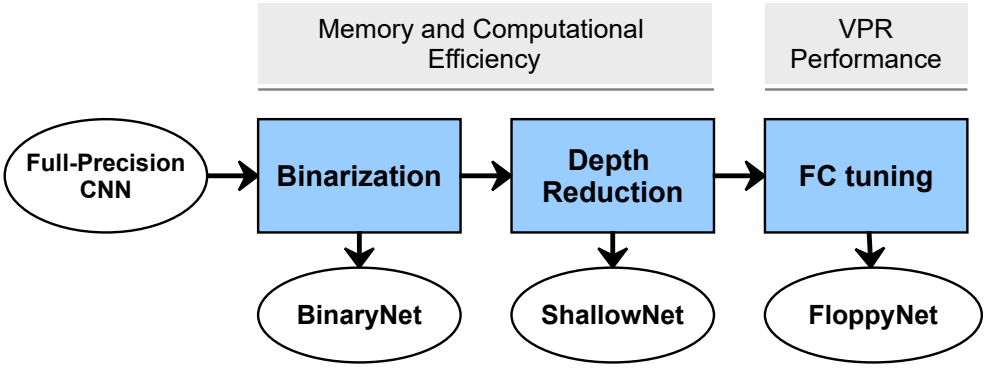


Fig. 5.2. The diagram shows the three transformation steps to obtain FloppyNet and the related by-product: BinaryNet and ShallowNet.

5.2 From CNNs to BNNs in Three Steps

This section describes the steps to turn a CNN into a compact yet effective feature extractor for VPR while introducing BNN training and design principles. Fig. 5.2 shows the proposed approach and the by-product of each step. Binarization reduces the model size and speeds up convolutions by enabling bitwise arithmetic. Depth reduction decreases the number of layers for further model size reduction and faster computation. The subsequent performance loss due to binarization and layer removal is countered chiefly by training the network with an appropriately sized fully-connected stage consisting of full-precision neurons.

5.2.1 First Step: Binarization

This study assumes that full-precision neural networks use 32-bit parameters and describes how binarization improves memory usage and computational speed compared to such networks. It should be noted that the methods described here are also applicable to 64-bit networks without any loss of generality.

The main advantages of binarization are the tiny model sizes to deploy and the high

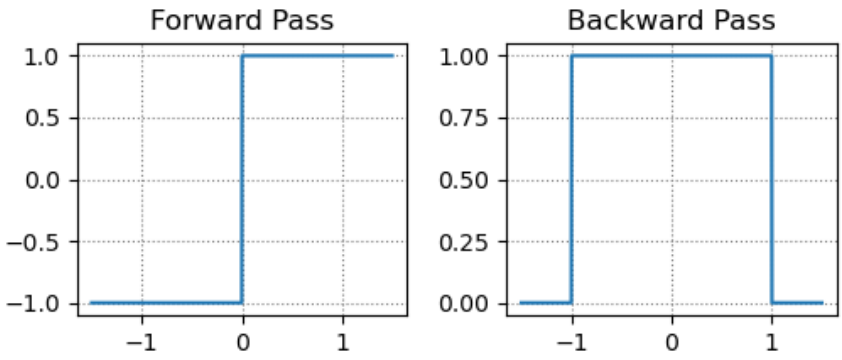


Fig. 5.3. Sign quantizer in forward and backward passes.

computational efficiency that the concatenation of multiple parameters into 32-bit variables enable. Indeed, a 32-bit weight requires four bytes, while a single bit is needed for a binary one. Hence, by concatenating 32 binary weights into a floating-point variable, the resulting model size is about 97% smaller than its full-precision counterpart. Moreover, bitwise operations between weights and activations are computed in parallel with 32 binary operands resulting in a significant speed-up in convolutions. However, optimizing a binary model with a reasonable performance gap from its full-precision counterpart requires applying specific techniques and some network structure adjustments. This section has the two-fold purpose of describing the implementation and design criteria the author has taken and giving a gentle introduction to BNNs.

Training and Binary Function

As discussed in Section 2.2.3, post-training binarization is unsuitable for obtaining reasonably accurate BNNs. However, training BNNs with backpropagation is not directly applicable as it requires sufficient precision to allow gradient accumulation to work [151]. Courbariaux *et al.* solved this problem [25] with Straight Through Estimator (STE) [150]. The fundamental idea of STE is that the quantization function is applied in

the forward pass but skipped during backpropagation. STE keeps a set of full-precision weights denoted as proxies (W_F), which are binarized (W_B) on the forward pass to make a prediction and compute a loss. Any function can be used as a binarization function. Courbariaux *et al.* used $sign$ function:

$$W_B = sign(W_F)\,, \tag{5.1}$$

In the backpropagation phase, W_F is updated accordingly to the loss gradient as in a regular network:

$$\frac{\partial L_{oss}}{\partial W_F} = \frac{\partial L_{oss}}{\partial W_B}\,. \tag{5.2}$$

Activations are binarized similarly to the weights but do not need proxies as they are recomputed entirely in every forward pass. Fig. 5.3 shows the plots for the binarization function. In the forward pass, it behaves as the $sign$ function performing binarization. In the backward pass, the function returns a clipped identity of the gradient. Clipping the gradient when activations exceed 1.0 improves a binary model's accuracy:

$$\frac{\partial L_{oss}}{\partial a_F} = \begin{cases} \frac{\partial L_{oss}}{\partial a_B}, & \text{if } |a_F| \leq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{5.3}$$

where a_F is the full-precision input to the activation function and a_B is the correponding binarized output. The binary models presented in this work use $sign$ as a quantizer for both weights and activations and are trained with Larq [173]. Larq is a framework built on Keras [210] that fully supports the training of BNNs with STE.

Encoding Values

Binary encoding of weights and activations reduces dot products to a series of bitwise operations. In particular, representing logical 0 and 1 with -1 and 1 renders convolutions and matrix multiplications a series of XNOR and pop-count operations [25]. However, a dedicated compute engine or specific hardware is required to exploit the efficiency of binary arithmetic [25, 151]. A conventional compute engine for full-precision networks stores binary weights into 32-bit registers. As a result, multiply-accumulate computations (MAC) in BNNs require the same time and resources as in a full-precision network because binary operands are stored in floating-point variables. Compute engines for BNNs use the *Single Instruction, Multiple Data (SIMD) within a register (SWAR)* method [151, 25], which concatenates 32 binary operands into a 32-bit register and evaluates them simultaneously using bitwise operations. Typically, a binary MAC is implemented as follows:

$$a_1 \mathrel{+}= \text{popcount}(\text{xnor}(a_o^{32}, w_1^{32}))\,, \tag{5.4}$$

where a_o^{32} and w_1^{32} are sets of 32 input activations and weights. Although operand concatenation enables the computation of multiple binary MACs in parallel, $32\times$ speed-up is unrealistic. This limitation depends on several factors, including instruction scheduling, CPU pipeline stalls, and the hardware architecture above anything else. General-purpose CPUs and GPUs have specialized instructions for computing a floating-point MAC in a single clock cycle. Conversely, for binary operations, no such support exists [172]. Hence, a binary MAC results from multiple instructions on many hardware platforms, such as Nvidia GPUs [143] and ARM processors [144]. Therefore, if we let c_b represents the number of clock cycles to compute a binary MAC for a given hardware platform, the

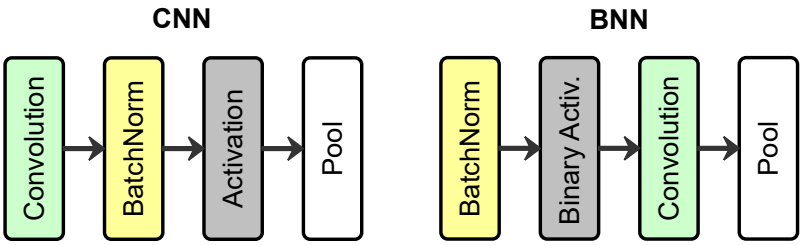


Fig. 5.4. A typical convolutional block in a CNN (left) and BNN (right).

obtainable speed-up is capped at $32/c_b$.

Batch Normalization

Batch Normalization (BatchNorm) [152] uses mini-batch statistics during training to adjust and scale activations. The central role of BatchNorm in full-precision networks is to speed up the training. In BNNs, BatchNorm is essential as it improves performance and helps training convergence [153, 154, 147]. It is worth mentioning that the parameters of BatchNorm layers cannot be binarized; however, they are few compared with the number of weights and contribute little to a model's size (Section 5.3.1).

Layer Order

As illustrated in Fig. 5.4, a convolutional block in a CNN consists of convolution, BatchNorm, activation, and pooling. BNNs achieve better performance if the order of the layers is as follows: BatchNorm, binary activation, convolution, and pooling [123]. This layer arrangement has a two-fold purpose. First, it allows for pooling from real values before binarization. Otherwise, the result would be a tensor dense in 'ones' which is proven to negatively affect the accuracy of a BNN [153]. Second, BatchNorm can replace bias as it works as a threshold for the subsequent layer [154, 147]. As bias parameters cannot be binarized, not using them reduces the memory and the number of full-precision MACs in

binary networks. The BNNs proposed in this thesis use only BatchNorm without biases, and convolutional blocks are as in Fig. 5.4.

First Layer Input

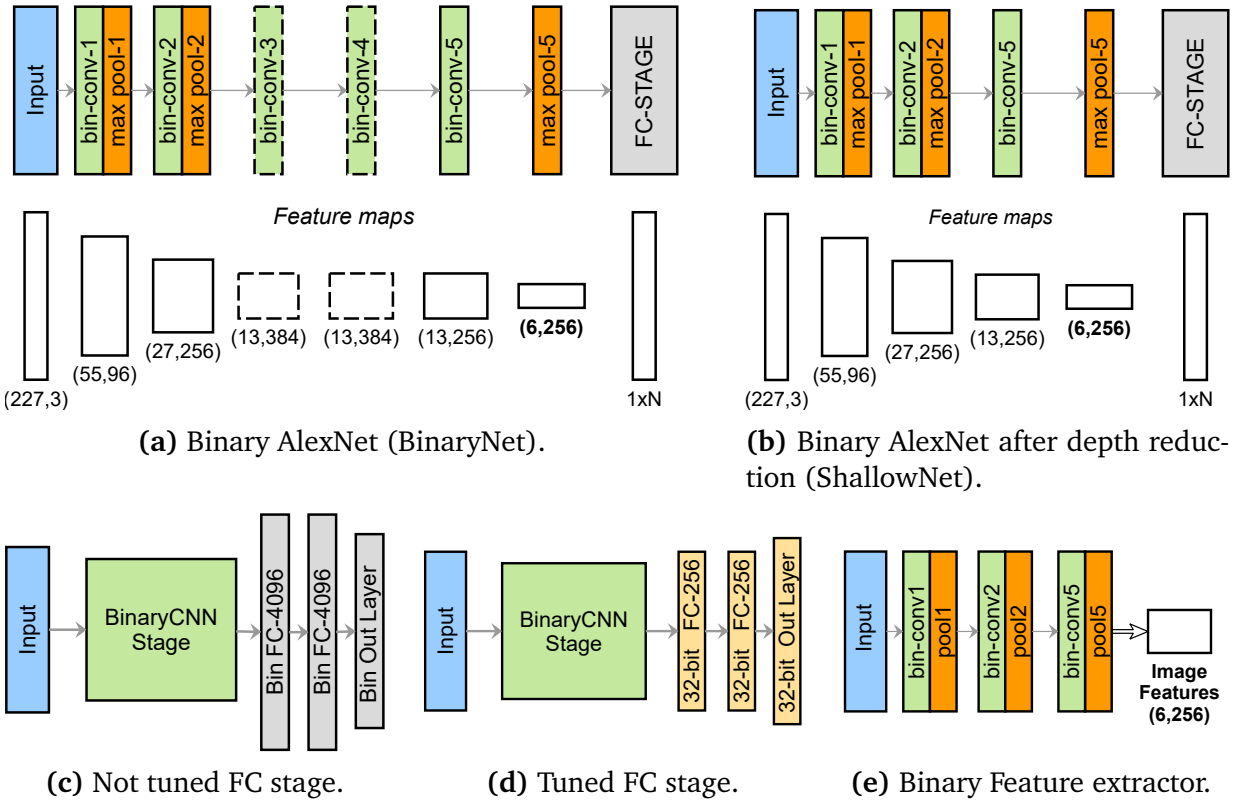
Full-precision inputs are recommended to improve a binary model's accuracy [151]. The model size is unaffected since the weights are binary. However, there is an impact on computational speed, which is generally acceptable when the convolution filters are few compared to deeper layers. Accordingly with this consideration, the binary networks presented in this work have the first convolutional layer directly connected to the input image with no binary activation and BatchNorm placed in the middle.

Padding

Convolutions are often padded with zeros in full-precision networks. This standard practice cannot be applied to BNNs that require padded values within the encoding set to enable bitwise operations. Zero-padding would expand the $\{-1, 1\}$ encoding to $\{-1, 0, 1\}$, rendering convolutions incompatible with bitwise operations. Therefore, all the BNNs presented here use one-padding accordingly with the weights and activation encoding employed.

5.2.2 Second Step: Depth Reduction

The primary motivation for depth reduction is to decrease the number of a model's parameters. Networks for classification are deep and can have dozens of convolutional levels [211]. However, VPR is a different task, and the author empirically demonstrate in Section 5.7 that achieving good performance in changing environments with fewer layers



(a) Binary AlexNet (BinaryNet).

(b) Binary AlexNet after depth reduction (ShallowNet).

(c) Not tuned FC stage.

(d) Tuned FC stage.

(e) Binary Feature extractor.

Fig. 5.5. Binarization (a), depth reduction (b) and FC tuning (c, d) applied to AlexNet. Depth reduction consists of removing *conv3* and *conv4* layers. The three pooling layers are kept to maintain the exact shape of the *pool5* feature map (d).

is possible. Not only the model size but also the computational efficiency of a network benefits from depth reduction. For example, the best model is obtained by removing the two intermediate convolutional layers from an AlexNet-like CNN, as shown in Figs. 5.5a and 5.5b. This operation decreases by 66% the amount of the weight, yielding significant model size and MACs reduction (Section 5.3.2). The network resulting from depth reduction is denoted by ShallowNet (Figs. 5.2 and 5.5b).

5.2.3 Third Step: Fully-Connected Stage Tuning

BNNs are highly optimized for classification. The fully-connected (FC) stage of classifiers is often populated with many neurons. AlexNet [26] and VGG-16 [212], for example, include 4096 units in each layer. When it comes to training a model for VPR, the hyper-parameters of the FC layers should be revised. For example, the top-performing binary model obtained used 256 32-bit neurons per FC layer during the training phase (Fig. 5.5d). Binary weights are a source of gradient noise [151] that renders the training more complex and longer to complete [213]. A 32-bit FC reduces the number of binary weights that need to be learned, making the training more stable. Smoother training has a lower chance of overshooting a loss function's minimum, facilitating a model's optimization. It is relevant mentioning that FC stage tuning is applicable only when VPR is carried out with convolutional features (Fig. 5.5e). This is the case of the proposed FloppyNet, which uses *pool5* features for VPR, as detailed in Section 5.3.2.

5.3 Binary Neural Networks for VPR

This sections details the BNNs employed in the experiments and defines the comparison baseline to evaluate the BNN resulting from the proposed approach describes in Section 5.2.

5.3.1 CNN Baseline and BinaryNet

The starting CNN is based on the AlexNet archetype [26], which is one of the most used network types for VPR [2, 96, 71, 72]. AlexNet-type networks' structure consists

TABLE 5.1: THE LAYER STRUCTURE OF BASELINE AND ITS BINARIZED VERSION, BINARYNET. THE TABLE IS SPLIT IN TWO ROWS FOR BETTER READABILITY.

Layers from conv1 to conv3

	conv1	**pool1**	**conv2**	**pool2**	**conv3**
Layer Setup	C(11,4,96)	P(2,2)	C(5,1,256)	P(2,2)	C(3,1,384)
Features Size	290400	69984	186624	43264	64896
Parameters (M)	0.03	0.04	0.65	0.65	1.54
Model Size (KiB)	136.5	137.3	2538	2540	5998
Total MACs (M)	105.42	105.42	553.31	553.31	702.83
Binarizable Par. (M)	0.03	0.03	0.65	0.65	1.53
Non-Binarizable Par.	0	0	192	192	704
Binary Model Size (KiB)	4.25	4.25	80	80	190
Binary Model Size (% of Baseline)	3.12	3.1	3.15	3.15	3.17

Layers from conv4 to fc7

	conv4	**conv5**	**pool5**	**fc6**	**fc7**
Layer Setup	C(3,1,384)	C(3,1,256)	P(2,2)	FC(4096)	FC(4096)
Features Size	64896	43264	9216	4096	4096
Parameters (M)	2.86	3.75	3.75	41.5	58.29
Model Size (KiB)	11186	14646	14648	162120	227704
Total MACs (M)	927.11	1076.63	1076.63	1114.38	1131.16
Binarizable Par. (M)	2.86	3.75	3.75	41.49	58.27
Non-Binarizable Par.	1472	2240	2240	2752	10944
Binary Model Size (KiB)	355	466	466	5076	7156
Binary Model Size (% of Baseline)	3.17	3.18	3.18	3.13	3.14

of several convolutional blocks (CB) alternated with pool layers followed by a fully-connected (FC) stage with one or more hidden layers.

The baseline CNN, denoted by ***Baseline*** in this chapter, is the same as a standard AlexNet [26] except for the use of BatchNorm and pool layers with a 2×2 non-overlapping kernel for higher accuracy [26]. Baseline is shown in Fig. 5.5a. It has five convolutional blocks followed by a fully-connected stage with two hidden layers, including 4096 neurons each. The detailed structure is described in Table 5.1 using the following notation. $C(k, s, h)$ indicates a convolutional block with kernel size k, stride s and h channels (filters). A similar notation is used for max pooling: $P(k, s)$. Fully-connected

TABLE 5.2: THE STRUCTURE OF FLOPPYNET. THE VALUES OF MODEL SIZE AND TOTAL MACS ARE INCREMENTAL.

FloppyNet and ShallowNet Features Extractors

	conv1	**pool1**	**conv2**	**pool2**	**conv5**	**pool5**
Layer Setup	C(11,4,96)	P(2,2)	C(5,1,256)	P(2,2)	C(3,1,256)	P(2,2)
1-bit parameters (M)	0.03	0.03	0.65	0.65	1.24	1.24
32-bits parameters (M)	0	0	192	192	704	704
Model Size (KiB)	4.25	4.25	80	80	154	154
Total MACs (M)	105.42	105.42	553.31	553.31	652.99	652.99
Param. % (BinaryNet)	100	100	100	100	33.1	33.1
Size Rate % (BinaryNet)	100	100	100	100	33.05	33.05
Size Rate % (Baseline)	3.12	3.1	3.15	3.15	1.05	1.05
MAC Rate % (BinaryNet)	100	100	100	100	60.7	60.7

layers are indicated with $FC(n)$, where n is the number of neurons. The model sizes and MACs reported in Table 5.1 are cumulative per network layer. For example, if the baseline is cut to use *fc6* features, the corresponding size of the model is 158.32 MiB and the MACs are 1.1 billion.

BinaryNet is the binary version of the baseline CNN. The bottom rows of Table 5.1 show the number of binarizable parameters and the corresponding model sizes. The remaining 32-bit parameters are due to BatchNorm. However, their contribution to the binary model size is negligible, around 0.6%. BinaryNet sizes vary from the 3.12% of Baseline at $conv1$, which is not preceded by a BatchNorm layer, to 3.18% at $pool5$.

5.3.2 FloppyNet and ShallowNet

FloppyNet consists of three binary convolutional blocks and three pool layers, as shown in Fig. 5.5e and detailed in Table 5.2. Binarization, jointly with depth reduction applied to Baseline, resulted in a compact model of 154 KiB. The layers removed in the depth reduction step are *conv3* and *conv4*. The output layer of FloppyNet is denoted as *pool5*

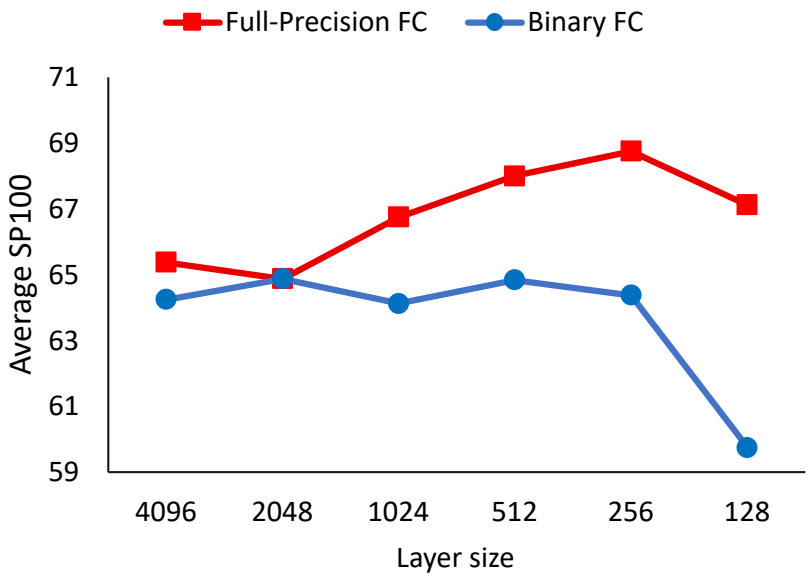


Fig. 5.6. Average S_{P100} across all datasets for full-precision and binary fully-connected stages at different layer sizes.

by convention. The author kept the same name as in the Baseline network since they have the same structure and provide feature vectors with the same shape and element number: $6 \times 6 \times 256 = 9216$ elements given an color input image of 227×227 pixels. The use of the last pooling stage for FloppyNet is motivated in Section 5.5, where the VPR skill resulting from binarization is analyzed layer by layer.

The primary motivation for FloppyNet is to reduce the model size and shorten the inference latency as much as possible while keeping a reasonable VPR accuracy. With two fewer convolutional layers, FloppyNet uses 33% of the memory of BinaryNet at the *pool5* layer, computing 39% fewer MACs (Table 5.2). Binarization and depth reduction steps cause a performance loss that is mainly compensated by tuning the FC stage properly for the training. The best FloppyNet model was obtained with 256 full-precision neurons in both *fc6* and *fc7*. Fig. 5.6 shows the VPR performance of the proposed model for various FC sizes. The performance peak corresponds to 256 full-precision neurons. This training approach's effectiveness is demonstrated in Section 5.6.1, where FloppyNet is compared

against its twin ShallowNet, which is, conversely, trained without tuning the FC stage.

Finally, it is relevant highlighting that FloppyNet's structure remains similar to the CNN from which it is originated (Fig. 5.5). Specifically, it is a general-purpose network such as its ancestor AlexNet but trained on a specific dataset to address VPR effectively.

5.4 Experimental Setup

This section details the experimental setup (including evaluation criteria, training, and test datasets) used to assess the VPR performance of the binary neural networks presented in Section 5.3.

5.4.1 VPR Performance Evaluation

Visual place recognition is cast as a retrieval task (Chapter 3). Reference images showing already visited locations are searched to find the best match with the robot camera's current view, namely the query image. VPR is considered successful when a query image is paired with one of the correct reference images. The image descriptors used to match images are obtained by L_2-normalization of a network's layer output:

$$D = \frac{\hat{X}_l}{||\hat{X}_l||_2}\,, \tag{5.5}$$

where $\hat{X}_l$ is the output of the l^{th} layer. Descriptors are compared using Euclidean distance; the shorter the distance, the higher the similarity between two images.

$$d = ||D_1 - D_2||_2\,, \tag{5.6}$$

where D_1 and D_2 are the image descriptors to be compared. The reference image with the shortest distance from the query is regarded as the current location.

Following the approach described in Chapter 3, VPR is evaluated on a whole dataset with S_{P100} index (Eq. 3.8) along with Z-test validation (Eq. 3.10) as needed.

5.4.2 Memory Allocation Efficiency

VPR performance is also evaluated in relation to memory requirements. Memory efficiency is defined as the ratio of the model size to S_{P100}:

$$\eta_m = \frac{M_{size}}{S_{P100}}\,. \tag{5.7}$$

η_m measures the memory cost per S_{P100} point, expressing the trade-off between memory usage and VPR performance. The lower η_m, the more efficiently the model uses memory, M_{size}. Memory efficiency is proposed to generalize the trade-off analysis between accuracy and parameter density [122] to low-precision networks whose memory footprint also depends on weight quantization. Hence, the use of model size instead of the number of weights in Eq. 5.7. Moreover, η_m can be applied to networks having the same structure but different weight precision to determine the relationship between VPR performance and quantization, providing additional information to characterize a low-precision network or choose the optimal quantization for an application.

5.4.3 Processing Time and Power Usage

The processing time per image, T_i, and power usage, P_w, are used to determine the computational and energy efficiency of deployed models. T_i is the time required to

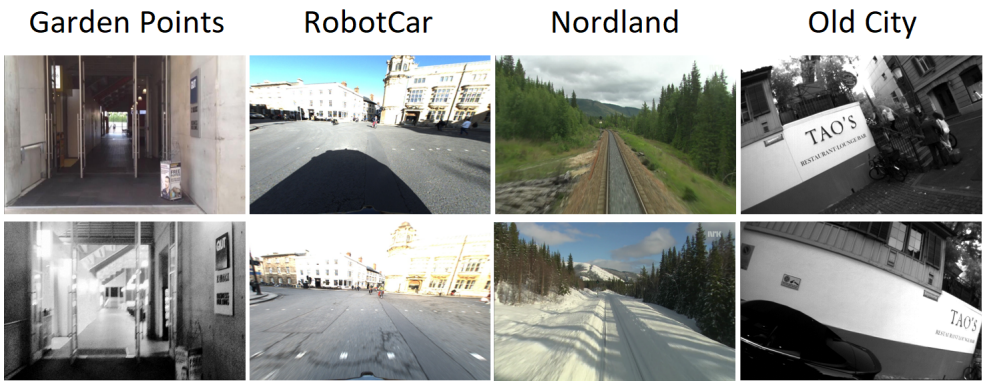


Fig. 5.7. A corresponding image pair from each test dataset.

process an image. For a neural network, it corresponds to the inference latency. The time intervals to load an image and consume the output are excluded from the measurement so that T_i reflects the actual computational effort for an image representation. P_w is measured directly on the hardware platform. It is used to determine the inference energy cost:

$$E_i = \int_{T_i} P_w(t)dt\,. \tag{5.8}$$

5.4.4 Training Data

The dataset used to train all the models is Places365 [92, 214]. It is a place-themed dataset consisting of 1,803,460 images divided into 365 categories with between 3068 and 5000 images in each category. The validation set used for the experiments includes 100 images per location class.

TABLE 5.3: TEST DATASETS AND GROUND TRUTH TOLERANCE.

Dataset	Appearance Variation		Reference Images	Query Images	Ground Truth
	Viewpoint	Condition			
GardenPoints (Day-Night Right)	Mild to moderate lateral shift	Night-Day	200	200	± 2 frames
Norland (Summer-Winter)	None	Seasons	1622	1622*	± 5 frames
Old City	Extreme 6-DOF	None	5408	5641*	by authors
RobotCar Cross-Seasons	Mild lateral shift	Illumination Dynamic Elements	206	202*	± 5 frames

(*) A RANDOM SUBSET OF 200 IMAGES IS USED FOR THE EXPERIMENTS.

5.4.5 Test Data

On long-term runs, the working space undergoes changes that alter the appearance of places a robot visits at different times or from different directions. In order to provide comprehensive results, experiments use four datasets to include several types of appearance variations. All datasets have two subsets that correspond to different traversals of the environment, of which Fig. 5.7 shows some examples. The reference dataset (I_M) represents the previous knowledge of the environment, and the query dataset (I_Q) represents the current traversal. The datasets include a different number of query images. To compute fair average performance indicators such as the average S_{P100}, 200 query images are randomly sampled from each of them for a total of 800 images. The test datasets were previously presented in Section 2.1.5. For the reader's convenience, their characteristics are summarized below and in Table 5.3.

GardenPoints [8] includes three loops of the Queensland University of Technology (QUT). The experiments employed Right-Day and Right-Night to test VPR under illumination changes and mild lateral shifts. Ground truth is built by frame correspondences

with a tolerance of ± 2 frames [110].

Nordland [116] includes a set of four traversals captured along a rail track in Norway in every season. The experiments employed Summer and Winter journeys as reference and query datasets, respectively. The ground truth is built with a tolerance of ± 5 frames [110].

Old City [4] is a urban dataset with two traversals showing the same location from different perspectives to generate 6-DOF viewpoint variations. The ground truth data is available from the authors [196].

RobotCar Cross-Seasons [1] is a subset of the Oxford RobotCar dataset [115] consisting of two sequences of 206 sunny query images and 202 dusk reference images recorded on board a car driving in an urban environment. This dataset includes illumination changes, mild lateral viewpoint shifts, and dynamic elements such as pedestrians, cars, and shadows. Ground truth is built by frame correspondences with a tolerance of ± 5 frames.

5.5 Breaking Down BNNs: Analyzing Layer Performance

The first question to answer when a convolutional network is employed as a feature extractor is: *"which layer is the most suitable to build a distinctive image descriptor?* This section provides a VPR performance assessment of the features from every layer in both Baseline network and its binary counterpart, BinaryNet. CNNs can learn features at different levels of abstraction. Convolutional features retain some spatial information. However, as the depth increases, pool layers induce the loss of such spatial information in favor of translation invariance. In fully-connected layers, the activation of a neuron

TABLE 5.4: S_{P100} [%] FOR EVERY LAYER IN BASELINE (TOP) AND BINARYNET (BOTTOM).

Baseline Image Features

	conv1	**pool1**	**conv2**	**pool2**	**conv3**	**conv4**	**conv5**	**pool5**	**fc6**	**fc7**
Garden Point	14.5	29	56.5	62.5	67	79.5	66	84	73	56.5
RobotCar	80.1	89.3	87.4	90.3	91.3	92.2	82	93.2	90.8	74.3
Nordland	69	75.5	86.5	91	91	95	54	85	40	29
Old City	7	11	7	9	11	18	13	33	39	37
Average	42.7	51.2	59.4	63.2	65.1	71.2	53.8	**73.8**	60.7	49.2

BinaryNet Image Features

	conv1	**pool1**	**conv2**	**pool2**	**conv3**	**conv4**	**conv5**	**pool5**	**fc6**	**fc7**
Garden Point	3	11.5	43	51.5	75.5	74.5	76	79.5	66.5	39.5
RobotCar	61.2	73.3	81.4	81.9	82.4	82.4	83	83	84	63.6
Nordland	53	60	71.5	71	77.5	77	72	71.5	38.5	14
Old City	9	10.5	11.5	15	16	15	20.5	23	49	44.5
Average	31.6	38.8	51.9	54.9	62.9	62.2	62.9	**64.3**	59.5	40.4

depends on every neuron in the previous level. Hence, the spatial information vanishes while improving the invariance to viewpoint changes and translation [89]. As BNNs' features have yet to be investigated for use in VPR applications, a second question needs an answer: *"how does binarization impact the features and their VPR matching performance?"*

The answers to these questions are given in Table 5.4, which shows S_{P100} for every layer of Baseline and BinaryNet. In Baseline, fully-connected layers and deeper convolutions handle viewpoint changes better than the initial layers. *Fc6* and *fc7* obtain the highest performance under the extreme viewpoint changes that characterize Old City. *Pool5* is the best on GardenPoints, which includes mild viewpoint shifts other than day-light variations. On the other hand, shallower layers deal better with conditional changes. The best layer for the seasonal variation of Nordland is *conv4* with $S_{P100} = 95\%$. These results partially confirm the findings of a previous study on AlexNet [89], which indicates *conv3* as the best layer to deal with conditional changes while *pool5* and, in

some cases, *fc6* as the best choice to deal with viewpoint changes. Moving to BinaryNet, it is straightforward to notice how binarization negatively affects VPR performance, but the characteristics of the layers are more or less unchanged. Table 5.4 shows that *pool5* achieves the highest performance on the same dataset as for Baseline. Similarly, fully-connected layers outperform the others on Old city.

The average S_{P100} reported at the bottom of Table 5.4 is computed across all the datasets as they formed a single environment to simulate a workspace exhibiting various appearance changes. From this perspective, *pool5* is the layer that guarantees the best performance across the four datasets. The average S_{P100} is 73.8% for Baseline and 64.3% for its binary counterpart. The author considers the gap acceptable (9.5%), especially considering that BinaryNet requires only 3.18% of the memory of Baseline at *pool5* (Table 5.1). The main goal of addressing conditional changes and results obtained from the layer analysis, suggested the use of *pool5* as an output layer for FloppyNet (Section 5.3.2).

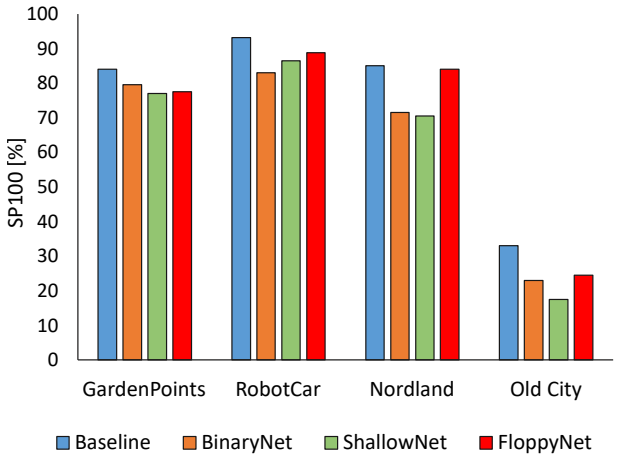
5.6 VPR Performance Evaluation

FloppyNet is compared against several other networks. These include Baseline, BinaryNet and ShallowNet from Section 5.3.1, HybridNet [5], VGG-16 [212], CALC [94], and a 8-bit implementation of FloppyNet. ShallowNet is like FloppyNet but trained with regular fully-connected layers of 4096 binary neurons, as described in Section 5.2.2. HybridNet is a version of AlexNet [26] with an additional convolutional block trained on ImageNet [87] and tuned on SPED [5]. To avoid training data influencing the results, the author trained an HybridNet model by replacing ImageNet with Places365, the

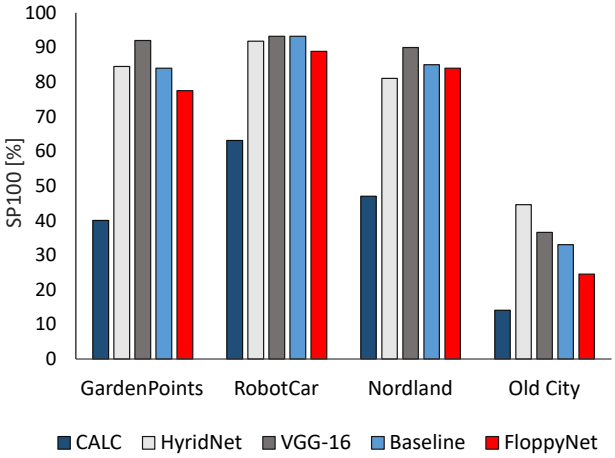
TABLE 5.5: MODELS' PERFORMANCE AND EFFICIENCY FOR RASPBERRY PI 4 IMPLEMENTATIONS.

Network	bits	S_{P100} (avg)	Params [M]	Size [KiB]	η_m [KiB]	MACs [M]	T_i [ms]	FPS	P_w [w]	E_i [mj]
HybridNet	32	75.4	5.07	16957	224.89	1098 M	137.8	7.26	2.616	360.5
Baseline	32	73.8	3.75	14648	198.48	1077 M	132.4	7.55	2.616	346.4
VGG-16	32	77.9	14.7	57487	737.96	15.3 B	992.1	0.45	2.616	2595
BinaryNet	1	64.3	3.75	466	7.25	1077 M	21.6	46.3	2.536	54.8
ShallowNet	1	62.9	1.24	154	2.45	653 M	18.2	54.95	2.536	46.2
FloppyNet	1	68.7	1.24	154	2.24	653 M	18.2	54.95	2.536	46.2
FloppyNet-8	8	71.4	1.24	1213	16.99	653 M	33.2	30.12	2.876	95.5
FloppyNet-32	32	72.9	1.24	4843	66.43	653 M	76.1	13.14	2.616	199.1
CALC	32	40.5	0.137	537	13.26	186 M	45.1	22.17	2.616	118.0

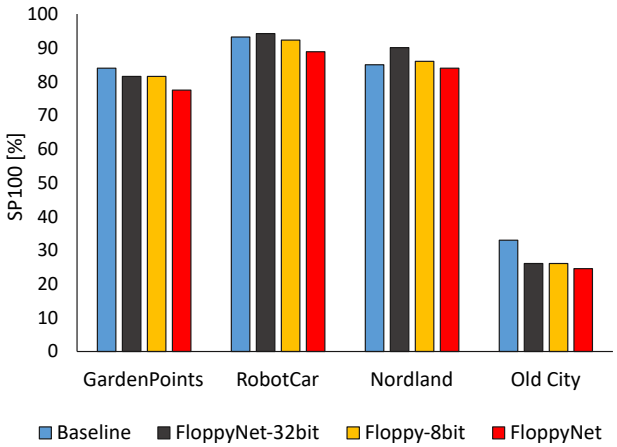
dataset used to train the other networks considered for the experiments. The results for HybridNet are obtained with the deepest convolutional layer available. VGG-16 [93] is a very deep network if compared to FloppyNet since it includes 13 convolutional blocks. It is relevant to include VGG-16 in the comparison because several multi-staged VPR methods widely use it as a feature extractor. Some examples are R-MAC [96], Cross-Region-Bow [2] and NetVLAD [71]. The VGG-16 model has been trained from scratch using Places365, and the features used for the tests are from the very last pool layer. CALC [94] is a lightweight CNN designed to address VPR with low resource requirements. CALC includes about 137 K parameters: a small fraction of 3.75 M of Baseline and 1.24 M of FloppyNet (Table 5.5). The results are given for the model trained on Place365 shared by the authors. The 8-bit version of FloppyNet is included in the comparison to show that BNNs also scale well to 8-bit quantization, demonstrating the potential applicability of binarization as a more efficient yet effective approach for VPR .



(a) Baseline, BinaryNet and ShallowNet.



(b) Several full-precision networks.

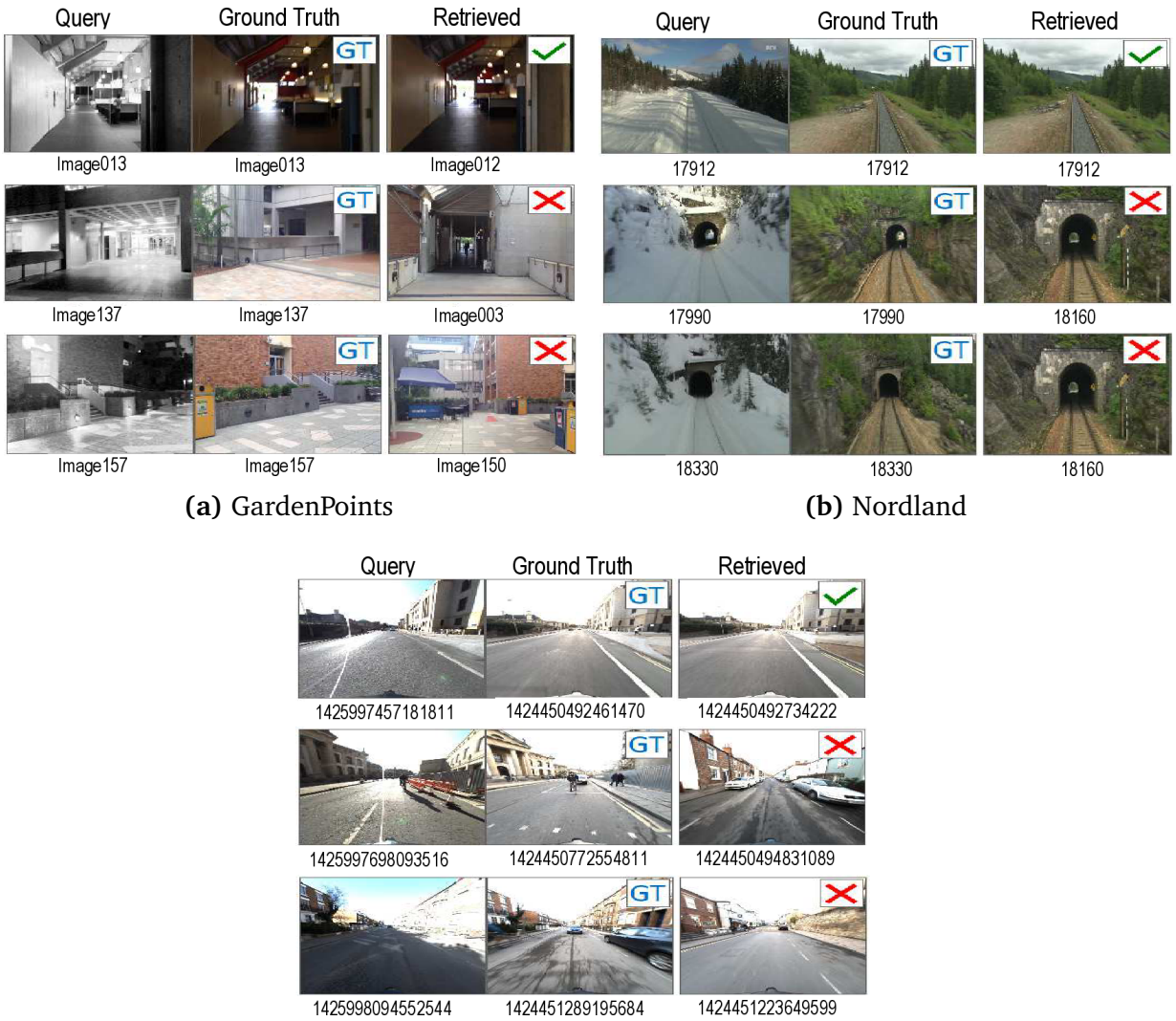


(c) Binary to 8-bit and full-precision.

Fig. 5.8. FloppyNet is compared against several CNNs and BNNs.

5.6.1 Comparison with the baseline

FloppyNet aims to achieve similar performance as the starting Baseline network with higher efficiency. Fig. 5.8a shows comparative results between FloppyNet, Baseline and the intermediate design steps: BinaryNet and ShallowNet. Binarization and depth reduction negatively impact the VPR performance. BinaryNet and ShallowNet score the lowest S_{P100} on every dataset, exhibiting a substantial gap from Baseline. FloppyNet outperforms BinaryNet and ShallowNet on every dataset confirming that tuning the fully-



(a) GardenPoints **(b)** Nordland

(c) RobotCar Cross-Seasons

Fig. 5.9. Some significant query results from GardenPoints (a), Nordland (b), and RobotCar Cross-Seasons (c) datasets.

connected stage during the training (Section 5.2.3) mitigates the performance loss due to binarization and depth reduction.

Baseline generally has better performance than FloppyNet. However, the difference is slight on GardenPoints, Nordland, and Robotocar due to a few places with some particular characteristics. The most difficult locations to recognize for FloppyNet are those presenting substantial viewpoint variations. An analysis on GardenPoints mismatches

confirms such a FloppyNet's weakness. The S_{P100} gap on GardenPoints is due mainly to lateral shift, which is generally mild but in a few locations, where FloppyNet fails while Baseline succeeds (Fig. 5.9a). This FloppyNet's shortcoming is also reflected by the S_{P100} gap of 8.5% from Baseline on Old City, which is where the performance difference is the largest. FloppyNet scores almost the same S_{P100} as Baseline on Nordland, 84% vs 85%. The mismatches are mainly between locations showing tunnel entrances. Fig. 5.9b shows two examples of mismatch. FloppyNet retrieves the same reference image on two different queries including a tunnel. FloppyNet scores a good S_{P100} on RobotCar as well. The S_{P100} difference with Baseline is caused by a series of wrong matches in the two locations shown in Fig. 5.9c. The high illumination contrast and the occlusions due to dynamic elements, such as cars and shadows, are possibly the cause of FloppyNet's failures.

5.6.2 Comparison with CNNs

Full-precision and deeper networks obtain better VPR performance than the proposed binary network (Fig. 5.8b). Substantial gaps are exhibited on Garden Points and Old City by VGG-16 and HybridNet, respectively. On the other hand, those two networks are far larger than FloppyNet even without considering the weights' precision. Table 5.5 shows additional data on the average VPR performance across the four datasets. FloppyNet scores an average S_{P100} of 68.7% using 1.24 M binary parameters and computing 653 M MACs. The highest average S_{P100} is achieved by VGG-16, 77.9%. However, VGG-16 includes 14.7 M weights and computes 15.3 B floating-point MACs resulting in two orders of magnitude longer inference latency. The small number of parameters penalizes CALC, which FloppyNet outperforms by a large margin on every dataset. Moreover, its low

	1-bit vs 8-bit	1-bit vs 32-bit	8-bit vs 32-bit
GardenPoints	-2.6	-0.23	2.1
Nordland	-0.5	-2.2	-1.4
Old City	-1.3	-1.5	0.71
RobotCar	-2	-3	-1.7

Fig. 5.10. Pairwise $Z@0.5$ scores for FloppyNet (1-bit), FloppyNet-8 (8-bit) and FloppyNet-32 (32-bit). Underlined values indicate a confidence interval $\geq 95\%$.

VPR performance is not compensated by a sufficient computational efficiency as CALC is about two times slower than FloppyNet, as shown in Table 5.5 and discussed later in Section 5.8.

5.6.3 Weight Quantization: Impact on Performance

Fig. 5.8c presents a comparison between FloppyNet model against its 8-bit and full-precision versions to examine the impact of weight quantization on place recognition capabilities. Increasing the quantization bits has a positive effect on VPR performance. FloppyNet-8bit slightly outperforms FloppyNet in every test scenario and nearly closes the gap with Baseline except on Old-City. Indeed, the reduced depth affects VPR performance on significant viewpoint changes, accordingly to the finding of the layers analysis in Section 5.5. The larger performance gap occurs for GardenPoints, 6%. These performance differences are validated by the Z scores shown in Fig. 5.10. Indeed, FloppyNet is comparable to the 8-bit version on Nordland and Old-City while inferior on GardenPoints and Robocar. On Old-City FloppyNet is also comparable to the 32-bit model with a $|Z|$ value of 1.5. As subsequently discussed in Sections 5.6.4 and 5.8, this slightly lower performance of FloppyNet is largely compensated by its higher memory and computational

efficiency compared to the 8-bit and 32-bit versions. Finally, increasing a model's precision from 8 to 32 bits does not significantly improve the VPR capabilities of the proposed network. As shown in Fig. 5.8c, 8-bit and 32-bit models score similar S_{P100} on every dataset and the Z scores confirm their close performance level except on GardenPoints, where the 8-bit model is the best.

5.6.4 Weight Quantization: Impact on Memory Efficiency

Table 5.5 shows the memory efficiency for all the compared networks. S_{P100} is the average score on the four datasets. Binary networks have smaller η_m values compared to any full-precision network. FloppyNet requires 2.24 KiB per S_{P100} point, while CALC, the most memory-efficient CNN, requires 13.26 KiB. ShallowNet has the same size as FloppyNet but has a lower S_{P100}, hence it uses memory less efficiently: $\eta_m = 2.45$ KiB. FloppyNet-8bit performs better on average than the 1-bit model by 2.7% but requires about eight times the memory. The considerably higher η_m of 16.99 KiB reflects a trend indicating that an increase in the model size does not correspond to a proportional increase in the VPR performance.

5.7 Binarization, Depth Reduction and FC-256

Fig. 5.11 shows S_{P100} relatively to Baseline resulting from using binarization (Bin), depth reduction (Depth), and FC stage tuning (FC256) separately and their relevant combinations. The features used to obtain the results are from the *pool5* layer.

Depth reduction (Depth) yields a full-precision network with better performance on Nordland and RobotCar. Depth reduction makes the output layer of a model retaining

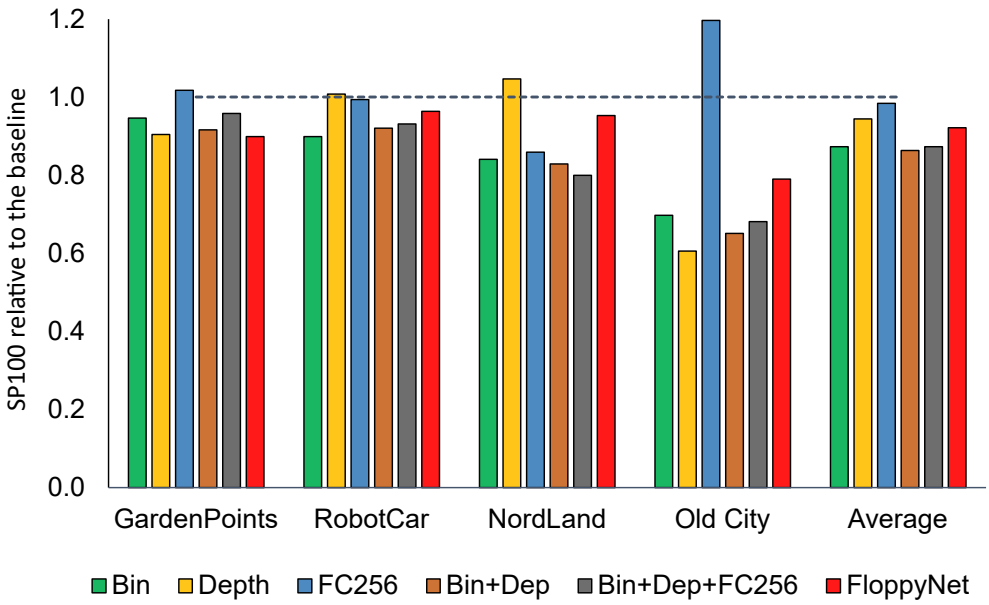


Fig. 5.11. S_{P100} relative to Baseline (dotted line) of several combinations of binarization, depth reduction, and FC-Tuning.

more spatial information compared to Baseline. Shallower layers are more suitable to deal with conditional changes (Table 5.4). Hence, the better performance on Nordland and Robotcar datasets that present significant conditional changes while none or mild viewpoint variations. Training a full-precision model with 256 neurons in the FC stage (FC256) helps VPR significantly to tackle extreme 6-DOF viewpoint variations. FC256 model achieves 22% higher S_{P100} on Old City than the original model trained with 4096 neurons in the FC stage. In classifiers, the FC stage is usually sized as large as possible to maximize accuracy while avoiding overfitting. Conversely, the empirical evidence shows that VPR benefits from a smaller FC stage. These results suggest that the FC stage has different roles in classification and VPR tasks.

Depth reduction does not help S_{P100} scores of the binarized models. The values of S_{P100} for binary (bin) and shallow binary networks (bin+dep) are very close to each other on every test dataset (Nordland in particular), which is rather unexpected consid-

ering the full-precision case (Depth). The red bars in Fig. 5.11 represent FloppyNet, which implements all the steps of the proposed approach. The addition of FC stage tuning counters the S_{P100} loss due to binarization and depth reduction in every tested scenario (except GardenPoints) supporting the effectiveness of the proposed training method.

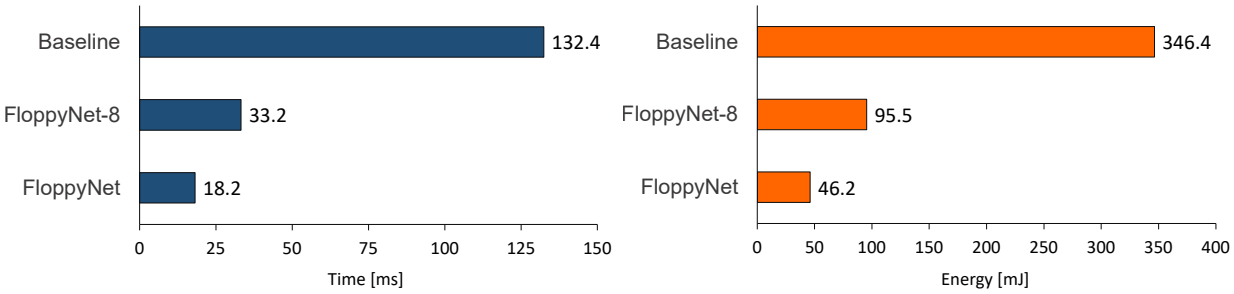
5.8 Computing and Energy Usage Benchmarks

The framework used to deploy binary models is Larq Compute Engine (LCE) [172]. LCE consists of a model compiler and a kernel to compute binary convolutions within the TensorFlow Lite runtime environment (TFlite) [145, 142]. LCE has been a natural choice as it is part of the Larq ecosystem [173] used to train the binary models presented in this work. The 8-bit and full-precision implementations use the built-in TFlite compute kernel.

Processing time indicates a deployed model's time to complete an inference, namely to compute an image representation. Energy usage is determined from the power usage of a deployed model using Eq. 5.8. The platform employed for the experiments is a Raspberry PI4 (RPI4) that sits on an ARMv8 Cortex-A72 running at 1.5 GHz [215]. The operating system is Ubuntu 20.04 Linux, 64-bit.

5.8.1 Processing Time and Computation Speed-Up

Fig. 5.12a compares the inference latency of Baseline and two FloppyNet implementations: 1-bit and 8-bit. The values reported in the figure are an average of 100 runs using four threads, namely employing all the cores available in the RPI4's CPU. FloppyNet has



(a) Processing time. **(b)** Energy usage.

Fig. 5.12. Processing time (a) and energy usage (b) for one image. The measurements are obtained with 4 threads on a Raspberry PI4.

a latency of $18.2\,ms$, which is about seven times shorter than Baseline's ($132.4\,ms$).

Table 5.5 shows the processing time for all the tested models. FloppyNet is considerably faster than any other full precision CNN including CALC, which computes only 186 M MACs, 28.5% of FloppyNet's MACs. However, it takes a considerably longer processing time than the proposed network, $45.1\,ms$ against $18.2\,ms$, while achieving considerably lower VPR performance (Fig. 5.8b). BinaryNet's latency enables an analysis of the speed-up contributions provided by binarization and depth reduction. BinaryNet has the same structure as Baseline except for the binary weights and executes in $21.6\,ms$, resulting in six times faster than its full-precision counterpart. Depth reduction removes two convolutions from BinaryNet, shortening the execution by a further 16% to the $18.2\,ms$ measured for FloppyNet. Finally, latency measures demonstrate that binarization also scales well to 8-bit quantization. FloppyNet-8bit completes an inference in $33.2\,ms$, resulting in almost twice slower as FloppyNet.

5.8.2 Energy Usage

The power usage of a deployed model is measured with an ampere meter connected to the USB-C power port of the RPI4. The energy spent per image, E_i, depends on the power absorbed by the RPI4, P_w, and processing time, T_i (Eq. 5.8). P_w is stable during an inference depending only on a model's weights precision because of the computational kernel used: LCE for binary models and TFlite for 8-bit and full-precision models. Hence, Eq. 5.8 becomes $E_i = P_w T_i$, where P_w is the constant power usage measured during image processing. P_w is $2.54\,W$, $2.62\,W$ and $2.88\,W$ for binary, 32-bit and 8-bit models. As T_i varies in a broader range than P_w, E_i depends mainly on a model's processing time (Fig. 5.12). The rightmost columns of Table 5.5 show the power absorption and energy usage for every deployed model. BNNs consume less energy than any other considered network. In particular, FloppyNet spends $46.2\,mJ$ per image, which is considerably more energy-efficient than the 8-bit implementation and Baseline usage of $95.5\,mJ$ and $346.4\,mJ$, respectively.

5.9 Comparison with Handcrafted Descriptors

This section compares FloppyNet with several handcrafted image descriptors relevant for VPR applications [28]. They include HOG [43], GIST [10] and CoHOG [60]. The results show that the proposed BNN has significantly better VPR capabilities while having comparable or higher computational efficiency than the considered handcrafted descriptors.

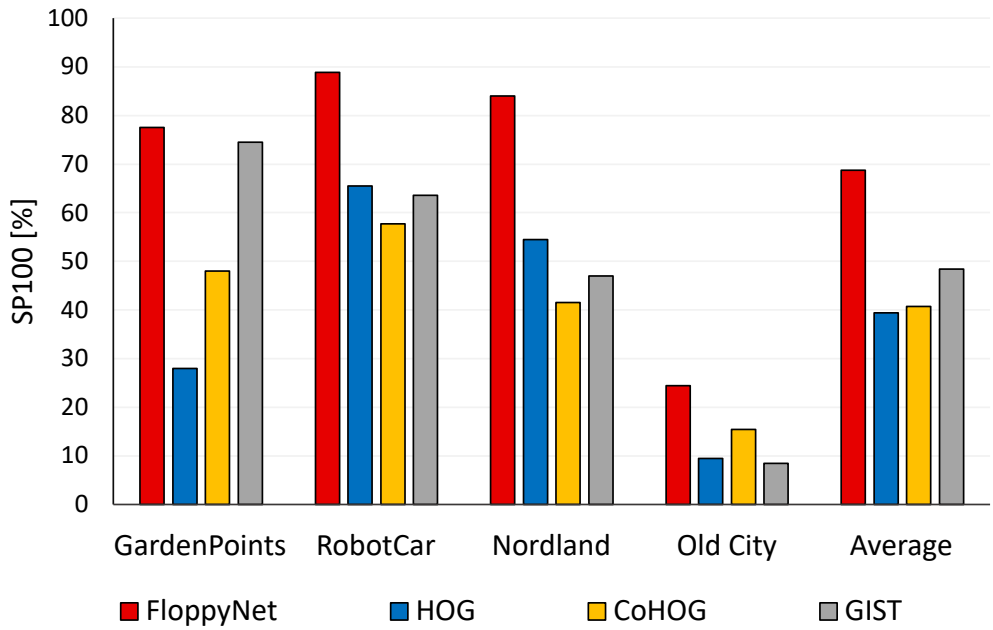


Fig. 5.13. VPR performance of HOG, CoHOG and GIST compared to FloppyNet.

5.9.1 Handcrafted Descriptors Setup

The platform employed for measuring the processing time is a Raspberry PI4 (RPI4), as in the previous section. The handcrafted descriptors setting used for the experiments are as follows. For HOG [43] is used the OpenCV 4.5.0 implementation with a cell size of 16×16 and block size of 32×32, as suggested in [28]. The input image size is set to 256×256 pixels, which is similar to FloppyNet's input size of 227×227 pixels. CoHOG [60] is tested using the code shared by the authors with their recommended settings [216]: cell size of 16×16, 8 bins and entropy threshold of 0.4. The image size is 256×256 pixels. Gist [10] is available in the *C* library *Lear's GIST* through the *pyleargist* python wrapper [217]. Gist is used with the parameters indicated by the authors [218]: 4 blocks and 8 orientations per scale. The image size is set to 128×128 pixels to keep the Gist's latency comparable with the other methods. Indeed, using 256×256 images extends

Gist's processing time by roughly four times.

5.9.2 Results Discussion

Figs. 5.13 and 5.14 shows the comparison results. FloppyNet achieves substantially better VPR performance on every dataset scoring an average S_{P100} of 68.7%. Gist achieves the highest VPR performance among the handcrafted descriptors but results in the slowest one taking $105\,ms$ to process an image.

The proposed BNN is the fastest technique when running using all the four RPI4's cores. However, Gist and HOG implementations cannot run on multiple threads. Hence, the inference latency for FloppyNet running on a single thread is reported in Fig. 5.14 for a fair comparison. Only HOG executes faster than the proposed network taking $20.4\,ms$ to process an image instead of the $39.1\,ms$ required by FloppyNet (1-thread). On the other hand, HOG exhibits a wide VPR performance gap on every dataset. HOG scores an average S_{P100} of 39.4%, whereas FloppyNet achieves $S_{P100} = 68.7\%$. The author concludes that the shorter latency of HOG does not compensate for the poor VPR performance it achieves compared to the proposed binary network.

5.10 Summary and Further Considerations on Processing Time

This chapter presented FloppyNet, a compact binary network to address VPR. FloppyNet achieves slightly lower or comparable VPR performance to deeper and full-precision networks in changing environments having a drastically smaller model size and substantial

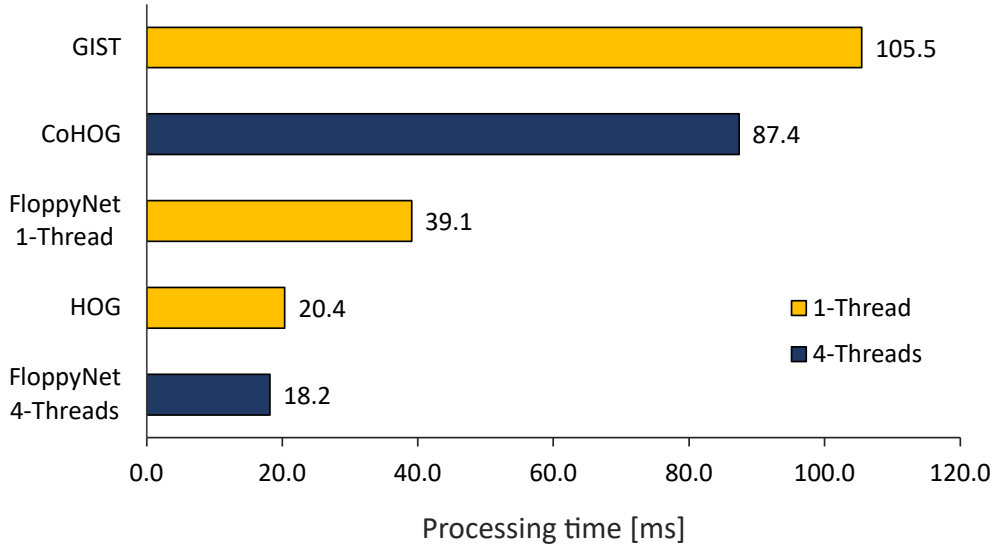


Fig. 5.14. Processing time of HOG, CoHOG and GIST compared to FloppyNet.

computational speed-up. Such a lightweight network offers several opportunities for embedded systems and edge computing in general. FloppyNet may be employed to enable VPR on very cheap hardware or replace standard CNNs to free up resources to allocate for additional functionalities to improve a robot's navigation system or increase the frame rate on low-cost embedded applications. For example, NetVLAD is a two-stage image descriptor that uses VGG-16 to extract image features that are subsequently post-processed to compute a robust image representation. VGG-16 is a large network that requires a relatively long time to extract image features. If a BNN such as FloppyNet is used instead, NetVLAD's memory requirements and computational efficiency would improve dramatically. This example suggests that a natural extension of this work investigates the applicability of BNNs in multi-stage descriptors that use a CNN as a feature extractor. This research hypothesis is further discussed later in Chapter 7 with other possible research developments.

A second developing direction for this research arises from the processing time measurements discussed in Section 5.8.1. Binarization alone improves the speed by six times, taking the processing time from $134.4\,ms$ of Baseline to $21.6\,ms$ of BinaryNet. Depth reduction removes almost 40% of the MACs from BinaryNet, but the execution time decreases only by 16%. Hence, a large share of the remaining computational effort occurs elsewhere in the network. The next chapter investigates this problem extensively, identifying the first convolutional layer as the slowest stage of FloppyNet and providing a solution to shorten its processing time significantly.

Chapter 6

Highly Efficient Binary Neural Networks for Visual Place Recognition[1]

Chapter 5 demonstrates how BNNs can be employed successfully as a more efficient alternative to CNNs to address VPR in changing environments. FloppyNet is seven times faster than AlexNet and twice as fast as CALC, a tiny network designed for efficient loop closure detection. However, FloppyNet's processing time and energy usage can be improved significantly by addressing the first convolutional layer, which is not completely binarized for better performance. In particular, the first layer is the slowest network stage, requiring a large share of the entire computational effort to process an image. This chapter shows how to combine *depthwise separable factorization* and *binarization* to design an in-place replacement for the first convolutional layer of a BNN to improve computational and energy efficiency. The best model presented here performs similarly to FloppyNet, requiring 50% of the processing time and energy usage.

[1]This work was published and presented at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 5493-5500. DOI: 10.1109/IROS47612.2022.9981978.

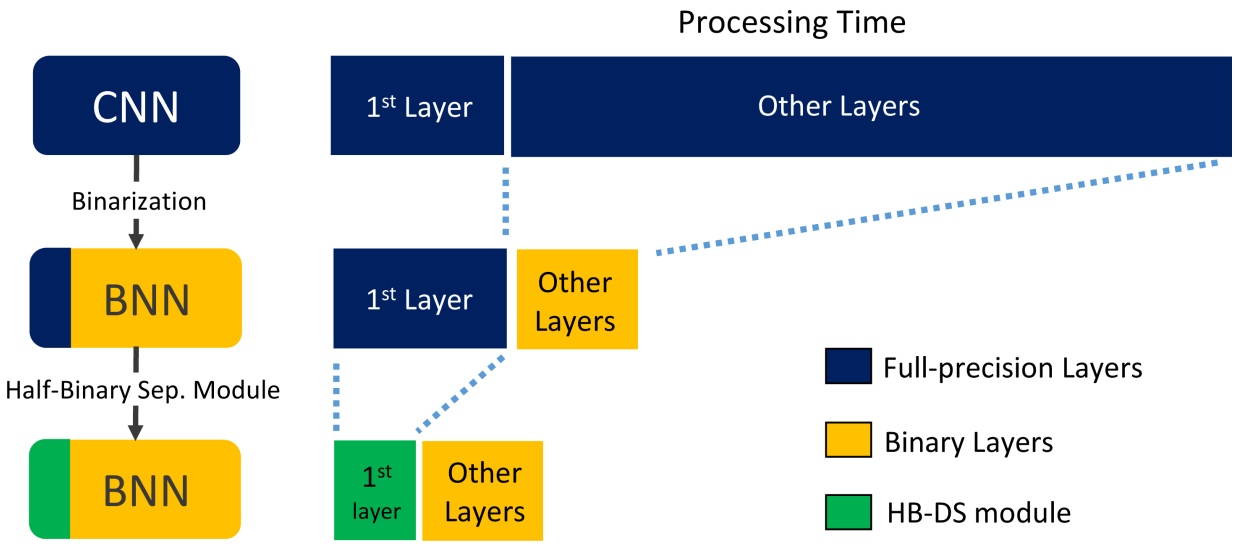


Fig. 6.1. The first layer bottleneck problem in BNNs addressed by the HB-DS module.

6.1 The First Layer Bottleneck Problem

Mobile robots need to track their position within the workspace to operate autonomously. As part of the navigation system, place recognition is fundamental in the localization process. Considering that mobile robots often sit on resource-constraint hardware and are battery-powered, computational and energy efficiency are not less important than VPR accuracy. In Chapter 5, BNNs [25] are proposed as a more efficient yet effective alternative to CNNs for enabling VPR in resource-constraint contexts. The key idea of BNNs is using a single bit to encode weights and activations, allowing compact model sizes and bitwise operations to achieve high computational efficiency, as described in Section 5.2.1. Although a BNN can be one order of magnitude faster than a CNN, there is a substantial margin for improvement. For better performance, the first layer of BNNs takes high-precision inputs [151]. Hence, the first convolution is incompatible with bitwise operations resulting in the most inefficient stage of a BNN, as exemplified in Fig. 6.1.

Such a bottleneck problem is particularly relevant for VPR as many techniques use a relatively small number of convolutions [94, 90, 2, 96]. Therefore, the first layer computes a significant part of the total operations to process an image that cannot be binarized without impacting the VPR performance. For example, FloppyNet has only three convolutions and spends about 84% of the entire processing time on the first one, which has binary weights but takes high-precision inputs.

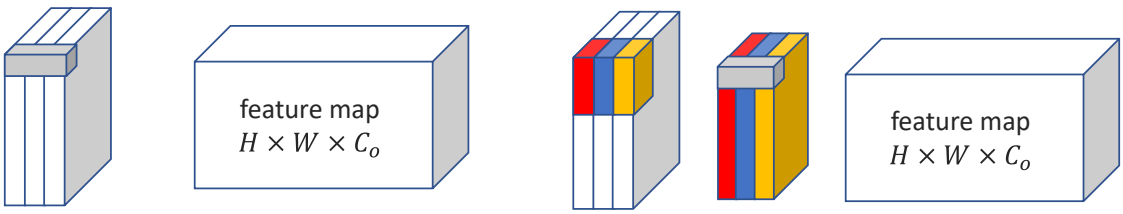
This chapter addresses the bottleneck of the first convolutional layer in BNNs by proposing the *Half-Binary Depthwise Separable* (HB-DS) module. HB-DS combines depthwise separable factorization [132, 133] with binarization to enhance the computational efficiency of a BNN without affecting the VPR performance. The HB-DS module is then used to design a BNN that can be tuned to train models to several performance-efficiency trade-offs to meet various application requirements. The best model trained[2] achieves the same VPR accuracy as FloppyNet requiring 50% of the time and energy.

6.2 Unblocking the Latency Bottleneck

The first convolutional layer is crucial for a BNN's performance A common practice in BNN design is using high-precision inputs because binarization negatively affects performance [151]. Consequently, the first convolutional layer is incompatible with bit-wise operations resulting in the slowest stage of a BNN, as demonstrated in Section 6.4.3.

This section presents *Half-Binary Depthwise Separable* module (HB-DS) to address the first layer bottleneck problem. It is a configurable module that can replace the first convolution in a BNN without significantly adjusting the existing network architecture.

[2]https://github.com/bferrarini/Half-Binary-Depthwise-Separable-Module

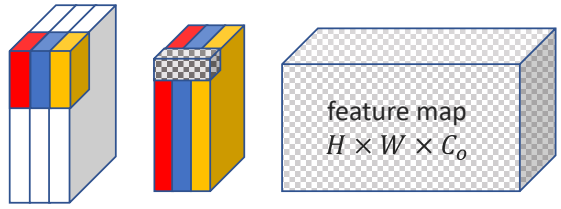


convolution: $(3, k, s, c_o)$

(a) Convolution.

depthwise: $3 \times (k, s, 1)$ + pointwise: $(3,1,1,c_o)$

(b) Depthwise separable factorization.



depthwise: $3 \times (k, s, c_o)$ + pointwise: $(3,1,1,c_o)^{bin}$

(c) HB-DS module.

Fig. 6.2. A standard convolution (a) compared to depthwise separable factorization (b) and HB-DS module (c).

6.2.1 Depthwise Separable Convolutions

The proposed approach uses depthwise separable factorization to split a convolution into two separate layers: a depthwise convolution and a pointwise convolution [132, 133]. The depthwise convolution operates on the input channels individually, resulting in an output feature map with the same number of channels as the input. The pointwise layer consists of a 1×1 convolution that builds a new map from the depthwise layer's features. Fig. 6.2 illustrates the idea underlying depthwise separable decomposition. This factorization has the purpose of reducing the computational complexity of the original convolution. The term *complexity* is used here as a synonym for the number of multiply-accumulate operations (MACs) computed by a convolution. Hence, having fewer MACs means lower computational complexity. Let us assume a convolutional layer takes an

input tensor $T_{in} = h_i \times w_i \times c_i$ and uses a kernel, $k \times k$, to output a feature map $T_{out} = h_o \times w_o \times c_o$. The computational cost is:

$$C_{conv} = (k^2 \cdot c_i) \cdot h_o \cdot w_o \cdot c_o \ , \tag{6.1}$$

where $(k^2 \cdot c_i)$ is the cost for a single element in T_{out}. A depthwise convolution convolves the input channels individually, creating a feature map having the same depth, c_i, as the input tensor. Fig. 6.2b shows an example of a depthwise convolution processing a three-channel tensor (e.g., a color image). The computational cost of a depthwise convolution is as follows:

$$C_{depth} = k^2 \cdot c_i \cdot h_o \cdot w_o \ . \tag{6.2}$$

The subsequent pointwise stage is a standard convolution with $k = 1$:

$$C_{point} = c_i \cdot h_o \cdot w_o \cdot c_o \ . \tag{6.3}$$

The total computational cost of a depthwise separable convolution is:

$$C_{sep} = C_{depth} + C_{point} = c_i \cdot h_o \cdot w_o \cdot (k^2 + c_o) \ . \tag{6.4}$$

Compared to a standard convolution, the depthwise separable factorization reduces the complexity by:

$$\frac{C_{conv}}{C_{sep}} = \frac{k^2 c_o}{k^2 + c_o} \ . \tag{6.5}$$

The larger the kernel, more effective is the depthwise separable factorization.

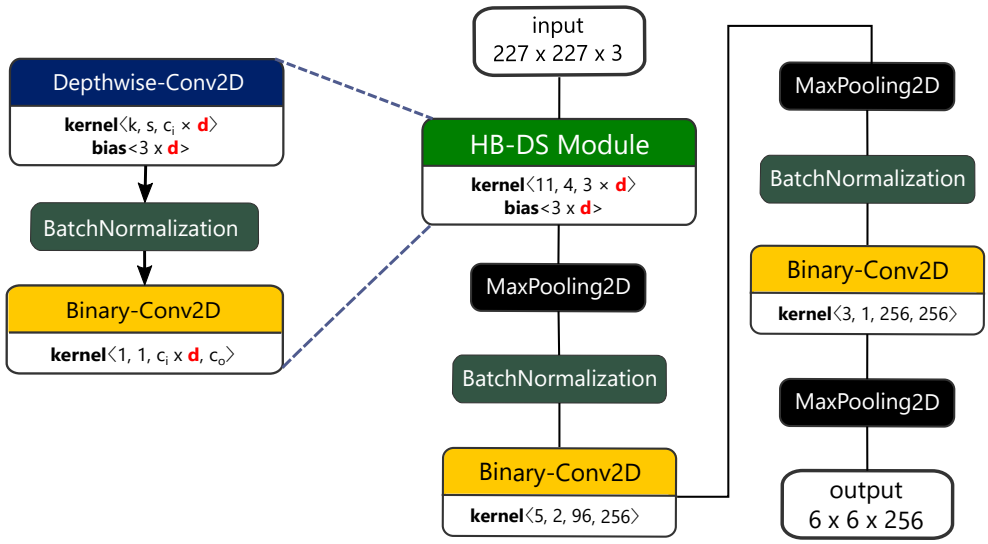


Fig. 6.3. HB-DS module implementation (left) and its placement as a first stage in FloppyNet (right). d denotes the depth multiplier, k the kernel size, s the stride, c_i the input channels and c_0 the output channels.

6.2.2 Half-Binary Depthwise Separable Convolutions

The *Half-Binary Depthwise Separable* module (HB-DS) proposed here consists of a depthwise separable factorization whose pointwise convolution is binarized. (Fig. 6.2c). As a result, the depthwise layer takes full precision inputs preventing performance loss, while the subsequent pointwise convolution is binary for high computational speed. The share of binary MACs in HB-DS is:

$$\frac{C_{point}}{C_{sep}} = \frac{c_o}{k^2 + c_o} \,. \tag{6.6}$$

Conversely, the full precision MAC are those in the depthwise convolution:

$$\frac{C_{depth}}{C_{sep}} = \frac{k^2}{k^2 + c_o} \,. \tag{6.7}$$

If $c_o > k^2$ the effect of binarization is dominant on factorization. Conversely, the complexity reduction is primarily due to factorization. The implementation of the HB-DS module

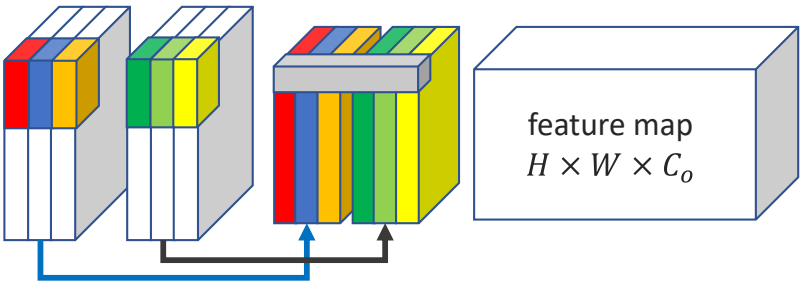


Fig. 6.4. Depthwise separable factorization with a depth multiplier of 2.

is illustrated in Fig. 6.3 (left). A batch normalization layer [152] is placed before the binary convolution to improve a model's accuracy and training convergence [153].

HB-DS includes a *depth multiplier*, d, to make it tunable for different scenarios. The depth multiplier is number of kernels the depthwise convolution uses on the input to create a ticker, and then, more information-rich feature map of $d \cdot c_i$ channels. Fig. 6.4 shows an example for $d = 2$. The application of a depth multiplier increases the computational complexity of the deptwise convolution by d times. Therefore, Eq. 6.5 is rewritten as follows:

$$\frac{C_{conv}}{C_{sep}} = \frac{k^2 c_o}{d\,k^2 + c_o}\,. \tag{6.8}$$

On the other hand, the VPR performance of a BNN improves as d increases. Section 6.4.3 demonstrates the use of d as a tuning parameter to obtain various trade-offs while keeping HB-DS faster than an ordinary convolutional layer.

6.2.3 BNN Setup for HB-DS Benchmarking

The network used for the experiment is a version of FloppyNet enhanced with HB-DS, as illustrated in Fig. 6.3. The HB-DS module uses the same hyperparameters as the original FloppyNet's convolution: a 11×11 kernel, stride of 4 and 96 output channels.

The rest of the network is as in the original design to highlight the impact of HB-DS on VPR performance and efficiency.

6.3 Experimental Setup

The proposed network is trained with several depth multipliers, d. The resulting models are assessed on VPR under various appearance changes. A model's efficiency is evaluated using processing time and energy usage as criteria.

6.3.1 VPR Performance

VPR is cast as a retrieval problem, as in the previous chapters. A query image representing the current robot's camera view is compared to the reference images showing the previously visited locations. The image descriptor for a model is obtained from the vectorized output of a convolutional or pooling layer by L_2-normalization:

$$D = \frac{\hat{X}_l}{||\hat{X}_l||_2} \, , \tag{6.9}$$

where $\hat{X}_l$ is the output of the l^{th} layer. The similarity between the two images is determined using cosine:

$$s = \frac{D_1 \, D_2}{||D_1|| \, ||D_2||} \, . \tag{6.10}$$

The reference image scoring the highest similarity with the query is regarded as the current location. VPR performance is measured on a dataset using several criteria, including S_{P100} (Eq. 3.8), average EP (Eq. 3.3) and AUC.

TABLE 6.1: TEST DATASETS AND GROUND TRUTH TOLERANCE.

Dataset	Appearance Variation		Reference Images	Query Images	Ground Truth
	Viewpoint	Condition			
GardenPoints (Day-Night Right)	Mild to moderate lateral shift	Night-Day	200	200	± 2 frames
SPED Test	None	Night-Day; Seasons; Dynamic Elements	1000	200	± 5 frames
RobotCar Cross-Seasons	Mild lateral shift	Illumination Dynamic Elements	206	202	± 5 frames
Norland (Summer-Winter)	None	Seasons	1622	1622	± 5 frames
Old City	Extreme 6-DOF	None	5408	5641*	by authors
Combined	All above		8436	1000	Mixed

(*) ONLY A SUBSET OF 200 IMAGES IS USED FOR OLD CITY TO KEEP THE EXPERIMENTS WITHIN A REASONABLE DURATION.

6.3.2 Processing Time and Energy Usage

The processing time, T_i, and power usage, P_w, are evaluated following the same criteria as is in Section 5.4.3. They are acquired from deployed models and VPR algorithms running on a test hardware platform. T_i is the time required to elaborate an input image. The image loading and preprocessing (e.g. reshaping) are excluded so that T_i reflects the actual computational complexity of a network or an algorithm.

The energy per image processed, E_i, indicates the energy spent to compute a single image representation. It is determined from the power usage with Eq. 5.8, which is rewritten below, knowing that the Raspberry Pi4 used for the experiments absorbs a stable amount of power during an inference (Section 5.8.2):

$$E_i = T_i \, P_w \, . \tag{6.11}$$

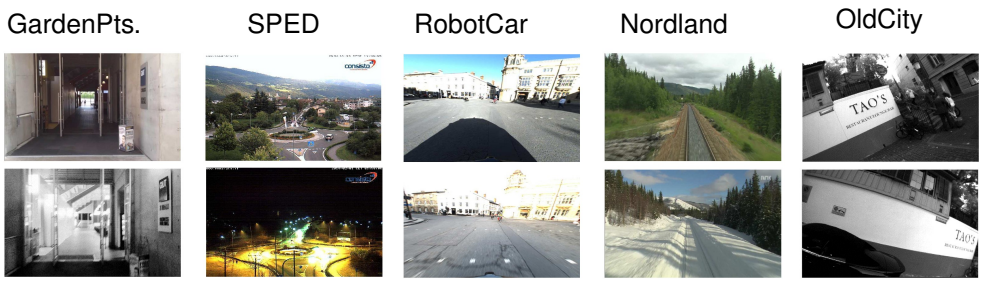


Fig. 6.5. A matching pair from every test dataset.

6.3.3 Training Data

All the binary models are trained from scratch using Place365 [92] within the Larq framework [219]. Places365 is a place-themed dataset consisting of 1,803,460 images divided into 365 classes, including between 3068 and 5000 samples. The validation set includes 100 images per category.

6.3.4 Test Data

VPR assessment is carried out under various image variations that a robot encounters over extended runs using five datasets, each containing one or more appearance changes. They include: GardenPoints [8], 200 places randomly sampled from SPED [5], the Cross-Season sequence from RobotCar [1], Nordlands [116] and Old City [4]. Table 6.1 summarizes each dataset's characteristics and ground truth criteria. All of them include a reference set representing the knowledge of the environment and a query set representing the current view of a robot's camera. Fig. 6.5 shows some examples of matching pairs. *Combined* is a sixth dataset included to simulate a large complex environment. The reference set is the union of the other five datasets; the query set includes 200 randomly sampled images from each one for a total of 1000 queries. The combined dataset is intended to provide more realistic global performance measures than averaging the

results from five datasets tested individually.

6.4 Results Discussion

This section presents the results for the BNN described in Section 6.2.3, discussing both the VPR performance and efficiency. The experiments include several models of the proposed BNN trained with different depth multipliers, d. By convention, HB***X***-FN denotes the use of HB-DS with $d = X$. The computational time and energy usage are measured from models deployed on a Raspberry Pi4 (RPI4) [215] using Larq-Compute-Engine (LCE) [172]. In the first part of this section, HB12-FN is compared to other VPR techniques. The author selected HB12-FN as the best-balanced model between computation and performance. However, the experiments involved multiple models trained with different d values. The second part of this section discusses the depth multiplier as a tuning parameter presenting the results for various values of d.

6.4.1 Comparative Analysis

The VPR networks and methods included in the comparison are FloppyNet (Chapter 5), VGG-16 [93], CALC [94], CoHOG [60] and HOG [43]. The CNNs and setting of the hand-crafted techniques are the same used in Section 5.9. The VGG-16 model used for the experiment is optimized on Place365. The results for CALC are obtained with the model trained on Place365 shared by the authors. For CoHOG, the experiments employed the source code and the parameters shared by the authors for an image size of 256×256 pixels [216]. Finally, HOG is used for 256×256 pixel images with a cell size of 16×16 and a block size of 32×32, as suggested in [28].

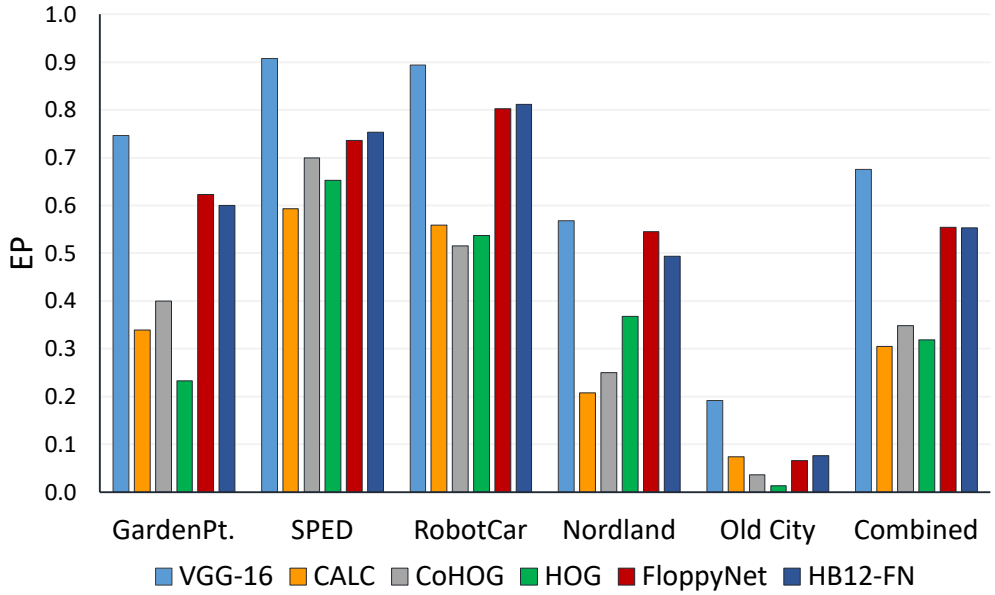


Fig. 6.6. VPR performance on different appearance changes.

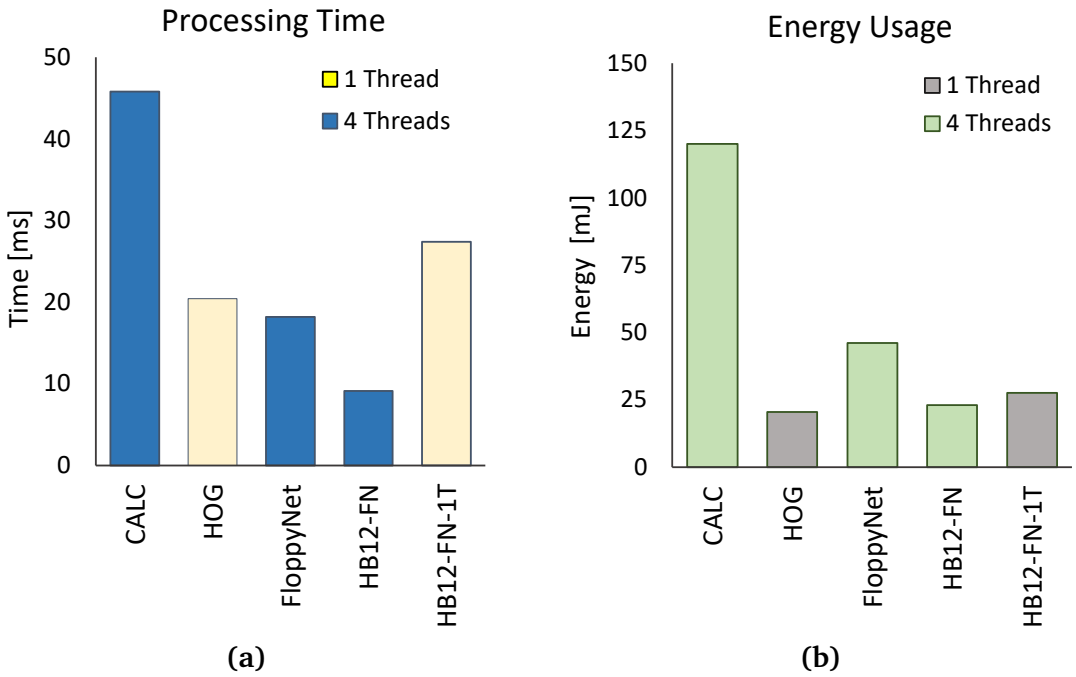


Fig. 6.7. Processing time (a) and energy usage (b) of the proposed BNN compared to other VPR methods.

TABLE 6.2: VPR MEASUREMENTS ARE GIVEN FOR THE COMBINED DATASET. T_i AND E_i ARE MEASURED ON A RASPBERRY PI4.

VPR	Type	T_i [ms]	E_i [mJ]	VPR (on Combined)		
				avg. EP	avg. AUC	S_{P100}[%]
VGG-16	CNN	995.7	2608.7	0.676	0.679	80.5
CALC	CNN	45.8	120	0.306	0.323	37.1
CoHOG	Trainless	87.4	210.6	0.349	0.37	42.5
HOG	Trainless	20.4	20.6	0.318	0.335	38.6
FloppyNet	BNN	18.2	46.2	0.554	0.568	67.3
HB12-FN	BNN	9.1	23.1	0.553	0.566	67.2
HB12-FN-1T	BNN	27.4	27.7	0.553	0.566	67.2

Fig. 6.6 shows the EP score for all the considered VPR methods. HB12-FN and FloppyNet perform equally well on the Combined dataset[3], while on GardenPoints and Nordland, the latter achieves slightly higher performance. HB12-FN outperforms by a substantial margin the other lightweight techniques: HOG, CALC, and CoHOG. VGG-16 captures the highest EP score in every environmental condition and on the Combined dataset, as also confirmed by the S_{P100} and AUC reported in Table 6.2. These results are not surprising considering the VGG-16's large size and depth. However, VGG-16 requires about $1s$ to compute an image descriptor, resulting in two orders of magnitude slower than any considered BNN. The complete set of T_i is reported in Table 6.2 while Fig. 6.7a compares a selection of the most efficient techniques: the BNNs, CALC, and HOG. HB12-FN is the fastest one taking $9.1\,ms$ to process an image, 50% of FloppyNet's inference time. Considering these two BNNs have similar VPR performance, HB-DS significantly speeds up the original FloppyNet's computation. The HOG implementation used for the experiments (OpenCV 4.5.0) can run only on a single thread. For a fair comparison, Fig. 6.7a shows the processing time of the proposed model for 1-thread execution (1T) using yellow bars. HOG takes about $7\,ms$ less than HB12-FN-1T to compute a descriptor.

[3]The corresponding $|Z|@0.5$ is lower than 1.96.

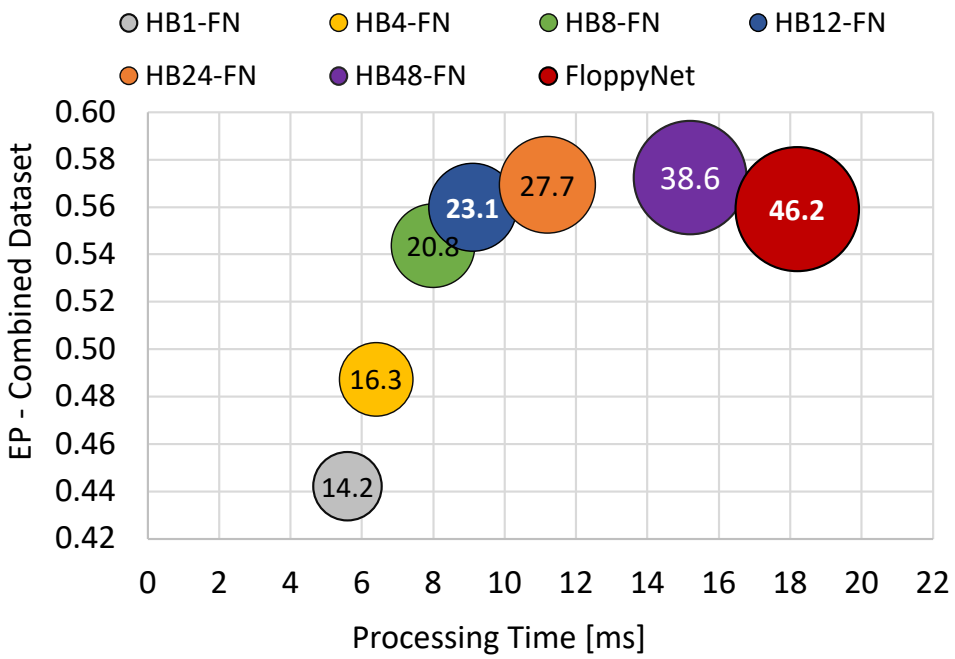


Fig. 6.8. EP versus processing time for several depth multipliers. The circles' area represents the energy usage in mJ per processed image.

However, their VPR performance is very different. While HOG scores $EP = 0.318$, HB12-FN achieves 0.553. Such a performance gap corresponds to 28.6% less place correctly recognized in the Combined dataset (S_{P100} column in Table 6.2). The author evaluate this gap is too wide to consider HOG as a good alternative to the proposed BNN.

6.4.2 Energy Usage

The energy usage, E_i, is reported in Table 6.2 and Fig. 6.7b. E_i is determined using Eq. 6.11 from the average power usage measured directly from a RPI4 on 100 consecutive runs. RPI4 has an approximately continuous power usage during runtime. Thus, E_i is mainly influenced by the number of active CPU cores and interference time. To this end, the processing time reduction due to HB-DS contributes to energy saving, which

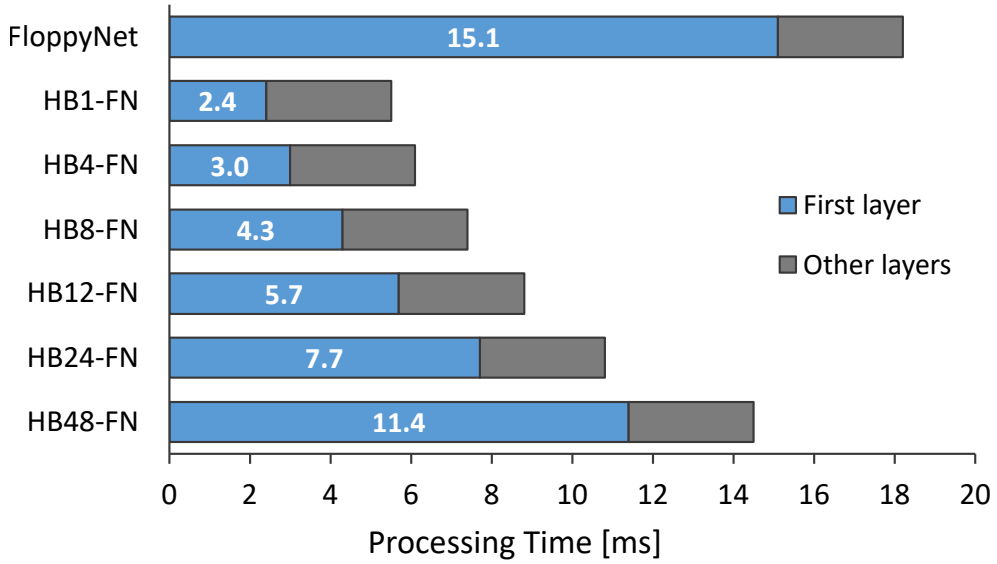


Fig. 6.9. Processing time for several depth multipliers.

is essential for battery-supplied robotic platforms. The positive effect of HB-DS is well depicted by the energy difference between FloppyNet and HB12-FN as they differ only in the first stage. HOG is the most energy-efficient technique. However, as motivated above, HOG has too low VPR performance to be considered instead of HB12-FN.

6.4.3 Depth Multiplier as a Tuning Parameter

The HB-DS design enables the performance tuning of a BNN by acting only on the depth multiplier, d, without changing any other network parameter. Fig. 6.8 plots the EP score on the Combined dataset versus the processing time for several depth multipliers. The circles' diameter represents the energy usage in mJ per processed image. HB12-FN, the model selected for the comparison presented above, reaches the same VPR performance as FloppyNet, spending one-half of the processing time and energy. Nevertheless, the VPR performance can be further improved by increasing d. For example, HB24-FN and

TABLE 6.3: PERFORMANCE AND EFFICIENCY FOR SEVERAL IMPLEMENTATION OF THE PROPOSED BNN. T_i AND E_i ARE MEASURED ON A RASPBERRY PI 4.

BNN	First Stage						T_i [ms]		E_i [mJ]	VPR (Combined Dataset)		
	Structure (k,s,c_o,d)	d	Params		MAC [M]							
			32 bit	1 bit	32bit	1bit	First	Total		EP	AUC	S_{P100}[%]
HB1-FN	HD-BS(11,4,96,1)	1	366	288	1.1	0.87	2.4	5.6	14.2	0.442	0.458	53.2
HB4-FN	HD-BS(11,4,96,4)	4	1464	1152	4.4	3.5	3.0	6.4	16.3	0.487	0.511	57.5
HB8-FN	HD-BS(11,4,96,8)	8	2928	2304	8.8	7.0	4.3	7.7	19.6	0.554	0.566	67.8
HB12-FN	HD-BS(11,4,96,12)	12	4392	3456	13.2	10.5	5.7	9.1	23.1	0.553	0.566	67.2
HB24-FN	HD-BS(11,4,96,24)	24	8784	6912	26.4	20.9	7.7	10.9	27.7	0.569	0.587	68.1
HB48-FN	HD-BS(11,4,96,48)	48	17568	13824	52.7	41.8	11.4	15.2	38.6	0.575	0.591	69.0
FloppyNet	C(11,4,96)	N/A	34848	0	105.4	0	15.1	18.2	46.2	0.554	0.568	67.3

HB48-FN outperform HB12-FN by a small margin at the cost of longer processing time and energy usage. If the target application has enough resources, HB24-FN and HB48-FN might be good options. On the opposite side, lower d values can find application in reducing a BNN's complexity to fit extremely resource constraint hardware or saving energy to extend battery life. For example, the EP loss from HB12-FN to HB1-FN is 0.11, which corresponds to -14% correctly matched places in the Combined dataset (S_{P100} in Table 6.3). While HB12-FN is possibly the best option in many scenarios, HB1-FN might be preferred when energy saving is a strict requirement as it spends less energy than HB12-FN to process an image: $14.2\ mJ$ against $23.1\ mJ$. It is worth mentioning that HB1-FN, which holds the worst VPR performance among binary models, outperforms HOG, CALC, and CoHOG while achieving higher computational efficiency (Tables 6.2 and 6.3). Finally, Fig. 6.9 shows T_i for several depth multipliers. The blue bars represent the time spent on the first layer, which depends on d. The gray bars are for the other layers. FloppyNet uses a regular non-binary convolution as a first stage. The time spent on the first convolution is 84% of the entire processing time. The networks using HB-DS have a more fair distribution of the latency between the first and the other layers proving that the proposed approach mitigates the bottleneck problem of BNNs.

6.5 Summary and Further Applications for HB-DS

BNNs are an efficient class of deep neural networks using binary arithmetic to speed up convolutions. The slowest stage in a BNN is the first convolutional layer, non-binary for higher accuracy. This chapter proposed an enhanced version of FloppyNet requiring one-half of the resources of the original version. Such a significant efficiency boost is obtained using the newly proposed HB-DS, which significantly reduces the number of non-binary operations in BNNs. Another feature of HB-DS is enabling performance tuning of a BNN by acting only on a single parameter to train binary models suitable for different deploy scenarios. An extension of this work is investigating HB-DS for BNN designed and optimized for different tasks than image matching and different network architectures (e.g., ResNet [131]) and hyperparameters. Also, Eqs. 6.7 and 6.6 suggest further research: the kernel size and the number of output channels greatly influence the number and distribution of non-binary operations in HB-DS. Future experiments will investigate different kernel sizes and depths to provide new insights into the HB-DS module's capabilities.

Chapter 7

Conclusions and Future Directions

Visual Place Recognition is a well-established and relevant topic for both academia and industry, with a vast literature dedicated to this subject. As highlighted in this thesis, dealing with appearance changes can make VPR difficult, but the challenge is even harder when faced with limited runtime resources. This is a common scenario in the real world, given the widespread use of tiny robots for autonomous navigation and the need to complete long-term operations, which can be susceptible to dynamic viewing conditions.

This thesis primarily addresses the problem of improving the efficiency of VPR by exploring different approaches to build distinctive image representations while keeping the computational effort as low as possible. The first approach investigated in this manuscript employs local features computed by lightweight, handcrafted algorithms. While they exhibit satisfying performance in matching images captured from a different perspective, they fail under environmental changes such as those due to illumination and seasonality. Conversely, deep learning-based techniques achieve better accuracy under those changes but at a considerably higher computational cost. As a result of this obser-

vation, the main contribution of this thesis is an efficient image feature extractor based on Binary Neural Networks. The best model trained achieves competitive performance to standard Convolutional Neural Networks under conditional changes while requiring only a fraction of the computational effort. However, the proposed binary network is quite weak against viewpoint changes because it relies only on convolutional features to build an image representation. In the author's opinion, this is the most relevant problem left open by this thesis. Hence, the work presented here should be extended to multi-staged VPR techniques to improve viewpoint invariance. An example is NetVLAD, which achieves good performance on both viewpoint and conditional changes using a VLAD-like stage to process convolutional features. One of the author's aims for his post-doc research is to investigate this direction, providing BNNs with a post-processing stage to achieve both viewpoint and conditional invariance.

All the contributions presented in this manuscript are peer-reviewed and published in international conferences and journals after being through a series of revisions. Compared to the published papers, the chapters in this manuscript have a few changes and additions to provide more comprehensive insights into the problems investigated by the author and make this thesis self-contained as much as possible. Chapter 3 extends the corresponding paper providing some theoretical background on PR-Curves and details on how they plotted that could not be included in the published version to keep the manuscript length within the page limit. The results presented in Chapter 4 have been reworked using the framework introduced in Chapter 3, while the conclusions drawn are unchanged. This chapter's content is also enriched with a new comparison between local image descriptors and CNN-based techniques to demonstrate the higher efficiency of the former and the better VPR performance under conditional changes of the latter.

Chapters 5 and 6 differ only in layout, text, and figure enhancements from their published versions.

7.1 Summary of Contributions

This thesis presents the author's research performed in visual place recognition to achieve the PhD requirements The significant contributions presented in this thesis are summarized below.

- Chapter 3 addresses the problem of VPR evaluation by proposing a framework specifically designed for this domain. It is based on a new metric, Extended Precision (EP), designed to extend the well-established R_{P100} to measure low-level performance, which cannot be captured otherwise. An evaluation framework is then built around EP, which offers two types of analysis. The first evaluates the overall performance of VPR on a whole dataset and its consistency showing the maximum and minimum guarantee EP score. The second compares VPR techniques using the McNemar's test to confirms whether the performance difference between VPR techniques is statistically reliable within a confidence interval. The utilization of the proposed framework is demonstrated by comparing five VPR methods on various appearance changes.

- Convolutional Neural Networks (CNN) enable state-of-the-art VPR performance but are computationally intensive and, therefore, unsuitable for resource-constraint platforms. This chapter investigates the local feature descriptor suitability in the challenging context of flying drones characterized by 6-DOG viewpoint changes

and severe hardware limitations. The experiments are conducted for several local feature descriptors ranging from the well-established SIFT to the recent AKAZE, including the computationally efficient ORB. The results demonstrated that local feature descriptors have comparable performance to the state-of-the-art CNN-based methods for this application and considerably faster processing. However, local features are susceptible to conditional changes and are outperformed by learned VPR techniques.

- Binary Neural Networks (BNN) are investigated to handle conditional changes within a resource budget. Chapter 5 first demonstrates that BNNs can be used for VPR and then proposes a transformation pipeline to turn a CNN into a BNN optimized for VPR applications. This pipeline is applied to AlexNet to obtain FloppyNet, a compact BNN with three layers and a model size of just 154 KiB. Deployed on a Raspberry Pi4, FloppyNet runs seven times faster than AlexNet, using about 13% of the energy per processed image.

- The analysis of FloppyNet's layers showed that most of the processing time is spent on the first convolution, which is not fully binary for better VPR performance. Chapter 6 proposes an in-place replacement for the first convolutional layer, HB-DS, to address such a computational bottleneck. HB-DS combines depthwise factorization and binarization to reduce as much as possible the number of non-binary operations. This module is then used on FloppyNet to obtain a BNN twice as fast without any accuracy loss.

7.2 Future Directions

This section suggests possible developments of the research presented in this thesis and discusses other gaps the author identified during his investigation over the doctorate years.

- Chapter 3 proposed an evaluation framework to provide a reliable performance comparison between VPR techniques. It uses McNemar's test to compare VPR methods on a dataset with two possible outcomes. The performance difference is certified within a specific confidence interval if $|Z|$ is above a threshold. Otherwise, the two techniques are regarded as comparable. Such a test is designed to assess VPR techniques on a dataset. However, it could be used to create a benchmark dataset that reliably identifies performance differences without returning a low-value of $|Z|$. Testing data has been a concern for data scientists for a long time [182]. The framework proposed in Chapter 3 could be adapted to tune, build or assess a VPR benchmark dataset. This idea consists in selecting a *training set* of VPR methods representative of various approaches and testing them on a dataset. This dataset is then enriched with new places or pruned over multiple iterations until the comparative results pass McNemar's test. The expected result is a highly discriminative dataset to test any VPR technique within the proposed evaluation framework.

- Chapter 5 proposed a BNN as an efficient global feature extractor for VPR using convolutional features. State-of-the-art methods often use at least a second stage to perform region matching (CrossRegion-Bow), geometrical verification (R-MAC), or to obtain a highly distinctive global image descriptor (NetVLAD). FloppyNet fails

under extreme viewpoint variation because it uses convolutional features alone without a post-processing or pooling stage. An extension of this thesis will investigate the employment of a lightweight BNN in multi-stage VPR approaches to achieve higher viewpoint invariance. The authors consider this open problem a priority to complete the research started with his PhD to address VPR efficiently and effectively under any appearance changes.

- The HB-DS module proposed in Chapter 6 is tested for VPR only with a standard VGG-type architecture considering a series of convolutional and pooling layers. HB-DS can likely improve the efficiency of any network type (e.g., ResNet [131]). As the use of a non-binary first convolution is a common practice [147], it is worth investigating the use of HB-DS with different network architectures and vision applications other than VPR, such as object detection, image segmentation, and classification.

7.3 Closing Remarks

Despite the hard work and advancements made by the research community in Visual Place Recognition, there is still a need for further improvements. This thesis aims to contribute to this progress by exploring ways to extend the use of greedy computational approaches to low-end hardware platforms. While this research represents only a small step toward future advancements, the author hopes that the findings presented here can encourage others to build upon this work and improve Visual Place Recognition.

Bibliography

[1] M. M. Larsson, E. Stenborg, L. Hammarstrand, M. Pollefeys, T. Sattler, and F. Kahl, "A cross-season correspondence dataset for robust semantic segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 9524–9534.

[2] Z. Chen, F. Maffra, I. Sa, and M. Chli, "Only look once, mining distinctive landmarks from convnet for visual place recognition," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 9–16.

[3] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, "Visual place recognition with repetitive structures," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 883–890.

[4] F. Maffra, L. Teixeira, Z. Chen, and M. Chli, "Real-time wide-baseline place recognition using depth completion," *IEEE Robotics and Automation Letters*, 2019.

[5] Z. Chen, A. Jacobson, N. Sünderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford, "Deep learning features at scale for visual place recognition," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3223–3230.

[6] M. Zaffar, S. Ehsan, M. Milford, and K. D. McDonald-Maier, "Memorable maps: A framework for re-defining places in visual place recognition," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2020.

[7] R. Sahdev and J. K. Tsotsos, "Indoor place recognition system for localization of mobile robots," in *2016 13th Conference on Computer and Robot Vision (CRV)*, 2016, pp. 53–60.

[8] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4297–4304.

[9] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 118–126.

[10] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progress in brain research*, vol. 155, pp. 23–36, 2006.

[11] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2015.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, 2013.

[14] "Parrot web page," https://www.parrot.com/us/drones/anafi-ai, accessed: 2022-01-30.

[15] "DJI Online Store," https://store.dji.com/shop/dji-mini?from=store_homepage, accessed: 2022-01-30.

[16] S. Garg, T. Fischer, and M. Milford, "Where Is Your Place, Visual Place Recognition?" in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4416–4425. [Online]. Available: https://www.ijcai.org/proceedings/2021/603

[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] "Google Maps Rome Images," https://www.google.com/maps/@41.8897454,12.4910251,3a,75y,80.19h,97.07t/data=!3m6!1e1!3m4!1sONC7Y-S9VYlHiPENkSeVrg!2e0!7i13312!8i6656, accessed: 2023-03-09.

[19] K. Liu and H. Ou, "A light-weight lidar-inertial slam system with high efficiency and loop closure detection capacity," in *2022 International conference on advanced robotics and mechatronics (ICARM)*. IEEE, 2022, pp. 284–289.

[20] M. Waheed, M. Milford, K. McDonald-Maier, and S. Ehsan, "Switchhit: A probabilistic, complementarity-based switching system for improved visual place recognition in changing environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 7833–7840.

[21] M. Zaffar, A. Khaliq, S. Ehsan, M. Milford, and K. McDonald-Maier, "Levelling the playing field: A comprehensive comparison of visual place recognition approaches under changing conditions," in *IEEE ICRA Workshop on Dataset Generation and Benchmarking of SLAM Algorithms for Robotics and VR/AR*. IEEE, 2019.

[22] S. Schubert and P. Neubert, "What makes visual place recognition easy or hard?" Jun. 2021, arXiv:2106.12671 [cs]. [Online]. Available: http://arxiv.org/abs/2106.12671

[23] S. Ehsan, A. Clark, A. Leonardis, A. Khaliq, M. Fasli, K. McDonald-Maier *et al.*, "A generic framework for assessing the performance bounds of image feature detectors," *Remote Sensing*, vol. 8, no. 11, p. 928, 2016.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: http://www.nature.com/articles/nature14539

[25] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep Convolutional Neural Networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[27] C. Leng, H. Zhang, B. Li, G. Cai, Z. Pei, and L. He, "Local feature descriptor for image matching: A survey," *IEEE Access*, vol. 7, pp. 6424–6434, 2019.

[28] M. Zaffar, S. Garg, M. Milford, J. Kooij, D. Flynn, K. McDonald-Maier, and S. Ehsan, "Vpr-bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change," *International Journal of Computer Vision*, pp. 1–39, 2021.

[29] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1437–1451, 2018.

[30] Z. Zeng, J. Zhang, X. Wang, Y. Chen, and C. Zhu, "Place recognition: An overview of vision perspective," *Applied Sciences*, vol. 8, no. 11, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/11/2257

[31] V. Polizzi, R. Hewitt, J. Hidalgo-Carrió, J. Delaune, and D. Scaramuzza, "Data-Efficient Collaborative Decentralized Thermal-Inertial Odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 681–10 688, Oct. 2022, conference Name: IEEE Robotics and Automation Letters.

[32] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[33] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," *arXiv preprint arXiv:1603.09114*, 2016.

[34] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[35] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *null*. IEEE, 2003, p. 1470.

[36] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[37] K. Mikolajczyk and K. Mikolajczyk, "Scale & affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, oct 2004.

[38] 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.

[39] H. Winnemöller, J. E. Kyprianidis, and S. C. Olsen, "XDoG: An eXtended difference-of-Gaussians compendium including advanced image stylization," *Computers & Graphics*, vol. 36, no. 6, pp. 740–753, Oct. 2012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S009784931200043X

[40] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[41] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 2564–2571.

[42] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International workshop on automatic face and gesture recognition*, vol. 12, 1995, pp. 296–301.

[43] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893 vol. 1.

[44] E. Stumm, C. Mei, and S. Lacroix, "Probabilistic place recognition with covisibility maps," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4158–4163.

[45] W. Zhang and J. Kosecka, "Image based localization in urban environments," in *Third international symposium on 3D data processing, visualization, and transmission (3DPVT'06)*. IEEE, 2006, pp. 33–40.

[46] A. Haar, *Zur theorie der orthogonalen funktionensysteme*. Georg-August-Universitat, Gottingen., 1909.

[47] A. C. Murillo, J. J. Guerrero, and C. Sagues, "Surf features for efficient robot localization with omnidirectional images," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3901–3907.

[48] M. Cummins and P. Newman, "Appearance-only slam at large scale with fab-map 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[49] D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6314, pp. 778–792, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-642-15561-1_56

[50] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, Dec. 2013, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/0278364913499193

[51] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 IEEE international conference on computer vision (ICCV)*. Ieee, 2011, pp. 2548–2555.

[52] F. Maffra, Z. Chen, and M. Chli, "Viewpoint-tolerant place recognition combining 2d and 3d information for uav navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2542–2549.

[53] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 3951, pp. 430–443, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/11744023_34

[54] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015, conference Name: IEEE Transactions on Robotics.

[55] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, conference Name: IEEE Transactions on Robotics.

[56] X. Wu, C. Sun, L. Chen, T. Zou, W. Yang, and H. Xiao, "Adaptive ORB feature detection with a variable extraction radius in RoI for complex illumination scenes," *Robotics and Autonomous Systems*, vol. 157, p. 104248, Nov. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889022001439

[57] A. Dietsche, L. Ott, R. Siegwart, and R. Brockers, "Visual Loop Closure Detection for a Future Mars Science Helicopter," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 014–12 021, Oct. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9894657/

[58] P. F. Alcantarilla and T. Solutions, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *IEEE Trans. Patt. Anal. Mach. Intell*, vol. 34, no. 7, pp. 1281–1298, 2011.

[59] H. Madokoro, K. Sato, and N. Shimoi, "Indoor Scene and Position Recognition Based on Visual Landmarks Obtained from Visual Saliency without Human Effect," *Robotics*, vol. 8, no. 1, p. 3, Jan. 2019. [Online]. Available: http://www.mdpi.com/2218-6581/8/1/3

[60] M. Zaffar, S. Ehsan, M. Milford, and K. McDonald-Maier, "Cohog: A light-weight, compute-efficient, and training-free visual place recognition technique for changing environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1835–1842, 2020.

[61] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 04, pp. 669–688, 1993.

[62] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via Convolutional Neural Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.

[63] B. Kumar, G. Carneiro, I. Reid *et al.*, "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5385–5394.

[64] S. Ehsan, A. Clark, and K. McDonald-Maier, "Rapid online analysis of local feature detectors and their complementarity," *Sensors*, vol. 13, no. 8, pp. 10 876–10 907, 2013.

[65] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[66] F. Rahutomo, T. Kitasuka, and M. Aritsugi, "Semantic cosine similarity," in *The 7th international student conference on advanced science and technology ICAST*, vol. 4, no. 1, 2012, p. 1.

[67] M. Cummins and P. Newman, "Appearance-only slam at large scale with fab-map 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[68] C. Cadena and J. Neira, "A learning algorithm for place recognition," in *The ICRA Workshop on Long-term Autonomy*, 2011.

[69] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*. IEEE Computer Society, 2010, pp. 3304–3311.

[70] R. Arandjelovic and A. Zisserman, "All about vlad," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

[71] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: CNN architecture for weakly supervised place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.

[72] A. Khaliq, S. Ehsan, Z. Chen, M. Milford, and K. McDonald-Maier, "A holistic visual place recognition approach using lightweight cnns for significant viewpoint and appearance changes," *IEEE Transactions on Robotics*, pp. 1–9, 2019.

[73] H. Jegou and A. Zisserman, "Triangulation Embedding and Democratic Aggregation for Image Search," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, Jun. 2014, pp. 3310–3317. [Online]. Available: https://ieeexplore.ieee.org/document/6909819

[74] Z. Gao, J. Xue, W. Zhou, S. Pang, and Q. Tian, "Democratic Diffusion Aggregation for Image Retrieval," *IEEE Transactions on Multimedia*, vol. 18, no. 8, pp. 1661–1674, Aug. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7469838/

[75] H. Jégou and O. Chum, "Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening," in *European conference on computer vision*. Springer, 2012, pp. 774–787.

[76] N. Sünderhauf and P. Protzel, "Brief-gist-closing the loop by simple means," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1234–1241.

[77] A. C. Murillo and J. Kosecka, "Experiments in place recognition using gist panoramas," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 2009, pp. 2196–2203.

[78] G. Singh and J. Kosecka, "Visual loop closing using gist descriptors in manhattan world," in *ICRA Omnidirectional Vision Workshop*, 2010, pp. 44–48.

[79] Y. Liu and H. Zhang, "Visual loop closure detection with a compact image descriptor," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 1051–1056, iSSN: 2153-0866.

[80] D. Gabor, "Theory of communication. part 1: The analysis of information," *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[81] C. McManus, B. Upcroft, and P. Newmann, "Scene signatures: Localised and point-less features for localisation," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[82] N. Sunderhauf and P. Protzel, "BRIEF-Gist - closing the loop by simple means," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Francisco, CA: IEEE, Sep. 2011, pp. 1234–1241. [Online]. Available: http://ieeexplore.ieee.org/document/6094921/

[83] M. Zaffar, A. Khaliq, S. Ehsan, M. Milford, K. Alexis, and K. D. McDonald-Maier, "Are state-of-the-art visual place recognition techniques any good for aerial robotics?" *CoRR*, vol. abs/1904.07967, 2019. [Online]. Available: http://arxiv.org/abs/1904.07967

[84] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *European conference on computer vision*. Springer, 2014, pp. 392–407.

[85] Z. Chen, O. Lam, A. Jacobson, and M. Milford, "Convolutional Neural Network-based Place Recognition," 2014.

[86] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural Codes for Image Retrieval," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, vol. 8689, pp. 584–599, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10590-1_38

[87] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[88] M. Lopez-Antequera, R. Gomez-Ojeda, N. Petkov, and J. Gonzalez-Jimenez, "Appearance-invariant place recognition by discriminatively training a convolutional neural network," *Pattern Recognition Letters*, vol. 92, pp. 89–95, Jun. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167865517301381

[89] Y. Hou, H. Zhang, and S. Zhou, "Convolutional neural network-based image representation for visual loop closure detection," in *2015 IEEE International Conference on Information and Automation*, 2015, pp. 2238–2245.

[90] D. Bai, C. Wang, B. Zhang, X. Yi, and X. Yang, "Sequence searching with cnn features for robust and fast visual place recognition," *Computers & Graphics*, vol. 70, pp. 270–280, 2018.

[91] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1643–1649.

[92] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[93] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[94] N. Merrill and G. Huang, "Lightweight unsupervised deep loop closure," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[95] B. Dongdong, W. Chaoqun, B. Zhang, Y. Xiaodong, Y. Xuejun *et al.*, "CNN feature boosted seqslam for real-time loop closure detection," *Chinese Journal of Electronics*, vol. 27, no. 3, pp. 488–499, 2018.

[96] G. Tolias, R. Sicre, and H. Jégou, "Particular Object Retrieval With Integral Max-Pooling of CNN Activations," in *ICLR 2016*, ser. International Conference on Learning Representations, San Juan, Puerto Rico, May 2016, pp. 1–12. [Online]. Available: https://hal.inria.fr/hal-01842218

[97] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conference on Computer Vision*. Springer, 2014, pp. 346–361.

[98] S. Hausler, S. Garg, M. Xu, M. Milford, and T. Fischer, "Patch-NetVLAD: Multi-Scale Fusion of Locally-Global Descriptors for Place Recognition," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, Jun. 2021, pp. 14 136–14 147. [Online]. Available: https://ieeexplore.ieee.org/document/9577552/

[99] M. Chancán, L. Hernandez-Nunez, A. Narendra, A. B. Barron, and M. Milford, "A hybrid compact neural architecture for visual place recognition," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 993–1000, 2020.

[100] B. L. McNaughton, F. P. Battaglia, O. Jensen, E. I. Moser, and M.-B. Moser, "Path integration and the neural basis of the'cognitive map'," *Nature Reviews Neuroscience*, vol. 7, no. 8, pp. 663–678, 2006.

[101] B. Arcanjo, B. Ferrarini, M. Milford, K. D. McDonald-Maier, and S. Ehsan, "An efficient and scalable collection of fly-inspired voting units for visual place recognition in changing environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2527–2534, 2022.

[102] S. Hussaini, M. Milford, and T. Fischer, "Spiking neural networks for visual place recognition via weighted neuronal assignments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4094–4101, 2022.

[103] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608097000117

[104] E. P. Frady, G. Orchard, D. Florey, N. Imam, R. Liu, J. Mishra, J. Tse, A. Wild, F. T. Sommer, and M. Davies, "Neuromorphic nearest neighbor search using intel's pohoiki springs," in *Proceedings of the neuro-inspired computational elements workshop*, 2020, pp. 1–10.

[105] G. Berton, R. Mereu, G. Trivigno, C. Masone, G. Csurka, T. Sattler, and B. Caputo, "Deep visual geo-localization benchmark," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 5396–5407.

[106] M. Waheed, M. Milford, K. McDonald-Maier, and S. Ehsan, "Improving visual place recognition performance by maximising complementarity," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5976–5983, 2021.

[107] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.

[108] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[109] J. Smith, "Image retrieval evaluation," in *Proceedings. IEEE Workshop on Content-Based Access of Image and Video Libraries (Cat. No.98EX173)*, 1998, pp. 112–113.

[110] S. Lowry and M. J. Milford, "Supervised and unsupervised linear learning techniques for visual place recognition in changing environments," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 600–613, 2016.

[111] K. Vidanapathirana, P. Moghadam, B. Harwood, M. Zhao, S. Sridharan, and C. Fookes, "Locus: Lidar-based place recognition using spatiotemporal higher-order pooling," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5075–5081.

[112] Z. C. Lipton, C. Elkan, and B. Naryanaswamy, "Optimal thresholding of classifiers to maximize f1 measure," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II 14*. Springer, 2014, pp. 225–239.

[113] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li, "Deep learning for content-based image retrieval: A comprehensive study," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 157–166.

[114] Z. Chen, L. Liu, I. Sa, Z. Ge, and M. Chli, "Learning context flexible attention model for long-term visual place recognition," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4015–4022, 2018.

[115] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017. [Online]. Available: http://dx.doi.org/10.1177/0278364916679498

[116] S. Skrede, "Nordlandsbanen: minute by minute, season by season https://nrkbeta.no/2013/01/15/nordlandsbanen-minute-by-minute-season-by-season/," accessed: 2021-11-06.

[117] J. Mount and M. Milford, "2d visual place recognition for domestic service robots at night," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4822–4829.

[118] A. Karki, C. Palangotu Keshava, S. Mysore Shivakumar, J. Skow, G. Madhukeshwar Hegde, and H. Jeon, "Tango: A deep neural network benchmark suite for

various accelerators," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, pp. 137–138.

[119] I. Palit, Q. Lou, R. Perricone, M. Niemier, and X. S. Hu, "A uniform modeling methodology for benchmarking dnn accelerators," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–7.

[120] C. Xia, J. Zhao, H. Cui, X. Feng, and J. Xue, "Dnntune: Automatic benchmarking dnn models for mobile-cloud computing," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 16, no. 4, pp. 1–26, 2019.

[121] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[122] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[123] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[124] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, "Back to simplicity: How to train accurate bnns from scratch?" *CoRR*, vol. abs/1906.08637, 2019. [Online]. Available: http://arxiv.org/abs/1906.08637

[125] H. N. Mhaskar and T. Poggio, "Deep vs. shallow networks: An approximation theory perspective," *Analysis and Applications*, no. 06, pp. 829–848, 2016.

[126] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989. [Online]. Available: https://doi.org/10.1007/BF02551274

[127] J. Ba and R. Caruana, "Do deep nets really need to be deep?" *Advances in neural information processing systems*, vol. 27, 2014.

[128] Y. N. Dauphin and Y. Bengio, "Big Neural Networks Waste Capacity," Mar. 2013, arXiv:1301.3583 [cs]. [Online]. Available: http://arxiv.org/abs/1301.3583

[129] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[130] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016.

[131] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[132] L. Sifre and P. S. Mallat, "Rigid-motion scattering for image classification author," *English. Supervisor: Prof. Stéphane Mallat. Ph. D. Thesis. Ecole Polytechnique*, 2014.

[133] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[134] S. Hanson and L. Pratt, "Comparing biases for minimal network construction with back-propagation," *Advances in neural information processing systems*, vol. 1, 1988.

[135] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[136] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: optimal brain surgeon," in *Proceedings of the 5th International Conference on Neural Information Processing Systems*, 1992, pp. 164–171.

[137] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, 2015, pp. 1135–1143.

[138] H. Van Nguyen, K. Zhou, and R. Vemulapalli, "Cross-Domain Synthesis of Medical Images Using Efficient Location-Sensitive Deep Network," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. Frangi, Eds. Cham: Springer International Publishing, 2015, vol. 9349, pp. 677–684, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-24553-9_83

[139] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/hash/2afe4567e1bf64d32a5527244d104cea-Abstract.html

[140] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," Feb. 2016, arXiv:1510.00149 [cs]. [Online]. Available: http://arxiv.org/abs/1510.00149

[141] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[142] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "Tensorflow lite micro: Embedded machine learning on tinyml systems," *CoRR*, vol. abs/2010.08678, 2020. [Online]. Available: https://arxiv.org/abs/2010.08678

[143] "CUDA Toolkit Documentation," https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#arithmetic-instructions, accessed: 2020-07-26.

[144] "Arm Cortex-A76 Software Optimization Guide," https://developer.arm.com/documentation/swog307215/a, accessed: 2021-03-03.

[145] "Tensorflow Lite," https://www.tensorflow.org/lite, accessed: 2021-03-05.

[146] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," *arXiv e-prints*, pp. arXiv–1412, 2014.

[147] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, 2019.

[148] D. Saad and E. Marom, "Training feed forward nets with binary weights via a modified chir algorithm," *Complex Systems*, vol. 4, no. 5, 1990.

[149] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015.

[150] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, 2013. [Online]. Available: http://arxiv.org/abs/1308.3432

[151] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[152] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[153] M. Alizadeh, J. Fernández-Marqués, N. D. Lane, and Y. Gal, "An empirical study of binary neural networks' optimisation," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJfUCoR5KX

[154] E. Sari, M. Belbahri, and V. P. Nia, "A study on binary neural networks initialization," *CoRR*, vol. abs/1909.09139, 2019. [Online]. Available: http://arxiv.org/abs/1909.09139

[155] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *CoRR*, vol. abs/1606.06160, 2016. [Online]. Available: http://arxiv.org/abs/1606.06160

[156] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *Thirty-First AAAI conference on artificial intelligence*, 2017.

[157] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," *Advances in neural information processing systems*, vol. 30, 2017.

[158] F. Li and B. Liu, "Ternary weight networks," *CoRR*, vol. abs/1605.04711, 2016. [Online]. Available: http://arxiv.org/abs/1605.04711

[159] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *CoRR*, vol. abs/1612.01064, 2016. [Online]. Available: http://arxiv.org/abs/1612.01064

[160] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "LEARNED STEP SIZE QUANTIZATION," 2020.

[161] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, and N. Kwak, "LSQ+: Improving low-bit quantization through learnable offsets and better initialization," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, WA, USA: IEEE, Jun. 2020, pp. 2978–2985. [Online]. Available: https://ieeexplore.ieee.org/document/9151058/

[162] "CUDA," https://developer.nvidia.com/cuda-zone, accessed: 2020-07-26.

[163] M. Flynn, "Very high-speed computing systems," *Proceedings of the IEEE*, vol. 54, no. 12, pp. 1901–1909, 1966.

[164] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://arxiv.org/pdf/1908.09791.pdf

[165] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Structured binary neural networks for accurate image classification and semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 413–422.

[166] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *CoRR*, vol. abs/1512.01274, 2015. [Online]. Available: http://arxiv.org/abs/1512.01274

[167] R. C. Whaley and A. Petitet, "Minimizing development and maintenance costs in supporting persistently optimized blas," *Software: Practice and Experience*, vol. 35, no. 2, pp. 101–121, 2005.

[168] J. Fromm, M. Cowan, M. Philipose, L. Ceze, and S. Patel, "Riptide: Fast end-to-end binarized neural networks," in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 379–389. [Online]. Available: https://proceedings.mlsys.org/paper/2020/file/2a79ea27c279e471f4d180b08d62b00a-Paper.pdf

[169] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[170] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[171] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, "{TVM}: An automated end-to-end optimizing compiler for

deep learning," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 578–594.

[172] T. Bannink, A. Hillier, L. Geiger, T. de Bruin, L. Overweel, J. Neeven, and K. Helwegen, "Larq compute engine: Design, benchmark, and deploy state-of-the-art binarized neural networks," 2020.

[173] L. Geiger and P. Team, "Larq: An open-source library for training binarized neural networks," *Journal of Open Source Software*, vol. 5, no. 45, p. 1746, Jan. 2020. [Online]. Available: https://doi.org/10.21105/joss.01746

[174] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 65–74.

[175] "PYNQ," http://www.pynq.io/, accessed: 2021-04-26.

[176] A. Pappalardo, "Xilinx/brevitas," https://doi.org/10.5281/zenodo.3333552.

[177] K. A. Tsintotas, L. Bampis, and A. Gasteratos, "The Revisiting Problem in Simultaneous Localization and Mapping: A Survey on Visual Loop Closure Detection," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–25, 2022, conference Name: IEEE Transactions on Intelligent Transportation Systems.

[178] C. Masone and B. Caputo, "A Survey on Deep Visual Place Recognition," *IEEE Access*, vol. 9, pp. 19 516–19 547, 2021, conference Name: IEEE Access.

[179] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, "Towards life-long visual localization using an efficient matching of binary sequences from images," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 6328–6335.

[180] N. V. Shirahatti and K. Barnard, "Evaluating image retrieval," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 955–961.

[181] S. Ehsan, N. Kanwal, A. F. Clark, and K. D. McDonald-Maier, "Improved repeatability measures for evaluating performance of feature detectors," *Electronics letters*, vol. 46, no. 14, pp. 998–1000, 2010.

[182] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR 2011*, 2011, pp. 1521–1528.

[183] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, "Appearance-based loop detection from 3d laser data using the normal distributions transform," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 23–28.

[184] M. Zaffar, S. Garg, M. Milford, J. Kooij, D. Flynn, K. McDonald-Maier, and S. Ehsan, "Vpr-bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change," *International Journal of Computer Vision*, vol. 129, no. 7, pp. 2136–2174, Jul 2021. [Online]. Available: https://doi.org/10.1007/s11263-021-01469-5

[185] S. Ehsan, A. F. Clark, B. Ferrarini, N. U. Rehman, and K. D. McDonald-Maier, "Assessing the performance bounds of local feature detectors: Taking inspiration from electronics design practices," in *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2015, pp. 166–169.

[186] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[187] J. L. Fleiss, B. Levin, and M. C. Paik, *Statistical methods for rates and proportions*. John Wiley & Sons, 2013.

[188] A. L. Edwards, "Note on the "correction for continuity" in testing the significance of the difference between correlated proportions," *Psychometrika*, vol. 13, no. 3, pp. 185–187, 1948.

[189] A. Agresti, *An introduction to categorical data analysis*, third edition ed., ser. Wiley series in probability and statistics. Hoboken, NJ: John Wiley & Sons, 2019.

[190] "Standard Normal Distribution Tables," https://getcalc.com/statistics-normal-distribution-table.htm, accessed: 2023-03-09.

[191] Z. Chen, A. Jacobson, N. Sünderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid, and M. Milford, "Amosnet and hybridnet implementation," https://github.com/scutzetao/DLfeature_PlaceRecog_icra2017, 2017, accessed: 2019-09-04.

[192] G. Tolias, R. Sicre, and H. Jégou, "R-mac implementation," https://github.com/gtolias/rmac, 2016, accessed: 2019-09-04.

[193] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "netvlad implementation," https://github.com/Relja/netvlad, 2016, accessed: 2019-09-04.

[194] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, "Visual place recognition with repetitive structures," in *CVPR*, 2013.

[195] Z. Chen, F. Maffra, I. Sa, and M. Chli, "Cross-region-bow implementation," https://github.com/scutzetao/IROS2017_OnlyLookOnce, 2016, accessed: 2019-09-04.

[196] "V4RL Wide-baseline Place Recognition Dataset," https://github.com/VIS4ROB-lab/place_recognition_dataset_ral2019, accessed: 2019-04-04.

[197] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1647–1650, 2016.

[198] T. Villa, F. Gonzalez, B. Miljievic, Z. Ristovski, and L. Morawska, "An overview of small unmanned aerial vehicles for air quality measurements: Present applications and future prospectives," *Sensors*, vol. 16, no. 7, p. 1072, 2016.

[199] J. Li, Y. Bi, M. Lan, H. Qin, M. Shan, F. Lin, and B. M. Chen, "Real-time simultaneous localization and mapping for uav: a survey," in *Proc. of International micro air vehicle competition and conference*, 2016, pp. 237–242.

[200] S. Khattak, C. Papachristos, and K. Alexis, "Keyframe-based direct thermal-inertial odometry," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3563–3569.

[201] D. O. Wheeler, D. P. Koch, J. S. Jackson, T. W. McLain, and R. W. Beard, "Relative navigation: A keyframe-based approach for observable gps-degraded navigation," *IEEE Control Systems Magazine*, vol. 38, no. 4, pp. 30–48, 2018.

[202] N. Sünderhauf, P. Neubert, and P. Protzel, "Are we there yet? challenging seqslam on a 3000 km journey across all four seasons," in *Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA)*, 2013, p. 2013.

[203] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "Kaze features," in *European Conference on Computer Vision*. Springer, 2012, pp. 214–227.

[204] S. M. Omohundro, *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[205] S. Ehsan, A. Clark, B. Ferrarini, and K. McDonald-Maier, "Jpeg, blur and uniform light changes image database," http://vase.essex.ac.uk/datasets/index.html, 2012, accessed: 2022-10-15.

[206] B. Ferrarini, S. Ehsan, A. Leonardis, N. U. Rehman, and K. D. McDonald-Maier, "Performance characterization of image feature detectors in relation to the scene content utilizing a large image database," *IEEE Access*, vol. 6, pp. 8564–8573, 2018.

[207] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *British Machine Vision Conf. (BMVC)*, 2013.

[208] J. Guevara Diaz, "PyVLAD," https://github.com/mxbi/PyVLAD, accessed: 2019-04-04.

[209] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.

[210] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[211] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, pp. 1–62, 2020.

[212] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[213] M. Toneva, A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon, "An empirical study of example forgetting during deep neural network learning," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BJlxm30cKm

[214] "Places365 DevKit," https://github.com/zhoubolei/places_devkit, accessed: 2020-07-26.

[215] "Raspberry Pi 4 Tech Specs," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/, accessed: 2021-03-06.

[216] "CoHOG source code," https://github.com/MubarizZaffar/VPR-Bench/tree/main/VPR_Techniques/CoHOG_Python, accessed: 2021-11-06.

[217] "pyleargist," https://pypi.org/project/pyleargist/, accessed: 2021-11-06.

[218] "GIST MATLAB implementation," http://people.csail.mit.edu/torralba/code/spatialenvelope/, accessed: 2021-11-06.

[219] L. Geiger and P. Team, "Larq: An open-source library for training binarized neural networks," *Journal of Open Source Software*, vol. 5, no. 45, p. 1746, 2020.