Doctoral Thesis

Scalable Life-long Visual Place Recognition

Submitted by:

Anh-Dzung DOAN

Supervised by:

Prof. Tat-Jun CHIN

Dr. Yasir LATIF

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Computer and Mathematical Sciences

School of Computer Science

January 2022

Dedicated to my parents,
for their unconditional love and endless support

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time

Signed: _____

Date: 18 February 2021

THE UNIVERSITY OF ADELAIDE

Abstract

School of Computer Science

Doctor of Philosophy

Scalable Life-long Visual Place Recognition

by Anh-Dzung Doan

Visual place recognition (VPR) is the task of using visual inputs to determine if mobile robots are visiting a previously observed place or exploring new regions. To perform convincingly, a practical VPR algorithm must be robust against appearance changes, due to not only short-term (e.g., weather, lighting) and long-term (e.g., seasons, vegetation growth, etc) environmental variations, but also "less cyclical" changes (construction and roadworks, updating of signage, façades and billboards, etc). Such appearance changes invariably occur in real life. It motivates our thesis to fill this research gap.

To this end, we firstly investigate probabilistic frameworks to effectively exploit the temporal information from visual data which is in the form of videos. Inspired by Bayes Filter, we propose two VPR methods that respectively perform filtering on discrete and continuous domains, where the temporal information is efficiently used to improve VPR accuracy under appearance changes. Given the fact that the appearance of operational environments uninterruptedly and indefinitely changes, a promising solution for VPR to deal with appearance changes is to continuously accumulate images to incorporate new changes into the internal environmental representation. This demands a VPR technique that is scalable on an ever growing dataset. To this end, inspired by Hidden Markov Models (HMM), we develop novel VPR techniques, that can be efficiently updated and compressed, such that the recognition of new queries can exploit all available data (including recent changes) without suffering from the linear growth in time and space complexity. Another approach to address the scalability issue in VPR is map summarization, which only keeps imformative 3D points in a topometric map, according to predefined constraints. In this thesis, we define timestamp as another constraint. Accordingly, we formulate a repeatability predictor (RP) as a regressor, that predicts the repeatability of an interest point as a function of time. We show that the RP can be used to significantly alleviate the degeneration of VPR accuracy from map summarization.

The contributions of this thesis not only fill the gap within current state of VPR research; but, more importantly, also enable a wide range of applications, such as, self-driving cars, autonomous robots, augmented reality, and so on.

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor, Tat-Jun Chin, for his guidance, patience, and support throughout these years. The completion of my thesis would not have been possible without the nurturing of him. He accepted me to his group when no one knew me, took me under his wing, educated me and gradually shaped me from a student to an independant researcher. I will never forget all scientific discussions that we had, all night-ters that we pulled together before the conference deadlines, and all priceless lessons on how to write a paper and give a presentation. I thank him for his unconditional support and encouragement when my research got tough. I am also extremely thankful for his advice in career path and life, that will follow me forever. Thank you so much, TJ, for everything you did for me. Being your student is sincerely my privilege.

I am extremely grateful to my co-supervisor, Yasir Latif, for his advice, revisions and comments in my research. His profound knowledge and insighful feedback have always been inspiring, that led me going through all challenges of the PhD journey. I regard him not only as a supervisor, but also as an older brother, who was always willing to listen to all of the difficulties that I encountered throughout these years. I always feel grateful for all of the advice and encouragement that he gave me. I would like to extend my sincere thanks to Yasir's wife, Zoona. She always treated me like one of their family members, especially every Christmas dinner, that truly helped me feel less lonely while living alone in Australia.

I would like to express my gratitude to Ian Reid for the scientific discussions and paper revisions. His passion and vision in research have always been my inspiration.

I would like to show my great appreciation to Soohyun Bae, who was the mentor of my internship. Soohyun was always responsible and provided a lot of insightful advice to my research and life. Being his mentee was a wonderful experience to me. I wished I could have a chance to learn more from him in the future.

Also, I would like to thank Daniyar Turmukhambetov, Gabriel Brostow, Grace Tsai, Michael Firman, Victor Adrian Prisacariu, and Eric Brachmann for helpful discussions during my internship. Working with you was truly my pleasure.

I would like to thank my Vietnamese friends Toan Do, Toan Tran, Huu Le, and Trung Pham for helping me settle down in Australia and helpful scientific discussions. I would like to thank Cuong Nguyen, Quang Tran, Anh Bui, Anh Nguyen, Huy Tran, Tuan Hoang, Hoa Nguyen, and Kiet Wong for hanging out with me when I was under pressure, and also providing me with a lot of helpful advice.

I would like to thank Alvaro, Bo, and Zhipeng for passing me a lot of research experience, and advice about my career path.

I would also like to thank Huangying, Kejie, Caiming, Tian Zhi, Samya, Mahsa, Mehdi, Yifan, Xinyu, Hayden, Rafa, Dong Gong, Gabriel, Pulak, Yuankai, Saroj, and Andrew for being very good colleagues, and sharing to me a lot of research and life experience.

Special thanks also to my lab mates and friends Daqi, Shin Fang, CK, Yutong, Dandan, Violetta, Yu Liu, Wei Liu, Hai-Ming, Haokui, and Qingsen. We shared a lot of beautiful memories together, including happiness when papers get accepted and sadness when receiving negative reviewers' comments. I will never forget every all-nighter that we pulled together before conference deadlines, every scientific discussion that helped us learn from each other, and every party that we threw together in restaurants, in Daqi's home, in Yutong's home, and (sometimes and secretly) in the lab. Thank you so much for being a part of my PhD journey, being with you was a wonderful gift for me.

I am also grateful to the University of Adelaide for providing the scholarship to make my Ph.D study possible. I extend this thank to the Australian Centre for Robotic Vision and the Australian Institute for Machine Learning for facilitating my research.

Finally, I owe my deepest gratitude to my parents. This thesis would have been impossible without their unconditional love and support. Even though our life was so tough, my parents always worked hard such that I could go to school without any worry. They taught me how to be a good person, always be willing to help people in need, appreciate to every favour someone did for me, and be tough when facing challenges. I am deeply indebted to my parents.

Publications

This thesis is based on the content of following conference and journal papers:

- Anh-Dzung Doan, Yasir Latif, Thanh-Toan Do, Yu Liu, Shin-Fang Ch'ng, Tat-Jun Chin, and Ian Reid. Visual Localization Under Appearance Change: A Filtering Approach. In *International Conference on Digital Image Computing: Techniques and Applications* (DICTA), 2019.

  (This work won **APRS/IAPR Best paper award**)

- Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, Yu Liu, Thanh-Toan Do, and Ian Reid. Scalable Place Recognition Under Appearance Change for Autonomous Driving. In *International Conference in Computer Vision (ICCV)*, 2019.

- Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, Yu Liu, Shin-Fang Ch'ng, Thanh-Toan Do, and Ian Reid. Visual Localization Under Appearance Change: Filtering Approaches. *Neural Computing and Applications (NCAA)*, 2020.

- Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, and Ian Reid. HM^4: Hidden Markov Model with Memory Management for Visual Place Recognition. *IEEE Robotics and Automation Letters (RA-L)*, 2020.

- Anh-Dzung Doan, Daniyar Turmukhambetov, Yasir Latif, Tat-Jun Chin, and Soohyun Bae. Learning to Predict Repeatability of Interest Points. In *International Conference on Robotics and Automation (ICRA)*, 2021.

In addition, I co-authored the below paper:

- Yasir Latif, Anh-Dzung Doan, Tat-Jun Chin, and Ian Reid. SPRINT: Subgraph Place Recognition for INtelligent Transportation. In *International Conference on Robotics and Automation (ICRA)*, 2020.

The technical detail about G2D is also made publicly available

- Anh-Dzung Doan, Abdul Mohsi Jawaid, Thanh-Toan Do, and Tat-Jun Chin. G2D: from GTA to Data. *arXiv preprint arXiv:1806.07381*, 2019.

Contents

Declaration v

Abstract vii

Acknowledgements ix

Publications xi

Contents xi

1 Introduction 1
- 1.1 Schematic of visual place recognition . . . 2
  - 1.1.1 Image representation . . . 3
  - 1.1.2 Representation of environment . . . 4
  - 1.1.3 Belief generation . . . 5
  - 1.1.4 Map update . . . 5
- 1.2 Why scalable life-long visual place recognition? . . . 6
- 1.3 Main contributions . . . 8
- 1.4 Thesis outline . . . 10

2 Literature Review 13
- 2.1 Image representation . . . 13
  - 2.1.1 Hand-crafted methods . . . 13
  - 2.1.2 Learning methods . . . 14
- 2.2 Life-long mapping . . . 15
  - 2.2.1 Remembering and forgetting data . . . 16
  - 2.2.2 Multiple representations of the environment . . . 16
- 2.3 Belief generation . . . 17
- 2.4 Research gaps . . . 18

3 Filtering Approaches for Visual Localization 23
- 3.1 Introduction and research questions . . . 26
- 3.2 Problem definition . . . 29
- 3.3 Observation encoder . . . 29
- 3.4 Visual localization with HMM . . . 30

3.4.1 Place hypotheses from HMM . . . 31
3.4.1.1 Transition matrix . . . 32
3.4.1.2 Observation model . . . 33
3.4.2 Estimating 6 DoF pose . . . 33
3.4.3 Overall algorithm . . . 33
3.5 Monte Carlo-based visual localization (MCVL) . . . 34
3.5.1 Motion model . . . 34
3.5.2 Estimating noisy measurements . . . 36
3.5.3 Updating particle weights and Resampling . . . 36
3.5.4 Overall algorithm . . . 37
3.6 G2D: From GTA to Data . . . 38
3.6.1 Scripthook V . . . 38
3.6.2 Constructing trajectory . . . 39
3.6.2.1 Sparse Trajectory (user-defined) . . . 40
3.6.2.2 Dense Trajectory (generated automatically) . . . 41
3.6.3 Retracing a dense trajectory . . . 42
3.6.4 Changing the environmental conditions . . . 42
3.6.5 Unit Conversion . . . 42
3.7 Synthetic data . . . 43
3.8 Experiments . . . 44
3.8.1 Implementation details . . . 45
3.8.2 Datasets . . . 46
3.8.3 Comparison between VLAD and Bag of Word (BoW) . . . 46
3.8.4 The benefit of post-processing . . . 47
3.8.5 Sum and democratic aggregation . . . 47
3.8.6 Comparison to competitors . . . 48
3.8.6.1 Synthetic dataset . . . 48
3.8.6.2 Oxford RobotCar . . . 49
3.9 Conclusion . . . 51

4 Scalable Visual Place Recognition 53
4.1 Introduction and research questions . . . 56
4.2 Problem setting . . . 58
4.2.1 Overall aims . . . 58
4.3 Map representation . . . 59
4.4 Place recognition using HMM . . . 60
4.4.1 State transition model . . . 61
4.4.2 Observation model . . . 61
4.4.2.1 Image representation . . . 61
4.4.2.2 Computing likelihood . . . 62
4.4.3 Inference using matrix computations . . . 63
4.4.3.1 Inference . . . 63
4.4.3.2 Computational cost . . . 63
4.5 Scalable place recognition based on HMM . . . 63
4.5.1 Map intialization . . . 64
4.5.2 Map update and compression . . . 64
4.5.2.1 Culling new places . . . 65

4.5.2.2 Combining old places . . . 65
4.5.3 Updating the observation model . . . 66
4.5.4 Overall algorithm . . . 66
4.6 Experiments . . . 67
4.6.1 Performance with and without updating the database . . . 67
4.6.2 Map maintenance and visiting unknown regions . . . 68
4.6.3 Comparison against state of the art . . . 70
4.7 Conclusion . . . 71

5 Hidden Markov Model with Memory Management and Polytope VLAD 73
5.1 Introduction and research questions . . . 76
5.2 Background: HMM for VPR . . . 79
5.3 Achieving scalability with HM^4 . . . 80
5.3.1 Compact image representation . . . 80
5.3.2 Coarse respresentation of full database . . . 80
5.3.2.1 Topological map . . . 80
5.3.2.2 Feature space clustering . . . 81
5.3.2.3 Topological submap . . . 82
5.3.3 Two-tiered HMM inference . . . 82
5.3.4 Updating database representation . . . 84
5.3.4.1 Updating topological map . . . 84
5.3.4.2 Updating coarse representation . . . 85
5.3.5 Overall algorithm . . . 86
5.3.6 Complexity analysis: . . . 87
5.4 Polytope VLAD . . . 87
5.4.1 Background: VLAD . . . 87
5.4.2 Polytope VLAD . . . 87
5.4.2.1 Clustering polyVLAD . . . 88
5.4.3 Inverted index for efficient distance computation . . . 89
5.4.4 Complexity analysis . . . 90
5.5 Experiments . . . 90
5.5.1 Datasets . . . 90
5.5.2 Implementation . . . 90
5.5.3 Evaluation . . . 91
5.5.4 Ablation study . . . 91
5.5.5 Comparison to baseline . . . 94
5.6 Conclusion . . . 94

6 Learning to Predict Repeatability of Interest Points 97
6.1 Introduction and research questions . . . 100
6.2 Learning repeatability predictor . . . 102
6.2.1 Parameterizing repeatability functions . . . 102
6.2.2 Repeatability predictor . . . 103
6.2.3 Training loss . . . 104
6.2.4 Constructing ground truth . . . 104
6.3 Application in map summarization . . . 105
6.3.1 Map summarization . . . 106

6.3.1.1 3D point representation . . . 106
6.3.1.2 Sampling 3D points . . . 106
6.3.2 Online visual localization . . . 106
6.4 Experiments . . . 107
6.4.1 Predicting repeatability function . . . 107
6.4.1.1 Datasets . . . 108
6.4.1.2 Results . . . 109
6.4.2 Map summarization for VL . . . 111
6.4.2.1 Datasets . . . 111
6.4.2.2 Results . . . 111
6.5 Conclusion . . . 112

7 Conclusions **115**
7.1 Contributions . . . 115
7.2 Future directions . . . 116

Bibliography **119**

Chapter 1

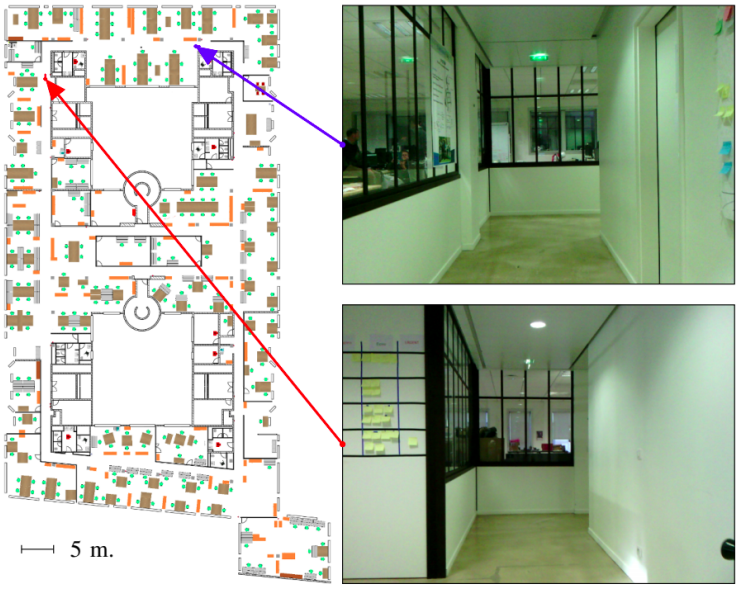
Introduction

Imagine visiting a new city as a tourist and being lost at night. While you are walking around and trying to figure out your position, you turn at a corner and see a theatre that you visited yesterday. The moment you recognize that theatre, you not only know where you are but also know exactly how to get back to your hotel. Your brain has successfully matched what you are currently seeing to a place from your memory, in order to help you locate yourself in the world. This is the process of *visual place recognition* (VPR): the ability of identifying a previously seen location based solely on visual information. In other words, VPR matches current images to images in the past. To operate effectively, a general VPR system must have a memory to store visual information (generally called a *map*) and be able to generate localization hypotheses based on current observations. VPR is an indispensable tool for life-long robotic operation, allowing them to localize across time.

Similar to other perception problems, VPR is a challenging problem because the appearance of the operational environment changes continuously and indefinitely across time. While the same place might look different at different times (see Fig. 1.1a), the problem is further complicated by the fact that different places can also look similar, a problem termed *perceptual aliasing*. An example of this phenomenon can be seen when different corridors in a building share the same appearance; see Fig. 1.1b. Dealing with appearance changes and visual ambiguity are two main problems that need to be addressed by any VPR method.

This thesis presents methods that allow VPR to deal with problems arising during life-long operation. This chapter presents an overall view of the problem, identifies research gaps and highlights our approach towards addressing them. In Sec. 1.1, we revisit each sub-component in VPR. Next, in Sec. 1.2, we discuss the reason why a life-long visual

(a)



(b)

Fig. 1.1. **(a)** Images are representing a same place but under different appearance **(b)** Perceptual aliasing: Two different places that share a similar visual appearance. Courtesy: Google Streetview [66], Freiburg dataset [126], Bonn dataset [173], and Nowakowski et al. [132].

place recognition should be scalable for long-term autonomy. Sec. 1.3 will summarize the contributions of this thesis and finally Sec. 1.4 will outline the thesis structure.

1.1 Schematic of visual place recognition

Fig. 1.2 describes the standard VPR system, which contains four main modules:

- **Image representation:** This module receives the raw visual data input, processes it, and then outputs the discriminative representation that describes the appearance of places. The major requirement for this module is to ensure the representation of a particular place is invariant to appearance changes; see Sec. 1.1.1.

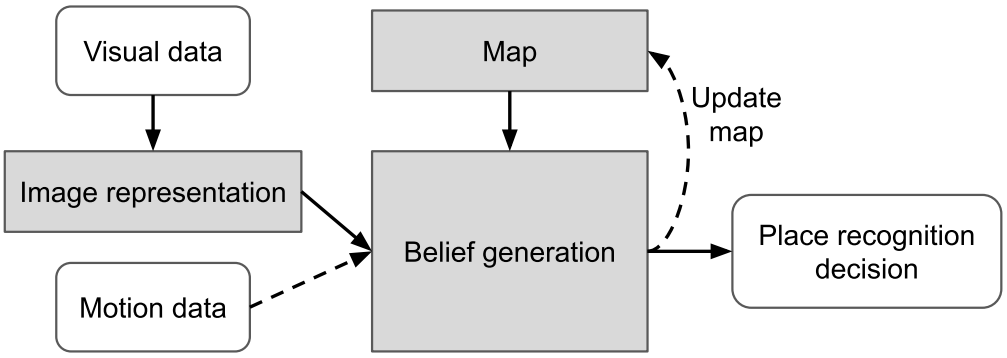


Fig. 1.2. The conventional pipeline of visual place recognition system. The visual data is processed via the *image representation* component. The *belief generation* module uses the map, output of image representation, and motion data (if available) to make the VPR decision. The new data is optionally updated to the map if there are new changes in the environment.

- **Map:** This component is the internal representation of the environment. Typically, there are three types of map, i.e., image database, topological map, and topometric map; each of which has its own advantages and disadvantages. Sec. 1.1.2 provides details about different types of map.
- **Belief generation:** Given the processed visual input and motion data (if available), belief generation aims to determine if mobile robots are either revisiting a specific place within the environment, or exploring a new region; see Sec. 1.1.3.
- **Map update:** VPR is demanded to remain accurate and robust throughout its lifetime, thus it is necessary to continuously update the map to represent the latest state of the environment. Sec. 1.1.4 will describe this module in greater detail.

1.1.1 Image representation

The first step of designing a VPR framework is to decide how to represent visual data. The simplest choice is using raw images processed by image processing techniques (e.g., SeqSLAM [118]), which surprisingly demonstrates its capability in representing appearance under extreme natural changes, but comes with its sensitivity to viewpoint changes [163]. To overcome this issue, other VPR methods select local features as the backbone of their image representation because of their invariance to rotation, translation and scaling factors. For example, Churchill and Newman [28] use FAST [140] to detect invariant local regions inside images, and then describe those by Binary Robust Independent Elementary Features (BRIEF) [22]. Se et al. [154] employ scale-invariant feature transform (SIFT) [102] in their feature tracking. However, extracting SIFT is computationally demanding, that is inappropriate if a VPR system requires a real-time processing. Cummins and Newman [31] thus use speeded up robust features

(SURF) [13]—an alternative to SIFT that is faster in computation. Recently, algorithmic development of VPR witnesses the popularity of replacing hand-crafted local features by learned ones [44, 136, 181, 36], which have shown a great potential in VPR.

As the environment becomes very large (e.g., city scale), local features are computationally expensive and memory inefficient. Large-scale VPR methods instead [31, 109, 62] employ Bag of Word (BoW) to represent every image by a single vector. The idea of BoW is to build a vocabulary by partitioning feature space into a finite number of visual words (typically using K-means). In FAB-MAP 2.0, a vocabulary with the size of 100,000 visual words is required to achieve a reasonable VPR accuracy on 1,000 km [31]. The improved versions of BoW include Bags of Binary Words [54] and vector of locally aggregated descriptors (VLAD) [79], whose efficiency in VPR is also demonstrated in literature [54, 167, 7, 41].

1.1.2 Representation of environment

In VPR, a *map* can be defined in three different ways:

- *Image database* is the simplest form of map, which only contains the appearance information of each place in the environment. With image databases, VPR is ordinarily performed using image retrieval techniques [54, 167, 7]. This type of map also allows us to use computationally effective indexing methods [122, 78, 84]. For example, Schindler et al. [149] employ a vocabulary tree to develop an effective VPR for the city-scale map, and FAB-MAP 2.0 [31] demonstrates its efficiency in 1000-km path by using an inverted index.

- *Topological map* defines the environment as a graph, whose nodes represent distinct places and edges encode the topological relations between places (e.g., transition probabilities). Representative methods for VPR using topological map include SeqSLAM [118], network flow [127], and Bayesian filtering [40]. Another interesting approach is based on the observation that loop closure hypotheses are naturally sparse, Latif et al. [96] formulate VPR as a sparse L_1-minimization problem using topological information.

- *Topometric map* includes metric information (e.g., 6 DoF poses) on edges and nodes, and optionally position of 3D points or objects within each node. Feature-based visual SLAM [123, 34, 20] builds sparse 3D points from local interest points detected in images. In some applications, with the aim of recovering 6 DoF poses of query images, descriptors are also associated with 3D points for establishing 2D-3D correspondences [28, 108, 154]. In contrast with feature-based methods, direct

visual SLAM systems [129, 50, 51, 49] construct dense or semi-dense 3D point cloud using photometric errors. Each 3D point corresponds to pixels observed across image frames within video. By virtue of leveraging metric information, some VPR systems, e.g., CAT-SLAM [109] and SMART [133] demonstrate a significant boost in terms of localization accuracy.

1.1.3 Belief generation

The major task of *belief generation* is to provide a belief distribution or confidence score, that determines how much the current visual input matches a particular place in the map, given the visual input, and motion data (if available). If the place of the map and the visual input share a similar appearance, there is a greater likelihood that they are representing a same physical place.

However, environments with repetitive scene might cause perceptual aliasing [147], i.e., failure in distinguishing two distinct places due to the similarity of their appearances (see Fig. 1.1b). Another issue is the changes of the environmental conditions might lead to different appearances of a same place (see Fig. 1.1a). Generally, a belief generation method is required to precisely deal with perceptual aliasing as well as correctly match images regardless of being captured under distinct environmental conditions.

1.1.4 Map update

The robustness of VPR is challenged by the changes of appearance due to environmental variations. The short-term changes include higher frequency environmental variability, such as, time of day, weather fluctuations, and pedestrian density. Apart from it, long-term aspects, e.g., seasons, growth of vegetation, also contribute toward appearance changes. Another significant contributor to environmental variations, which is "less cyclical", is human activities, such as construction and roadworks, updating of signage, façades and billboards, as well as abrupt changes to traffic rules that affect traffic flow. Fig. 1.1a are examples of different appearances of a physical place.

The fact that environmental changes occur continuously and indefinitely demands a map that is able to self-update to faithfully represent the latest state of environment. Inspired by human memory system, some methods decide what information should be remembered or forgotten [35, 93]. Their basic idea is to maintain landmarks that remain unchanged over long time period while eliminate those that are no longer present in the environment. The main challenge in this strategy is finding efficient mechanism in deciding whether a landmark should be kept or not, otherwise we will "forget" landmarks

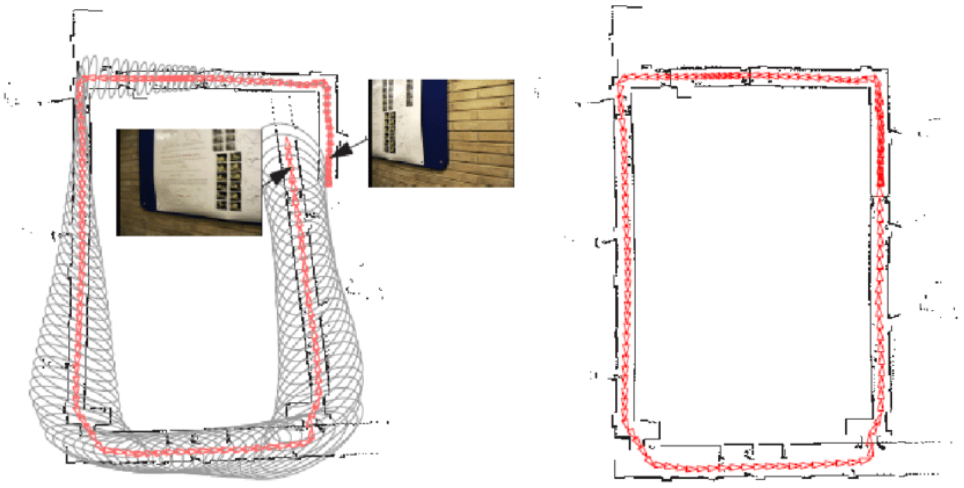
helpful for VPR. Another approach is to store multiple representations of a place [117, 28]. Typically, they propose to always perform SLAM in parallel to other tasks to ensure robots can constantly update the latest changes of environment. However, as the data is continuously accumulated, the fundamental challenge which arises from this approach demands VPR being scalable in the unboundedly growing map.

1.2 Why scalable life-long visual place recognition?

Long-term autonomy, one of the long-standing goals of robotics, is essential for mobile robots to play a vital role in every aspect of human life. It enables robots to carry out required actions without any human supervision, such as, helping disabled people, taking care of elderly, assisting surgeon in operation, and so on. The first step towards autonomy is observing and reasoning about the surrounding environment, e.g., the types of objects in the environment, their relative distances, and motion profiles. These tasks in turn depend on the ability of a robot to estimate its own pose in the environment, *a.k.a* localization. As robots require a prior map before performing required tasks, their typical operation starts with constructing the map, and then carrying out localization. Alternatively, robots simultaneously build the map and localize themselves with respect to the map, that is dubbed *Simultaneous Localization and Mapping* (SLAM). In both strategies, localization occupies a key role, i.e., given a sensor input, localization identifies if a robot is visiting a specific previously observed place or exploring an unknown area.

Because of a wide range of attractive properties, camera sensors have gained popularity in robotic research [21, 89, 33, 59]. Firstly, cameras are a low cost and low maintenance alternatives to other types of sensors (e.g, LiDAR), which can reduce the price of robots to consumers. Secondly, compared to LiDAR, cameras provide richer information while consuming smaller amounts of energy, that significantly benefits the tasks of scene understanding in robotic systems. Thirdly, from the aesthetic perspective, embedding small and light-weight cameras into robots does not change their outer appearance. Therefore, localization using cameras as a primary sensor has been a popular research topic for decades [105, 92, 59], which is referred to as *visual place recognition* (VPR).

A typical application of VPR is loop closure detection within SLAM systems [123, 113, 139, 151]. Specifically, the objective of SLAM is to consistently construct a global representation of the environment. Without "loop closure", SLAM reduces to odometry, whose pose estimate quickly drifts after few meters [21]. Hence, it is advised that one should often "close the loops" while exploring the environment to correct the drift of local odometry measurement. This correction procedure is typically done via pose graph



(a) (b)

Fig. 1.3. **(a)** Before closing loop, the uncertainty of pose estimate (gray eclipses) increases with the length of traversal, leading to a drift in the pose estimate, even though the robot revisits its starting place. This sequentially causes a large error in the mapping phase. **(b)** VPR is used to detect the loop, and then pose graph optimization or bundle adjustment are performed to produce a consistent map. Courtesy: Ho and Newman [73]

.

optimization [91] or bundle adjustment [69, Chapter 18]; see Fig. 1.3. In this way, VPR is an efficient approach to provide loop closure hypotheses for SLAM systems.

Unfortunately, as the appearance variability of environment occurs at vastly different time scales, this poses a great challenge for life-long VPR. Specifically, the natural factors, including time of day, fluctuation of weathers, different seasons and the growth of vegetation, lead to cyclical appearance variations. At another extreme, more unpredictable and less "cyclical" changes are caused by human activities, such as construction works, abrupt changes of traffic flow, updating of signage, façades and billboards. This demands a life-long VPR that is able to accurately match two images showing a same place under a distinct appearance; see Fig. 1.1a.

As the appearance of environment continuously and unpredictably changes, a large amount of research effort has been aimed at developing VPR systems, that remain robust and accurate over lifetime operation. A notable approach is performing sequence-to-sequence matching, pioneered by SeqSLAM [118]. SeqSLAM shows that sequence-based matching is more resilient to appearance changes but comes with its drawback, i.e., sensitivity to view-point changes and differences in sequence velocities [163]. The potential of SeqSLAM has motivated the development of a wide range of techniques that effectively exploit the sequence-to-sequence matching, such as flow network [126], graph search [172], temporal filtering [40], and Hidden Markov Models [41].

Another paradigm is employing machine learning techniques (deep learning in particular) to either extract visual "fingerprints" [145, 166], that are robust against natural appearance changes, or transfer the appearance of a place from a seen condition to unseen ones [95, 6]. In spite of showing a great potential, the robustness of this approach against unpredictable appearance changes caused by human activities, such as updating of façades/billboards/signage, construction works, as well as the abrupt changes of traffic flow, is unproven.

Among existing approaches, continuous data collection has been shown to be the most promising strategy [28, 41, 46]. The basic idea is to continuously accumulate data at high frequency (e.g., weekly) to update the VPR systems. This data collection scheme can be done via a fleet of service vehicles (e.g., taxis [60]) or crowd-sourced services, such as amateur mappers [176] or webcams [76]. Unfortunately, this strategy imposes a fundamental challenge about scalability for life-long VPR

Definition 1.1 (Scalability)**.** Once a VPR system updates new data, the time and space complexity of update and inference processes must grow slower than the database size.

To this end, this thesis will present strategies to deal with scalability issues that arise during life-long VPR. Specifically, our approaches aim to address the following:

- Sequence-to-sequence matching.
- Computational cost and memory usage of VPR inference with the increase in the map size.
- Highly efficient updating or retraining the VPR algorithm on new data to respond quickly to environmental changes.

1.3 Main contributions

We develop novel VPR algorithms for life-long operation, that not only effectively model the temporal information of the visual input, but also satisfactorily meet the scalability demands. Below is the summary of our contributions during the course of this thesis

1. We revisit Bayes Filter and derive two filtering methods: i) on discrete domain with Hidden Markov Model, and ii) on continuous domain with Monte Carlo-based visual localization. Our approaches consider each image as a noisy measurement which is represented as a fixed (low) dimensional vector. In the experiment, a comparison between our methods is made to gain a better understanding about

the performance of probabilistic inference in VPR. In addition, we show that probabilistic inference significantly outperforms deep learning-based regressors in the task of recovering 6 DoF camera poses (see Chapter 3).

2. We present G2D—a software that enables capturing videos from Grand Theft Auto V (GTA V), a popular role playing game set in an expansive virtual city. The target users of our software are computer vision researchers who wish to collect hyper-realistic computer-generated imagery of a city from the street level, under controlled 6 DoF camera poses and varying environmental conditions (weather, season, time of day, traffic density, etc.). G2D accesses/calls the native functions of the game; hence users can directly interact with G2D while playing the game. Specifically, G2D enables users to manipulate conditions of the virtual environment on the fly, while the gameplay camera is set to automatically retrace a predetermined 6DOF camera pose trajectory within the game coordinate system. Concurrently, automatic screen capture is executed while the virtual environment is being explored (see Chapter 3) [1].

3. Using G2D, we collect a synthetic dataset (with accurate 6 DoF ground truth poses) by simulating 59 vehicles which run independently in different routes and environmental conditions. Also, those vehicles are simulated to run in different times of day and weather conditions. The times and weathers in training sequences are uniformly distributed from 1am to 11pm, and in 7 different weather conditions (snowy, foggy, clear, overcast, cloudy, sunny and rainy). Five sequences in different times and weathers are also collected for the testing phase (see Chapter 3) [2].

4. In long-term operations, mobile robots need to continuously accumulate images to maintain adequate samples of the conditions and incorporate new changes into the map, which demands a VPR technique scalable on an unboundedly growing database. We propose a novel VPR technique that can be efficiently retrained and compressed, such that the recognition of new queries can exploit all available data (including recent changes) without suffering from visible growth in computational cost. Underpinning our method is a novel temporal image matching technique based on HMM (see Chapter 4).

5. We incorporate the mechanism of two-tiered memory management into HMM to obtain a novel temporal image matching method which achieves the space-time scalability. Our algorithm, named HM^4, exploits temporal look-ahead to transfer promising candidate images between passive storage and active memory when needed. The inference process takes into account both promising images and a

[1] G2D and its source code are publicly available at https://github.com/dadung/G2D
[2] The dataset is made publicly at https://sites.google.com/view/g2d-software/home

coarse representations of the full database. We show that this allows constant space-time complexity of inference for a fixed coverage area. The coarse representations can also be updated incrementally to absorb new data (see Chapter 5).

6. Even though HM^4 offers a scalable solution for VPR, the computational hardware on mobile robots are very limited whereas it also has to simultaneously process many tasks (e.g., recognition, detection, segmentation) for the scene understanding purpose. Therefore, a VPR system is also required to be lightweight, which motivates us to derive polyVLAD—a compact image representation inspired by Locality Sensitive Hashing [5]. Our compact image representation greatly reduces the memory footprint of HM^4. Also, with the use of inverted index, it significantly reduces the inference time of HM^4 (see Chapter 5).

7. In some robotic contexts (e.g., obstacle avoidance, planning and navigation), it is necessary to recover full 6 DoF camera poses, hence a topometric map with 3D points are required for this task. In large-scale environments, it is infeasible to store a full-scale map on the onboard hardware, a compact representation of the full context map should thus be obtained through the map summarization [45, 121, 19, 108], which performs 3D point sampling according to predefined criteria. Different from those typical approaches, we present a novel localization pipeline that selects 3D points likely being repeatable at the localization run-time. Our map summarization scheme is built upon a repeatability predictor which is able to predict the repeatability of an interest point as a function of time. Through the experiment, localization run-time demonstrates itself as a good criterion in preventing the deterioration of map summarization in VPR accuracy (see Chapter 6).

1.4 Thesis outline

The rest of this thesis is organized as follows

- Chapter 2 provides an overview about recent development of visual place recognition, mainly focusing on place recognition in life-long operation.
- Chapter 3 investigates Bayesian filtering for VPR. Concretely, we propose visual localization with Hidden Markov Model and Monte Carlo-based visual localization, that respectively perform filtering on discrete and continuous domains. In addition, we present G2D (from GTA to data)—an open-source software that assists researchers in collecting synthetic data from the computer game GTA V. Using G2D, we collect a synthetic dataset for VPR.

- Chapter 4 contributes a scalable VPR technique based on Hidden Markov Model, that can be effectively retrained and compressed such that the computational cost does not increase visibly with the growth of map size. The experiment shows a great potential of our technique for large-scale VPR.

- Chapter 5 proposes coupling Hidden Markov Model with two-tier memory management, dubbed HM^4, that allows a constant time and space inference as a new data is updated to refine the map. In addition, a novel compact image representation (polyVLAD) is proposed to reduce the memory footprint of HM^4. The experiment shows a lightweight and scalable VPR system when combining HM^4 and polyVLAD.

- Chapter 6 introduces a novel pipeline for VPR that examines the run-time of VPR to perform the map summarization. Underpinning our strategy is a repeatability predictor that is able to predict the repeatability of an interest points as a function of time.

- Chapter 7 summarizes the main contributions of this thesis and suggests future research directions.

Chapter 2

Literature Review

Early VPR systems assumed that the state of environment remains unchanged, which is clearly invalid over a long time span. As alluded in chapter 1, the appearance variations caused by natural factors and abrupt human activities lead to ever-larger uncontrolled environment. This results in the failure of those systems and prevents its applicability in practice. Therefore, a large research effort has been spent in developing VPR algorithms that remain accurate and stable over a long time period, including invariant image representations (Sec. 2.1), life-long mapping frameworks (Sec. 2.2), and robust belief generation modules (Sec. 2.3). This chapter reviews recent advancement of VPR for life-long operation, and then draws a conclusion in Sec. 2.4 to clarify the contribution of this thesis to the research gap.

2.1 Image representation

It is evident that the visual appearance are vastly distinct under different conditions (see Fig. 1.1a), hence an ideal image representation should remain invariant to the changes of environment. In literature, there are two main approaches to achieve an invariant image representation: hand-crafted and learning approaches.

2.1.1 Hand-crafted methods

For more than a decade, local keypoint detectors, e.g., SIFT [102], SURF [13], ORB [142], BRISK [99], and FREAK [3] have been popular choices in VPR systems [52, 93, 28, 123, 151]. Their primary idea is to detect local keypoints within images (e.g., corners) which remain repeatable under affine transformations and multi-scale representations. This ensures those keypoints are invariant to translation, rotation and scaling factors. Every

keypoint is then described by a descriptor [102, 13, 142], that is extracted from a local patch sampled around the keypoint. A pair of keypoint and descriptor is typically referred as a *local feature* in literature. Due to the importance of local features, their robustness has been widely investigated [115, 116, 153, 11]. In particular, Stylianou et al. [161] argue that environmental changes (including physical, weather, and illumination changes) affect to the physical appearance of 3D points. Their experiment consequently shows a decay in terms of the feature matching performance, which is crucially caused by the deficiency of feature detectors. Similarly, Furgale and Barfoot [52] show that under the extreme lighting changes, SURF detector fails to detect sufficient features, leading to the failure of teach-and-repeat localization system.

Another issue of representing images by local features is scalability. Schindler et al. [149] point out that 30,000 images generate about 100,000,000 SIFT features ($\approx$ 12GB memory). This is infeasible for life-long operation of mobile robots because their onboard hardware has a limited computation and memory capacity.

Therefore, Sivic and Zisserman [158] introduce BoW, followed by VLAD proposed by Jegou et al. [79]. Their basic idea is firstly embedding local features to higher dimension for improving their discriminativeness, and then aggregating them to obtain a single vector for representing the image. Inspired from this idea, Gálvez-López and Tardos [54] develop the binary version of BoW for fast VPR. Due to the efficiency of their technique, it is widely used by many existing SLAM systems [123, 139, 23] for loop closure detection.

As shown that the performance of keypoint detectors are negatively influenced by the illumination change [52], Torii et al. [167] densely extract SIFT features on images and employ VLAD for the image representation. Their experiment shows the efficiency of this approach for extreme lightning changes (day-night time).

2.1.2 Learning methods

Alternatively, recent attempts replace hand-crafted detectors and descriptors by learned ones. Learned descriptors use triplet-based loss to train Siamese deep networks on image patches, such as, Deepdesc [157], L2-net [165], Hardnet [119], and LogPolarDesc [48]. To further improve the performance of learned descriptors, some other works explore geometry constraints (e.g., GeoDesc [107]) or incorporate global and geometric contexts (e.g., ContextDesc [106]). Apart from learned descriptors, the keypoint detectors are also formulated to learning approaches, such as, TILDE [170], Quad-networks [148], and Key.Net [12].

Above methods follow the *detect-then-describe* strategy, i.e., finding local interest points first, then describing them. This strategy undergoes a performance degradation when the environmental conditions severely changes. This is because illumination variations, mainly caused by environmental changes, negatively influence to the pixel intensities, that makes feature detection unstable. To address this issue, D2-Net [44] propose the *detect-and-describe* strategy, that trains a deep network to simultaneously detect and describe local features. Revaud et al. [136] argues interest points that are highly repeatable are not necessarily reliable for matching, hence they propose R2D2 network to only extract reliable local features. D2-Net and R2D2 demonstrate a great potential of *detect-and-describe* strategy in VPR.

As indicated in Sec. 2.1.1, if only using local features, developing large-scale VPR becomes infeasible, hence applying BoW or VLAD on learned local features is a reasonable strategy. Gong et al. [64] extract CNN features on multi-scale patches of images, and use VLAD to aggregate them. They demonstrate a significant improvement in terms of accuracy in the tasks of image classification and retrieval. Similarly, Razavian et al. [155] use off-the-shelf OverFeat network with BoW and VLAD.

Above methods still separate feature extraction with feature aggregation, thus Arandjelovic et al. [7] reformulate VLAD [79] to the differentiable version, which is then plugged into CNN for the end-to-end training. The efficiency of this strategy motivates its further adaptation to place recognition with LiDAR [169], and also the attempts in developing a number of differentiable formulations of VLAD, such as, ActionVLAD [61], NeXtVLAD [100], Spatial Pyramid NetVLAD [182], SeqNet [57], and Patch-NetVLAD [71].

Apart from it, another line of works also aims to achieve a robust image representation, such as, transferring image domains (e.g., from daytime appearance to nighttime appearance) using GANs [134, 6, 95], finding latent feature maps invariant to natural changes [160], and incorporating semantic information [58].

2.2 Life-long mapping

The endless changes of appearance demand robots that can continually adapt its map to faithfully represent the latest state of the environment. Those changes might remain for a long period (e.g., the update of façades and billboards) or short time span (e.g., pedestrian or traffic density), hence a life-long VPR system should be able to decide what information should be remembered or forgotten. For cyclical changes (e.g., time of day, weathers, and seasons), a feasible strategy is to maintain multiple representations

of a particular place. This section will present the recent advancement in maintaining a life-long map for VPR.

2.2.1 Remembering and forgetting data

Interesting works in this approach are the biologically inspired mapping techniques based on concepts found in human memory system, i.e., sensory memory, short-term memory, and long-term memory. Dayoub and Duckett [35] propose rehearsal and recall operations to manage information within short- and long-term memory. In particular, the visual input is firstly processed by the selective attention that determines what information should be moved from sensor memory to short-term memory. The rehearsal mechanism selects stable features to be transferred to long-term memory, while forgetting useless features. Similarly, the recall stage in long-term memory also forget features if they are no longer present within the environment. The idea of memory model is also used in other VPR systems [93, 120], which show its efficiency in maintaining a life-long map.

Hafez et al. [67] propose a reliability weight for each feature. After multiple times of exploring the environment, the reliability weights are updated according to feature correspondences between query and database images. The features with low reliability weights are eliminated while those with high reliability weights are kept in the map. Likewise, Dymczyk et al. [47] only select stable features for updating the map. Their criteria for being a "stable" feature include: being consistently re-detected, being repeatable under illumination and viewpoint changes, and having discriminative descriptors that is easily matched.

2.2.2 Multiple representations of the environment

A notable work in this approach is RatSLAM [117]. Their argument is there is no breakpoint in the changes of the environment, hence they propose to run SLAM in parallel to other tasks to ensure robots able to constantly update the changes and maintain the latest internal representation of the world. Glover et al. [62] present a hybrid large-scale VPR system by coupling the mapping framework of RatSLAM and the data association of FAB-MAP.

Works from Dymczyk et al. [47] and McManus et al. [114] collect data from multiple traversals to discover stable features in a particular scene, which are used to train classifiers to determine stable features for localization. Churchill and Newman [28] propose plastic map, which consists of a set of visual experiences. An experience is basically the output of visual odometry, and there is an localizer associated with an experience.

During the test time, their plastic map will create a new experience if it is unable to be localized, which implicitly updates a new representation of each place. Despite effectiveness and elegance of this method, its drawback is a large number of localizers will be accumulated in the long-term operation. To overcome that issue, Linegar et al. [101] propose "path memory" to select relevant experiences that robots is likely to localize against in next steps. Their method improves VPR accuracy without suffering the growth of computational requirement.

Bürki et al. [19] update new data to the map regardless of the output of localization component. Their motivation is if a new data is unsuccessfully localized, it certainly represents a new appearance of environment, hence it should be updated to the map. By contrast, even the new data is successfully localized, it likely reveals information being helpful for future localization, thus incorporating it to the map is also necessary. However, because this strategy always accumulates new data, the map summarization [45] must be performed after the map update to maintain a bounded-size map.

2.3 Belief generation

The simplest form of VPR is to exhaustively compare the distance between query image and every database image, then return the best match [167, 7]. If recovering 6 DoF pose of the query image is demanded, we can establish 2D-3D correspondences for solving Perspective-n-Point inside RANSAC iterations [145]. If VPR returns an incorrect place, recovering 6 DoF pose will definitely output a wrong prediction; thus a large research effort has been spent to improve the VPR accuracy.

More advanced methods exploit the topological information to resolve the case that visual sensor data is unable to be matched, especially due to changing environment [118, 127, 172]. Specifically, SeqSLAM [118] formulates the matching of image sequences as dynamic time warping. [118] shows that sequence-based matching is more resilient to appearance changes but comes with its own shortcoming (e.g., sensitivity to view-point changes and differences in sequence velocities [163]). The variants of SeqSLAM eliminate the assumption of constant velocity through searching on the dissimilarity matrix [83], or using odometry as an additional input [133]. Naseer et al. [127] address the temporal matching problem by constructing a flow network and estimating its minimum cost flow. The starting and ending places within the traversal are respectively the source and sink nodes, where the source node generates the flow while the sink node consumes it. Vysotska and Stachniss [172] formulate temporal matching as the graph search, in which the shortest path corresponds to image matches. Then, they further extend their strategy to multi-sequence maps [171]. Also, some recent works concurrently exploit

probabilistic frameworks for VPR. In particular, Xu et al. [180] propose a convergence detector to determine if the posterior is converged. They also further extend their method by adding an "off-map" state to identify whether the robot has left the map or not [179].

2.4 Research gaps

Even though VPR has been a popular research topic for decades, there are several existing gaps which we wish to discuss in this section, and how the research in this thesis fill those.

- The succeed of machine learning (deep learning in particular) in challenging computer vision problems motivates the development of learning approaches toward addressing VPR, including: local features [44, 36], 2D-3D correspondences regression [156, 15], differentiable RANSAC [16, 14], and absolute pose regression [87, 17]. In spite of the fact that empirical proofs confirm state-of-the-art performance of learning approach [166, 82], there is an existing debate that deep networks only memorize data instead of learning useful knowledge [184, 42]. This argument is also empirically validated by Sattler et al [146], who shows absolute pose regression by deep networks [87, 17] is closely related to retrieval-based VPR [167].

  Furthermore, recent development of deep networks focuses on point estimate, while its uncertainty modeling is under-investigated [86]. Those drawbacks raise a question "should deep learning be the central component within robotic systems?". As discussed in chapter 1, the operating environment is largely uncontrolled and the appearance changes happen continuously and indefinitely, thus a life-long VPR will frequently encounter out-of-distribution test samples. Compared to realistic scenarios, benchmarks from [166, 82] mainly focus on natural changes that cyclically happen, yet lack of abrupt changes caused by human activities. Hence, the practicability of deep learning should be investigated more carefully.

  In addition, due to the sensitivity in the applications of robotic systems (e.g., an accident caused by self-driving cars will most likely risk human lives), it is compulsory to model uncertainty within their predictions. It motivates us to employ Bayesian framework, which is the so-called language of uncertainty [53], in our development of VPR techniques. Chapter 3 shows that Bayesian filter and Hidden Markov Model inference can efficiently explore temporal information, and demonstrates their superior performance over learning techniques.

- As indicated, recent benchmarks [145, 166] claim "long-term localization", but they only focus on cyclical changes (e.g., time of day, weather, season), while "less cyclical" changes (e.g., construction and roadworks, updating of signage, façades and billboards) and abrupt changes (e.g., traffic density) are ignored. This prevent the practicability of methods benchmarked in [145, 166] in realistic scenarios. However, from literature, to accommodate long-term evolution in appearance, the strategy that continuously accumulates data to refine VPR algorithms demonstrates its greater potential [117, 47, 114, 19]. Unfortunately, this poses a new challenge, i.e., the unbounded growth of map size, that demands a *scalable solution* for VPR. Specifically, the computational cost and memory requirement of inference should not increase linearly with the map size; and equally crucially, "absorbing" new data must also be efficient.

  There are several recent works that aim to addressing the scalable issue of VPR. Le et al. [97] employ tree structure to make VPR prediction, but their training complexity linearly increases with dataset size. Hence, they propose a compressed training protocol which reduces their method similar to hashing techniques [38, 37, 63]. Garg and Milford [56] propose a quantization-based method to compress image representations to compact codes. With the usage of a hash table, they can achieve a constant querying complexity.

  Even though those methods [97, 56] achieve a promising localization accuracy in experiments, they suffer from two major drawbacks. Firstly, they require retraining their hash functions to "absorb" new data while data collection of VPR operates continuously, hence the scalability in their map update is not satisfied. Secondly, their methods do not offer the theoretical guarantees of similar images compressed to a same code—leading to a debate regarding its safety in long-term operation, in which mobile robots likely encounter unseen data frequently. Note that Sablayrolles et al. [144] emphasize the importance of evaluating learning-based compression methods on a benchmark with out-of-distribution testing samples.

  The approach that is inherited from a fundamental hashing theory is from Lowry and Andreasson [104], who use Locality Sensitive Hashing (LSH) [26] to compress VLAD vectors [79] to binary codes. However, their technique directly applies the hyperplane LSH to VLAD vectors without further exploring the characteristics of locally aggregated vectors on unit sphere to gain a more appropriate compression scheme. Furthermore, as new data is accumulated, the memory complexity of their method linearly increases with the database size, which does not meet the scalable demand of life-long VPR.

To resolve those existing issues, chapter 4 presents a novel temporal image matching technique based on Hidden Markov Models (HMM), which is efficiently retrained and compressed, such that the inference of new queries can exploit all available data (including recent appearance changes), without suffering from the visible growth of computational cost. Coupling with the two-tiered memory management, chapter 5 presents Hidden Markov Model with Memory Management (dubbed as HM^4), that exploits temporal look-ahead to transfer promising candidate images between passive storage and active memory when needed. To represent the full topological map, it constructs a coarse representation of the map. The HMM inference is adapted to take into account both promising images and the coarse representation. We show that HM^4 allows a constant time and memory needed for inference provided that the coverage area is fixed. Our method also quickly updates new data to refine the map and its coarse representation. To further achieve a lightweight system, inspired by LSH [5], we derive a novel compact image representation by exploiting the characteristics of VLAD vectors on unit sphere. This yields polyVLAD, which greatly reduces memory footprints of image presentations while still maintaining an excellent localization accuracy. Also, employing the inverted index structure allows a fast distance computation between polyVLAD vectors, leading to a significant improvement in the VPR inference time. Combining HM^4 with polyVLAD offers us a lightweight and scalable VPR system.

- Mobile robots are usually equipped a limited onboard hardware, thus map summarization is necessary to obtain a compact map for large-scale environment. [25] formulates the problem to K-cover algorithm, which selects a minimal subset of 3D points such that each database image sees at least K number of points regardless of its description. To address this issue, [24] uses weighted K-cover algorithm, which considers the discriminative power of 3D point descriptors. Instead of decimating points for space coverage, [45, 151] sample 3D points w.r.t observation frequency and low uncertainty. Generally, existing works select or eliminate 3D points according to predefined criteria, which have not regarded the time span as a constraint in their sampling process. However, we empirically observe that in urban area, many 3D points are easily detected in daytime, but disappear when it comes to the nighttime. In contrast, some other 3D points becomes easier to be detected at night because the lights from urban area are on. Based on this observation, chapter 6 proposes a novel VPR scheme that periodically invokes the map summarization part. During the VPR session, map summarization samples highly repeatable 3D points according to the current run-time of VPR. This sampling process relies on a repeatability predictor that can predict the repeatability of an

interest point as a function of time. The repeatability scores of interest points are then used to determine if their corresponding 3D point is repeatable at the current VPR run-time.

Chapter 3

Filtering Approaches for Visual Localization

This chapter is based on the content of following conference and journal papers

- Anh-Dzung Doan, Yasir Latif, Thanh-Toan Do, Yu Liu, Shin-Fang Ch'ng, Tat-Jun Chin, and Ian Reid. Visual Localization Under Appearance Change: A Filtering Approach. In *International Conference on Digital Image Computing: Techniques and Applications* (DICTA), 2019.
- Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, Yu Liu, Shin-Fang Ch'ng, Thanh-Toan Do, and Ian Reid. Visual Localization Under Appearance Change: Filtering Approaches. *Neural Computing and Applications (NCAA)*, 2020.

Statement of Authorship

Title of Paper	Visual Localization Under Appearance Change: Filtering Approaches
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, Yu Liu, Shin-Fang Ch'ng , Thanh-Toan Do, Ian Reid, *"Visual Localization Under Appearance Change: Filtering Approaches"*, Neural Computing and Applications (NCAA) 2020. Anh-Dzung Doan, Yasir Latif, Thanh-Toan Do, Yu Liu, Shin-Fang Ch'ng, Tat-Jun Chin, Ian Reid, *"Visual Localization Under Appearance Change: A Filtering Approach"*, International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2019.

Principal Author

Name of Principal Author (Candidate)	Anh-Dzung Doan		
Contribution to the Paper	Developed the methods, conducted the experiments, and wrote the papers.		
Overall percentage (%)	55		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	19 February 2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Yasir Latif		
Contribution to the Paper	Supervised the development of the methods, provided the suggestions, and revised the papers.		
Signature		Date	23/02/2021

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Proposed the general direction, supervised the development of the methods, provided the suggestions, and revised the papers.		
Signature		Date	21 Feb 2021

Name of Co-Author	Yu Liu		
Contribution to the Paper	Helped with the experiments and proofread the papers		
Signature		Date	Feb 22, 2021

Name of Co-Author	Shin-Fang Ch'ng		
Contribution to the Paper	Provided suggestions and proofread the papers		
Signature		Date	22 Feb 2021

Name of Co-Author	Thanh-Toan Do		
Contribution to the Paper	Provided suggestions and proofread the papers		
Signature		Date	22-Feb-2021

Name of Co-Author	Ian Reid		
Contribution to the Paper	Provided suggestions and proofread the papers		
Signature		Date	23/2/21

3.1 Introduction and research questions

To carry out higher level tasks such as planning and navigation, a robot needs to maintain, at all times, an accurate estimate of its position and orientation with respect to the environment. VPR methods normally assume that the appearance remains unchanged from when the map is generated to the present time when the robot needs to localize itself. However, as the operational time span of the robot increases, the appearance of the environment inevitably changes. This poses a great challenge for VPR methods as the underlying assumption of static appearance is violated due to continual changes of environment, e.g., weather condition, time of day, construction sites, updating of façades and billboards, etc.

A popular approach towards dealing with appearance change is to observe as many variations as possible of each location and carry a representation that can model them [117, 28]. Unfortunately, its bottleneck is the significant amount of time needed to capture sufficient appearance variations. For example, recent benchmarks [145, 74, 111] have taken years to collect data that satisfactorily covers natural variability of appearance.

An orthogonal approach to address appearance change, pioneered by SeqSLAM [118], is to consider sequence-to-sequence matching instead of matching a single query image to previously observed images. SeqSLAM showed that sequence-based matching is more resilient to appearance changes but comes with its own shortcoming (e.g., sensitivity to view-point changes and differences in sequence velocities [163]).

Another paradigm to overcome this challenge is to extract "fingerprints" which are robust against appearance changes [167, 7]. Yet, this approach has been only shown its efficiency in "natural" changes, e.g., weather conditions, or time of day. Another line of works leverage deep learning for establishing 2D-3D correspondences or regressing absolute poses [14, 87, 85]. However, the major drawback of these approaches is focusing on the case of a single query image, while the camera, in robotics setting,is operating continuously once the vehicle is moving and hence, it is realistic to expect a video (sequence of images) as an input to VPR methods.

Therefore, in this chapter, we ask this following research question:

Research question 1 *How to effectively model temporal information?*

Even though recent algorithms [29, 17] demonstrate a potential of temporal consistency in improving the localization accuracy, they are not supported by any fundamental theory in modeling temporal information. Hence, to comprehensively answer this research questions, we break it down into a couple of sub-questions

RQ1.1 *How should we develop probabilistic-based algorithms?*

We revisit Bayes filter [164, Chapter 2] as the backbone in developing our VPR algorithms. Specifically, Bayes filter maintains belief distributions over states (i.e., places) conditioned on observed measurements (e.g., images). Belief distributions are recursively computed from measurement and motion data, that consist of two steps: the motion model receives the motion information to predict the belief at time t based on the prior belief at time $t-1$, and then observation model takes the noisy measurement to update the belief. Basically, Bayes filter is derived from Bayes rule. For more detail of mathematical derivation, we refer readers to [164, Chapter 2].

To effectively reason over an image sequence (video), we propose two visual localization (VL) methods:

- *Visual localization with Hidden Markov Model* (Sec. 3.4): the inference is performed on image indices to retrieve place hypotheses from which query frames are possibly captured. The 6 DoF camera poses of query frames are then interpolated from those place hypotheses. The temporal information is handled by a transition matrix to effectively exploit the temporal continuity of query sequences.
- *Monte Carlo-based visual localization* (Sec. 3.5): the Monte Carlo principle is leveraged to approximate the probability distribution of 6 DoF camera poses incrementally. We show that for the case of driving on urban streets, a simple motion model is enough to successfully track the vehicle's pose.

The above VL methods rely on a novel *observation encoder* (Sec. 3.3) which generates a fixed (low) dimensional vector representation for each image. We experimentally show that such representation is more resilient to appearance variations along with minor changes in the viewpoint, while being compact in size. A comprehensive analysis about our proposed observation encoder is presented in Sec. 3.8.3, Sec. 3.8.4, and Sec. 3.8.5.

Sec. 3.8.6 shows the comparison between *visual localization with Hidden Markov Model* (discrete domain) and *Monte Carlo-based visual localization* (continuous domain). Also, our methods are compared against deep learning methods in the large-scale environment with significant appearance changes, i.e., the traversal distance of the vehicle is about 10 km.

RQ1.2 *How to cost-effectively evaluate VPR methods?*

(a) Clear (b) Rain (c) Snow

(d) Clear (e) Rain (f) Snow

Fig. 3.1. Images under different conditions with the same camera pose. The first and second rows respectively correspond to day (12:00) and night (23:00) time.

The testing of a broad range of robotic vision algorithms require datasets with accurate 6 DoF groundtruth poses, e.g., SfM (structure from motion) [159], visual SLAM (simultaneous localization and mapping) [34], camera pose estimation [98]. Furthermore, to thoroughly assess the performance of the algorithms on realistic operating conditions, it is vital to use datasets captured under varying conditions. Indeed, recent studies [153, 185] show that the efficacy of local feature detectors, which form the first input to many computer vision algorithms, reduces significantly under different environmental conditions (weather, time-of-day, traffic density, etc.).

Unfortunately, collecting the necessary image datasets under varying conditions is extremely costly and time-consuming. This has been a persistent obstacle towards developing and testing computer vision algorithms that are robust and reliable under varying operating conditions.

To answer this research question, Sec. 3.6 presents G2D, an image simulator software that exploits the detailed virtual environment in Grand Theft Auto V (GTA V). G2D allows users to collect hyper-realistic computer-generated imagery of an urban scene, under controlled 6 DoF camera poses and varying environmental conditions (weather, season, time of day, traffic density, etc.). Users directly interact with G2D while playing the game; specifically, users can manipulate conditions of the virtual environment on the fly, while the gameplay camera is set to automatically retrace a predetermined 6 DoF camera pose trajectory within the game coordinate system. Concurrently, automatic screen capture is executed while the virtual environment is being explored. The output

of G2D is a set of images with 6 DoF groundtruth camera poses, captured under varying conditions; see Fig. 3.1.

3.2 Problem definition

We define the map $\mathcal{D} = \{\mathcal{V}^1, ..., \mathcal{V}^n\}$, where $\mathcal{V}^i = \{(I_1^i, s_1^i), ..., (I_T^i, s_T^i)\}$ is an image sequence (video) i. Every frame I_t^i is associated with a 6 DoF camera pose s_t^i, which can be attained by conducting Visual SLAM [20] or Structure from Motion [152].

For the online inference, given a query video $\mathcal{Q} = \{Q_1, ..., Q_T\}$, our goal is to estimate the camera pose s_t for each Q_t with respect to the map $\mathcal{D}$.

As our both methods rely on the observation encoder which maps every image to a single vector, we are going to describe it first in Sec. 3.3. Next, Sec. 3.4 and Sec. 3.5 respectively detail our proposed visual localization methods using HMM and Monte Carlo principle. In Sec. 3.6, we present G2D—a software to assist researchers in collecting data from the computer game Grand Theft Auto V (GTA V). Using G2D, we create the synthetic dataset, which simulates the setting of data collection by multiple vehicles under different time of day and environmental conditions. Finally, experimental results and conclusion are respectively presented in Sec. 3.8 and Sec. 3.9.

3.3 Observation encoder

We seek for every image I a nonlinear function $\tau(I)$ that maps I to a vector in a fixed dimensional space. To do so, we first densely compute SIFT features: $\{x_i \in \mathbb{R}^d \,|\, i = 1, ..., m\}$ over the image, followed by RootSIFT normalization [9]:

1. $L1$ normalize every SIFT vector $x_i = x_i / ||x_i||_1$
2. square root each element $x_i = \sqrt{x_i}$

where, RootSIFT normalization makes the Euclidean distance calculation among SIFT features equivalent to computing the Hellinger kernel.

Subsequently, we employ the embedding function VLAD [80] to embed SIFT features into a higher dimensional vector space. In particular, given a vocabulary learned by K-means: $\mathcal{C} = \{c_k \in \mathbb{R}^d \,|\, i = 1, ..., K\}$, every SIFT feature is embedded as follows:

$$\phi_{VLAD}(x_i) = [..., 0, x_i - c_j, 0, ...] \in \mathbb{R}^D \tag{3.1}$$

where c_j is the nearest visual word to x_i, and $D = K \times d$. Note that different from Bag of Word (BoW), which embeds the feature vector as follows:

$$\phi_{BoW}(x_i) = [..., 0, 1, 0, ...] \in \mathbb{R}^K \tag{3.2}$$

where only j^{th} component of $\phi_{BoW}(x_i)$ non-zero means that the nearest neighbor of feature x_i is visual word c_j; VLAD considers the residual between a visual word and its nearest feature. Do et al. [39] show that VLAD is a simplified version of local coordinate coding [183], which tries to approximate a nonlinear classification function by a linear function.

From a set of embedded vectors: $\{\phi(x_i) \in \mathbb{R}^D \,|\, i = 1, ..., m\}$, we aggregate them by the sum pooling to obtain a single vector representing the image I:

$$\tau(I) = \sum_{i=1}^{m} \phi(x_i) \tag{3.3}$$

In literature, there are several other ways for this aggregation step [81, 124], but for simplification, we choose sum pooling and show that it can obtain good performance in practice (see Sec. 3.8.5).

One drawback of using local features in image retrieval is that the background features (e.g., trees, sky, road, etc) significantly outnumber features from informative objects (e.g., buildings). To alleviate this, we apply PCA projection, whitening and L2-normalization [77], which limit the impact of background features in the vector $\tau(I)$ (see Sec. 3.8.4).

3.4 Visual localization with HMM

Let the map $\mathcal{D} = \{I_j, s_j\}_{j=1}^M$ be the database of M images after "unrolling" all videos $\mathcal{V}^i = \{(I_1^i, s_1^i), ..., (I_T^i, s_T^i)\}$. We regard each image index j as a *"place"*. If the field of view of two images are overlap sufficiently (e.g., if they are temporally close in their source video), we will assign a transition probability between these two image indices. The detail of computing a full transition matrix of $\mathcal{D}$ will be given in Sec. 3.4.1.1.

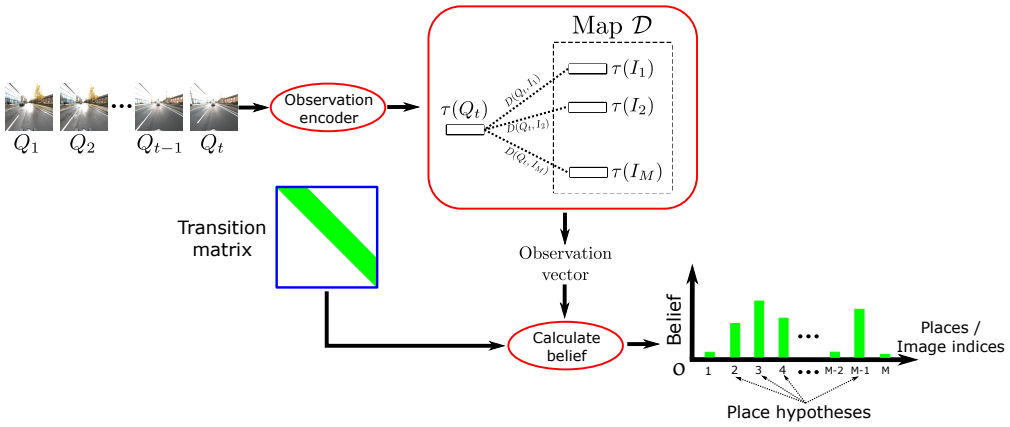


Fig. 3.2. The diagram of our proposed HMM for visual localization. Assume the map $\mathcal{D}$ has M images after "unrolling" all videos $\mathcal{V}^i = \{(I_1^i, s_1^i), ..., (I_T^i, s_T^i)\}$, where each image indices is regarded as a physical place. Places with top highest belief probabilities are selected as place hypotheses, which are then used for estimating 6 DoF pose (see Sec. 3.4.2).

Given a query video $\mathcal{Q} = \{Q_1, ..., Q_T\}$, our goal is to find place hypotheses $\mathcal{H}_t$ for every Q_t (see Sec. 3.4.1). Then, the 6 DoF camera pose of every Q_t will be estimated through place hypotheses $\mathcal{H}_t$ (see Sec. 3.4.2)

3.4.1 Place hypotheses from HMM

To find the place hypotheses on query video $\mathcal{Q}$, we model $\mathcal{Q}$ using a Hidden Markov Model (HMM). We regard each image Q_t as a noisy measurement of *place state* v_t, where $v_t \in \{j\}_{j=1}^M$, and as mentioned above, $\{j\}_{j=1}^M$ is set of image indices (or places) of database $\mathcal{D}$

A Hidden Markov Model (HMM) consists of three parameters $\{\mathbf{E}, \mathbf{O}, \pi\}$, which respectively are transition matrix, observation model and initial state probabilities. Size of HMM parameters are: $\mathbf{E} \in \mathbb{R}^{M \times M}$, $\mathbf{O} \in \mathbb{R}^{M \times T}$ and $\pi \in \mathbb{R}^M$. Specifically,

$$\mathbf{E}(j_1, j_2) = P(v_t = j_2 | v_{t-1} = j_1) \tag{3.4}$$

where, the value at row j_1 and column j_2 of matrix $\mathbf{E}$ represents the probability of transitioning from place j_1 to place j_2.

The observation matrix is defined as follows:

$$\mathbf{O}(j,t) = P(Q_t|v_t = j) \tag{3.5}$$

which represents the likelihood probability of the frame I_j matched against query frame Q_t. The recursive belief $\mathbf{p}_t = P(v_t|Q_{1:t})$ is computed in the following:

$$\mathbf{p}_t = \eta\, \mathbf{o}_t \circ \mathbf{E}^T \mathbf{p}_{t-1} \tag{3.6}$$

where, $\mathbf{p}_t(j)$ is the belief probability of place j, given queries up to time t (i.e., $Q_{1:t}$). $\circ$ is Hadamard product, $\mathbf{o}_t$ is the t^{th} column of $\mathbf{O}$, η is the normalizing constant to ensure $\sum \mathbf{p}_t = 1$. At time $t = 1$, $\mathbf{p}_1 = \pi$, where $\pi(j) = 1/M$.

The place hypotheses $\mathcal{H}_t$ at time t are top k places with largest $\mathbf{p}_t(j)$:

$$\mathcal{H}_t = \{j \mid \text{top } k \text{ largest } \mathbf{p}_t(j)\} \tag{3.7}$$

3.4.1.1 Transition matrix

Given an video $\mathcal{V}^i = \{I_1^i, ..., I_T^i\}$, a transition matrix E^i of size $T \times T$ is formed as follows:

$$E^i(r,c) = \begin{cases} 1 & 0 \leq c - r \leq \mathbf{v}_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

where, $E^i(r,c)$ is the element at row r^{th} and column c^{th}. $\mathbf{v}_{\max}$ is the maximum velocity of the vehicle. The transition matrix $\mathbf{E}$ of database $\mathcal{D}$ is then created by concatenating every E^i in the diagonal direction:

$$\mathbf{E} = \begin{bmatrix} E^1 & & & \\ & E^2 & & \\ & & \ddots & \\ & & & E^n \end{bmatrix} \tag{3.9}$$

Subsequently, transition matrix $\mathbf{E}$ is normalized to ensure the summation of every row of $\mathbf{E}$ equals to 1, i.e., $\sum_c \mathbf{E}(r,c) = 1$

3.4.1.2 Observation model

The distance between Q_t to every database image $I_j \in \mathcal{D}$ is computed in the following

$$D(Q_t, I_j) = \left\| \tau(Q_t) - \tau(I_j) \right\|_2^2 \tag{3.10}$$

where, $\tau(.)$ is the observation encoder (see Sec. 3.3). The observation vector $\mathbf{o}_t$ is then calculated:

$$\mathbf{o}_t = \exp\left(-\frac{D}{\sigma}\right) \tag{3.11}$$

3.4.2 Estimating 6 DoF pose

Due to every image in $\mathcal{D}$ associated with a corresponding 6 DoF camera pose, mean-shift algorithm is employed on $\mathcal{H}_t$ over the translational part of their poses. The largest cluster is then selected to calculate the mean of translation and rotation [112], which is regarded as predicted 6 DoF pose s_t of the image query Q_t.

3.4.3 Overall algorithm

The Algorithm 3.1 presents the method of visual localization with HMM. Given the query Q_t, we calculate its fixed high-dimensional vector representation $\tau(Q_t)$. The Euclidean distances between Q_t and every database frames I_j is computed, which is then used to estimate the observation vector $\mathbf{o}_t$. The belief $\mathbf{p}_t$ is subsequently computed which is utilized to form place hypotheses $\mathcal{H}_t$. The predicted pose of Q_t is finally estimated from place hypotheses $\mathcal{H}_t$.

Algorithm 3.1 Visual localization with HMM

Require: Q_t, $\mathcal{D}$, and $\mathbf{E}$
1: Compute $\tau(Q_t)$ as Sec. 3.3
2: Compute $D\left(Q_t, I_j\right)$, $\forall I_j \in \mathcal{D}$ using equation (3.10).
3: Compute observation vector $\mathbf{o}_t$ using equation (3.11)
4: Compute belief $\mathbf{p}_t$ using Eq. (3.6)
5: Form place hypotheses $\mathcal{H}_t$ using Eq. (3.7)
6: Estimate 6 DoF pose s_t of Q_t (Sec. 3.4.2)
7: **return** s_t

3.5 Monte Carlo-based visual localization (MCVL)

Let the 6 DoF camera pose of Q_t be given by:

$$s_t = [r_t,\ \Omega_t]^T \tag{3.12}$$

where, r_t and Ω_t represent the 3D position and Euler orientation respectively at time t. Given motion $u_{1:t}$ and noisy measurement $z_{1:t}$ up to time t, we would like to estimate probability distribution $p(s_t|u_{1:t}, z_{1:t})$. However, $p(s_t|u_{1:t}, z_{1:t})$ can be an arbitrary distribution, thus we leverage Monte Carlo principle to address this issue.

The idea of Monte Carlo-based visual localization is to represent the probability distribution $P(s_t|u_{1:t}, z_{1:t})$ with a set of N particles. Each particle maintains an estimate of 6 DoF pose at time t:

$$\mathcal{S}_t = \{s_t^{[1]}, s_t^{[2]}, ..., s_t^{[N]}\} \tag{3.13}$$

with a set of corresponding weights:

$$\mathcal{W}_t = \{w_t^{[1]}, w_t^{[2]}, ..., w_t^{[N]}\} \tag{3.14}$$

The poses of particles and their corresponding weights are respectively updated according to the motion u_t (Sec. 3.5.1) and noisy measurement z_t (Sec. 3.5.2) at time t. In the Sec. 3.5.1, we also justify the use of a simple motion model. Fig. 3.3 shows an overview of MCVL.

3.5.1 Motion model

In the autonomous driving scenario when navigating the roads of an urban area[1], the motion of the car is fairly restricted, i.e., it largely stays on a road network on an approximately 2D plane[2] [141]. While the road surface still allows significant scope for

[1] In more "localized" operations such as parking, where highly accurate 6 DoF estimation is required, it is probably better to rely on the INS.

[2] More fundamentally, the car is a non-holonomic system [178].

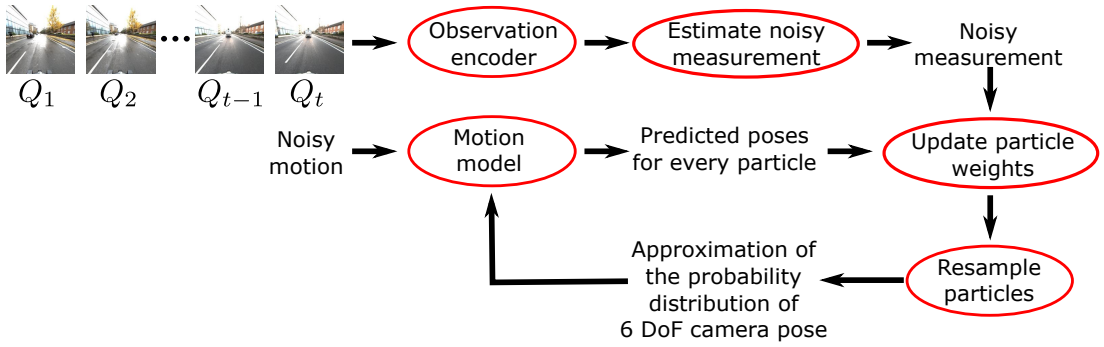


Fig. 3.3. The diagram of our proposed Monte Carlo-based visual localization (MCVL).

movement (cf. Fig. 3.7), relative to the size of the map, the motion of the car is rather limited[3]. This is echoed by the recent work of [145], who observed that there is "*lower variation in viewpoints as the car follows the same road*". Hence, a Monte Carlo scheme with a simple motion model suffices to track the 6 DoF rigid motion of the vehicle. In many cities, the road networks are complex Euclidean graphs. In fact, it is well known that using (visual) odometry alone, it is possible to accurately track a car on a 2D road map [18][4]. More fundamentally, this suggests that temporal continuity in the testing sequence (which is fully exploited by our method) strongly benefits VL.

Mathematically, for each particle, we model the noisy motion consisting of linear velocity $v_t^{[i]}$ and angular velocity $\psi_t^{[i]}$ as the following:

$$\begin{aligned} v_t^{[i]} &\sim \mathcal{N}(\mu_v, \Sigma_v) \\ \psi_t^{[i]} &\sim \mathcal{N}(\mu_\psi, \Sigma_\psi) \end{aligned} \tag{3.15}$$

where, μ_v and μ_ψ respectively represent the linear and angular velocities. The accelerations are modeled by the noise covariance matrices Σ_v and Σ_ψ. For each particle, their motion in each time step is given by:

$$u_t^{[i]} = [v_t^{[i]},\ \psi_t^{[i]}]^T \tag{3.16}$$

In practice, the μ_v, μ_ψ, Σ_v, and Σ_ψ can be either manually tuned, or estimated from training data [88].

While 3D positions can be easily updated by using simple additions, we convert two Euler angles to the Direction Cosine Matrix (DCM), multiply two matrices and convert

[3] On uneven or hilly roads, accelerometers can be used to estimate the vertical motion, hence, VL can focus on map-scale navigation.

[4] The method of [18] will give ambiguous results on non-informative trajectories, e.g., largely straight routes. Hence, VL is still crucial.

the result back to the Euler representation, to stay on the 3D manifold of valid rotations. Let $\varphi(.)$ be a function that maps an Euler representation to DCM and $\varphi^{-1}(.)$ is its inverse mapping. Our motion model for each particle is then given by:

$$s_t^{[i]} = \begin{bmatrix} r_{t-1}^{[i]} + v_t^{[i]} \\ \varphi^{-1}\left(\varphi(\psi_t^{[i]}).\varphi(\Omega_{t-1}^{[i]})\right) \end{bmatrix} \tag{3.17}$$

The experimental results will show that our motion model can properly handle the temporal evolution of camera pose in an image sequence. In case there is mismatch between the actual motion and the one predicted by the motion model, such as during emergency maneuvers, the discrepancy would be reflected in the enlarged covariance estimate and resolved once motion returns to within normal bounds.

3.5.2 Estimating noisy measurements

The similarity between the query and database images is calculated using L-2 distance:

$$D(Q_t, I_i) = \left\| \tau(Q_t) - \tau(I_i) \right\|_2^2 \tag{3.18}$$

where, $\tau(.)$ is the observation encoder (see Sec. 3.3). Afterwards, top R database images with smallest distances are retrieved. Next, mean-shift algorithm is applied on R retrieved images over the translational part of their poses. We then select the largest cluster, and calculate the mean of translation and rotation [112], which is viewed as a noisy measurement z_t from the image query I_t.

3.5.3 Updating particle weights and Resampling

For every particle, its weight is computed as the following:

$$w_t^{[i]} = p\left(z_t | s_t^{[i]}\right) \propto e^{-\frac{1}{2}(z_t - s_t^{[i]})^T \Sigma_o^{-1} (z_t - s_t^{[i]})} \tag{3.19}$$

where, Σ_o is a covariance matrix which describes the noise of the measurement obtained by observation encoder. Then, all particle weights are normalized to ensure their summation equal to 1:

$$\forall i, w_t^{[i]} = \frac{w_t^{[i]}}{\sum_{j=1}^{n} w_t^{[j]}} \tag{3.20}$$

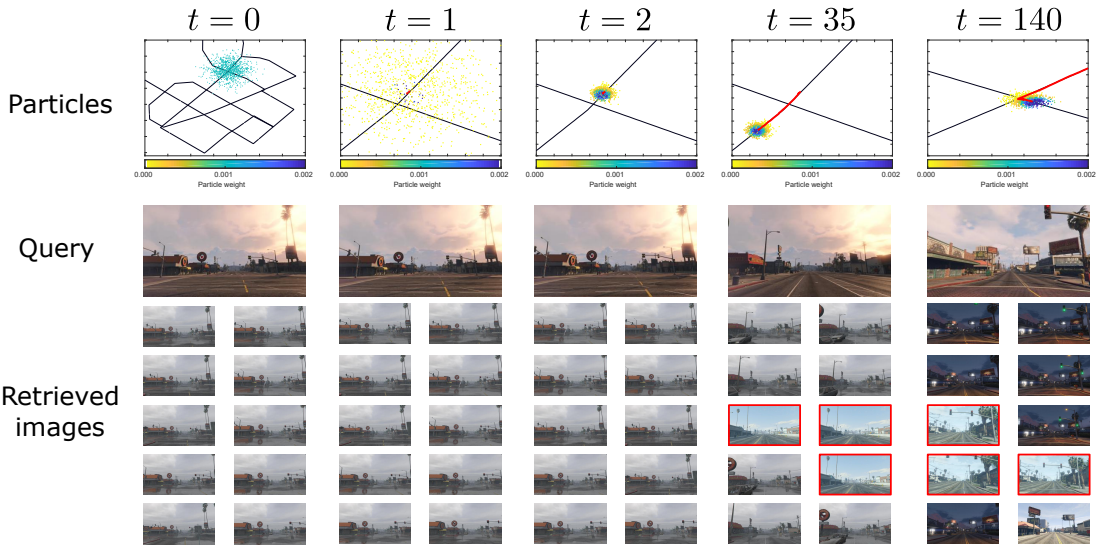


Fig. 3.4. An example of MCVL. 1st and 2nd rows respectively are the particle distribution and query image, from 3rd to 7th rows are the retrieval results. Red bounding boxes indicate mistakes. The color bar describes the particle weights. Red lines are the camera pose predicted by MCVL. At the first frame, we randomly generates 1,000 particles with weight 0.001, then particles which are inconsistent with measurements are vanished. Eventually, the ambiguity is resolved, and the particles track the movement of the vehicle. It is also worth mentioning that the query and retrieved images are from different time and weather conditions. In particular, the query images are from 6:26pm-cloudy, and the correct retrieved images are from 11:24am-rainy, 9:15pm-clear, and 1:28pm-sunny conditions.

Finally, we resample particles based on their weights by stochastic universal sampling [177]. This resampling step prevents the degeneracy problem, which can occur in the long-term localization scenario. Fig. 3.4 shows the filtering process performed by our proposed method. At the first iteration, hypotheses are randomly generated. Hypotheses with small weights vanish if they are inconsistent with the noisy measurement. Finally, the ambiguity is resolved, and the particles successfully track the vehicle. It is worth noting that in the example shown, the query and retrieved images are from different times and weather conditions.

3.5.4 Overall algorithm

Algorithm 3.2 summarizes the proposed Monte Carlo-based visual localization method. The critical benefit of maintaining a set of particles is leveraging Monte Carlo principle to approximate the probability distribution $P(s_t|u_{1:t}, z_{1:t})$. As the number of particles N are sufficiently large, this set of particles are equivalent to a representation of probability density function. The motion model ensures the temporal smoothness of the vehicle's movement.

Algorithm 3.2 Monte Carlo-based Visual Localization (MCVL)

Require: $\mathcal{S}_{t-1}$, $\mathcal{W}_{t-1}$, and Q_t
$\overline{\mathcal{S}_t} = \mathcal{S}_t = \emptyset$
$\overline{\mathcal{W}_t} = \mathcal{W}_t = \emptyset$
Estimate z_t from Q_t (Sec. 3.5.2).
for $i = 1$ to N **do**
  sample $v_t^{[i]}$ and $\psi_t^{[i]}$ as equation (3.15).
  compute $s_t^{[i]}$ as equation (3.17).
  compute $w_t^{[i]}$ as equation (3.19).
  $\overline{\mathcal{S}_t} = \overline{\mathcal{S}_t} + < s_t^{[i]} >$
  $\overline{\mathcal{W}_t} = \overline{\mathcal{W}_t} + < w_t^{[i]} >$
end for
normalize $w_t^{[i]}$ as equation (3.20).
for $i = 1$ to n **do**
  draw i with probability $w_t^{[i]}$ (Sec. 3.5.3).
  add $s_t^{[i]}$ to $\mathcal{S}_t$.
  add $w_t^{[i]}$ to $\mathcal{W}_t$.
end for
return $\mathcal{S}_t$

3.6 G2D: From GTA to Data

An intuitive summary of the functionality of G2D is given in Fig. 3.5. Overall, creating sparse trajectory is a user-defined step with the assistance of trajectory tool. After obtaining a sparse trajectory, a dense trajectory could be constructed in two manners: automatic-defined or user-defined orientations. The option of user-defined orientation gives a permission to define the image appearance as users preference. With the obtained dense trajectory, users could retrace that one and simultaneously collect the image set with its camera pose set. Apart from trajectory tool, condition tool could change the conditions of the game, i.e., weather, time and traffic density. Because condition tool and trajectory tool work independently, **several image sets** with a **consistent camera pose set** under **different conditions** could be collected. More details are provided in the rest of this document, as well as in the project page: https://sites.google.com/view/g2d-software/home.

3.6.1 Scripthook V

G2D is based on Scripthook V [5], a library developed by Alexander Blade that provides access to the native functions of GTA V (the usage Scripthook V distinguishes our G2D from Richter et al. [138, 137]), a wide range of fascinating mods are available [6], e.g.,

[5] http://www.dev-c.com/gtav/scripthookv/
[6] https://www.gta5-mods.com/

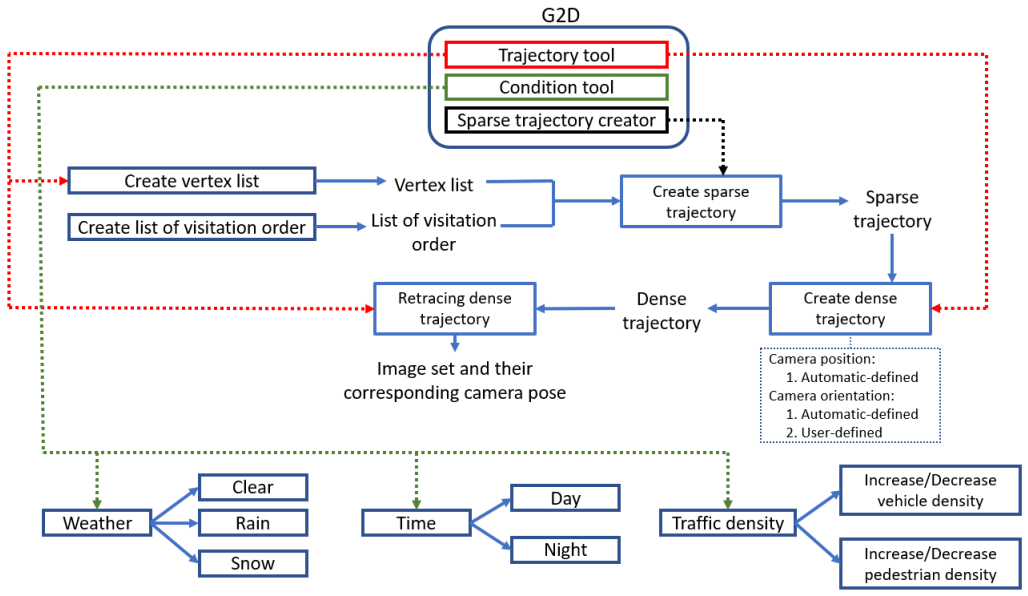


Fig. 3.5. An illustration of all functions of G2D. G2D contains 3 separate tools: trajectory tool, condition tool and sparse trajectory creator. Specifically, due to the ability of accessing to the native functions of GTA V, the condition tool enables users to easily manipulate the time, weather and traffic density. In addition, trajectory tool and sparse trajectory creator assist users to create a dense trajectory and finally collect a set of images with their corresponding camera poses. It is worth to note that because condition tool and trajectory tool are separate tools, G2D enables users to collect several image sets with a consistent camera pose set under varying conditions.

Invisibility Cloak [7] that can make the protagonist invisible. The list of native functions supported by Scripthook V could be found at http://www.dev-c.com/nativedb/.

3.6.2 Constructing trajectory

G2D defines two types of camera trajectories: *sparse* and *dense* trajectories. A sparse trajectory consists of a set of vertices (a set of positions on the "top down" 2D map of the virtual environment), and an order in which to visit the vertices. Users specify sparse trajectories. Then, given a sparse trajectory, a dense trajectory is generated automatically by G2D. Basically, the tool traces a continuous path along the dense trajectory and captures the scene as observed from the gameplay camera at **60 frames per second**[8], along with the 6DOF camera pose at each frame. In other words, G2D still guarantees the normal operation of the game as well as the collected dataset is in the standard video rate.

Fig. 3.6 shows an example, while more details are available in the following.

[7] https://www.gta5-mods.com/scripts/invisibility-cloak

[8] We measure this performance from a workstation Intel(R) Core(TM) i7-6700 @ 3.40GHz, RAM 16GB, NVIDIA GeForce GTX 1080 Ti and the maximum graphical configuration for GTA V

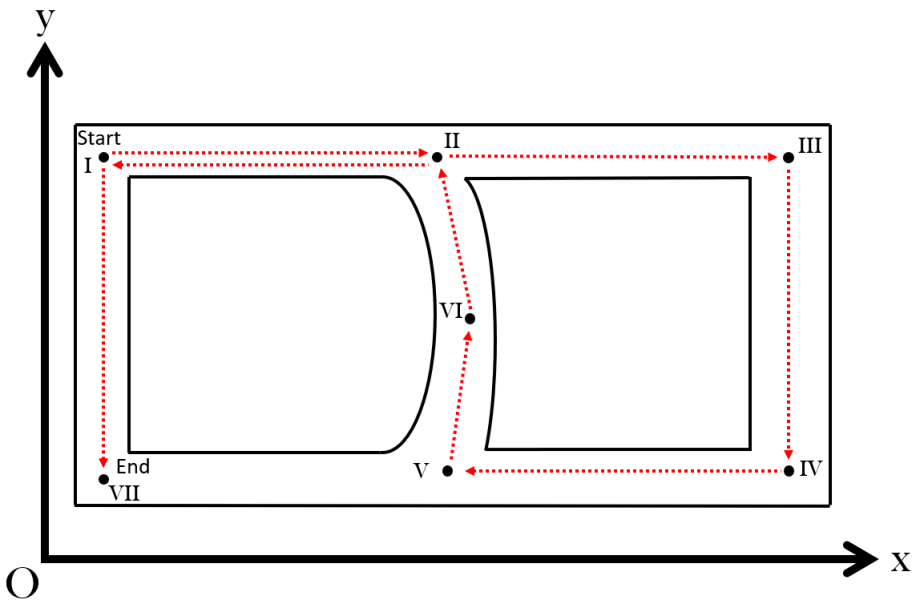


Fig. 3.6. Illustrating sparse and dense trajectories on a 2D map/coordinate system. Points labelled with Roman numerals indicate the vertices of a sparse trajectory; the order of visitation of the vertices is given by the red arrows. The sampled frames along with the 6DOF camera pose at each frame while automatically traversing the sparse trajectory gives rise to a dense trajectory.

3.6.2.1 Sparse Trajectory (user-defined)

The vertices $\{I, II, \dots\}$ of a sparse trajectory are defined by their coordinates on a 2D map $\{(x_I, y_I), (x_{II}, y_{II}), \dots\}$. The order of visitation is specified using an index

$$\begin{aligned} ORDER_I &= \{a, b, c, \dots\} \\ ORDER_{II} &= \{x, y, z, \dots\} \\ &\vdots \end{aligned}$$

where a, b, c are integers indicating the order in which vertex I is visited in the desired sequence (similarly x, y, z for vertex II). For the example in Fig. 3.6, the index would

be

$$ORDER_I = \Big\{1, 8\Big\}$$
$$ORDER_{II} = \Big\{2, 7\Big\}$$
$$ORDER_{III} = \Big\{3\Big\}$$
$$ORDER_{IV} = \Big\{4\Big\}$$
$$ORDER_V = \Big\{5\Big\}$$
$$ORDER_{VI} = \Big\{6\Big\}$$
$$ORDER_{VII} = \Big\{9\Big\}$$

In G2D, the vertex and order of visitation are specified in the files `vertex.txt` and `vertex_order.txt` respectively.

3.6.2.2 Dense Trajectory (generated automatically)

Given a user-specified sparse trajectory, G2D moves the protagonist of the game to automatically follow the trajectory. The orientation (rotation) of the camera while the movement is being executed can be specified in two modes:

- First-person view mode: G2D attaches the gameplay camera to the "eyes" of the protagonist, and the viewing direction always points forward without the need to handle camera orientations by the user.
- Third-person view mode: While the protagonist automatically moves along the trajectory, the user can use the mouse to control the orientation of the camera.

While the environment is being explored, G2D calls the relevant native functions and performs the necessary computations to obtain the 6DOF pose of the gameplay camera at each frame. Every 6DOF pose is stored in line-by-line manner within the file `trajectory_dense.txt`. Each line within `trajectory_dense.txt` has the following format:

```
<protagonist position XYZ> <camera position XYZ> <camera rotation XYZ>
```

It is worth noting that because the dense trajectory is simply an editable text file, users could easily open the dense trajectory and make some manual modifications to create a noisy version of trajectory, hence a more challenging dataset could be generated.

3.6.3 Retracing a dense trajectory

With the obtained dense trajectory, G2D opens the file `trajectory_dense.txt`, sequentially loads each line within that file and then sets 6 DoF pose to the camera object. With each 6 DoF value, G2D performs a screenshot of the screen rendered by the camera object.

G2D stores all image data along with their corresponding 6 DoF pose within `6dpose_list.txt` as the following format:

```
<image file name 1> <camera position XYZ> <camera rotation XYZ>
<image file name 2> <camera position XYZ> <camera rotation XYZ>
                                . . .
<image file name N> <camera position XYZ> <camera rotation XYZ>
```

All the images and their 6 DoF pose are automatically and fully generated from the predetermined dense trajectory as the explanation in Sec. 3.6.2.2. Therefore, before carrying out the function of retracing the dense trajectory, users could change the environmental conditions (i.e. weathers, time and traffic density) as their preference to attain their desired dataset.

3.6.4 Changing the environmental conditions

G2D provides the functions that could support users to change environmental conditions. There are three different settings regarding environmental conditions:

- Regarding the weather, G2D allows user to select between clear, rain or snow.
- In terms of the time, G2D enables users change time of day from 0:00 to 23:00.
- With regard to the traffic density, G2D assists users to increase or decrease two types of traffic density, i.e. vehicle and pedestrian. The density value varying from 0 to 1 represents from none to normal numbers of pedestrians/vehicles on the road.

3.6.5 Unit Conversion

In the testing of some algorithms, it may be useful to conduct metric conversion of the distances in the 2D map of the game. To this end, we perform the following trick:

- Make the game protagonist walk in the environment, and record the positions (in the 3D game coordinate) of the protagonist at every 2-3 steps.
- Calculate the average walking-stride length of the protagonist in unit-distances of the 3D game coordinate (roughly 0.9 units based on the 3D game coordinate that we used).

According to [70], the average walking stride length for a male adult is about 0.762 meters, hence 1 unit-distance in the 3D game coordinate is equal to about 0.85 meters.

3.7 Synthetic data

This section describes our synthetic dataset collected from the computer game GTV A using our software G2D (see Sec. 3.6). Note that there is an increasing recognition of the value of synthetic data towards building autonomous driving systems [168]. Our dataset can be found in the project page: https://sites.google.com/view/g2d-software/home.

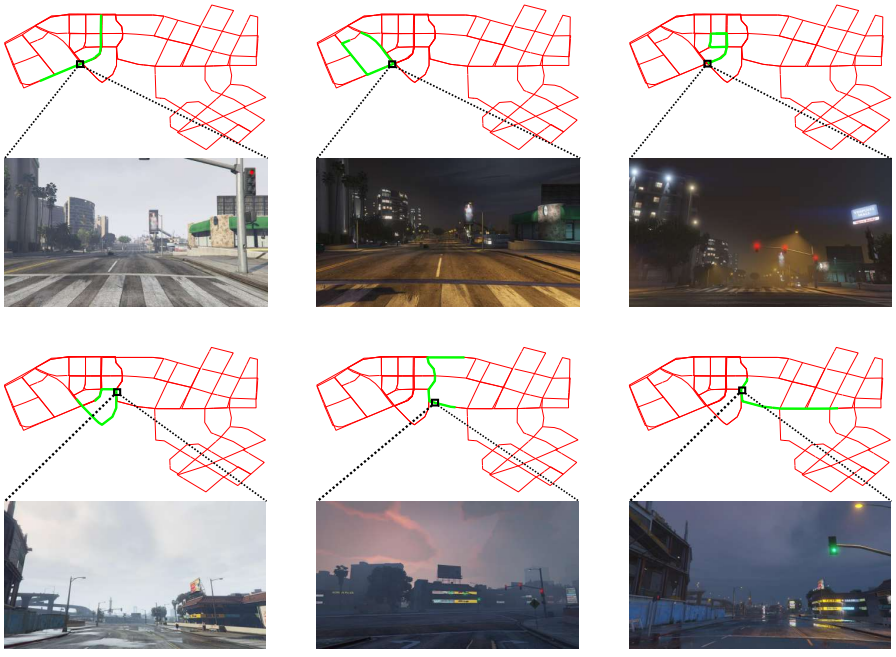


Fig. 3.7. Our synthetic dataset simulates image sequences collected from multiple vehicles under different environmental conditions. The red lines indicate the road network of a part of a city, while the green lines indicate a few of the trajectories taken by different data collection vehicles. Sample frames at similar locations are shown—note the differences in *pose* and environmental conditions between the images. Testing data will also be an image sequence recorded from a vehicle along the road network.

We simulate that there are 59 vehicles which run independently in different routes and environmental conditions. Fig. 3.7 shows the setting of our synthetic dataset. The

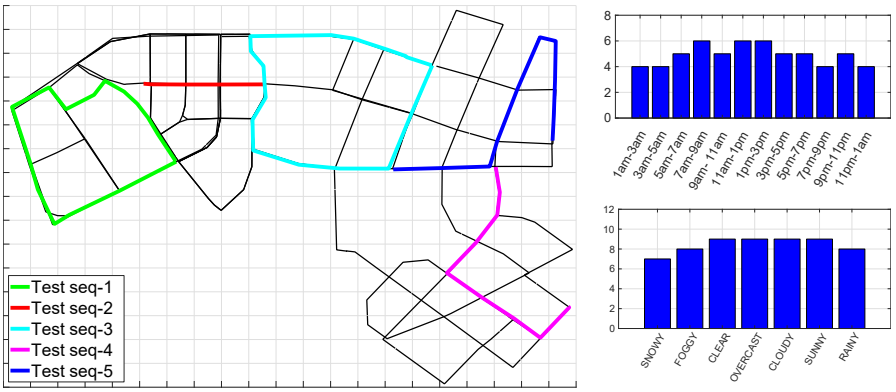


Fig. 3.8. Bird's eye view and distribution of environmental conditions of our synthetic dataset. We simulate that there are 59 vehicles running in different routes, distinct time and weathers. Grid lines are every 100m. In training set, the coverage area is about 3.36km^2, the time and weather conditions are uniformly distributed. The statistics of testing sequences is shown in Table 3.1

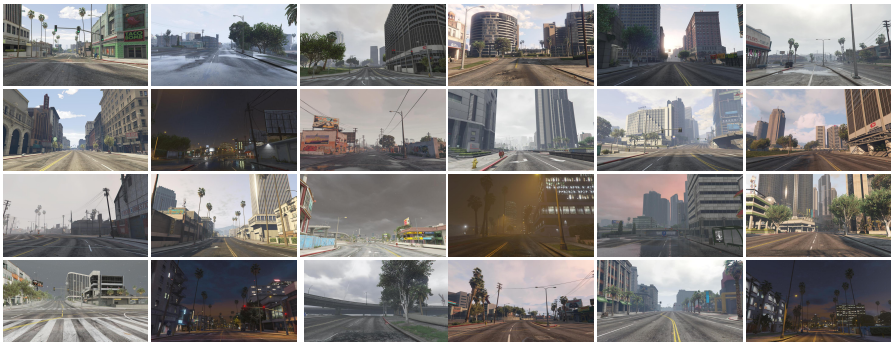


Fig. 3.9. Training (1st-3rd rows) and testing samples (4th row) in our synthetic dataset.

bird's-eye view of our synthetic dataset is illustrated in Fig. 3.8, the coverage area is about 3.36km^2. Also, those vehicles are simulated to run in different times of day and weather conditions. The distributions of times and weather conditions are shown in two histograms of Fig. 3.8. The times and weathers in training sequences are uniformly distributed from 1am to 11pm, and in 7 different weather conditions (snowy, foggy, clear, overcast, cloudy, sunny and rainy). Five sequences in different times and weathers are also collected for the testing phase. The statistics information and trajectory of the testing sequences are described in Table 3.1 and Fig. 3.8 respectively. We sub-sampled on the training sequences at 2Hz and the testing sequences at 4Hz. Examples from training and testing sequences are shown in Fig. 3.9.

3.8 Experiments

To validate the performance of our proposed VL methods, we conduct experiments on synthetic as well as real datasets. For quantitative results, the translation error is

Sequences	# images	Time & Weather	Traversal distance
Test seq-1	1451	9:36 am, snowy	1393.03m
Test seq-2	360	10:41 pm, clear	359.74m
Test seq-3	1564	11:11 am, rainy	1566.93m
Test seq-4	805	6:26 pm, cloudy	806.56m
Test seq-5	1013	3:05 pm, overcast	1014.91m

TABLE 3.1: Statistics of the testing sequences in the synthetic dataset

calculated as the Euclidean distance $||c_{est} - c_{gt}||_2$. The orientation error $|\alpha|$ is computed as the angular difference $2\,cos(|\alpha|) = trace(R_{gt}^{-1}\,R_{est})$ between estimated and ground truth camera rotation matrices R_{est} and R_{gt}.

A query frame is deemed as *correctly localized* if its predicted pose is within a threshold of (Xm, Y°) from its ground truth pose. We define following thresholds: $(1m, 5^\circ)$, $(5m, 10^\circ)$, $(10m, 20^\circ)$, $(15m, 30^\circ)$,$(20m, 40^\circ)$, and $(50m, 100^\circ)$

3.8.1 Implementation details

In the observation encoder, we extract SIFT feature at 4 different scales with region width of 16, 24, 32 and 40, over a grid with spacing of 2 pixels. The SIFT features are embedded and aggregated using VLAD to obtain a single vector of length $16,384$, which is then projected to a $4,096$ dimensional space via PCA, whitened and L2-normalized. To train the visual vocabulary, we randomly sample $5,000,000$ SIFT features in sequences "28/11/2014, 12:07:13" and "02/12/2014, 15:30:08" of Oxford RobotCar [111], and train a visual vocabulary size of $K = 128$.

In the visual localization with HMM, we set $\mathbf{v}_{\max} = 5$ and $\sigma = 0.06$. The number of place hypotheses are $|\mathcal{H}_t| = 20$. the initial belief $\mathbf{p}_0$ is uniformly initialized. This approach is termed as HMM in following sections.

In MCVL, in case prior information of initial camera pose is unknown, ones typically initialize particles according to uniform distribution. However, since a rough camera pose can be inferred from image retrieval, it is reasonable to assume that high (and small) probability for initial camera pose closer (and further) to that rough camera pose. Therefore, particles are initialized from Gaussian distribution with the mean from the noisy measurement in the first frame. The covariance matrices for initializing 3D location $r_{t=1}^{[i]}$ and orientation $\Omega_{t=1}^{[i]}$ respectively are $diag([10, 10, 10]^T)$ and $diag([0.001, 0.001, 1]^T)$. The parameters of motion model are set as the following: $\Sigma_o = diag([5, 5, 5, 0.0001, 0.0001, 0.001]^T)$, $\Sigma_v = diag([1, 1, 0.01]^T)$, $\Sigma_\psi = diag([0.0001; 0.00001; 0.01]^T)$, $\mu_v = [0.1, 0.1, 0.01]^T$, $\mu_\psi = [0.001, 0.00001, 0.01]^T$, where $diag$ is a function that

Route	Purpose	Recorded
Alternate route (1 km)	Training	26/6/2014, 9:24:58
	Training	26/6/2014, 8:53:56
	Unlabeled	14/5/2014, 13:50:20
	Unlabeled	14/5/2014, 13:46:12
	Query	23/6/2014, 15:41:25
Full route (10 km)	Training	28/11/2014, 12:07:13
	Training	02/12/2014, 15:30:08
	Unlabeled	12/12/2014, 10:45:15
	Query	09/12/2014, 13:21:02

TABLE 3.2: The split of training and testing sequences in Oxford RobotCar dataset.

outputs a square diagonal matrix with the elements of input vector on the main diagonal. The number of particles are fixed to 1000. The numbers of retrieved images $R = 20$.

We select CNN-based 6-DoF camera pose regression approaches as our competitors: PoseNet [87], MapNet [17] and MapNet+PGO [17]. Following suggestion of [17], PoseNet uses ResNet-34 [72], adds a global average pooling layer, and parameterizes camera orientation as logarithm of a unit quaternion. MapNet+PGO employs pose graph optimization (PGO) to fuse the absolute poses (produced by MapNet) and relative odometry (from visual odometry) during the inference. PoseNet is implemented in Tensorflow, MapNet and MapNet+PGO's implementations are provided by the authors. Parameters of PoseNet and MapNet are set as the suggestion of the authors.

3.8.2 Datasets

We conduct the experiment using our synthetic dataset (see Sec. 3.7), and Oxford RobotCar [111]. For Oxford RobotCar, we follow the configuration suggested by MapNet [17]. The split of training and testing sequences are summarized in Table 3.2. The training and testing sequences are recorded in different dates, which ensure a significant difference in their appearances. The experiment is conducted on the alternate and full routes with the length of 1 km and 10 km respectively. MapNet [17] is initially trained with the training set, and then fine-tuned in an unlabeled dataset, PoseNet [87] is trained with the training set, and our methods only use training set as the database $\mathcal{D}$.

3.8.3 Comparison between VLAD and Bag of Word (BoW)

Fig. 3.10a compares the localization errors between VLAD and BoW. For BoW, the visual vocabulary of size $4,096$ is trained using the $5,000,000$ SIFT features as described in Sec. 3.8.1. For each image, we also extract SIFT features in the same setting as

Sec. 3.8.1, the RootSIFT normalization (Sec. 3.3) is subsequently performed. BoW representation is formed for each image, it is then followed by the $L2$ normalization: $x_i = x_i/||x_i||_2$. Here, since VLAD is a higher order representation than BoW, it significantly outperforms BoW.

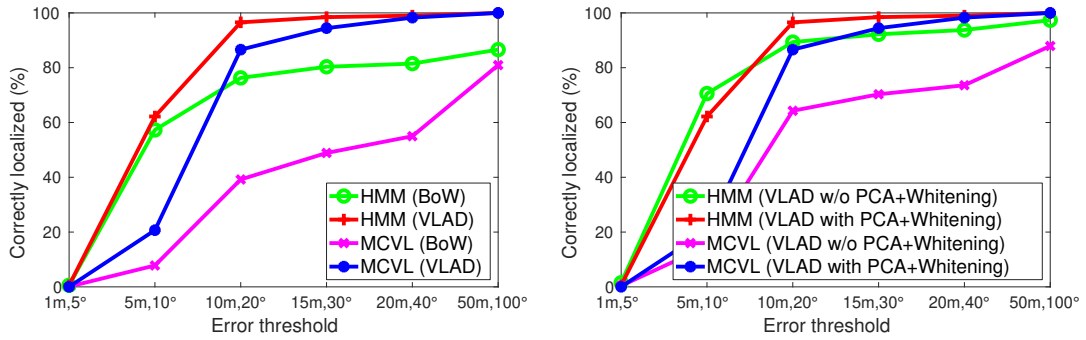


Fig. 3.10. Percentage of correctly localized query frames in Oxford Robotcar. **(a)** Comparison between VLAD and BoW, **(b)** The benefit of post-processing

3.8.4 The benefit of post-processing

Sec. 3.3 claims using PCA and whitening (post-processing) can reduce the impacts of background features. Fig. 3.10b verifies that this post-processing indeed improves the localization accuracy. Note that without PCA projection, the dimensionality of each image representation is $16,384$ which also slows down the inference time.

3.8.5 Sum and democratic aggregation

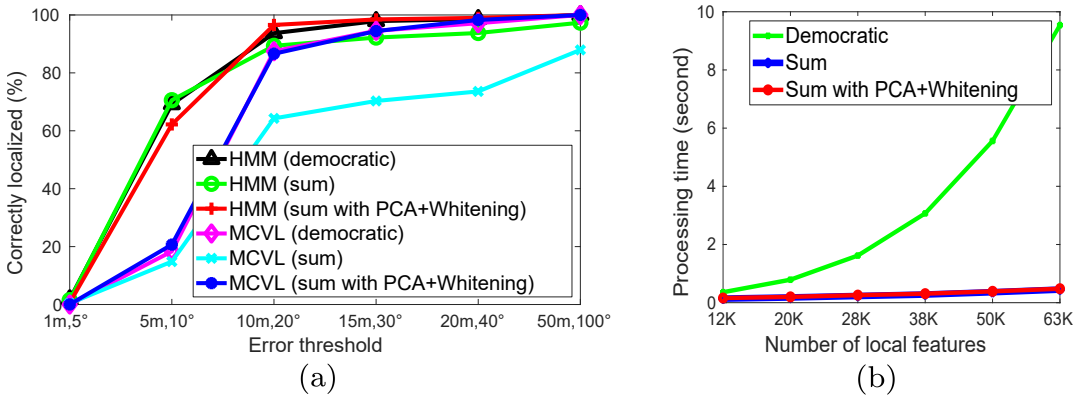


Fig. 3.11. Comparison between sum and democratic aggregations in Oxford RobotCar. **(a)** Localization accuracy, **(b)** Time of encoding the observation with respect to number of local SIFT features.

This section verifies the practical value of using sum aggregation over democratic aggregation. For democratic aggregation, we use the implementation provided by [81]. The

purpose of democratic aggregation is to equalize the contribution of embedded vectors, reducing the impact of background features, since background features (e.g., tree, sky, road, etc) usually dominate the image. Therefore, from Fig. 3.11a, democratic aggregation outperforms sum aggregation. However, thanks to post-processing (PCA and whitening), it improves sum aggregation to be comparable to democratic aggregation.

In addition, Fig. 3.11b shows that the processing time of democratic aggregation significantly becomes slower when the number of local SIFT features increase, while the processing time of sum aggregation combined with post processing remains also constant. This is because when democratic aggregation is performed, it needs to solve an optimization problem. By contrast, the post processing is only the matrix multiplication which can be processed efficiently in parallel. Hence, due to the computational complexity of democratic aggregation, its practicality is limited.

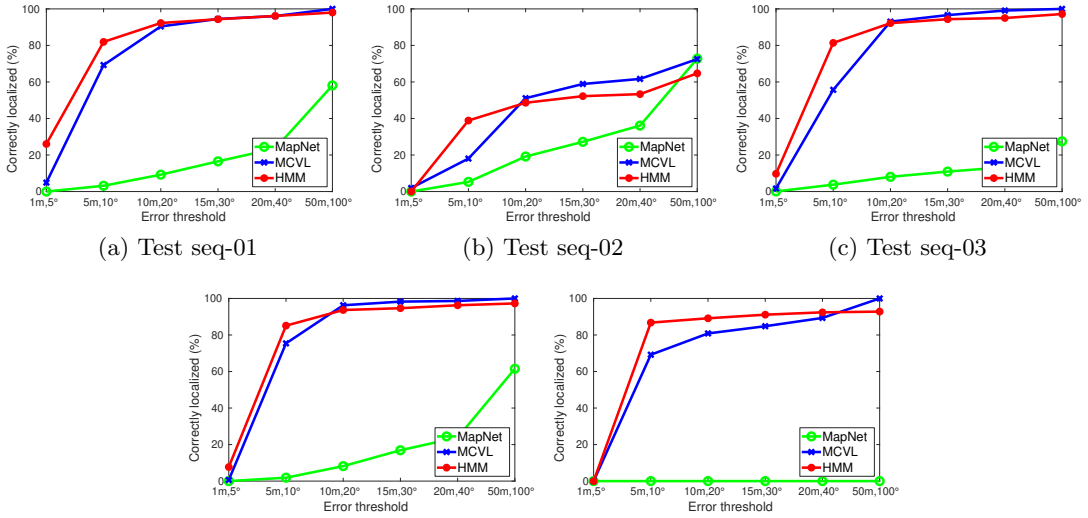
3.8.6 Comparison to competitors

(a) MapNet (b) HMM (c) MCL

Fig. 3.12. Results on our synthetic dataset. Green lines represented ground truth and predicted trajectories are given in red.

3.8.6.1 Synthetic dataset

As can be seen from Fig. 3.12, MapNet struggles to produce a reasonable result. This is because MapNet formulates the problem as an image to pose regression, whose underlying assumption of constant appearance is violated when the appearance of the environment changes. Moreover, repeated structures such as trees, sky, and road surfaces can leads to ambiguities for MapNet. In contrast, our observation encoder applies state-of-the-art normalization techniques to reduce the negative impact from repetitive objects. Hence as shown in Fig. 3.13, HMM and MCVL significantly outperform MapNet. Although HMM is slightly better than MCVL regarding the percentage of correctly localized queries, MCVL produces a more smooth trajectory than HMM (see Fig. 3.12).

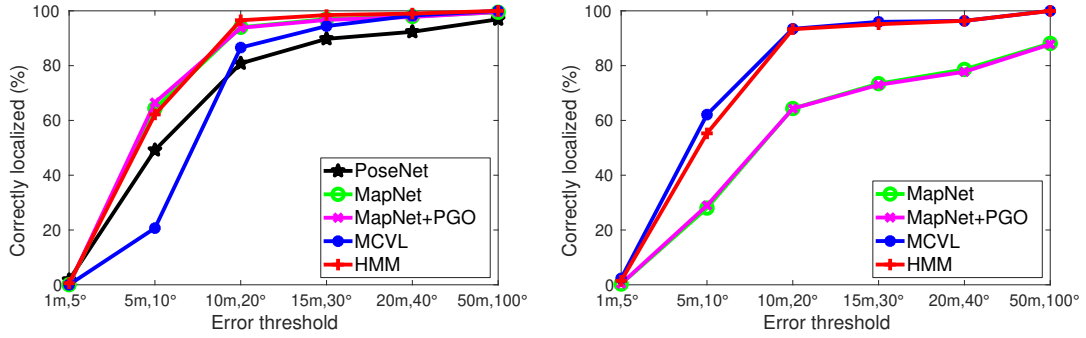


(a) Test seq-01 (b) Test seq-02 (c) Test seq-03

(d) Test seq-04 (e) Test seq-05

Fig. 3.13. Percentage of correctly localized query frames in our synthetic dataset.

3.8.6.2 Oxford RobotCar

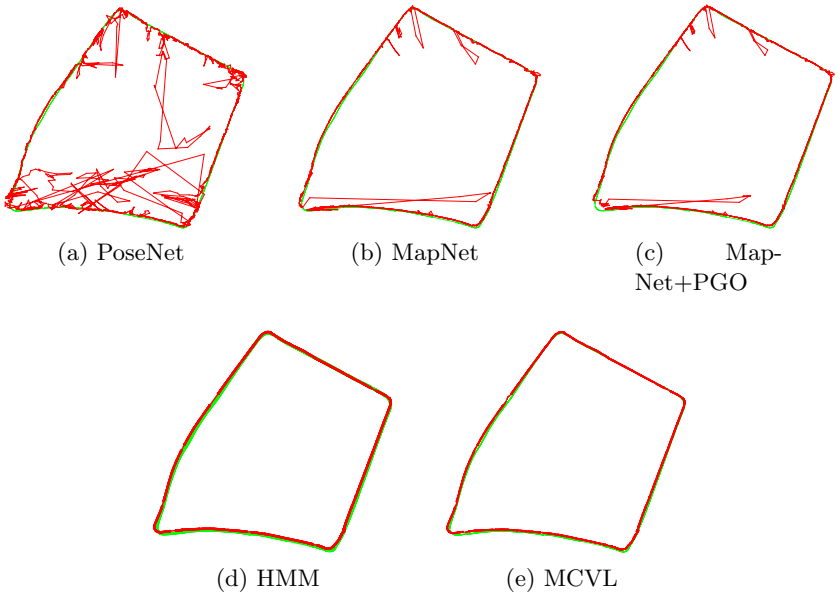


(a) Alternate route (1km) (b) Full route (10km)

Fig. 3.14. Percentage of correctly localized query frames in Oxford Robotcar.

We compare our proposed method to state-of-the-art approaches, i.e., PoseNet [87], MapNet and MapNet+PGO [17]. In particular, PoseNet directly regresses 6 DoF camera pose from an input image. MapNet receives videos as training data, hence its loss function minimizes absolute pose per image as well as the relative pose between consecutive images, it is then followed by a fine-tuning step on unlabeled data with their visual odometry (VO). MapNet+PGO, in the inference step, fuses the prediction of MapNet with VO by using pose graph optimization (PGO) to ensure the temporal smoothness.

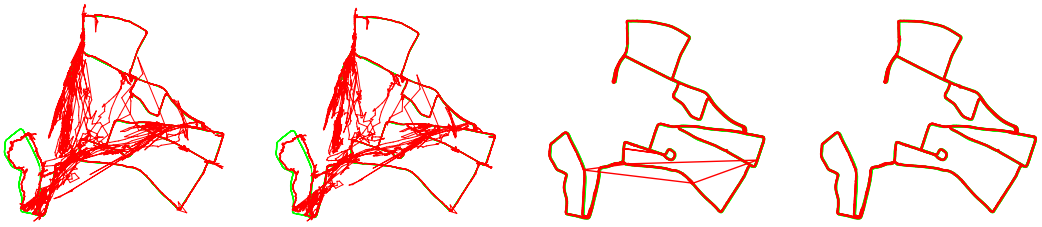
Results on alternate route (1km) are shown in Fig. 3.14a. MapNet, MapNet+PGO and MCVL share a comparable performance, which outperforms PoseNet and HMM. However, Fig. 3.15 shows that MapNet, MapNet+PGO and PoseNet are unable to output a smooth trajectory, while our methods (i.e., MCVL and HMM) can produce



(a) PoseNet (b) MapNet (c) MapNet+PGO

(d) HMM (e) MCVL

Fig. 3.15. Results on alternate route (1km) in Oxford RobotCar dataset. The green lines are ground truth, and red lines are predicted trajectories.



(a) MapNet (b) MapNet +PGO (c) HMM (d) MCVL

Fig. 3.16. Results on full route (10km) in Oxford RobotCar dataset. The green lines are ground truth, and red lines are predicted trajectories.

a smooth prediction. One possible reason which PGO is unable to produce a smooth prediction is there are some existing outliers from the predictions of both MapNet and VO which will probably make PGO stuck in a local minimum.

In the large-scale setting, which is full route (10km), PoseNet is not reported since it can not give a reasonable result. In general, we observe the results on full route are consistent with those on alternate route, i.e., HMM and MCVL output more smooth predictions, compared to MapNet and MapNet+PGO. Therefore, HMM and MCVL significantly outperform MapNet and MapNet+PGO in terms of percentage of correctly localized queries (see Fig. 3.14b).

3.9 Conclusion

In this chapter, we provide an empirical answer for the **Research question 1** *How to effectively model temporal information?* To convincingly deal with this question, we highlight our contributions that address sub-questions posed in Sec. 3.1.

RQ1.1 *How should we develop probabilistic-based algorithms?*

We employ a probabilistic framework—Bayes filter [164, Chapter 2], which has a step of motion prediction to effectively exploit the temporal information. Typically, the motion prediction relies on control data, but we show that even a random motion model can offer a good localization accuracy.

We propose two filtering methods: visual localization with HMM and Monte Carlo-based visual localization, that respectively exploit the discrete and continuous domains for VPR. Their observation models rely on a novel observation encoder that employs local features and encoding technique to represent a noisy measurement (i.e., image) as a fixed (low) dimensional vector.

The experiment shows that filtering on continuous domain produces a slightly better prediction than on discrete domain. In addition, our filtering methods significantly outperform deep learning methods in a large-scale dataset.

RQ1.2 *How to cost-effectively evaluate VPR methods?*

We present G2D that could be utilized to collect the dataset in Grand Theft Auto V. By virtue of the capability of accessing to the native functions, G2D allows users to control the various environmental conditions within the game, i.e. weather, time and traffic density. In addition, G2D samples a set of images with their corresponding camera poses in the game coordinate. Apart from it, G2D is open-sourced, hence users could modify it as their preference to collect the desired dataset. Using G2D, we create a synthetic dataset for VPR, that simulate multiple vehicles running different routes, time of day, and weathers to build a map.

Chapter 4

Scalable Visual Place Recognition

This chapter is based on the content of following conference paper

- Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, Yu Liu, Thanh-Toan Do, and Ian Reid. Scalable Place Recognition Under Appearance Change for Autonomous Driving. In *International Conference in Computer Vision (ICCV)*, 2019.

Statement of Authorship

Title of Paper	Scalable Place Recognition Under Appearance Change for Autonomous Driving
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, Yu Liu, Thanh-Toan Do, Ian Reid, *"Scalable Place Recognition Under Appearance Change for Autonomous Driving"*, International Conference in Computer Vision (ICCV), 2019

Principal Author

Name of Principal Author (Candidate)	Anh-Dzung Doan		
Contribution to the Paper	Developed the method, conducted the experiments, and wrote the paper		
Overall percentage (%)	60		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	19 February 2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Yasir Latif		
Contribution to the Paper	Supervised the development of the method, provided the suggestions, and revised the paper.		
Signature		Date	23/02/2021

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Proposed the general direction, supervised the development of the method, provided the suggestions, and revised the paper.		
Signature		Date	21 Feb 2021

Name of Co-Author	Yu Liu		
Contribution to the Paper	Helped with the experiment, and proofread the paper		
Signature		Date	Feb 22, 2021

Name of Co-Author	Thanh-Toan Do		
Contribution to the Paper	Provided the suggestions, and proofread the paper		
Signature		Date	22-Feb-2021

Name of Co-Author	Ian Reid		
Contribution to the Paper	Provided the suggestions, and proofread the paper		
Signature		Date	23/2/2021

4.1 Introduction and research questions

To perform convincingly, a practical VPR algorithm must be robust against appearance changes in the operating environment. These can occur due to higher frequency environmental variability such as weather, time of day, and pedestrian density, as well as longer term changes such as seasons and vegetation growth. A realistic VPR system must also contend with "less cyclical" changes, such as construction and roadworks, updating of signage, façades and billboards, as well as abrupt changes to traffic rules that affect traffic flow (this can have a huge impact on VPR if the database contains images seen from only one particular flow [32, 54]). Such appearance changes invariably occur in real life.

To meet the challenges posed by appearance variations, one paradigm is to develop VPR algorithms that are inherently robust against the changes. Methods under this paradigm attempt to extract the "visual essence" of a place that is independent of appearance changes [7]. However, such methods have mostly been demonstrated on more "natural" variations such as time of day and seasons.

Another paradigm is to equip the VPR algorithm with a large image dataset that was acquired under different environmental conditions [28]. To accommodate long-term evolution in appearance, however, it is vital to continuously accumulate data and update the VPR algorithm. To achieve continuous data collection cost-effectively over a large region, one could opportunistically acquire data using a fleet of service vehicles (e.g., taxis, delivery vehicles) and amateur mappers. Indeed, there are street imagery datasets that grow continuously through crowdsourced videos [128, 68]. Under this approach, it is reasonable to assume that a decent sampling of the appearance variations, including the recent changes, is captured in the ever growing dataset.

Under continuous dataset growth, the key to consistently accurate VPR is to "assimilate" new data quickly. This demands a VPR algorithm that is *scalable*. Specifically, the computational cost of testing (i.e., performing VPR on a query input) should not increase visibly with the increase in dataset size. Equally crucially, updating or retraining the VPR algorithm on new data must also be highly efficient. With this motivation, in this chapter, we ask:

Research question 2 *Can we develop scalable VPR for life-long operation?*

In realistic scenarios, the visual data is usually in the form of videos, and chapter 3 has shown the efficiency of Bayesian techniques in modelling the temporal information. Therefore, in this chapter, we develop a novel VPR technique based on Hidden Markov Models (HMMs) that is scalable in both training and testing such that it is able to

continuously grow due to the incorporation of new sequences in the dataset. Our method will be investigated according to four following sub-questions

RQ2.1 *Can we discover missing knowledge in the map?*

Mapping for large-scale environment typically requires multiple vehicles, and the sub-maps from vehicles are aligned together to obtain a full map. However, the alignment stage, no matter how accurate, will always produce a topological map with missing connections between nodes. Sec. 4.4 shows that our VPR inference can be used to "connect" nodes from different videos provided that they are representing a same physical place.

RQ2.2 *How to effectively absorb new data?*

Our method includes a topologically sensitive compression procedure that can update the system efficiently, *without* using GNSS positioning information or computing visual odometry. Also, the observation model rely on a compact image representation and hierarchical tree structure, that can be updated quickly (Sec. 4.5).

Arguably, VPR algorithms based on deep learning [17, 29] can accommodate new data by simply appending it to the dataset and fine-tuning the network parameters. However, as we will show later, this fine-tuning process is still too costly to be practical, and the lack of accurate labels in the testing sequence can be a major obstacle.

The experimental results show the scalability of our method in updating new data, compared to deep learning algorithms (Sec. 4.6.3).

RQ2.3 *Can we eat a lot but never get fat?*

The experiment demonstrates that our compression technique can improve accuracy by continuous adaption to new data, while maintaining the computational efficiency (Sec. 4.6.1).

The rest of the chapter is organized as follows: We respectively provide the problem setting and map representation in Sec. 4.2 and Sec. 4.3. Next, we briefly introduce how to perform VPR using HMM in Sec. 4.4. We provide the detail of our compression technique when updating new data in Sec. 4.5. Finally, the experimental results are presented in Sec. 4.6, and the conclusion of this chapter is drawn in Sec. 4.7.

4.2 Problem setting

We first describe our adopted setting for VPR for autonomous driving. Let $\mathcal{D} = \left\{\mathcal{V}_1, \ldots, \mathcal{V}_M\right\}$ be a dataset of M videos, where each video

$$\mathcal{V}_i = \{I_{i,1}, I_{i,2}, \ldots, I_{i,N_i}\} = \{I_{i,j}\}_{j=1}^{N_i} \tag{4.1}$$

is a time-ordered sequence of N_i images. In the proposed VPR system, $\mathcal{D}$ is collected in a distributed manner using a fleet of vehicles instrumented with cameras. Since the vehicles could be from amateur mappers, accurately calibrated/synchronized GNSS positioning may not be available. However, we do assume that the camera on all the vehicles face a similar direction, e.g., front facing. The query video is represented as

$$\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_T\} \tag{4.2}$$

which is a temporally-ordered sequence of T query images. The query video could be a new recording from one of the contributing vehicles (recall that our database $\mathcal{D}$ is continuously expanded), or it could be the input from a "user" of the VPR system, e.g., an autonomous vehicle.

4.2.1 Overall aims

For each $Q_t \in \mathcal{Q}$, the goal of VPR is to retrieve an image from $\mathcal{D}$ that was taken from a similar location to Q_t, i.e., the FOV of the retrieved image overlaps to a large degree with Q_t. As mentioned above, what makes this challenging is the possible variations in image appearance.

In the envisioned VPR system, when we have finished processing $\mathcal{Q}$, it is appended to the dataset

$$\mathcal{D} = \mathcal{D} \cup \{\mathcal{Q}\}, \tag{4.3}$$

thus the image database could grow unboundedly. This imposes great pressure on the VPR algorithm to efficiently "internalise" new data and compress the dataset. As an indication of size, a video can have up to 35,000 images.

4.3 Map representation

When navigating on a road network, the motion of the vehicle is restricted to the roads, and the heading of the vehicle is also constrained by the traffic direction. Hence, the variation in pose of the camera is relatively low [145, 141].

The above motivates us to represent a road network as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, which we also call the "map". The set of nodes $\mathcal{N}$ is simply the set of all images in $\mathcal{D}$. To reduce clutter, we "unroll" the image indices in $\mathcal{D}$ by converting an (i, j) index to a single number $k = N_1 + N_2 + \cdots + N_{i-1} + j$, hence the set of nodes are

$$\mathcal{N} = \{1, \ldots, K\}, \tag{4.4}$$

where $K = \sum_{i=1}^{M} N_i$ is the total number of images. We call an index $k \in \mathcal{N}$ a "place" on the map.

We also maintain a corpus $\mathcal{C}$ that stores the images observed at each place. For now, the corpus simply contains

$$\mathcal{C}(k) = \{I_k\}, \quad k = 1, \ldots, K, \tag{4.5}$$

at each cell $\mathcal{C}(k)$. Later in Sec. 4.5, we will incrementally append images to $\mathcal{C}$ as the video datatset $\mathcal{D}$ grows.

In $\mathcal{G}$, the set of edges $\mathcal{E}$ connect images that overlap in their FOVs, i.e., $\langle k_1, k_2 \rangle$ is an edge in $\mathcal{E}$ if

$$\exists I \in \mathcal{C}(k_1) \text{ and } \exists I' \in \mathcal{C}(k_2) \text{ such that } I, I' \text{ overlap.} \tag{4.6}$$

Note that two images can overlap even if they derive from different videos and/or conditions. The edges are weighted by probabilities of transitioning between places, i.e.,

$$w(\langle k_1, k_2 \rangle) = P(k_2 \mid k_1) = P(k_1 \mid k_2), \tag{4.7}$$

for a vehicle that traverses the road network. Trivially,

$$\langle k_1, k_2 \rangle \notin \mathcal{E} \text{ iff } P(k_2 \mid k_1) = P(k_1 \mid k_2) = 0. \tag{4.8}$$

It is also clear from (4.7) that $\mathcal{G}$ is undirected. Concrete definition of the transition probability will be given in Sec. 4.5. First, Sec. 4.4 discusses VPR of $\mathcal{Q}$ given a fixed $\mathcal{D}$ and map.

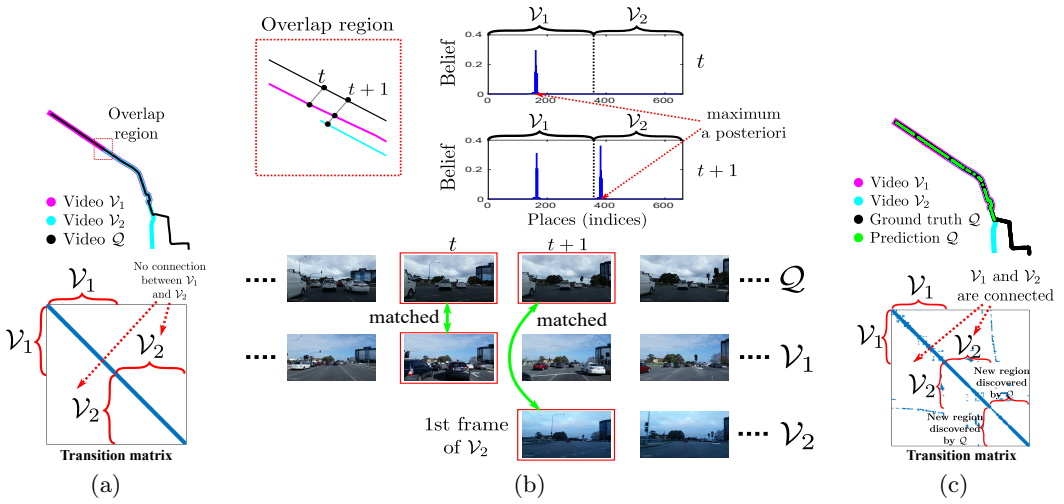


Fig. 4.1. An overview of our idea using HMM for place recognition. Consider dataset $\mathcal{D} = \{\mathcal{V}_1, \mathcal{V}_2\}$ and query $\mathcal{Q}$. Figure 4.1a: Because $\mathcal{V}_1$ and $\mathcal{V}_2$ are recorded in different environmental conditions, $\mathcal{V}_2$ cannot be matched against $\mathcal{V}_1$, thus there is no connection between $\mathcal{V}_1$ and $\mathcal{V}_2$. Query $\mathcal{Q}$ visits the place covered by $\mathcal{V}_1$ and $\mathcal{V}_2$, and then an unknown place. Figure 4.1b: Query $\mathcal{Q}$ is firstly localized against only $\mathcal{V}_1$. When it comes to the "Overlap region" at time $t+1$, it localizes against both $\mathcal{V}_1$ and $\mathcal{V}_2$. The image corresponding to MaxAP at every time step t is returned as the matching result. Figure 4.1c: A threshold decides if the matching result should be accepted, thus when Q visits an unseen place, the MaxAPs of $\mathcal{V}_1$ and $\mathcal{V}_2$ are small, we are uncertain about the matching result. Once $\mathcal{Q}$ is finished, the new place discovered by $\mathcal{Q}$ is added to the map to expand the coverage area. In addition, since Q is matched against both $\mathcal{V}_1$ and $\mathcal{V}_2$, we can connect $\mathcal{V}_1$ and $\mathcal{V}_2$.

4.4 Place recognition using HMM

To perform VPR on $\mathcal{Q} = \{Q_1, \ldots, Q_T\}$ against a fixed map $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and corpus $\mathcal{C}$, we model $\mathcal{Q}$ using a HMM [143]. We regard each image Q_t to be a noisy observation (image) of an latent *place state* s_t, where $s_t \in \mathcal{N}$. The main reason for using HMM for VPR is to exploit the temporal order of the images in $\mathcal{Q}$, and the high correlation between time and place due to the restricted motion (Sec. 4.3).

To assign a value to s_t, we estimate the belief

$$P(s_t \mid Q_{1:t}), \quad s_t \in \mathcal{N}, \tag{4.9}$$

where $Q_{1:t}$ is a shorthand for $\{Q_1, \ldots, Q_t\}$. Note that the belief is a probability mass function, hence

$$\sum_{s_t \in \mathcal{N}} P(s_t \mid Q_{1:t}) = 1. \tag{4.10}$$

Based on the structure of the HMM, the belief (4.9) can be recursively defined using Bayes' rule as

$$P(s_t|Q_{1:t}) = \eta P(Q_t|s_t) * \sum_{s_{t-1} \in \mathcal{N}} P(s_t|s_{t-1}) P(s_{t-1}|Q_{1:t-1}), \tag{4.11}$$

where $P(Q_t|s_t)$ is the observation model, $P(s_t|s_{t-1})$ is the state transition model, and $P(s_{t-1}|Q_{1:t-1})$ is the prior (the belief at the previous time step) [143]. The scalar η is a normalizing constant to ensure that the belief sums to 1.

If we have the belief $P(s_t \mid Q_{1:t})$ at time step t, we can perform VPR on Q_t by assigning

$$s_t^* = \arg\max_{s_t \in \mathcal{N}} \; P(s_t \mid Q_{1:t}) \tag{4.12}$$

as the place estimate of Q_t. Deciding the target state in this manner is called maximum *a posteriori* (MaxAP) estimation. See Fig. 4.1 for an illustration of VPR using HMM.

4.4.1 State transition model

The state transition model $P(s_t|s_{t-1})$ gives the probability of moving to place s_t, given that the vehicle was at place s_{t-1} in the previous time step. The transition probability is simply given by the edge weights in $\mathcal{G}$, i.e.,

$$P(s_t = k_2|s_{t-1} = k_1) = w(\langle k_1, k_2 \rangle). \tag{4.13}$$

Again, we defer the concrete definition of the transition probability to Sec. 4.5. For now, the above is sufficient to continue our description of our HMM method.

4.4.2 Observation model

Our observation model is based on image retrieval. Specifically, we use SIFT features [103] and VLAD [79] to represent every image. Priority search k-means tree [122] is used to index the database, but it is possible to use other indexing methods [78, 43, 10].

4.4.2.1 Image representation

For every image $I_k \in \mathcal{C}$, we seek a nonlinear function $\psi(I_k)$ that maps the image to a single high-dimensional vector. To do that, given a set of SIFT features densely extracted

from image I_k: $\mathbf{X}_k = \{x_k^h\} \in \mathbb{R}^{d \times H_k}$, where H_k is the number of SIFT features of image I_k. K-means is used to build a codebook $B = \{b_m \in \mathbb{R}^d \,|\, m = 1, ..., \mathtt{M}\}$, where $\mathtt{M}$ is the size of codebook. The VLAD embedding function is defined as:

$$\phi(x_k) = [..., 0, x_k^h - b_m, 0, ...] \in \mathbb{R}^D \tag{4.14}$$

where, b_m is the nearest visual word of feature vector x_k^h. To obtain a single vector, we employ sum aggregation:

$$\psi(I_k) = \sum_{i=1}^{H_k} \phi(x_k) \tag{4.15}$$

To reduce the impact of background features (e.g., trees, roads, sky) within the vector $\psi(I_k)$, we adopt rotation and normalization (RN) [81], followed by L-2 normalization. In particular, we use PCA to project $\psi(I_k)$ from D to D', where $D' < D$. In our experiment, we set $D' = 4,096$. Power-law normalization is then applied on rotated data:

$$\psi(I_k) := |\psi(I_k)|^\alpha sign(\psi(I_k)) \tag{4.16}$$

where, we set $\alpha = 0.5$.

Note that different from DenseVLAD [167] which uses whitening for post-processing, performing power-law normalization on rotated data is more stable.

4.4.2.2 Computing likelihood

We adopt priority search k-means tree [122] to index every image $I_k \in \mathcal{C}$. The idea is to partition all data points $\psi(I_k)$ into $\mathbf{K}$ clusters by using K-means, then recursively partitioning the points in each cluster. For each query Q_t, we find a set of L-nearest neighbor $\mathbf{L}(Q_t)$. Specifically, Q_t is mapped to vector $\psi(Q_t)$. To search, we propagate down the tree at each cluster by comparing $\psi(Q_t)$ to $\mathbf{K}$ cluster centers and selecting the nearest one.

The likelihood $P(Q_t|s_t)$ is calculated as follows:

- Initialize $P(Q_t|s_t = k) = e^{\frac{-\beta}{\sigma}}$, $\forall k \in \mathcal{N}$, where, we set $\beta = 2.5$ and $\sigma = 0.3$ in our experiment.
- For each $I_k \in \mathbf{L}(Q_t)$
  - Find node $\hat{k} = \mathcal{C}^{-1}(I_k)$, where $\mathcal{C}^{-1}$ is the inverse of corpus $\mathcal{C}$, which finds node $\hat{k}$ storing I_k.

- Calculate the probability: $\hat{p} = e^{\frac{-dist(Q_t, I_{\hat{k}})}{\sigma}}$, where *dist* is the distance between Q_t and I_k.
- If $\hat{p} > P(Q_t|s_t = \hat{k})$, then $P(Q_t|s_t = \hat{k}) = \hat{p}$.

4.4.3 Inference using matrix computations

4.4.3.1 Inference

The state transition model can be stored in a $K \times K$ matrix $\mathbf{E}$ called the *transition matrix*, where the element at the k_1-th row and k_2-th column of $\mathbf{E}$ is

$$\mathbf{E}(k_1, k_2) = P(s_t = k_2 \mid s_{t-1} = k_1). \tag{4.17}$$

Hence, $\mathbf{E}$ is also the weighted adjacency matrix of graph $\mathcal{G}$. Also, each row of $\mathbf{E}$ sums to one. The observation model can be encoded in a $K \times K$ diagonal matrix $\mathbf{O}_t$, where

$$\mathbf{O}_t(k, k) = P(Q_t \mid s_t = k). \tag{4.18}$$

If the belief and prior are represented as vectors $\mathbf{p}_t, \mathbf{p}_{t-1} \in \mathbb{R}^K$ respectively, operation (4.11) can be summarized as

$$\mathbf{p}_t = \eta \mathbf{O}_t \mathbf{E}^T \mathbf{p}_{t-1}, \tag{4.19}$$

where $\mathbf{p}_0$ corresponds to uniform distribution. From this, it can be seen that the cost of VPR is $\mathcal{O}(K^2)$.

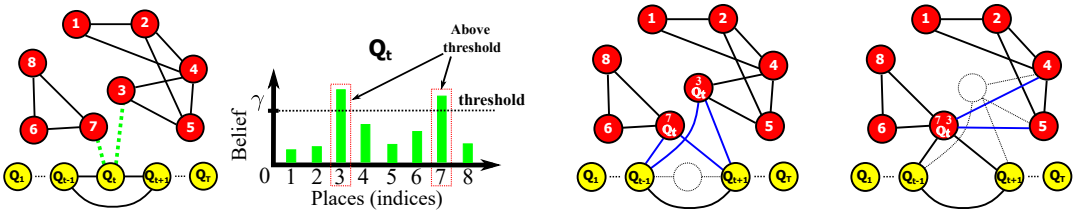
4.4.3.2 Computational cost

Note that $\mathbf{E}$ is a very sparse matrix, due to the topology of the graph $\mathcal{G}$ which mirrors the road network; see Fig. 4.3 for an example $\mathbf{E}$. Thus, if we assume that the max number of non-zero values per row in $\mathbf{E}$ is r, the complexity for computing $\mathbf{p}_t$ is $\mathrm{O}(rK)$.

Nonetheless, in the targeted scenario (Sec. 4.2), $\mathcal{D}$ can grow unboundedly. Thus it is vital to avoid a proportional increase in $\mathbf{E}$ so that the cost of VPR can be maintained.

4.5 Scalable place recognition based on HMM

In this section, we describe a novel method that incrementally builds and compresses $\mathcal{G}$ for a video dataset $\mathcal{D}$ that grows continuously due to the addition of new query videos.



(a) Matching (b) Culling (c) Combining

Fig. 4.2. An overview of our idea for scalable place recognition. Graph $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$, where $\mathcal{G}_1 = \{1, 2, 3, 4, 5\}$ and $\mathcal{G}_2 = \{6, 7, 8\}$ are disjoint sub-graphs. Query video $\mathcal{Q} = \{Q_1, ..., Q_T\}$ is matched against $\mathcal{G}$. Figure 4.2a: Q_t is matched with node $k = 3$ and 7 (dashed green lines), due to $\mathbf{p}_t(3)$, $\mathbf{p}_t(7) > \gamma$. Figure 4.2b: Q_t is added to node 3 and 7, new edges are created (blue lines) to maintain the connections between Q_{t-1}, Q_{t+1} and Q_t. Figure 4.2c: Node 3 and 7 are combined. New edges are generated (blue lines) to maintain the connections within the graph. Note that after matching query $\mathcal{Q}$ against $\mathcal{G}$, our proposed culling and combining methods connect two disjoint sub-graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ together.

We emphasize again that the proposed technique functions without using GNSS positioning or visual odometry.

4.5.1 Map intialization

Given a dataset $\mathcal{D}$ with one video $\mathcal{V}_1 = \{I_{1,j}\}_{j=1}^{N_1} \equiv \{I_k\}_{k=1}^{K}$, we initialize $\mathcal{N}$ and $\mathcal{C}$ as per (4.4) and (4.5). The edges $\mathcal{E}$ (specifically, the edge weights) are initialized as

$$w(\langle k_1, k_2 \rangle) = \begin{cases} 0 & \text{if } |k_1 - k_2| > W, \\ \alpha \exp\left(-\frac{|k_1 - k_2|^2}{\delta^2}\right) & \text{otherwise,} \end{cases} \tag{4.20}$$

where α is a normalization constant. The edges connect frames that are $\leq W$ time steps apart with weights based on a Gaussian on the step distances. The choice of W can be based on the maximum velocity of a vehicle.

Note that this simple way of creating edges will ignore complex trajectories (e.g., loops). However, the subsequent steps will rectify this issue by connecting similar places.

4.5.2 Map update and compression

Let $\mathcal{D} = \{\mathcal{V}_i\}_{i=1}^{M}$ be the current dataset with map $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and corpus $\mathcal{C}$. Given a query video $\mathcal{Q} = \{Q_t\}_{t=1}^{T}$, using our method in Sec. 4.4 we perform VPR on $\mathcal{Q}$ based on $\mathcal{G}$. This produces a belief vector $\mathbf{p}_t$ (4.19) for all t.

We now wish to append $\mathcal{Q}$ to $\mathcal{D}$, and update $\mathcal{G}$ to maintain computational scalability of future VPR queries. First, create a subgraph $\mathcal{G}' = (\mathcal{N}', \mathcal{E}')$ for $\mathcal{Q}$, where

$$\mathcal{N}' = \{K+1, K+2, \ldots, K+T\}, \tag{4.21}$$

(recall that there are a total of K places in $\mathcal{G}$), and $\mathcal{E}'$ simply follows Sec. 4.5.1 for $\mathcal{Q}$.

In preparation for map compression, we first concatenate the graphs and extend the corpus

$$\mathcal{N} = \mathcal{N} \cup \mathcal{N}', \ \mathcal{E} = \mathcal{E} \cup \mathcal{E}', \text{ and } \mathcal{C}(K+t) = \{Q_t\} \tag{4.22}$$

for $t = 1, \ldots, T$. There are two main subsequent steps: culling new places, and combining old places.

4.5.2.1 Culling new places

For each t, construct

$$\mathcal{M}(t) = \{k \in \{1, \ldots, K\} \mid \mathbf{p}_t(k) \geq \gamma\}, \tag{4.23}$$

where γ with $0 \leq \gamma \leq 1$ is a threshold on the belief. There are two possibilities:

- If $\mathcal{M}(t) = \emptyset$, then Q_t is the image of a new (unseen before) place since the VPR did not match a dataset image to Q_t with sufficient confidence. No culling is done.
- If $\mathcal{M}(t) \neq \emptyset$, then for each $k_1 \in \mathcal{M}(t)$,
  - For each k_2 such that $\langle K+t, k_2 \rangle \in \mathcal{E}$:
    - Create new edge $\langle k_1, k_2 \rangle$ with weight $w(\langle k_1, k_2 \rangle) = w(\langle K+t, k_2 \rangle)$.
    - Delete edge $\langle K+t, k_2 \rangle$ from $\mathcal{E}$.
  - $\mathcal{C}(k_1) = \mathcal{C}(k_1) \cup \mathcal{C}(K+t)$.

Once the above is done for all t, for those t where $\mathcal{M}(t) \neq \emptyset$, we delete the node $K+t$ in $\mathcal{N}$ and cell $\mathcal{C}(K+t)$ in $\mathcal{C}$, both with the requisite adjustment in the remaining indices. See Figs. 4.2a and 4.2b for an illustration of culling.

4.5.2.2 Combining old places

Performing VPR on $\mathcal{Q}$ also provides a chance to connect places in $\mathcal{G}$ that were not previously connected. For example, two dataset videos $\mathcal{V}_1$ and $\mathcal{V}_2$ could have traversed a common subpath under very different conditions. If $\mathcal{Q}$ travels through the subpath

under a condition that is simultaneously close to the conditions of $\mathcal{V}_1$ and $\mathcal{V}_2$, this can be exploited for compression.

To this end, for each t where $\mathcal{M}(t)$ is non-empty,

- $k_1 = \min \mathcal{M}(t)$.
- For each $k_2 \in \mathcal{M}(t)$ where $k_2 \neq k_1$ and $\langle k_1, k_2 \rangle \notin \mathcal{E}$:
  - For each k_3 such that $\langle k_2, k_3 \rangle \in \mathcal{E}$, $\langle k_1, k_3 \rangle \notin \mathcal{E}$:
    - Create edge $\langle k_1, k_3 \rangle$ with weight $w(\langle k_1, k_3 \rangle) = w(\langle k_2, k_3 \rangle)$.
    - Delete edge $\langle k_2, k_3 \rangle$ from $\mathcal{E}$.
  - $\mathcal{C}(k_1) = \mathcal{C}(k_1) \cup \mathcal{C}(k_2)$.

Again, once the above is done for all t for which $\mathcal{M}(t) \neq \emptyset$, we remove all unconnected nodes from $\mathcal{G}$ and delete the relevant cells in $\mathcal{C}$, with the corresponding index adjustments. Figs. 4.2c, 4.1a and 4.1c illustrate this combination step.

4.5.3 Updating the observation model

When $\mathcal{Q}$ is appended to the dataset, i.e., $\mathcal{D} = \mathcal{D} \cup \mathcal{Q}$, all vector $\psi(Q_t)$ need to be indexed to the k-means tree. In particular, we find the nearest leaf node that $\psi(Q_t)$ belongs to. Assume the tree is balanced, the height of tree is $(log\, N / log\, \mathbf{K})$, where $N = \sum N_i$, thus each $\psi(Q_t)$ needs to check $(log\, N / log\, \mathbf{K})$ internal nodes and one leaf node. In each node, it needs to find the closest cluster center by computing distances to all centers, the complexity of which is $O(\mathbf{K}D')$. Therefore, the cost for adding the query video $\mathcal{Q}$ is $O\left(T\mathbf{K}D'(log\, N / log\, \mathbf{K} + 1)\right)$, where $T = |\mathcal{Q}|$. Assume it is a complete tree, every leaf node contains $\mathbf{K}$ points, thus it has $N/\mathbf{K}$ leaf nodes. For each point $\psi(Q_t)$, instead of exhaustedly scanning $N/\mathbf{K}$ leaf nodes, it only needs to check $log\, N / log\, \mathbf{K} + 1$ nodes. Hence, it is a scalable operation.

4.5.4 Overall algorithm

Algorithm 4.1 summarizes the proposed scalable method for VPR. A crucial benefit of performing VPR with our method is that map $\mathcal{G}$ does not grow unboundedly with the inclusion of new videos. Moreover, the map update technique is simple and efficient, which permits it to be conducted for every new video addition. This enables scalable VPR on an ever growing video dataset. In Sec. 4.6.3, we will compare our technique with state-of-the-art VPR methods.

4.6 Experiments

We use a dataset sourced from Mapillary [128] which consists of street-level geo-tagged imagery; see supplementary material for examples. Benchmarking was carried out on the Oxford RobotCar [111], from which we use 8 different sequences along the same route; details are provided in supplementary material, and the sequences are abbreviated as Seq-1 to Seq-8. The initial database $\mathcal{D}$ is populated with Seq-1 and Seq-2 from the Oxford RobotCar dataset. Seq-3 to Seq-8 are then sequentially used as the query videos. To report the 6-DoF pose for a query image, we inherit the pose of the image matched using the MaxAP estimation. Following [145], the translation error is computed as the Euclidean distance $||c_{est} - c_{gt}||^2$. Orientation errors $|\theta|$, measured in degree, is the angular difference $2\,cos(|\theta|) = trace(R_{gt}^{-1}\,R_{est}) - 1$ between estimated and ground truth camera rotation matrices R_{est} and R_{gt}. Following [87, 85, 17, 175], we compare mean and median errors.

4.6.1 Performance with and without updating the database

Algorithm 4.1 Scalable algorithm for large-scale VPR.

Require: Threshold W for transition probability, threshold γ for VPR, initial dataset $\mathcal{D} = \{\mathcal{V}_1\}$ with one video.

1: Initialize map $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and corpus $\mathcal{C}$ (Sec. 4.5.1).
2: Create observation model (Sec. 4.4.2)
3: **while** there is a new query video $\mathcal{Q}$ **do**
4: Perform VPR on $\mathcal{Q}$ using map $\mathcal{G}$, then append $\mathcal{Q}$ to $\mathcal{D}$.
5: Create subgraph $\mathcal{G}'$ for $\mathcal{Q}$ (Sec. 4.5.2).
6: Concatenate $\mathcal{G}'$ to $\mathcal{G}$, extend $\mathcal{C}$ with $\mathcal{Q}$ (Sec. 4.5.2).
7: Reduce $\mathcal{G}$ by culling new places (Sec. 4.5.2).
8: Reduce $\mathcal{G}$ by combining old places (Sec. 4.5.2).
9: Update observation model (Sec. 4.5.3).
10: **end while**
11: **return** Dataset $\mathcal{D}$ with map $\mathcal{G}$ and corpus $\mathcal{C}$.

We investigate the effects of updating database on localization accuracy and inference time. After each query sequence finishes, we consider three strategies: i) `No update`: $\mathcal{D}$ always contains just the initial 2 sequences, ii) `Cull`: Update $\mathcal{D}$ with the query and perform culling, and iii) `Cull+Combine`: Full update with both culling and combining nodes. Mean and median 6-DoF pose errors are reported in Table 4.1. In general, `Cull` improves the localization accuracy over `No update`, since culling adds appearance variations to the map. In fact, there are several cases, in which `Cull+Combine` produces better results over `Cull`. This is because we consolidate useful information in the map (combining nodes which represent the same place), and also enrich the map topology (connecting nodes close to each other through culling). Inference times per query with

	No update	Cull	Cull+combine
Seq-3	6.59m, 3.28°		
Seq-4	7.42m, 4.64°	**5.80m**, 3.24°	6.01m, **3.11**°
Seq-5	16.21m, 5.97°	**15.07m**, **5.89**°	15.88m, 5.91°
Seq-6	26.02m, 9.02°	**18.88m**, **6.24**°	19.28m, 6.28°
Seq-7	31.83m, 17.99°	30.06m, 17.12°	**30.03m**, **17.05**°
Seq-8	25.62m, 22.38°	24.28m, 21.99°	**24.26m**, **21.54**°

	No update	Cull	Cull+combine
Seq-3	6.06m, 1.65°		
Seq-4	5.80m, 1.40°	**5.54m**, 1.39°	5.65m, **1.33**°
Seq-5	13.70m, 1.56°	13.12m, **1.52**°	**13.05m**, 1.55°
Seq-6	6.65m, 1.87°	**5.76m**, **1.75**°	6.60m, 1.85°
Seq-7	13.58m, 3.52°	11.80m, 2.81°	**10.87m**, **2.60**°
Seq-8	13.28m, 4.93°	**7.13m**, **2.31**°	7.15m, 2.47°

TABLE 4.1: Comparison between 3 different settings of our technique. Mean (top) and median (bottom) errors of 6-DoF pose on Oxford RobotCar are reported.

Sequences	No update	Cull	Cull+Combine
Seq-3	4.03		
Seq-4	**4.56**	5.05	4.82
Seq-5	**4.24**	5.06	4.87
Seq-6	3.81	4.03	**3.72**
Seq-7	3.82	4.18	**3.78**
Seq-8	3.77	3.91	**3.68**

TABLE 4.2: Inference time (ms) on Oxford RobotCar. `Cull+Combine` has comparable inference time while giving better accuracy (see Table 4.1) over `No update`.

different update strategies are given in Table 4.2. Without updating, the inference time is stable at ($\sim$ 4ms/query) between sequences, since the size of graph and the database do not change. In contrast, culling operation increases the inference time by about 1ms/query, and `Cull+Combine` makes it comparable to the `No update` case. This shows that the proposed method is able to compress the database to an extent that the query time after assimilation of new information remains comparable to the case of not updating the database at all.

4.6.2 Map maintenance and visiting unknown regions

Figure 4.3 shows the results on map maintenance with and without compression. Without compression, size of map $\mathcal{G}$ (specifically, adjacency matrix $\mathbf{E}$) grows continuously when appending a new query video $\mathcal{Q}$. In contrast, using our compression scheme, known places in $\mathcal{Q}$ are culled, and redundant nodes in $\mathcal{G}$ (i.e., nodes representing a same place) are combined. As a result, the graph is compressed.

Training sequences	VidLoc	MapNet	Our method
Seq-1,2	14.1h	11.6h	**98.9s**
Seq-3	-	6.2h	**256.3s**
Seq-4	-	6.3h	**232.3s**
Seq-5	-	6.8h	**155.1s**
Seq-6	-	5.7h	**176.5s**
Seq-7	-	6.0h	**195.4s**

TABLE 4.3: Training/updating time on the Oxford RobotCar.

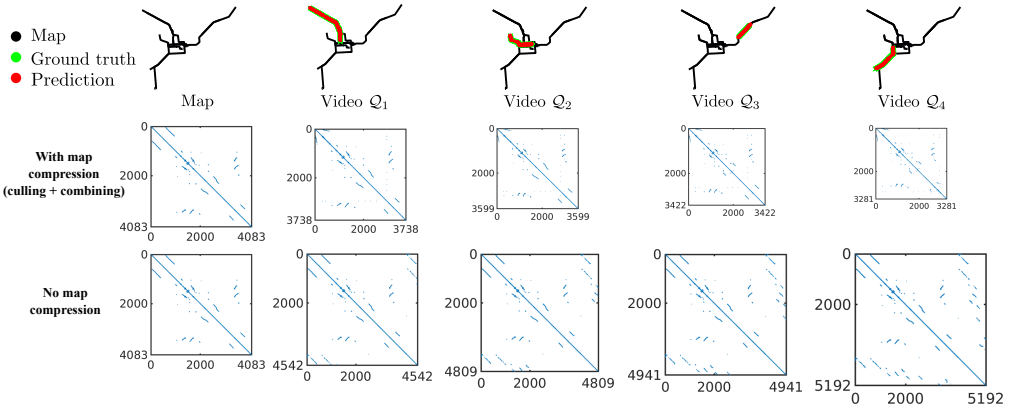


Fig. 4.3. Illustrating map maintenance w and w/o compression. After each query video $\mathcal{Q}$ finishes, we compress the map by culling known places in $\mathcal{Q}$ and combining old places on the map which represent the same place. Thus, the size of transition matrix is shrunk gradually. In contrast, if compression is not conducted, the size of transition matrix will continue increasing.

Methods	Seq-3	Seq-4	Seq-5	Seq-6	Seq-7	Seq-8
VidLoc	38.86m, 9.34°	38.29m, 8.47°	36.05m, 6.81°	51.09m, 10.75°	54.70m, 18.74°	47.64m, 23.21°
MapNet	9.31m, 4.37°	8.92m, 4.09°	17.19m, **5.72°**	26.31m, 9.78°	33.68m, 18.04°	26.55m, 21.97°
MapNet (update+ retrain)		8.71m, 3.31°	18.44m, 6.94°	28.69m, 10.02°	36.68m, 19.34°	29.64m, 22.86°
Our method	**6.59m, 3.28°**	**6.01m, 3.11°**	**15.88m**, 5.91°	**19.28m, 6.28°**	**30.03m, 17.05°**	**24.26m, 21.54°**

Methods	Seq-3	Seq-4	Seq-5	Seq-6	Seq-7	Seq-8
VidLoc	29.63m, **1.59°**	29.86m, 1.57°	31.33m, 1.39°	47.75m, **1.70°**	48.53m, 2.40°	42.26m, **1.94°**
MapNet	**4.69m**, 1.67°	**4.53m**, 1.54°	13.89m, **1.17°**	8.69m, 2.42°	12.49m, **1.71°**	8.08m, 2.02°
MapNet (update+ retrain)		5.15m, 1.44°	17.39m, 1.87°	11.45m, 3.42°	20.88m, 4.02°	11.01m, 5.21°
Our method	6.06m, 1.65°	5.65m, **1.33°**	**13.05m**, 1.55°	**6.60m**, 1.85°	**10.87m**, 2.60°	**7.15m**, 2.47°

TABLE 4.4: Comparison between our method, MapNet and VidLoc. Mean (top) and median (bottom) 6-DoF pose errors on the Oxford RobotCar dataset are reported.

Visiting unexplored area allows us to expand the coverage of our map, as we demonstrate using Mapillary data. We set $\gamma = 0.3$, i.e., we only accept the query frame which has the MaxAP belief ≥ 0.3. When the vehicle explores unknown roads, the probability of MaxAP is small and no localization results are accepted. Once the query sequence ends, the map coverage is also extended; see Fig. 4.4.

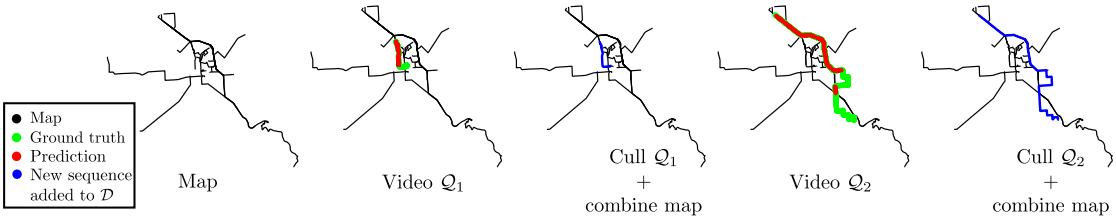


Fig. 4.4. Expanding coverage by updating the map. Locations are plotted using ground-truth GPS for visualization only.

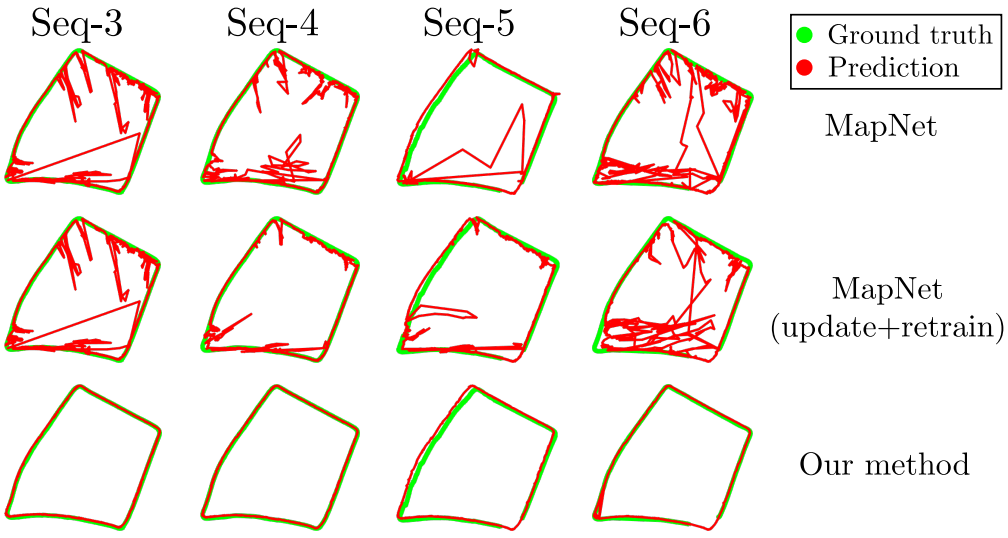


Fig. 4.5. Qualitative results on the RobotCar dataset.

4.6.3 Comparison against state of the art

Our method is compared against state-of-the-art localization methods: MapNet [17] and VidLoc [29]. We use the original authors' implementation of MapNet. VidLoc implementation from MapNet is used by the recommendation of VidLoc authors. All parameters are set according to suggestion of authors [1].

For map update in our method, `Cull+Combine` steps are used. MapNet is retrained on the new query video with the ground truth from previous predictions. Since VidLoc does not produce sufficiently accurate predictions, we do not retrain the network for subsequent query videos.

Our method outperforms MapNet and VidLoc in terms of the mean errors (see Table 4.4), and also has a smoother predicted trajectory than MapNet (see Fig. 4.5). In addition, while our method improves localization accuracy after updating the database (See Table 4.1), MapNet's results is worse after retraining (See Table 4.4). This is because MapNet is retrained on a noisy ground truth. However, though our method is

[1] Comparisons against [28] are not presented due to the lack of publicly available implementation.

qualitatively better than MapNet, differences in median error is not obvious: this shows that median error is not a good criterion for VL, since gross errors are ignored.

Note that our method mainly performs VPR; here, comparisons to VL methods are to show that a correct VPR paired with simple pose inheritance can outperform VL methods in presence of appearance change. The localization error of our method can likely be improved by performing SfM on a set of images corresponding to the highest belief.

Table 4.3 reports training/updating time for our method and MapNet and VidLoc. Particularly, for Seq-1 and Seq-2, our method needs around 1.65 minute to construct the k-means tree and build the graph, while MapNet and VidLoc respectively require 11.6 and 14.1 hours for training. For updating a new query sequence, MapNet needs about 6 hours of retraining the network, whilst our method culls the database and combine graph nodes in less than 5 minutes. This makes our method more practical in a realistic scenario, in which the training data is acquired continuously.

4.7 Conclusion

This chapter has answered the **Research question 2** *Can we develop scalable VPR for life-long operation?* The results from chapter 3 show the efficiency of probabilistic frameworks in modelling the temporal information, that inspires us to develop a scalable VPR technique based on Hidden Markov Model. To this end, we explore our technique according to following sub-questions

RQ2.1 *Can we discover missing knowledge in the map?*

We show that exploring belief provides us information useful for connecting nodes that might be missed due to imperfect mapping frameworks.

RQ2.2 *How to effectively absorb new data?*

The update stage has the complexity in proportion to the logarithm of the dataset size, which ensures a scalable update operation. The experiment shows our our accumulation process is much faster than that of deep learning methods.

RQ2.3 *Can we eat a lot but never get fat?*

We empirically show that our compression technique maintains a scalable map. It not only improves the accuracy localization as a result of continuous adaption to new data, but also ensures the computational cost does not grow visibly with the database size.

Chapter 5

Hidden Markov Model with Memory Management and Polytope VLAD

This chapter is based on the content of following journal paper

- Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, and Ian Reid. HM^4: Hidden Markov Model with Memory Management for Visual Place Recognition. *IEEE Robotics and Automation Letters (RA-L)*, 2020.

Statement of Authorship

Title of Paper	HM^4: Hidden Markov Model with Memory Management for Visual Place Recognition
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Anh-Dzung Doan, Yasir Latif, Tat-Jun Chin, Ian Reid, *"HM^4: Hidden Markov Model with Memory Management for Visual Place Recognition"*, IEEE Robotics and Automation Letters (RA-L), 2020

Principal Author

Name of Principal Author (Candidate)	Anh-Dzung Doan		
Contribution to the Paper	Developed the method, conducted the experiments, and wrote the paper		
Overall percentage (%)	60		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	18 February 2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Yasir Latif		
Contribution to the Paper	Supervised the development of the methods, provided the suggestions, and revised the paper.		
Signature		Date	23/02/2021

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Proposed the general direction, supervised the development of the methods, provided the suggestions, and revised the paper.		
Signature		Date	21 Feb 2021

Name of Co-Author	Ian Reid		
Contribution to the Paper	Provided suggestions and proofread the paper		
Signature		Date	23/2/2021

5.1 Introduction and research questions

Visual place recognition needs to be robust against appearance variability due to natural and man-made causes. The data collection should thus be an ongoing process to allow continuous appearance changes to be recorded. To build a data-driven VPR system that is robust against continuous appearance changes, a promising solution is to continuously accumulate data to refine the system [28, 41, 46]. While continuous data collection can be achieved via opportunistic or crowdsourced services (e.g., using taxi fleets), the ever-growing database demands a VPR algorithm scalable, i.e., the computational effort *and* memory usage (to be distinguished from long-term storage) for training and inference must grow slowly with the database size. With this above motivation, here we ask:

Research question 3 *Can we design a lightweight and scalable VPR system?*

In the context of autonomous driving, chapter 4 presents a promising VPR solution based on Hidden Markov Model (HMM). The basic idea is to exploit temporal continuity in the trajectory of the car to guide image matching. A key innovation is the usage of a state space model (a topological map) and an observation model (which uses an image indexing structure) that can be iteratively updated to efficiently "absorb" new information from newly appended images. This enables sublinear growth in *inference time* w.r.t. database size. Chapter 4 also shows that its updating procedure is much faster than the refinement process in end-to-end learning-based methods [17].

Those characteristics demonstrate a great potential of the HMM-based framework in achieving a completely scalable solution for VPR, hence we investigate this direction according to two following sub-questions:

RQ3.1 *Can we achieve sub-linear space-time complexity?*

It is apparent that the fundamental weakness of chapter 4 is the *linear memory complexity* of the observation model w.r.t. the database size. Specifically, while the image indexing structure [122] that underpins the observation model can be queried and updated efficiently, the whole indexing structure must be loaded in the main memory to support querying—this is equivalent to storing *all* database images in the main memory, which is infeasible in a lifelong operation.

A possible solution is to aggregate images (e.g., [131]) to remove redundancy and maintain a fixed-size database. However, image aggregation methods are imperfect and since the errors will be cumulated over time, the aggregated images will contain serious artefacts that affect localization accuracy.

A more direct solution is to progressively delete older images since they do not contain up-to-date appearance. However, devising an effective deletion scheme is nontrivial, since the images of different places could be refreshed at different rates (e.g., densely- versus sparsely-populated places), thus requiring careful tuning of place-dependent forgetting factors. More importantly, even if an effective deletion scheme can be built, the resulting database could still be too big to fit in memory—clearly a more fundamental treatment is required.

This chapter proposes a novel HMM-based VPR system called HM^4 (HMM with Memory Management), which employs a two-tiered memory concept—active memory (AM) and passive storage (PS)—to allow scalable operation; see Fig. 5.1, AM corresponds to fast main memory whose size is limited, while PS represents slower long-term memory storing the full image database.

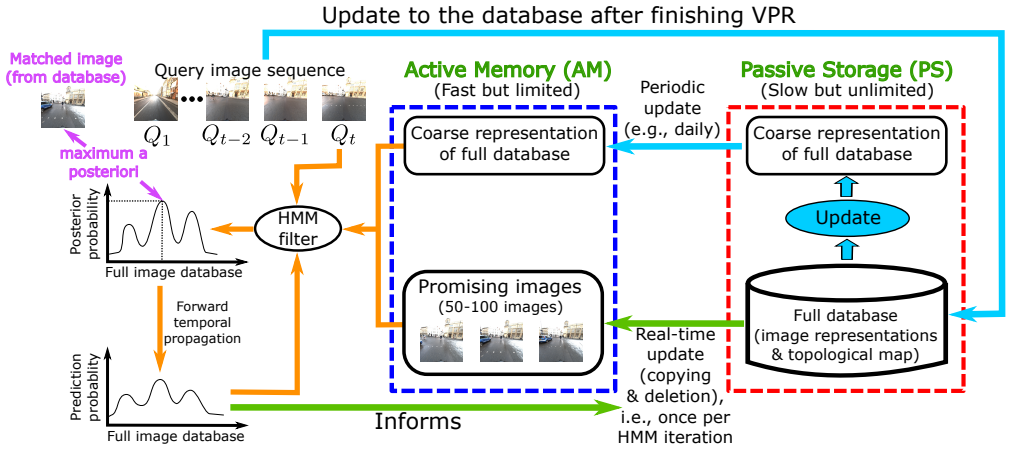


Fig. 5.1. HM^4 system for lifelong VPR, where, →: online inference, →: real-time update, and →: periodic update

Given a query video $\mathcal{Q}$, HM^4 compute the probability of matches between the current query image $Q_t \in \mathcal{Q}$ and the full database. For scalability, temporal reasoning and image search are tightly coupled, and the probabilities are computed in a layered manner:

- Forward temporal propagation is used to identify most promising database images to load to AM in real-time at each HMM iteration to replace the least promising ones. Typically, only a small number of promising images (50-100) are stored in AM at any time.
- A coarse representation of the full database in the form of feature space centroids and a topological submap (e.g., 7000 centroids and vertices for 17km traversal distance) are also loaded into AM, but these are updated at a lower frequency (e.g., daily) in PS to account for newly added image sequences to the database.

- The (posterior) probability of matches are recursively computed by the HMM filter, with transition and observation models that are evaluated exactly using the promising images, and approximately with bounded error using the coarse representations of the full database.

In effect, the data stored in AM is of constant size (provided the coverage area does not grow) thus ensuring HMM inference in constant time and memory, as well as robustness against appearance changes. Sec. 5.3 will provide the details.

In practice, we can implement AM on client side (e.g., self-driving cars) and PS on cloud whose storage can be seen as nearly infinite. The update operation is conducted via a stable and high-speed connection (e.g., 5G).

Our two-tiered memory concept for HMM-based VPR was inspired by RTAB-Map [93] – a loop closure detection for SLAM. RTAB-Map evaluates loop-closure hypotheses on images stored in AM (called "working memory" in [93]), which is a subset of the full image database judged to be the most promising by a heuristic. However, RTAB-Map does not maintain matching probabilities over the full database, thus, when none of the images in AM are good matches to the current query, the algorithm is lost and a hard reset is required. In contrast, HM^4 always maintains a probability distribution over the full database, thus allowing recognition of places not in the AM.

However, the strategy of memory management can only provide the scalable capability, while we empirically observe that the image representation of chapter 4 is the dominant factor in the memory requirement for VPR. Hence, we investigate the following sub-question

RQ3.2 *Can we find a compact yet informative image representation?*

Since efficiency of VLAD [79] has been shown in chapters 3 and 4, we derive its compact version called *polytope VLAD* (polyVLAD), which greatly reduces the memory footprint of HM^4. Specifically, to avoid the burstiness problem, locally aggregated vectors are L2-normalized, which allows us to use a cross-polytope to partition the unit sphere. Inspired by LSH [5], the compression scheme is conducted through random rotation matrices, which theoretically guarantees a small distance between two similar images (see [5, Theorem 1]). Also, we propose to employ inverted index for effectively computing distances between polyVLAD vectors, leading to a significant improvement in terms of the inference time.

The experiments show HM^4 offers a scalability solution for VPR. Combined with polyVLAD, a lightweight system is obtained while localization accuracy is not influenced.

This chapter is organized as follows: The HMM-based VPR is briefly reviewed in Sec. 5.2. Next, we provide the details of HM^4 and polyVLAD in Sec. 5.3 and Sec. 5.4 respectively. The experiments is presented in Sec. 5.5. Finally, we conclude the chapter in Sec 5.6.

5.2 Background: HMM for VPR

Let $\mathcal{D} = \{I_i\}_{i=1}^N$ be a dataset of images (e.g., video frames of street recordings from multiple vehicles). Following chapter 4, we regard each I_i as a "place" and define a topological map $\mathcal{G}$ over $\mathcal{D}$, where each I_i is a vertex of $\mathcal{G}$. Two vertices are connected by an edge in $\mathcal{G}$ if their FOVs overlap sufficiently (e.g., if they are temporally close in their source video, or if they are matched in previous iterations). Details on computing $\mathcal{G}$ in our method will be provided in Sec. 5.3.

Given a query video $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_T\}$, our goal is to match each frame $Q_t \in \mathcal{Q}$ with an image from $\mathcal{D}$ that corresponds to the same place. To this end, we define an HMM $\{\mathbf{E}, \mathbf{O}, \pi\}$, where $\mathbf{E} \in \mathbb{R}^{N\times N}$ is the transition matrix, $\mathbf{O} \in \mathbb{R}^{N\times T}$ is the observation matrix, and $\pi \in \{0,1\}^N$, with $\sum_i \pi_i = 1$ is the initial matching probabilities (assumed to be uniform). The place associated to Q_t is regarded as the random variable $s_t \in \{1, \ldots, N\}$. Element $\mathbf{E}(i,j)$ encodes the probability of moving from I_i to I_j between adjacent time steps $t-1$ and t

$$\mathbf{E}(i,j) = P(s_t = j \mid s_{t-1} = i), \tag{5.1}$$

while element $\mathbf{O}(i,t)$ encodes the probability of observing Q_t given the place I_i

$$\mathbf{O}(i,t) = P(Q_t \mid s_t = i). \tag{5.2}$$

Details on computing $\mathbf{E}$ and $\mathbf{O}$ in our method will be provided in Sec. 5.3.

Given the input sequence up to time t, i.e., $Q_{1:t} = \{Q_1, Q_2, \ldots, Q_t\}$, the aim is to calculate the posterior probabilities $P(s_t \mid Q_{1:t})$, for $s_t = 1, \ldots, N$, which can be represented as a vector

$$\mathbf{p}_t = \begin{bmatrix} P(1 \mid Q_{1:t}) & P(2 \mid Q_{1:t}) & \ldots & P(N \mid Q_{1:t}) \end{bmatrix}. \tag{5.3}$$

The posterior probabilities are recursively computed as

$$\mathbf{p}_t = \eta\, \mathbf{o}_t \circ \mathbf{E}^T \mathbf{p}_{t-1}, \tag{5.4}$$

where $\circ$ is the element-wise product, $\mathbf{o}_t$ is the t-th column of $\mathbf{O}$ and η is a normalizing constant to ensure $\sum_i \mathbf{p}_t(i) = 1$; for initialization, $\mathbf{p}_0 = \pi$. The VPR decision at time t is taken as the maximum *a posteriori* result

$$s_t^* = \underset{i \in \{1,\dots,N\}}{\operatorname{argmax}} \mathbf{p}_t(i) \tag{5.5}$$

For more details of the basic idea of HMM for VPR, see chapter 4.

From equation (5.4), it is clear that the time complexity of each HMM iteration is $O(Nr)$ (r is the maximum number of non-zero values in each column of $\mathbf{E}$), and memory complexity is $O(ND)$ (D is dimensionality of image representation), where, N increases unboundedly in lifelong operation.

5.3 Achieving scalability with HM^4

An overview of HM^4 was provided in Sec. 5.1 and Fig. 5.1. This section will describe the proposed VPR system in detail.

5.3.1 Compact image representation

Our system employs a new compact image representation called *polytope VLAD*, which converts each image to a feature vector encoded by 8192 bit. This provides a constant factor reduction in the computation and memory required. We postpone the description of polytope VLAD to Sec. 5.4; henceforth in this section, when we refer to an image, we mean specifically it is polytope VLAD vector.

5.3.2 Coarse respresentation of full database

To avoid loading the full database $\mathcal{D} = \{I_i\}_{i=1}^N$ into AM, we extract a summary of the database in the form of a topological map and feature space centroids.

5.3.2.1 Topological map

A topological map is built over $\mathcal{D}$ to summarize the "physical" connectivity of the images: in autonomous driving, the topological map reflects the road network of the coverage area. In HM^4, the topological map is equivalent to the transition matrix $\mathbf{E}$ (equation (5.1)) of the HMM. Moreover, $\mathbf{E}$ is built incrementally (sequence-by-sequence) as

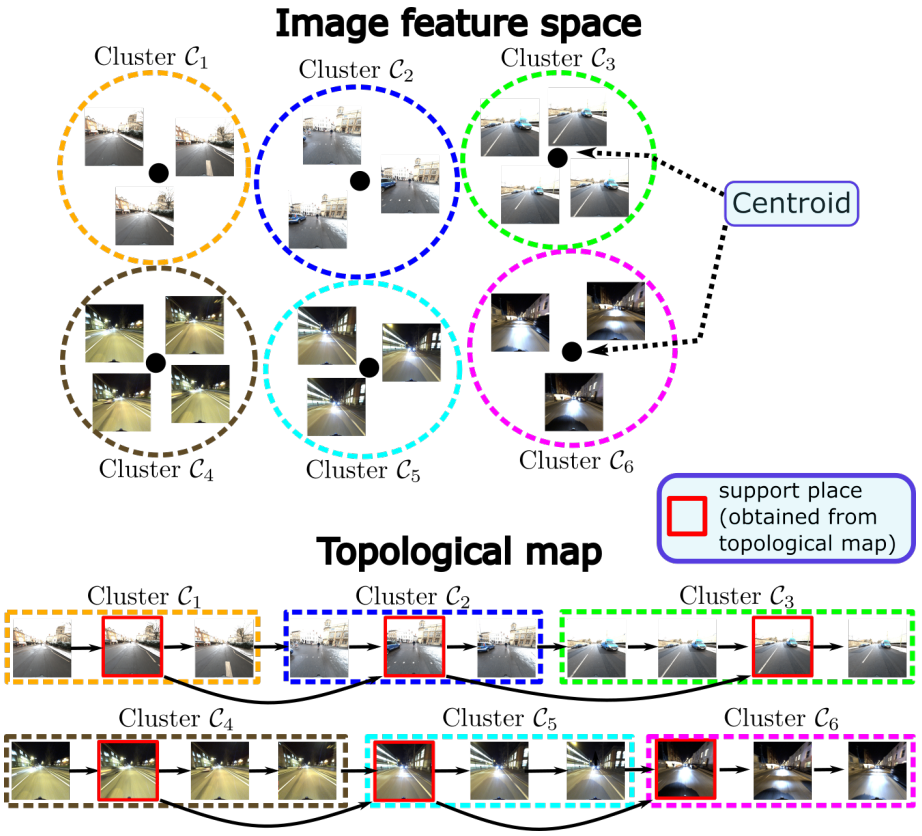


Fig. 5.2. Coarse representation of full database. The disjoint clusters are built on image feature space. Topological submap are created to coarsely represent the topological map.

new videos are appended to $\mathcal{D}$, using results of HMM inference. We will describe the initialization and updating of $\mathbf{E}$ later in Sec. 5.3.4. For now, assume $\mathbf{E}$ is available.

5.3.2.2 Feature space clustering

While $\mathbf{E}$ summarizes the physical connectivity of $\mathcal{D}$, the images are also clustered in the feature space to summarize their appearance. We partition $\mathcal{D}$ (in feature space) into K disjoint clusters $\{\mathcal{C}_1, ..., \mathcal{C}_K\}$, where each cluster $\mathcal{C}_k$ is a subset of the images in $\mathcal{D}$; for brevity, we also take

$$\mathcal{C}_k \subset \{1, \ldots, N\} \tag{5.6}$$

when we wish to refer to the indices of images that lie in each cluster. Each $\mathcal{C}_k$ contains a centroid B_k as a representative feature vector. We will be using the centroids to compute approximate distances between Q_t and the images in $\mathcal{D}$. If the chosen distance d is a

metric, it then holds that

$$\forall I_i \in \mathcal{C}_k, \quad |d(Q_t, I_i) - d(Q_t, B_k)| < d(I_i, B_k). \tag{5.7}$$

We then take $d(Q_t, I_i) \approx d(Q_t, B_k)$ for all $I_i \in \mathcal{C}_k$. To minimize the approximation error, we should find clusters $\{\mathcal{C}_k\}_{k=1}^K$ and centroids $\{B_k\}_{k=1}^K$ that minimize

$$\frac{1}{N} \sum_{k=1}^K \sum_{I_i \in \mathcal{C}_k} d(I_i, B_k). \tag{5.8}$$

For continuous feature spaces (e.g., I_i is a NetVLAD [7] vector) where d is the Euclidean distance, K-means algorithm can be used to perform the clustering. The proposed polytope VLAD, however, yields discrete feature vectors; we will discuss the appropriate metric in Sec. 5.4.

Also, similar to the construction of $\mathbf{E}$, the feature space clusters are computed incrementally as new videos are appended to $\mathcal{D}$, which we will describe in Sec. 5.3.4.

5.3.2.3 Topological submap

Note that the centroids $\{B_k\}_{k=1}^K$ do not generally correspond to feature vectors of actual images. To associate an image to each B_k, we seek the image in $\mathcal{C}_k$ that has the highest degree in $\mathbf{E}$, i.e.,

$$p_k = \arg\max_{i \in \mathcal{C}_k} \sum_j \mathbb{I}(\mathbf{E}(i,j) > 0). \tag{5.9}$$

We call I_{p_k} the *support place* of $\mathcal{C}_k$. Given the clusters and associated support places, we construct the topological submap of $\mathcal{D}$ as

$$\mathbf{E}_{sm} = \begin{bmatrix} \mathbf{E}(:, p_1) & \mathbf{E}(:, p_2) & \dots & \mathbf{E}(:, p_K) \end{bmatrix}. \tag{5.10}$$

In words, $\mathbf{E}_{sm}$ are the columns of $\mathbf{E}$ corresponding to the support places. See Fig. 5.2 for a high-level idea of the coarse representation of $\mathcal{D}$.

5.3.3 Two-tiered HMM inference

Our aim in this subsection is to perform HMM inference to obtain the posterior $\mathbf{p}_t$ (over the full database) for the current time t. Fig. 5.3 illustrates the data available to perform the inference in HM^4. Specifically, only the coarse representations of the full database—specifically, $\{B_k\}_{k=1}^K$ and $\mathbf{E}_{sm}$—are stored in AM, whereas the full database $\mathcal{D}$ and topological map $\mathbf{E}$ are stored in PS.

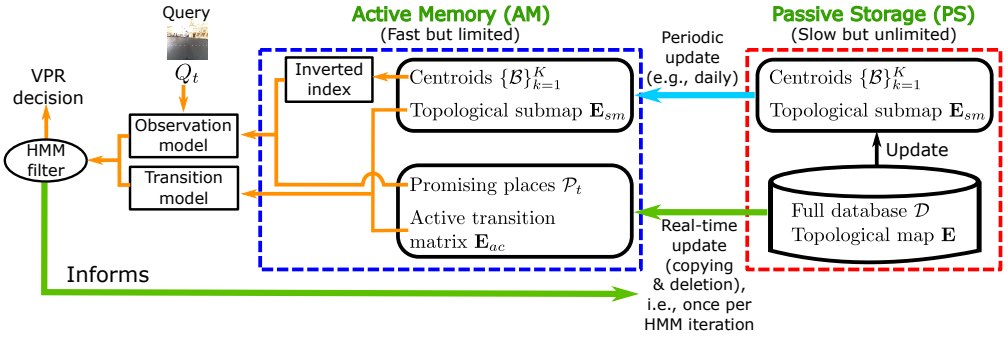


Fig. 5.3. Data available in AM to perform VPR inference, where, →: online inference, →: real-time update, and →: periodic update. Note that "Inverted index", which is used for polytope VLAD, will be described in Sec. 5.4. Before Sec. 5.4, we can ignore this component, and assume centroids are the input of observation model.

Noting that HMM inference is a recursive process, using the posterior probability $\mathbf{p}_{t-1}$ from the previous iteration, we identify a set of *promising places*:

$$\mathcal{P}_t = \mathcal{P}_t^{(1)} \cup \mathcal{P}_t^{(2)}, \tag{5.11}$$

where, $\mathcal{P}_t^{(1)} = \{i \in \{1,\dots,N\} \mid \mathbf{p}_{t-1}(i) \geq \zeta\}$, $\mathcal{P}_t^{(2)} = \{j \in \{1,\dots,N\} \mid \mathbf{E}(i,j) > 0, i \in \mathcal{P}_t^{(1)}\}$, and $\zeta \in [0,1]$ is a preselected threshold (see Sec. 5.3.5 on setting ζ). Intuitively, $\mathcal{P}_t$ consists of the set of places that Q_t likely corresponds to according to HMM propagation (before considering the appearance of Q_t), and the images that are adjacent to the likely places according to the topological map. It is vital to highlight that $\mathcal{P}_t$ is a tiny subset (e.g., $|\mathcal{P}_t| \leq 100$) of the full database $\mathcal{D}$.

We then obtain the *active* transition matrix as

$$\mathbf{E}_{ac} = \begin{bmatrix} \mathbf{E}(:,\mathcal{P}_{t,1}) & \mathbf{E}(:,\mathcal{P}_{t,2}) & \dots & \mathbf{E}(:,\mathcal{P}_{t,M}) \end{bmatrix} \in \mathbb{R}^{N \times M}, \tag{5.12}$$

where $\mathcal{P}_{t,m}$ is the m-th item of $\mathcal{P}_t$, and $M = |\mathcal{P}_t|$. In words, $\mathbf{E}_{ac}$ are the columns of $\mathbf{E}$ corresponding to the promising places. Since $\mathbf{E}$ resides in PS, $\mathbf{E}_{ac}$ is transferred to the AM at every HMM iteration. Also, while the size of $\mathbf{E}_{ac}$ depends on N, the matrix is very sparse, thus the transfer is cheap.

Given the current query image Q_t, our HMM observation model is based on computing the observation likelihood

$$L(Q_t, I_i) = \exp\left(-\frac{d(Q_t,I_i)}{\sigma}\right), \tag{5.13}$$

where I_i is an arbitrary database image, d is the feature space metric, and σ is a bandwidth parameter (see Sec. 5.3.5 for its setting). To avoid computing the likelihood

over the full database, we define the *active* observation vector

$$\mathbf{o}_{t,ac} = \Big[L(Q_t, I_{\mathcal{P}_{t,1}}), L(Q_t, I_{\mathcal{P}_{t,2}}), \ldots, L(Q_t, I_{\mathcal{P}_{t,M}})\Big]^T \in \mathbb{R}^M, \tag{5.14}$$

which requires only the promising images, and the *background* observation vector

$$\mathbf{o}_{t,bg} = \Big[L(Q_t, B_1), L(Q_t, B_2), \ldots, L(Q_t, B_K)\Big]^T \in \mathbb{R}^K, \tag{5.15}$$

which uses the feature space centroids of the full image database. As in the case of obtaining the active transition matrix, the feature vectors of the promising places will be transferred to AM at each HMM iteration.

We can now perform the inference: we modify the standard HMM update (5.4) to use only the available information in AM to obtain the matching confidence:

- Matching confidence in promising places $\mathcal{P}_t$:

$$\mathbf{q}_t(\mathcal{P}_{t,m}) = \mathbf{o}_{t,ac}(m).\mathbf{E}_{ac}(:,m)^T.\mathbf{p}_{t-1}, \tag{5.16}$$

- Matching confidence in other places computed from coarse representation:

$$\mathbf{q}_t(i) = \mathbf{o}_{t,bg}(k).\mathbf{E}_{sm}(:,k)^T.\mathbf{p}_{t-1}, \text{ if } i \notin \mathcal{P}_t \ \& \ i \in \mathcal{C}_k, \tag{5.17}$$

The posterior $\mathbf{p}_t$ is then obtained by normalizing:

$$\mathbf{p}_t = \frac{\mathbf{q}_t}{\sum \mathbf{q}_t}, \tag{5.18}$$

The VPR decision is finally made by equation (5.5).

5.3.4 Updating database representation

Given a query video $\mathcal{Q} = \{Q_t\}_{t=1}^T$ localized by VPR.

5.3.4.1 Updating topological map

We create its topological map

$$\mathbf{E}^{\mathcal{Q}}(i,j) = \begin{cases} \alpha \exp\frac{(j-i)^2}{\delta^2} & 0 \le j-i \le V_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{5.19}$$

where, α is normalization constant to ensure the summation of every row of $\mathbf{E}^{\mathcal{Q}}$ equals to one, $V_{\max}$ is the maximum velocity of the vehicle, and δ is preselected scale value

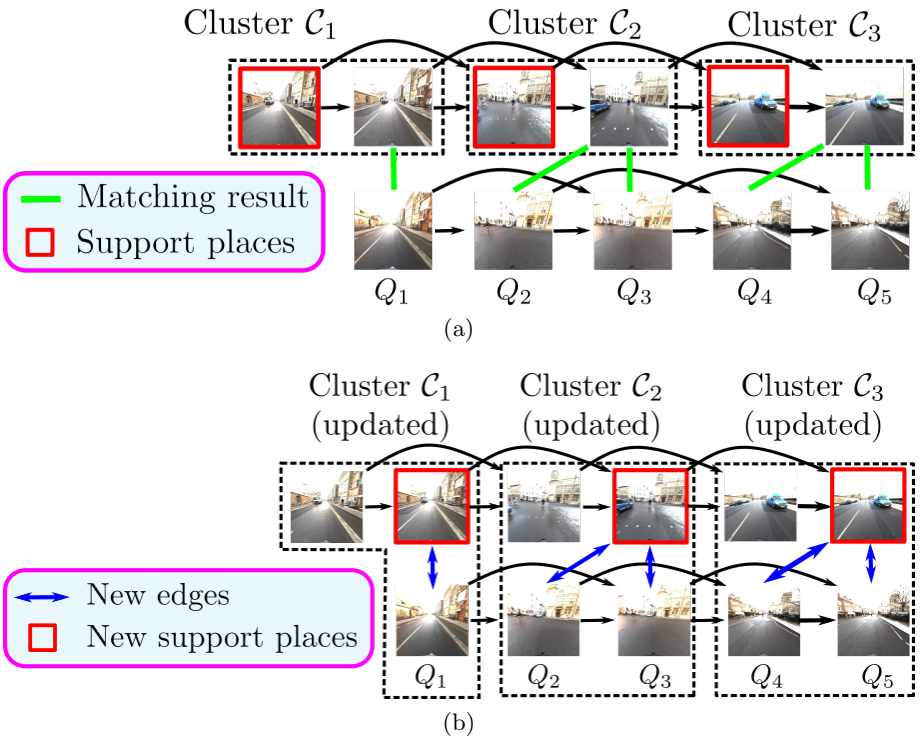


Fig. 5.4. **(a)** Matching results for $\mathcal{Q} = \{Q_i\}_{i=1}^{5}$. **(b)** Updating video $\mathcal{Q}$ to the database.

(see Sec. 5.3.5 for their settings). After that, the topological map $\mathbf{E}$ is expanded in the diagonal direction

$$\mathbf{E} = \begin{bmatrix} \mathbf{E} & 0_{N\times T} \\ 0_{T\times N} & \mathbf{E}^{\mathcal{Q}} \end{bmatrix}. \tag{5.20}$$

We create new edges for every localized query $Q_t \in \mathcal{Q}$ with its matched place i:

$$\begin{aligned} \mathbf{E}(t+N, i) &= \mathbf{E}(t+N, t+N) \\ \mathbf{E}(i, t+N) &= \mathbf{E}(i, i). \end{aligned} \tag{5.21}$$

$\mathbf{E}$ is then normalized to ensure the summation of each row equals to 1.

5.3.4.2 Updating coarse representation

For each $Q_t \in \mathcal{Q}$, we add it to its matched cluster, then compute new centroids:

$$B_k = \arg\min_{B_k} \textstyle\sum_{I_i \in \mathcal{C}_k} d(I_i, B_k) \tag{5.22}$$

For continuous feature space, where d is Euclidean distance, $B_k = \frac{1}{|\mathcal{C}_k|} \sum_{I_i \in \mathcal{C}_k} I_i$. For polytope VLAD, Sec. 5.4 will discuss how to compute B_k.

Algorithm 5.1 VPR with HM^4

Require: Unlimited query videos $\{\mathcal{Q}_i\}_{i=1}^{\infty}$, topological map $\mathbf{E}$, database $\mathcal{D}$, parameters V_{max}, δ, σ, ζ

Build clusters $\{\mathcal{C}\}_{k=1}^{K}$, centroids $\{B\}_{k=1}^{K}$, and topological submap $\mathbf{E}_{sm}$ (Sec. 5.3.2)
Copy centroids $\{B\}_{k=1}^{K}$, and topological submap $\mathbf{E}_{sm}$ from PS to AM
for each $\mathcal{Q} \in \{\mathcal{Q}_i\}_{i=1}^{\infty}$ **do**
    Initialize $\mathcal{P}_0 = \emptyset$
    for each frame $Q_t \in \mathcal{Q}$ **do**
        if $t = 1$ **then**
            Compute $\mathbf{o}_{t,bg}$ using Eq. (5.15)
            Compute matching confidence $\mathbf{q}_1$ using Eq. (5.17)
        else
            Seek $\mathcal{P}_t$ using Eq. (5.11)
            Build $\mathbf{E}_{ac}$ using Eq. (5.12)
            Copy images $\{I_i \,|\, i \in \mathcal{P}_t,\, i \notin \mathcal{P}_{t-1}\}$ from PS to AM
            Delete images $\{I_i \,|\, i \notin \mathcal{P}_t,\, i \in \mathcal{P}_{t-1}\}$ in AM
            Compute $\mathbf{o}_{t,ac}$ using Eq. (5.14), and $\mathbf{o}_{t,bg}$ using Eq. (5.15)
            Compute matching confidence $\mathbf{q}_t$ using (5.16) and (5.17)
        end if
        Compute belief $\mathbf{p}_t$ using Eq. (5.18)
        Find matched place/image i^* using Eq. (5.5)
    end for
    Store localized $\mathcal{Q}$ in PS for updating.
    if (On map update) **then**
        Update topological map $\mathbf{E}$, clusters $\{\mathcal{C}\}_{k=1}^{K}$, centroids $\{B\}_{k=1}^{K}$, and topological submap $\mathbf{E}_{sm}$ (Sec. 5.3.4)
        Copy new centroids $\{B\}_{k=1}^{K}$, new topological submap $\mathbf{E}_{sm}$ from PS to AM
    end if
end for

With new centroids and new topological map $\mathbf{E}$ expanded by $\mathbf{E}^{\mathcal{Q}}$, we build a new topological submap $\mathbf{E}_{sm}$ as described in Sec. 5.3.2.

Fig. 5.4 shows the high-level idea of the updating process. Note that this updating process also allows us to start with database $\mathcal{D}$ containing a single video first, then incrementally build topological map and coarse representation (sequence by sequence) using HMM inference.

5.3.5 Overall algorithm

The proposed algorithm for VPR is presented in Algorithm 5.1, and its high-level idea is shown in Fig 5.1. At the first frame ($t = 1$), $\mathbf{p}_0$ is uniformly distributed, hence we only use $\mathbf{o}_{t,bg}$ for computing belief $\mathbf{p}_1$. In our experiment, we set $V_{max} = 10$, $\delta = 3$, $\sigma = 0.03$, and $\zeta = 0.00015$.

5.3.6 Complexity analysis:

In standard HMM implementation, at each HMM iteration (each Q_t), the memory complexity is $O(ND)$, and time complexity of computing $\mathbf{o}_t$ is $O(ND)$, that of computing $\mathbf{p}_t$ is $O(Nr)$ (r is the numbers of non-zero values in each column of $\mathbf{E}$, which can be seen as a constant). In the context of sequentially updating new sequences, N grows *linearly and unboundedly.*

By contrast, our VPR algorithm has $O\left((K+|\mathcal{P}_t|)D\right)$ memory complexity, $O\left((K+|\mathcal{P}_t|)D\right)$ time complexity of computing $\mathbf{o}_t$ ($\mathbf{o}_{t,bg}$ and $\mathbf{o}_{t,ac}$), and $O\left((K+|\mathcal{P}_t|)r\right)$ time complexity of computing $\mathbf{p}_t$ (note $K+|\mathcal{P}_t| \ll N$). We experimentally show $K+|\mathcal{P}_t|$ remains almost constant when updating new sequences. In Sec. 5.4, with polytope VLAD, the time complexity of computing $\mathbf{o}_t$ can be slashed by a constant factor, i.e., $O\left(\frac{KD}{\text{const}}+|\mathcal{P}_t|D\right)$

5.4 Polytope VLAD

This section will describe polytope VLAD in details (Sec. 5.4.2), and the usage of inverted index for efficiently computing observation model (Sec. 5.4.3).

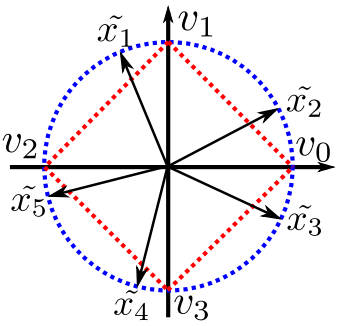
5.4.1 Background: VLAD

VLAD [79] is a feature aggregations scheme that given a set of local image features $\{f_i\}_{i=1}^{\mathbf{N}}$ (either hand-crafted [167] or deep learning [64] features), it assigns each local feature f_i to the closest cluster c_l of a vocabulary of size L, and accumulates residuals via: $x_l = \sum_{f_i \in \text{NN}(c_l)} f_i - c_l$

To avoid burstiness problem, before concatenating all x_l to form a VLAD vector, x_l is L2-normalized [8], thus each x_l is located on the unit sphere $S^{\mathbf{d}-1}$. Adapted the idea of LSH [5], we derive the "polytope VLAD" (polyVLAD) to compress vectors x_l into compact codes via a cross-polytope.

5.4.2 Polytope VLAD

Given a set of vectors $X = [x_1, ..., x_L]$ on a unit sphere $S^{\mathbf{d}-1}$, we employ a cross-polytope to partition the unit sphere $S^{\mathbf{d}-1}$. A cross-polytope is defined by a set of vertices: $V = \{v_i \,|\, i = 0, ..., 2\mathbf{d}-1\}$ that are all the permutations of $(\pm 1, 0, ..., 0)$, where v_i is a $\mathbf{d}$-dimensional vector. Let $R \in \mathbb{R}^{\mathbf{d}\times\mathbf{d}}$ be a random rotation matrix. We firstly rotate X by R: $\tilde{X} = RX = [\tilde{x_1}, ..., \tilde{x_L}]$, then each rotated vector $\tilde{x_l}$ is encoded by the



(a) polyVLAD idea

$\{B_1 = [3,0,2], B_2 = [0,2,0], B_3 = [2,1,3], B_4 = [3,2,0]\}$

index

$\mathcal{I}_0 = \{2\}$	$\mathcal{I}_1 = \{\}$	$\mathcal{I}_2 = \{3\}$	$\mathcal{I}_3 = \{1,4\}$
$\mathcal{I}_4 = \{1\}$	$\mathcal{I}_5 = \{3\}$	$\mathcal{I}_6 = \{2,4\}$	$\mathcal{I}_7 = \{\}$
$\mathcal{I}_8 = \{2,4\}$	$\mathcal{I}_9 = \{\}$	$\mathcal{I}_{10} = \{1\}$	$\mathcal{I}_{11} = \{3\}$

(b) An example of inverted index

Fig. 5.5. (a) A polytope (dashed red) with vertices v_0, v_1, v_2, v_3 is used to partition a unit sphere (dashed blue). **(b)** The size of inverted index is $D \times 2\mathbf{d}$ ($D = 3$, $\mathbf{d} = 2$). Centroid B_1 is indexed to $\mathcal{I}_3$, $\mathcal{I}_4$ and $\mathcal{I}_{10}$ using equation (5.26).

nearest vertex of the polytope V. The idea is illustrated in Fig. 5.5a and the encoding function is:

$$h_l = \underset{i}{\text{argmax}}(\tilde{x}_l^T.v_i)\,, \forall v_i \in V. \tag{5.23}$$

Rearranging all the vertex vectors v_i in a matrix: $V_{\mathbf{d}\times 2\mathbf{d}} = \begin{bmatrix}\mathbf{I}_{\mathbf{d}\times\mathbf{d}} & -\mathbf{I}_{\mathbf{d}\times\mathbf{d}}\end{bmatrix}$ where, $\mathbf{I}_{\mathbf{d}\times\mathbf{d}}$ is identity matrix of size $\mathbf{d} \times \mathbf{d}$, the product of $V^T.\tilde{x}_l$ is vector

$$u_l = [\tilde{x}_l^{[1]}, \ldots, \tilde{x}_l^{[\mathbf{d}]}, -\tilde{x}_l^{[1]}, \ldots, -\tilde{x}_l^{[\mathbf{d}]}]^T = [\tilde{x}_l^T, -\tilde{x}_l^T]^T \tag{5.24}$$

Due to the symmetric characteristic of u_l, we rewrite the encoding function (5.23):

$$h_l = \begin{cases} m^* - 1 & \tilde{x}_l^{[m^*]} \geq 0 \\ m^* + \mathbf{d} - 1 & \tilde{x}_l^{[m^*]} < 0 \end{cases} \tag{5.25}$$

where $m^* = \underset{m\in\{1,\ldots,\mathbf{d}\}}{\arg\max}(|\tilde{x}_l^{[m]}|)$ and h_l is the index of the closest vertex to $\tilde{x}_l$.

Algorithm 5.2 concretely describes the polyVLAD encoding scheme. With M different rotation matrices, M polyVLAD vectors are generated and then concatenated to obtain single polyVLAD vector I. The number of bits required to encode vector I is: $(\log_2\mathbf{d} + 1).L.M$. Rotation matrices are randomly sampled using Algorithm 5.3, which is based on Gram-Schmidt orthogonalization (line 4).

5.4.2.1 Clustering polyVLAD

As polyVLAD vectors are in discrete space, Jaccard distance (also metric) [90] is used. To build clusters $\{\mathcal{C}\}_{k=1}^K$ for coarse representation of HM^4, we use K-modes algorithm [75]

Algorithm 5.2 polyVLAD encoding

Require: Set of vectors X on a unit sphere $S^{\mathbf{d}-1}$, set of M rotation matrices $\mathbf{R} = \{R_1, ..., R_M\}$

Initialize polyVLAD vector $I = [\,]$
for r = 1 **to** M **do**
  Rotate vectors $\tilde{X} = R_r X$
  for each $\tilde{x}_l \in \tilde{X}$ **do**
    $m^* = \underset{m \in \{1,...,\mathbf{d}\}}{\text{argmax}} (|\tilde{x}_l^{[m]}|)$.
    Estimate h_l using equation (5.25)
    Concatenate $I = [I\ ,\ h_l]$
  end for
end for
return polyVLAD vector I

Algorithm 5.3 Randomly sampling rotation matrix

Require: feature dimension **d**

for i=1 **to d do**
  Sample vector r_i from **d**-dimensional Gaussian distribution
  for $j = 1$ **to** i -1 **do**
    $r_i = r_i - \frac{(r_i^T . r_j).r_j}{||r_j||_2}$
  end for
  $r_i = \frac{r_i}{||r_i||_2}$
end for
return Rotation matrix $R = [r_1, ..., r_{\mathbf{d}}]$

to minimize cost function (5.8). In updating process (Sec. 5.3.4), new centroids (equation (5.22)) are computed using Theorem 1 of [75].

5.4.3 Inverted index for efficient distance computation

In algorithm 5.1, computing $\mathbf{o}_{t,bg}$ (equation (5.15)) is most costly, as numbers of clusters are much larger than numbers of promising images ($K \gg |\mathcal{P}_t|$). Therefore, inverted index is used to index centroids $\{B_k\}_{k=1}^K$ (see Fig. 5.3).

Given a set of centroids $\{B_k\}_{k=1}^K$ and a query $Q_t = \left[q^{[1]}, ..., q^{[D]}\right]$ (note: $q^{[m]}$ is a scalar) represented by polyVLAD, we aim to calculate Jaccard distance from Q_t to every centroid B_k. We use an inverted index: $\{\mathcal{I}_0, ..., \mathcal{I}_{W-1}\}$, whose size is $W = 2\mathbf{d}LM = 2\mathbf{d}D$. Each $B_k = \left[b_k^{[1]}, ..., b_k^{[D]}\right]$ (note: $b_k^{[m]}$ is a scalar) is indexed:

$$\forall m = \{1, ..., D\},\ \mathcal{I}_{2\mathbf{d}(m-1)+b_k^{[m]}} = \mathcal{I}_{2\mathbf{d}(m-1)+b_k^{[m]}} \cup k \tag{5.26}$$

Fig. 5.5b is an example of inverted index. In the online inference, given a query: $Q_t = \left[q^{[1]}, ..., q^{[D]}\right]$. We first calculate similarity score between Q_t to every B_k:

- Initialize similarity score $\mathbf{S}(Q_t, B_k) = 0$ for every B_k
- For each $m \in \{1, ..., D\}$: $\mathbf{S}(Q_t, B_k) \leftarrow \mathbf{S}(Q_t, B_k) + 1$, for every $k \in \mathcal{I}_{2\mathbf{d}(m-1)+q^{[m]}}$

Jaccard distances are then computed: $d(Q, B_k) = 1 - \frac{\mathbf{S}(Q,B_k)}{D}$ for every B_k. Finally, $\mathbf{o}_{t,bg}$ is calculated from (5.13) and (5.15).

5.4.4 Complexity analysis

Assume that polytope vertex indices from $\{0, ..., 2\mathbf{d} - 1\}$ are distributed uniformly in each dimension m of $\{B_k\}_{k=1}^K$. The complexity of computing distance from Q_t to every B_k is $O(\frac{KD}{2\mathbf{d}})$, i.e., $2\mathbf{d}$ times faster than linear scan $O(KD)$. For some common local features, $\mathbf{d}$ is usually larger than 100 (e.g., $\mathbf{d} = 128$ for SIFT feature).

5.5 Experiments

5.5.1 Datasets

2 datasets are used:

- Oxford RobotCar [111]: 4 sequences ("26/06/2014, 09:24:58", "26/06/2014, 08:53:56", "23/06/ 2014, 15:41:25", and "23/06/2014, 15:36:04") are used and briefly referred to S1, S2, S3 and S4. The traversal distance in each sequence is about 1km.
- St Lucia [62]: 5 sequences ("10/09/2009, 08:45", "10/09/2009,10:00", "19/08/2009, 08:45", "21/08/2009, 10:10", and "21/08/2009, 12:10") are used and briefly referred to A1, A2, A3, A4 and A5. The traversal distance in each sequence is about 17km.

5.5.2 Implementation

We densely extract SIFT features at 4 scales with 16, 24, 32, 40-pixel region width, over a grid of 2-pixel spacing, and use vocabulary of size 128 for polyVLAD. 8 rotation matrices are sampled, resulting in each image being encoded by 8192 bits. We set number of clusters $K = 700$ and 7000 respectively for Oxford RobotCar and St Lucia. Other parameters are set as in Sec. 5.3.5. All experiments are conducted on a computer with Intel Core i7@3.4GHz (8 cores) and 16GB RAM, where we simulate PS as the hard disk and AM as the RAM.

5.5.3 Evaluation

Our method (denoted as `polyVLAD+HM`^{4}) is compared against baseline, i.e., `SPR` (described in chapter 4), DenseVLAD [167] and NetVLAD [7] incorporated to HMM framework (denoted as `DenseVLAD+HMM` and `NetVLAD+HMM`). A query image is regarded correctly localized if the distance from the matched place to the ground truth position is less than a threshold varied from 1m to 25m. We also measure memory (RAM) needed to store topological map and database for inference, as well as the inference time.

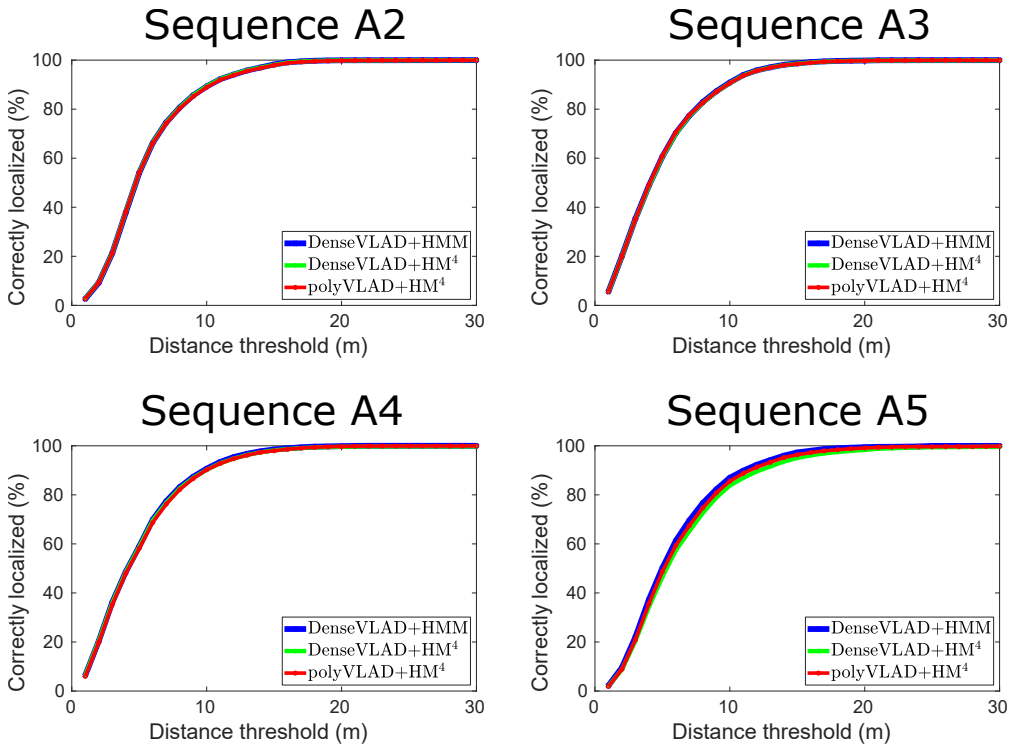
To simulate the scenario of continuously accumulating image data. In Oxford Robotcar, S1 forms the initial database; S2, S3 and S4 are sequentially used as the query sequences. In St Lucia, A1 is selected as initial database; A2, A3, A4 and A5 are sequentially used as the query sequences. After each query sequence finishes, we update to the database based on VPR decision. For `DenseVLAD+HMM` and `NetVLAD+HMM`, we update the topological map as described in Sec. 5.3.4, and append image representations to database.

5.5.4 Ablation study

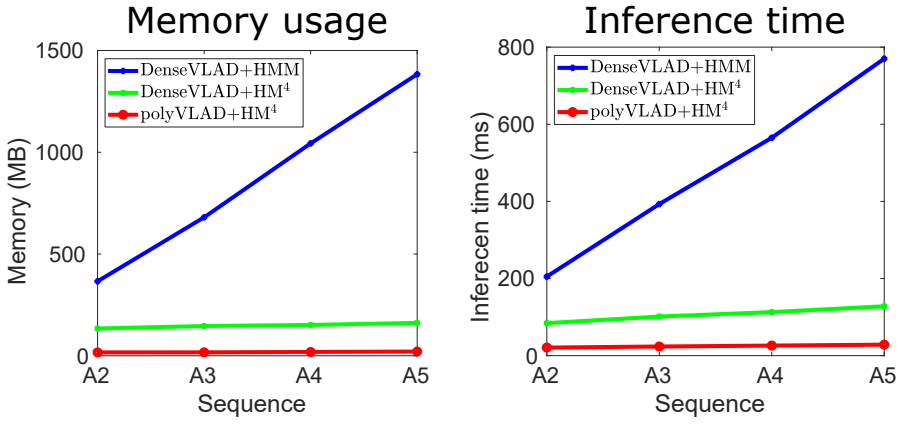
This experiment investigates the contribution of HM^4 and polyVLAD to the system. To this end, we use DenseVLAD incorporated with HM^4 (denoted as `DenseVLAD+HM`^{4}), which is then compared to `DenseVLAD+HMM` and `polyVLAD+HM`^{4}. Dataset St Lucia is used in this experiment

Fig. 5.6a demonstrates a comparable localization accuracy among these methods. More importantly, as shown in Fig. 5.6b, regardless the image representation method, HM^4 offers the scalability in terms of both memory usage and inference time (i.e., sublinear growth), while HMM shows a linear growth. With the use of polyVLAD, the memory usage and inference time is significantly reduced by a constant factor. This experimental result is also consistent to the complexity analysis in section 5.3.6.

In practice, if 1 sequence is collected every day to update the map, with only 5 sequences ($\sim$ operations in 5 days), the memory footprint & inference time of HMM increase about $3\times$ and $4\times$ respectively. Hence, the algorithms with linear complexity prevent their applicability in embedded systems with a small hardware capability (e.g., self-driving cars), which also must concurrently process many other tasks. Therefore, optimizing the memory & time complexity (as shown by `polyVLAD+HM`^{4}) in every task is crucial.



(a)



(b)

Fig. 5.6. Ablation study: **(a)** localization accuracy **(b)** growth in memory usage (in MB) & inference time (millisecond/image)

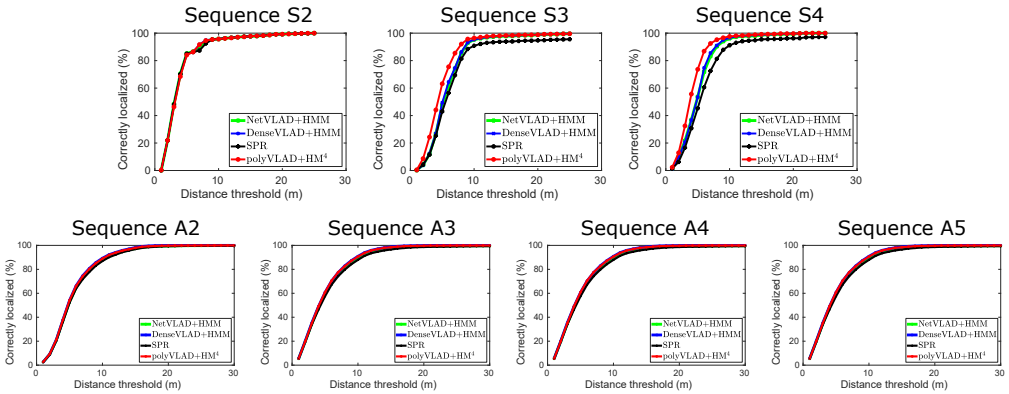
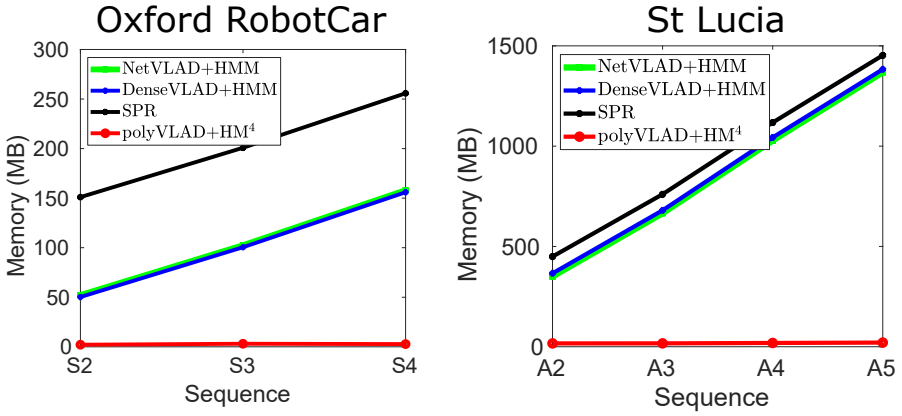
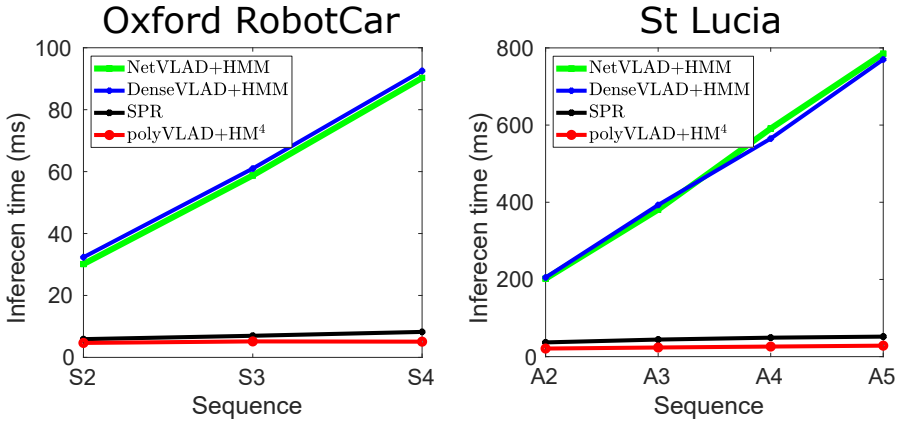


Fig. 5.7. Localization accuracy on Oxford RobotCar (top) and St Lucia (bottom)



(a) Growth in memory usage



(b) Growth in inference time

Fig. 5.8. The growth in **(a)** memory usage (in MB), and **(b)** inference time (millisecond/image).

5.5.5 Comparison to baseline

The growth in memory consumption and inference time is shown in Fig. 5.8, where the average value is reported, and the deviation is insignificant. The result is consistent in both Oxford RobotCar and St Lucia. Specifically, the memory usage and inference time of `NetVLAD+HMM` and `DenseVLAD+HMM` linearly grows when we update the new image sequences (Figs. 5.8a and 5.8b). For `SPR`, although its inference time is scalable (Fig. 5.8b), its memory usage still linearly grows (Fig. 5.8a) (Note that `SPR` requires more memory than `NetVLAD+HMM` and `DenseVLAD+HMM` because it also stores K-means tree for retrieval). By contrast, `polyVLAD+HM`^{4} shows a sublinear growth in both memory usage and inference time (Fig. 5.8)

Regarding the localization accuracy (Fig. 5.7), all methods share a comparable performance in general. `polyVLAD+HM`^{4} is slightly better than other methods in Oxford RobotCar.

5.6 Conclusion

In reality, robots must concurrently process many tasks (e.g., segmentation, recognition, localization, etc). Due to the limitation of embedded hardware, it requires those tasks must have the scalable capability. In VPR, as robots continuously move and accumulate data, the memory needed for VPR increases unboundedly. This motivates us to examine **Research question 3** *Can we design a lightweight and scalable VPR system?* To address this research question, the two following sub-question are concretely addressed

RQ3.1 *Can we achieve sub-linear space-time complexity?*

We propose HM^4 (HMM with Memory Management) handles the scalable problem through looking into the future and fetching just the right amount data needed to make VPR decisions. A coarse representation that summarizes the topological map is periodically updated (e.g., weekly or monthly) to be adapted for new changes of environment. We show that the size of coarse representation remains constant provided that the coverage area does not change. Those mechanisms ensure a scalable time and memory needed for inference.

RQ3.2 *Can we find a compact yet informative image representation?*

Inspired from LSH [5], polytope VLAD (polyVLAD) is derived to inherit theoretical guarantee of LSH that close data points have high probability to be hashed into a same code. This characteristic is important in robotic scenarios because robots frequently

encounter unseen samples (see Sec. 2.4), while the learning methods do not ensure to reliably work in this case.

The combination between HM^4 and polyVLAD yields a lightweight and scalable VPR method for the life-long operation, i.e., sequentially and continuously updating the map with new image sequences. In practice, our method can be deployed locally on an embedded hardware, in which the passive storage needs to be updated after few years. Another option is to implement our approach on a server-client architecture, in which the VPR task is shared between client and server. Also, this option leverages the nearly infinite storage of cloud and high-speed connection to transmit data between server and client.

Chapter 6

Learning to Predict Repeatability of Interest Points

This chapter is based on the content of following conference paper

- Anh-Dzung Doan, Daniyar Turmukhambetov, Yasir Latif, Tat-Jun Chin, and Soohyun Bae. Learning to Predict Repeatability of Interest Points. In *International Conference on Robotics and Automation (ICRA)*, 2021.

  (accepted)

Statement of Authorship

Title of Paper	Learning to Predict Repeatability of Interest Points
Publication Status	☐ Published ☑ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Anh-Dzung Doan, Daniyar Turmukhambetov, Yasir Latif, Tat-Jun Chin, Soohyun Bae, *"Learning to Predict Repeatability of Interest Points"*, IEEE International Conference on Robotics and Automation (ICRA), 2021

Principal Author

Name of Principal Author (Candidate)	Anh-Dzung Doan		
Contribution to the Paper	Developed the method, conducted the experiments, and wrote the paper.		
Overall percentage (%)	60		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	18 February 2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Daniyar Turmukhambetov		
Contribution to the Paper	Supervised the development of the method, provided the suggestions, and revised the paper.		
Signature		Date	19/02/2021

Name of Co-Author	Yasir Latif		
Contribution to the Paper	Provided suggestions and proofread the paper		
Signature		Date	23-02-2021

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Provided suggestions and proofread the paper		
Signature		Date	23 Feb 2021

Name of Co-Author	Soohyun Bae		
Contribution to the Paper	Proposed the general direction, supervised the development of the method, provided the suggestions, and revised the paper.		
Signature		Date	2/18/2021

6.1 Introduction and research questions

Chapter 5 has shown that it is vital to coarsely represent the full map to prevent HMM inference from getting lost. If VPR task requires to recover 6 DoF pose, it will need the topometric map with 3D points available. Coarsely representing a full map, which is so-called map summarization [45, 121, 19, 108], also allows the compressed map can be stored on the onboard hardware. A general strategy of existing map summarization techniques is to sample 3D points that are likely helpful for VPR as claimed by pre-defined criteria, that include: being re-detected frequently [46, 45, 25], low descriptor variance [45], discriminative descriptor that easily gets matched [47, 24], stable physical appearance over long period of time [47]. However, none of existing criteria take into account the run-time of VPR as a constraint for summarizing the map. This motivates us to ask:

Research question 4 *Can we summarize the map according to the run-time of VPR?*

It is well-known 3D points are triangulated from interest points detected from images. A criterion, namely *repeatability*, was introduced by Schmid et al. [150] to characterize local interest points. Since then, a considerable effort has been made to find a local interest point detector robust against appearance changes [103, 136, 44], whose basic strategy is to detect interest points with high repeatability scores. However, these approaches make a strong assumption, i.e., the interest points must be *repeatable* regardless of environmental changes w.r.t time span. In practice, according to [161], this assumption does not hold because environmental changes (including physical, weather, and illumination changes) affect to the physical appearance of 3D points. Consequently, the experiment in [161] shows a decay in terms of the feature matching performance, which is crucially caused by the degradation of repeatability.

Another strategy is to maintain only interest points stable for visual localization (VL), and eliminates interest points unable to be re-detected [47]. Nevertheless, these interest points likely disappear in a period of time but reappear after that. Hence, eliminating them can be seen as an extreme operation. Another issue is treating interest points frequently re-detected as stable features introduces a failure point, e.g., interest points on trees and road can be regularly re-detected but those are ambiguous points for VL.

Therefore, instead of finding permanently repeatable interest points for mapping, we investigate two following sub-questions

RQ4.1 *Can we predict if an interest point is repeatable at a particular time period?*

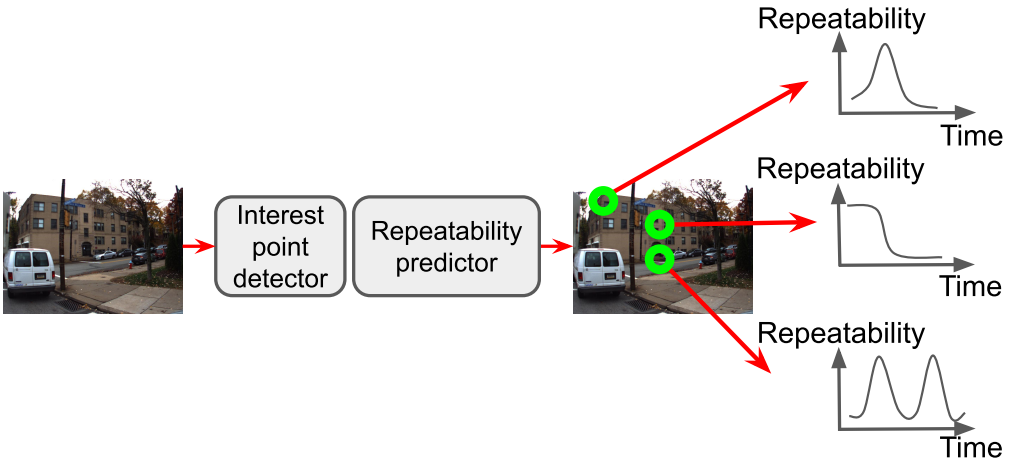


Fig. 6.1. Given a particular interest point detector, we wish to predict the repeatability of interest points (denoted as ⊙) as a function of time.

The idea is illustrated in Fig. 6.1, where given a set of interest points detected by a feature detector (e.g., SIFT [103], D2-Net [44] or R2D2 [136]), we predict the repeatability of interest points as functions of time. To this end, a repeatability predictor (RP) is formulated as a deep neural network regressor, which receives an interest point as the input, and outputs a vector of repeatability scores approximating the repeatability function. The ground truth data is built from images periodically captured at several viewpoints (e.g., building, house) over time. Because the number of repeatable interest points are the upper bound of number of correspondences found via feature matching, we derive the repeatability score at a particular timestamp from the matchability.

RQ4.2 *How to use the RP for map summarization?*

The main challenge in map summarization for VPR is which 3D points should be sampled. If this issue is not addressed properly, it will lead to a severe degradation of VPR accuracy. As the RP is capable of predicting which 3D points are potentially repeatable at a specific timestamp. It motives us to leverage its characteristics to the map summarization, i.e., selecting 3D points likely useful for VPR with respect to the time constraint—this, as a result, can alleviate the accuracy deterioration from the map summarization.

From the experiment, we observe that regressing repeatability function of unseen interest points is feasible, and gain more insight about the problem. Additionally, RP shows its great potential in VL through its superior to a standard competitor in the map summarization.

The rest of the chapter is organized as follows: Sec. 6.2 presents our learning process to train the RP. Next, the application of the RP is described in Sec. 6.3. The experimental

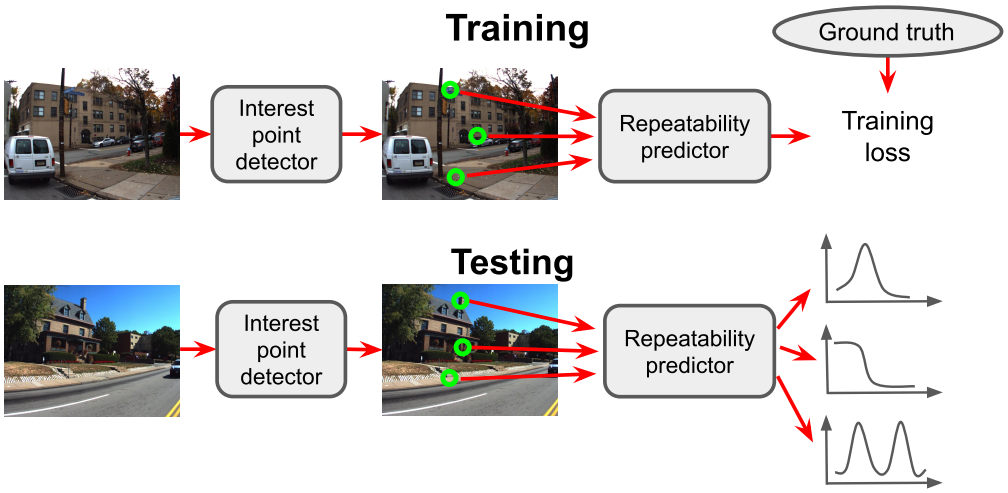


Fig. 6.2. The proposed system for learning the repeatability of the interest points.

results, that demonstrate the great potential of VPR and its application, are provided in Sec. 6.4. Finally, the conclusion of this chapter is drawn in Sec. 6.5.

6.2 Learning repeatability predictor

The proposed system for learning the repeatability of the interest points is shown in Fig. 6.2. Given an image, a feature detector extracts its interest points, each interest point is fed to the RP. During the predictor training, the RP is supervised by the ground truth repeatability function. The trained model is then utilized to predict the repeatability function of every interest point for the given test images. In the following sections, we will describe how we parameterize repeatability functions (Sec. 6.2.1), formulate RP as a deep neural network (Sec. 6.2.2), define training loss (Sec. 6.2.3), and build the ground truth (Sec. 6.2.4).

6.2.1 Parameterizing repeatability functions

As shown in Fig. 6.3, we represent a repeatability function over time with a discrete set of line segments. So, time is discretized into timestamps, then line segments are used to approximate the function between two consecutive timestamps. At each timestamp t_j, a repeatability score is stored as one element of a vector. Finally, we obtain the repeatability vector which approximates the repeatability function for the given time window.

Let $\triangle t$ be the interval between two consecutive timestamps and T be the number of timestamps. We can define T as one of

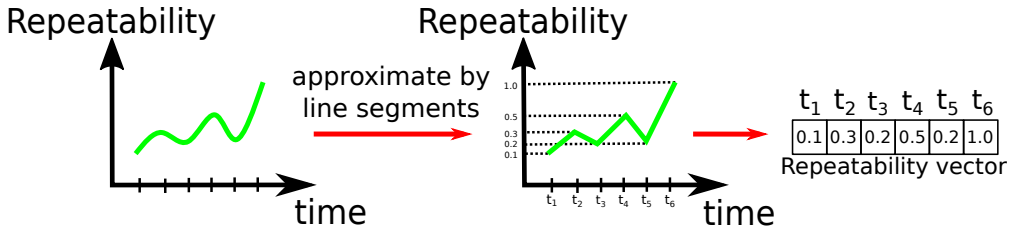


Fig. 6.3. Approximating repeatability function.

- $T = 24$ hours/day, $\triangle t = 1$ hour, then: $t_1 =$ 0:00, $t_2 =$ 1:00, ..., $t_{24} =$ 23:00.
- $T = 365$ days/year, $\triangle t = 1$ day, then: $t_1 =$ January 01, $t_2 =$ January 02, ..., $t_{365} =$ December 31.
- $T = 8760$ (= 24×365) hours/year, $\triangle t = 1$ hour, then: $t_1 =$ 0:00, January 01, $t_2 =$ 1:00, January 02, ..., $t_{8760} =$ 23:00, December 31.

6.2.2 Repeatability predictor

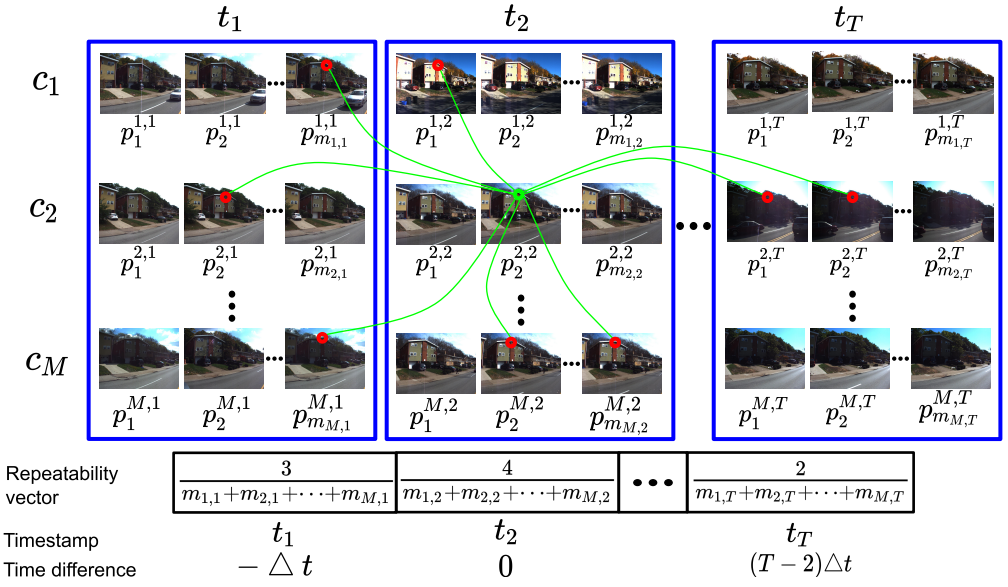


Fig. 6.4. Constructing repeatability vector for the interest point ○: the repeatability score at timestamp t_j is computed as $\frac{\text{number of inliers}}{\text{number of images}}$ within the corresponding block □ of t_j.

A deep neural network architecture, as described in Fig. 6.5, is used to construct the repeatability predictor (RP). For each interest point x in an image, the input of RP consists of the interest point's coordinate ($\in \mathbb{R}^2$), local patch ($\mathbb{R}^{64\times64\times3}$) centered at the interest point, and the time ($\mathbb{R}^d$) in which the image is captured, where:

- $d = 2$ if we represent times of day (hour and minute) or days of year (date and month)

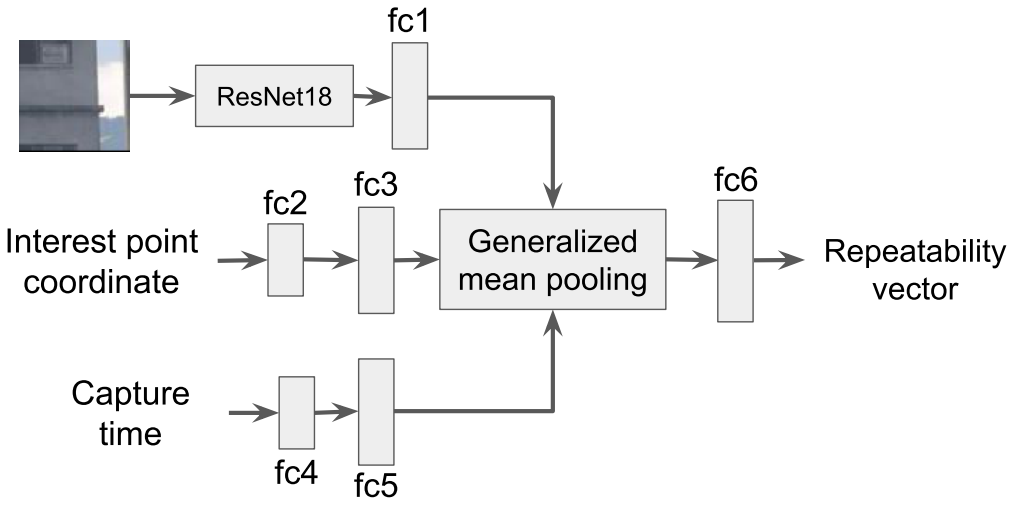


Fig. 6.5. Architecture of repeatability predictor.

- $d = 4$ if we represent hours of year (hour, minute, date and month).

The information of fully connected layers are: fc1 (64, relu), fc2 (32, relu), fc3 (64, relu), fc4 (32, relu), fc5 (64, relu), fc6 (D, sigmoid), where D is the dimensionality of repeatability vector. Outputs of fc1, fc3, and fc5 (denoted as z_1, z_2, and z_3) are pooled by the generalized mean pooling [135]:

$$\mathbf{z} = \left[\frac{1}{3}\left(z_1^p + z_2^p + z_3^p\right)\right]^{\frac{1}{p}}, \tag{6.1}$$

where p is a learnable parameter. If $p \to \infty$, it will become the max pooling; if $p = 1$, it will become the mean pooling.

6.2.3 Training loss

Given N interest points $\left\{x_i, y_i\right\}_{i=1}^{N}$ and T timestamps, for a given interest point x_i, $y_i = \left[y_i^1, y_i^2, \ldots, y_i^T\right]$ denotes its corresponding ground truth, where $y_i^j \in \mathbb{R}$ is the repeatability score at timestamp t_j. Similarly, $\hat{y}_i = \left[\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^T\right]$ denotes the corresponding prediction using RP. The neural network is trained by mean squared error: $\frac{1}{N}\sum_{i=1}^{N} ||y_i - \hat{y}_i||_2^2$

6.2.4 Constructing ground truth

The network with the aforementioned loss function is trained with the ground truth y for the interest point x. Fig. 6.4 illustrates the way of generating y. For a given set of

images viewing the local area (e.g., building, house) captured at $t_1, t_2, \ldots, t_T$, the images are grouped into difference cycles $c_1, c_2, \ldots, c_M$, where one cycle corresponds to one day (for T=24 hours/day) or one year (for T=365 days/year or T=8760 hours/year). Each image I_i is associated with a camera pose $p_i^{k,j}$ at cycle c_k and timestamp t_j. We also denote $m_{k,j}$ as the number of camera poses (images) at cycle c_k and timestamp t_j.

For each interest point x, we conduct feature matching to each remaining image as follows:

1. Find the closest interest point using Euclidean distance between feature descriptors
2. Verify if the closest interest point satisfies ratio test [103]
3. Conduct geometric verification to check if the matching pair is an outlier
4. Perform Structure from motion (SfM) and accept the matching pair if it can form a 3D point.

After obtaining set of inlier correspondences, the repeatability score y^j at timestamp t_j is calculated as follows:

$$y^j = \frac{\#\text{ of inliers}}{m_{1,j} + m_{2,j} + \cdots + m_{M,j}}. \tag{6.2}$$

Note that if the interest point x belongs to the image captured at timestamp t_j, we count itself to the number of inliers.

Also, our representation in the repeatability function can be seen as *time difference*, e.g., in Fig. 6.4, let $T = 24$ hours/day and $\triangle t = 1$ hour, interest point x (green point) belongs to the timestamp t_2 (= 1:00am), hence the repeatability scores at y^1 and y^T are respectively $\triangle t$ (= 1 hour) before 1:00am and $(T-2) \triangle t$ (= 22 hours) after 1:00am.

6.3 Application in map summarization

One of the potential applications that benefits from the trained repeatability predictor is map summarization for visual localization (VL). The pipeline is described in Fig. 6.6. Specifically, a full 3D map built by SfM is stored in the server. Given the current timestamp, using RP, the map summarization is performed to obtain a summary map, which is then transmitted to the client and used for online localization. If we set $T = 24$ hours/day and $\triangle t = 1$ hour, the summary map is updated on the hourly basis.

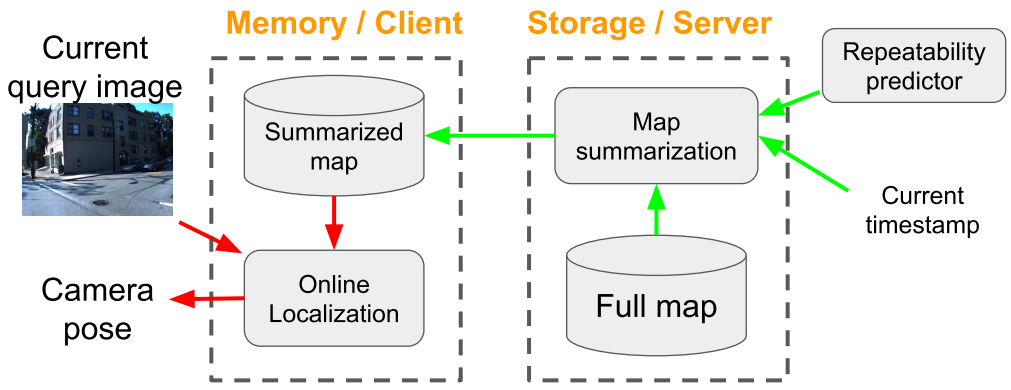


Fig. 6.6. Our proposed VL pipeline, where → is the online operation, and → is the periodic operation, which will be in hourly basis if number of timestamps are $T = 24$ hours/day, and $\triangle t = 1$ hour.

6.3.1 Map summarization

6.3.1.1 3D point representation

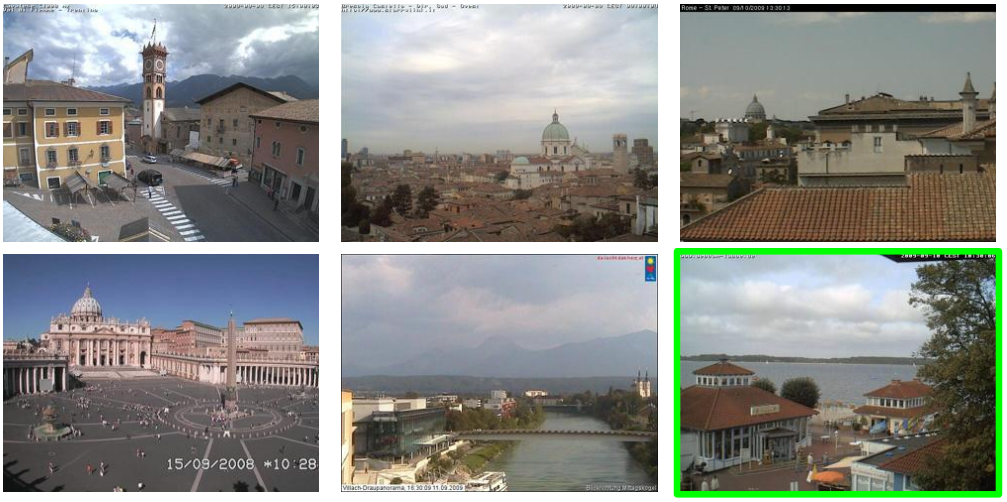
For every 3D point, we predict the repeatability functions for all interest points corresponding to the 3D point. Now, the repeatability of the 3D point is computed by the mean of all the repeatability functions for the interest points. Similarly, the mean of feature descriptors of 2D interest points is also used to represent the descriptor of the 3D point. This representation offers a compact way of storing the 3D map by cutting down the memory consumption on the descriptors and repeatabilities of interest points.

6.3.1.2 Sampling 3D points

Firstly, we partition the 3D map into several parts, and then individually prune 3D points in each part. This step prevents us from over-pruning 3D points in a particular part of the map, which would impair the localization accuracy in that part. So, for each part of the map, we compute repeatability score y_i^j of every 3D point $\mathbf{p}_i$ at the query timestamp t_j. Finally, we can remove 3D points with lowest repeatability score according to the pruning ratio.

6.3.2 Online visual localization

Given a query image, we firstly retrieve K-nearest images in the database. Note that all images are represented by NetVLAD [7], and the database images are indexed by KD-tree. From the summary map, we select 3D points observed by retrieved images as candidate 3D points. Then, 2D-3D correspondences between interest points of query image and candidate 3D points are established through comparing their feature descriptors



(a)

(b)

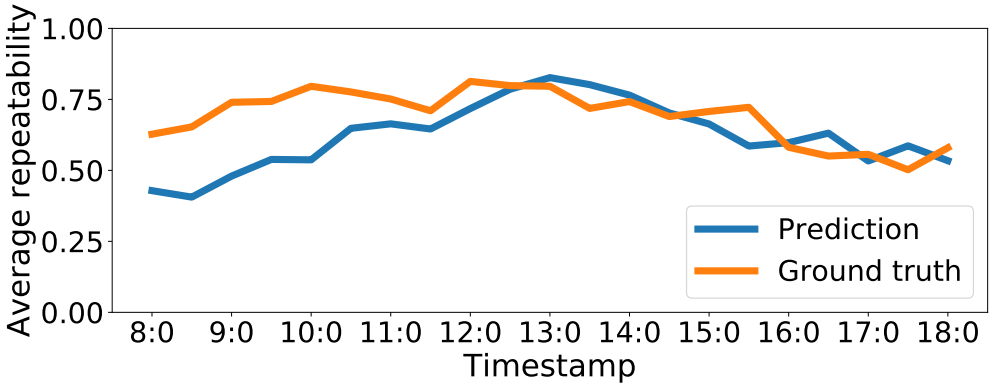
Fig. 6.7. Samples from (a) Webcam Clip Art and (b) Extended CMU Seasons datasets. Testing viewpoints are denoted as ▭.

and the ratio test [103] (Note that candidate 3D points are also indexed by KD-tree). Finally, the 6 DoF camera pose of query image is estimated via solving Perspective-n-Point with RANSAC.

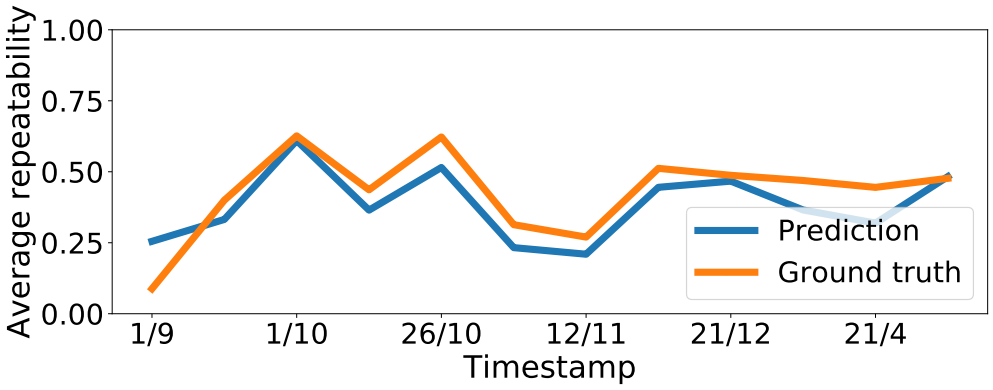
6.4 Experiments

6.4.1 Predicting repeatability function

This section investigates the performance of our algorithm on several datasets, including Webcam Clip Art dataset [94] and Extended CMU Seasons dataset [145]. We use SIFT detector & descriptor [103] for the experiments.



(a)



(b)

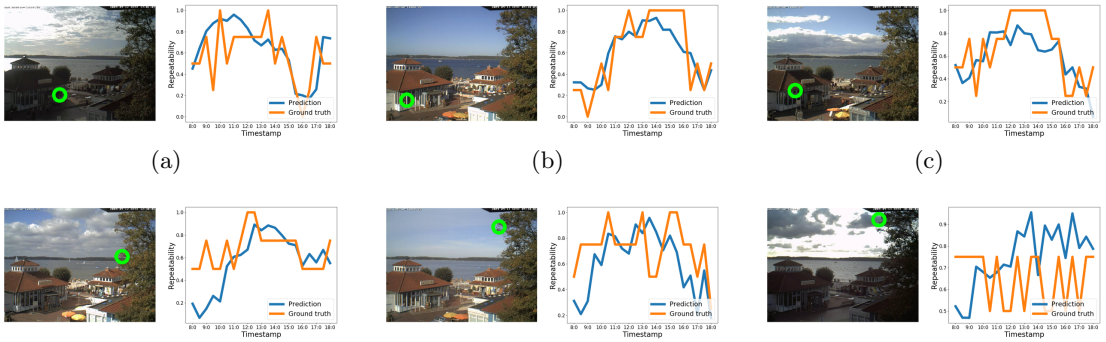
Fig. 6.8. Average repeatability over all testing samples on (a) Webcam Clip Art and (b) Extended CMU Seasons.

6.4.1.1 Datasets

Two datasets are used:

- Webcam Clip Art [94] has 54 difference viewpoints captured by webcam cameras during several years, and each viewpoint has about 10,000 images. Among the urban viewpoints, we manually select 8 viewpoints of buildings or houses, which are split to 7 training and 1 testing viewpoints (see samples in Fig. 6.7a).

  We set the number of timestamps to $T = 21$ hours/day and $\triangle t = 30$ minute between every consecutive timestamps, i.e., $t_1 = 8{:}00$, $t_2 = 8{:}30$, ..., $t_{21} = 18{:}00$. The number of cycles is $M = 4$ days. Because the webcam cameras are almost static, we make a minor change in the matching procedure (see Sec. 6.2.4): in the last step, if the absolute difference between two pixel coordinates is < 5 pixel, the matching pair is accepted as an inlier correspondence. Finally, we obtain 72,160 training and 4,217 testing interest points.



(a) (b) (c)

(d) (e) (f)

Fig. 6.9. Examples of RP predictions on testing set of Webcam Clip Art, where ○ is the interest point.

- Extended CMU Seasons dataset [145] is split to 25 separate regions with available ground truth 6 DoF camera poses from SfM. As the dataset is collected in the form of continuous trajectory, we group the images seeing the same viewpoint using available ground truth camera poses. In particular, region numbers 6, 7 and 9 are used for training; region 8 is used for testing. After grouping viewpoints, we have 73 training viewpoints and 17 testing viewpoint (see samples in Fig. 6.7b). We set the number of timestamps to $T = 12$ days/year and $\triangle t$ varies from 1 to 13 weeks, i.e., t_1 = March 04, t_2 = April 21, t_3 = July 28, t_4 = September 01, t_5 = September 15 , t_6 = October 01, t_7 = October 19, t_8 = October 26, t_9 = November 03, t_{10} = November 12, t_{11} = November 22, and t_{12} = December 21. The number of cycle is M=1 year. In the feature matching in Sec. 6.2.4, due to ground truth camera poses available, we simply perform triangulation instead of the full SfM, resulting in 45,870 training and 15,856 testing interest points.

6.4.1.2 Results

Fig. 6.8a shows the average repeatability function over all testing samples on Webcam Clip Art. Generally, in both ground truth and prediction curves, the repeatability score increases from the morning to noon, and gradually decreases as it gets close to the night time. This trend can also be seen in individual testing samples (see Fig. 6.9a-e). However, for testing sample with unclear ground truth trend (see Fig. 6.9f), RP struggles to learn its repeatability.

Fig. 6.8b shows the average repeatability function over all testing samples on Extended CMU Seasons. It is clear that the ground truth and the prediction curves share a similar trend, i.e., the repeatability score is high in spring, summer and autumn while it is low during winter. This trend mainly comes from interest points of discriminative objects,

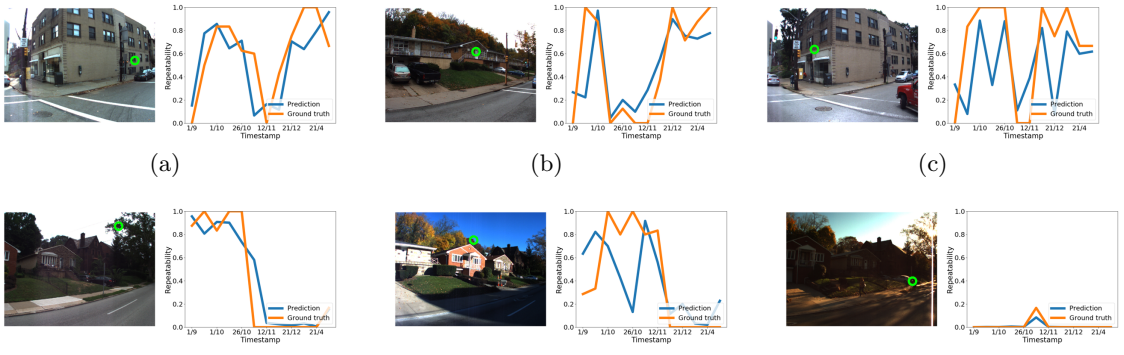


Fig. 6.10. Examples of RP predictions on testing set of Extended CMU Seasons, where ○ is the interest point.

e.g., buildings, houses (see Fig. 6.11), which RP can perform prediction reasonably. The concrete good examples are shown in Fig. 6.10a-b, but RP also fails in few examples (e.g., Fig. 6.10c).

For tree interest points, the basic trend is their repeatability drastically drops after winter (due to no leaves after winter), which RP often fails in predicting (see Fig. 6.11). Fig. 6.10d and Fig. 6.10e show the good and bad particular examples.

Interest points of dynamic objects (e.g., cars) are mostly repeatable at the capture time, i.e., the input timestamp (see Fig. 6.10f), yielding a good prediction in general (see Fig. 6.11).

For background interest points (e.g., road, sky), their repeatability shows an unclear trend, thus RP struggles to perform a reasonable performance (see Fig. 6.11).

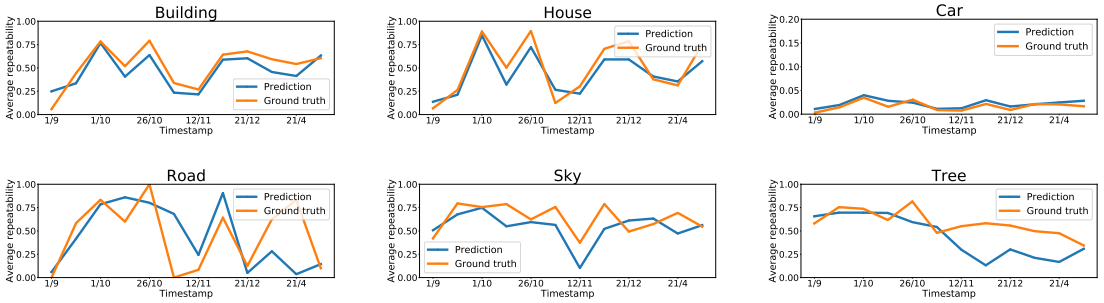


Fig. 6.11. Average repeatability in each category over all testing samples on Extended CMU Seasons.

Based on the comprehensive experiment, we empirically observe several challenges in this setup: 1) as shown in [161], sun direction is an important factor affecting the appearance of physical 3D points, which should be considered as an input, 2) Fig. 6.11 shows the performance of RP varies according to types of interest points. It suggests that the

TABLE 6.1: Sequences in Oxford RobotCar used to train RP.

Sequence	Timestamp
2015-02-13-09-16-26	$t_1 = 9{:}00$
2015-07-10-10-01-59	$t_2 = 10{:}00$
2015-03-17-11-08-44	$t_3 = 11{:}00$
2014-11-28-12-07-13	$t_4 = 12{:}00$
2014-11-18-13-20-12	$t_5 = 13{:}00$
2015-07-29-13-09-26	$t_6 = 14{:}00$
2015-05-19-14-06-38	$t_7 = 15{:}00$
2015-08-13-16-02-58	$t_8 = 16{:}00$
2015-07-14-16-17-39	$t_9 = 17{:}00$

predictor may benefit from semantics, and 3) another factor which greatly influences to the prediction result is the weather at the query time, e.g., at 12pm with a rainy and cloudy condition, the appearance might be darker than that at 5pm with a sunny and clear condition.

6.4.2 Map summarization for VL

6.4.2.1 Datasets

Oxford RobotCar dataset [111] is used to train the proposed network. Specifically, we set number of timestamps to $T = 9$ hours/day (from 9:00 to 17:00), $\triangle t = 1$ hour, and number of cycle $M = 1$. More detailed information of the Oxford RobotCar sequences is described in Table 6.1. To generate separate viewpoints, we manually select 23 viewpoints by taking the latitude and longitude, and the yaw angle of the vehicle seeing them, which, for convenience, are now denoted as *viewpoint latitude-longitude*, and *viewpoint yaw angle*. Afterward, using the exact ground truth pose of the Oxford RobotCar [110], we find images close to each viewpoint latitude-longitude $\leq$ 5m. To ensure images observing the same viewpoint, we only keep images whose yaw angle close to viewpoint yaw angle $\leq 45°$. Finally, SfM is conducted in each viewpoint; and only 18 viewpoints which have all images registered are retained for training RP.

For VL dataset, we select region 8 of Extended CMU Seasons [145]. Its 3D point cloud is used as the 3D map, and query images are captured at 04 March 2011.

6.4.2.2 Results

In Extended CMU Seasons, there is a capture timestamp associated with every query image. Utilizing that timestamp and RP trained on Oxford RobotCar, we summarize

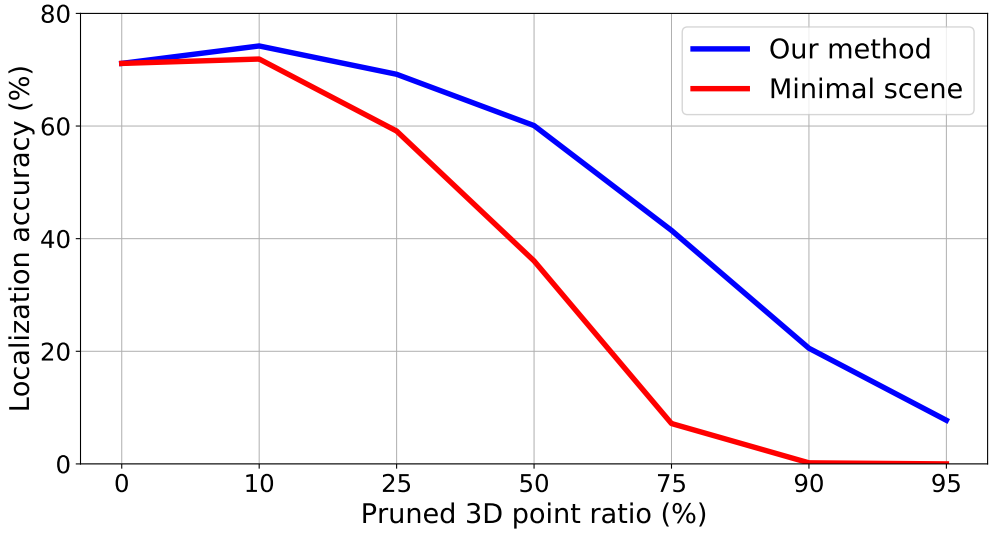


Fig. 6.12. Comparison of localization accuracy over the percentage of pruned 3D points

the 3D map as described in Sec. 6.3.1. In Fig. 6.12, where the localization accuracy is the percentage of query images localized $< 5\text{m}, 10°$, our method shows a superior accuracy to *Minimal scene* [25] in every pruned 3D point ratio. The reason is *Minimal scene* [25] does not consider the matching potential of 3D points in VL when selecting them, while our method regards the query time to sample highly repeatable 3D points, leading to a better performance.

6.5 Conclusion

In this chapter, we propose a novel map summarization scheme for VPR through answering

Research question 4 *Can we summarize the map according to the run-time of VPR?*

Existing map summarization methods rely on criteria derived from repeatability and matchability characteristics of interest points, while time span should be seriously considered as an additional constraint in the process of sampling useful 3D points for VPR. Specifically, this chapter explores following sub-questions

RQ4.1 *Can we predict if an interest point is repeatable at a particular time period?*

We propose a learning procedure that trains a repeatability predictor (RP) with the aim of predicting the repeatability of interest points as a function of time. Through a comprehensive experiment, we show that the RP is able to predict the repeatability

of an unseen interest point, also provide an insight analysis regarding this problem. It further suggests future works for this approach

RQ4.2 *How to use the RP for map summarization?*

The application of RP in summarizing topometric map for VPR is presented. The map summarization is periodically invoked to select 3D points that are likely repeatable at the VPR testing time. The experimental results demonstrate its great potential in the large-scale VPR.

Chapter 7

Conclusions

In this final chapter, we summarize our key contributions and also suggest possible research directions for future works.

7.1 Contributions

As mobile robots continuously operate in the environment, the visual input is usually in the form of videos. Chapter 3 revisits Bayes Filter to effectively exploit the temporal smoothness of image sequences. Also, G2D—an open-source software that assists researchers in collecting synthetic data for VPR is presented. As alluded in chapters 1 and 2, the continuous data collection is necessary for the life-long operation, which ensures the VPR system is able to observe as many appearance variations as possible. This strategy demands a VPR that is scalable, i.e., the computational effort and memory consumption must grow slowly or even remain constant with database size. Chapter 4 presents an HMM-based technique, whose map can be efficiently updated and compressed when "absorbing" new data. However, its observation model suffers from a linear increase in the space complexity, thus Chapter 5 proposes to couple HMM inference and two-tiered memory management—active memory (AM) and passive storage (PS). The basic idea is to use the forward temporal propagation to determine most promising places that will be transferred from PS to AM each HMM iteration. To avoid from getting lost, a coarse representation of the full map is stored on AM to approximate the belief on remaining places. In addition, inspired from LSH [5], chapter 4 proposes polytope VLAD (polyVLAD) that compresses the image representation to the compact code. The combination of HM^4 and polyVLAD offer a lightweight and scalable VPR system. In VPR with topometric map, the most expensive memory footprint is from storing 3D points, hence Chapter 6 presents a novel map summarization scheme, which

uses a repeatability predictor to predict the repeatability of an interest point as a function of time. Our technique will sample 3D points that are potentially repeatable at the VPR run-time. The experiment demonstrates that considering time constraint can alleviate the degradation of VPR accuracy from the map summarization.

7.2 Future directions

Based on answers provided to address the research questions in this thesis, we identify three possible research directions

1. *Investigating the robustness of Bayesian techniques against adversarial attacks.*

   In some robotic applications, a failure of operation might lead to a very serious consequence, for example, if a self-driving car makes an accident, it will risk many human lives. It is well-known that current state-of-the-art artificial intelligent (AI) technology is highly sensitive to adversarial attacks [4, 2]. It introduces a "Achilles heel" that can possibly be exploited by hackers. This is because most of existing AI technology rely on deep neural networks, which perform the point estimate in their inference, hence they are easily fooled by samples out of their training set [65, 130]. On the other hand, Bayesian techniques model every factors via probabilistic distributions, which can be used to quantify the uncertainty in their prediction. This characteristic makes the Bayesian framework become a potential approach in dealing with adversarial attacks [162]. Investigating VPR methods proposed in chapter 4 and chapter 5 in this direction will be a very interesting future work.

2. *Bridging the gap between simulation and reality.*

   Recently, deep learning methods significantly improves the performance of VPR [7, 27, 182]. However, it takes years and a considerable effort to collect sufficient training data for deep neural networks [111, 94, 145]. Fortunately, the development of graphic technology offers a feasibility in cost-effectively collecting hyper-realistic computer-generated imagery as well as a significant advancement in domain adaptation [30, 174, 55] has been made in literature. It suggests an interesting direction that we can completely train deep networks on simulation while effortlessly transferring its usage to real scenarios. This, as a result, can possibly reduce the cost of training and we can extensively test the VPR system before deploying it in reality.

3. *Investigating VPR in space*

   These days, space industry starts to pay attention to the development of AI technology through a number of competitions [1, 125]. Deploying robotic agents on

Moon or Mars needs to satisfy many constraints, e.g., the limitation in energy and computational resources, which clearly demand a VPR method that is scalable. In addition, as VPR system can only be tested on simulation, we need to ensure its performance does not change if deploying it in space. This challenge poses an attractive direction for future works.

Bibliography

[1] E. S. Agency. *Kelvins - ESA's Advanced Concepts Competition Website*. https://kelvins.esa.int/. Online; accessed 04 February 2021.

[2] N. Akhtar and A. Mian. "Threat of adversarial attacks on deep learning in computer vision: A survey". In: *IEEE Access* 6 (2018), pp. 14410–14430.

[3] A. Alahi, R. Ortiz, and P. Vandergheynst. "FREAK: Fast retina keypoint". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 510–517.

[4] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016).

[5] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. "Practical and optimal LSH for angular distance". In: *Advances in Neural Information Processing Systems* 28 (2015), pp. 1225–1233.

[6] A. Anoosheh, T. Sattler, R. Timofte, M. Pollefeys, and L. Van Gool. "Night-to-day image translation for retrieval-based localization". In: *IEEE International Conference on Robotics and Automation*. 2019, pp. 5958–5964.

[7] 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.

[8] R. Arandjelovic and A. Zisserman. "All about VLAD". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1578–1585.

[9] R. Arandjelovic and A. Zisserman. "Three things everyone should know to improve object retrieval". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2911–2918.

[10] A. Babenko and V. Lempitsky. "Tree quantization for large-scale similarity search and classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4240–4248.

[11] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5173–5182.

[12] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk. "Key. net: Keypoint detection by handcrafted and learned cnn filters". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5836–5844.

[13] H. Bay, T. Tuytelaars, and L. Van Gool. "Surf: Speeded up robust features". In: *European Conference on Computer Vision*. Springer. 2006, pp. 404–417.

[14] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. "Dsac-differentiable ransac for camera localization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6684–6692.

[15] E. Brachmann and C. Rother. "Learning less is more-6d camera localization via 3d surface regression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4654–4662.

[16] E. Brachmann and C. Rother. "Neural-guided RANSAC: Learning where to sample model hypotheses". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4322–4331.

[17] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. "Geometry-aware learning of maps for camera localization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2616–2625.

[18] M. A. Brubaker, A. Geiger, and R. Urtasun. "Lost! leveraging the crowd for probabilistic visual self-localization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3057–3064.

[19] M. Bürki, C. Cadena, I. Gilitschenski, R. Siegwart, and J. I. Nieto. "Appearance-based landmark selection for visual localization". In: *Journal of Field Robotics* 36.6 (2019), pp. 1041–1073.

[20] Á. P. Bustos, T.-J. Chin, A. Eriksson, and I. Reid. "Visual SLAM: Why bundle adjust?" In: *IEEE International Conference on Robotics and Automation*. 2019, pp. 2385–2391.

[21] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[22] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. "BRIEF: Computing a local binary descriptor very fast". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.7 (2011), pp. 1281–1298.

[23] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós. "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM". In: *arXiv preprint arXiv:2007.11898* (2020).

[24] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler. "Hybrid scene compression for visual localization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[25] S. Cao and N. Snavely. "Minimal scene descriptions from structure from motion models". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014.

[26] M. S. Charikar. "Similarity estimation techniques from rounding algorithms". In: *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*. 2002, pp. 380–388.

[27] 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: *IEEE International Conference on Robotics and Automation*. 2017, pp. 3223–3230.

[28] W. Churchill and P. Newman. "Experience-based navigation for long-term localisation". In: *The International Journal of Robotics Research* 32.14 (2013), pp. 1645–1661.

[29] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. "VidLoc: A deep spatio-temporal model for 6-dof video-clip relocalization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6856–6864.

[30] G. Csurka. "A comprehensive survey on domain adaptation for visual applications". In: *Domain adaptation in computer vision applications* (2017), pp. 1–35.

[31] M. Cummins and P. Newman. "Appearance-only SLAM at large scale with FAB-MAP 2.0". In: *The International Journal of Robotics Research* 30.9 (2011), pp. 1100–1123.

[32] M. Cummins and P. Newman. "FAB-MAP: Probabilistic localization and mapping in the space of appearance". In: *The International Journal of Robotics Research* 27.6 (2008), pp. 647–665.

[33] A. J. Davison. "FutureMapping: The computational structure of spatial AI systems". In: *arXiv preprint arXiv:1803.11288* (2018).

[34] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. "MonoSLAM: Real-time single camera SLAM". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067.

[35] F. Dayoub and T. Duckett. "An adaptive appearance-based map for long-term topological localization of mobile robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 3364–3369.

[36] D. DeTone, T. Malisiewicz, and A. Rabinovich. "Superpoint: Self-supervised interest point detection and description". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition workshops*. 2018, pp. 224–236.

[37] T.-T. Do, A.-D. Doan, and N.-M. Cheung. "Learning to hash with binary deep neural network". In: *European Conference on Computer Vision*. Springer. 2016, pp. 219–234.

[38] T.-T. Do, A.-D. Doan, D.-T. Nguyen, and N.-M. Cheung. "Binary hashing with semidefinite relaxation and augmented lagrangian". In: *European Conference on Computer Vision*. Springer. 2016, pp. 802–817.

[39] T.-T. Do, Q. D. Tran, and N.-M. Cheung. "FAemb: a function approximation-based embedding method for image retrieval". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3556–3564.

[40] A.-D. Doan, Y. Latif, T.-J. Chin, Y. Liu, S.-F. Ch'ng, T.-T. Do, and I. Reid. "Visual localization under appearance change: filtering approaches". In: *Neural Computing and Applications* (2020), pp. 1–14.

[41] A.-D. Doan, Y. Latif, T.-J. Chin, Y. Liu, T.-T. Do, and I. Reid. "Scalable place recognition under appearance change for autonomous driving". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9319–9328.

[42] P. Domingos. "Every Model Learned by Gradient Descent Is Approximately a Kernel Machine". In: *arXiv preprint arXiv:2012.00152* (2020).

[43] M. Douze, H. Jégou, and F. Perronnin. "Polysemous codes". In: *European Conference on Computer Vision*. Springer. 2016, pp. 785–801.

[44] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. "D2-net: A trainable cnn for joint description and detection of local features". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8092–8101.

[45] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart. "Keep it brief: Scalable creation of compressed localization maps". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2015, pp. 2536–2542.

[46] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale. "The gist of maps-summarizing experience for lifelong localization". In: *IEEE International Conference on Robotics and Automation*. 2015, pp. 2767–2773.

[47] M. Dymczyk, E. Stumm, J. Nieto, R. Siegwart, and I. Gilitschenski. "Will it last? Learning stable features for long-term visual localization". In: *International Conference on 3D Vision*. IEEE. 2016, pp. 572–581.

[48] P. Ebel, A. Mishchuk, K. M. Yi, P. Fua, and E. Trulls. "Beyond cartesian representations for local descriptors". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 253–262.

[49] J. Engel, V. Koltun, and D. Cremers. "Direct sparse odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (2017), pp. 611–625.

[50] J. Engel, J. Sturm, and D. Cremers. "Semi-dense visual odometry for a monocular camera". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1449–1456.

[51] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast semi-direct monocular visual odometry". In: *IEEE International Conference on Robotics and Automation*. 2014, pp. 15–22.

[52] P. Furgale and T. D. Barfoot. "Visual teach and repeat for long-range rover autonomy". In: *Journal of Field Robotics* 27.5 (2010), pp. 534–560.

[53] Y. Gal. "Uncertainty in deep learning". In: *University of Cambridge* 1.3 (2016), p. 4.

[54] D. Gálvez-López and J. D. Tardos. "Bags of binary words for fast place recognition in image sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197.

[55] Y. Ganin and V. Lempitsky. "Unsupervised domain adaptation by backpropagation". In: *International Conference on Machine Learning*. 2015, pp. 1180–1189.

[56] S. Garg and M. Milford. "Fast, compact and highly scalable visual place recognition through sequence-based matching of overloaded representations". In: *IEEE International Conference on Robotics and Automation*. 2020, pp. 3341–3348.

[57] S. Garg and M. Milford. "SeqNet: Learning descriptors for sequence-based hierarchical place recognition". In: *IEEE Robotics and Automation Letters* (2021).

[58] S. Garg, N. Suenderhauf, and M. Milford. "Semantic–geometric visual place recognition: a new perspective for reconciling opposing views". In: *The International Journal of Robotics Research* (2019).

[59] S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T.-J. Chin, I. Reid, S. Gould, P. Corke, and M. Milford. "Semantics for Robotic Mapping, Perception and Interaction: A Survey". In: *Foundations and Trends in Robotics* 8.1 (2020), pp. 1–222.

[60] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, et al. "A2d2: Audi autonomous driving dataset". In: *arXiv preprint arXiv:2004.06320* (2020).

[61] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. "Actionvlad: Learning spatio-temporal aggregation for action classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 971–980.

[62] A. J. Glover, W. P. Maddern, M. J. Milford, and G. F. Wyeth. "FAB-MAP+RatSLAM: Appearance-based SLAM for multiple times of day". In: *IEEE International Conference on Robotics and Automation*. 2010, pp. 3507–3512.

[63] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2012), pp. 2916–2929.

[64] 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.

[65] I. J. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and harnessing adversarial examples". In: *International Conference on Learning Representations* (2014).

[66] Google. *Google Streetview*. https://www.google.com/maps. Online; accessed 01 February 2021.

[67] A. A. Hafez, M. Singh, K. M. Krishna, and C. Jawahar. "Visual localization in highly crowded urban environments". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 2778–2783.

[68] M. Haklay and P. Weber. "Openstreetmap: User-generated street maps". In: *IEEE Pervasive Computing* 7.4 (2008), pp. 12–18.

[69] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[70] N. Hauptman. *The Average Walking Stride Length*. https://livehealthy.chron.com/average-walking-stride-length-7494.html. Online; accessed 18 January 2021.

[71] S. Hausler, S. Garg, M. Xu, M. Milford, and T. Fischer. "Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

[72] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[73] K. L. Ho and P. Newman. "Loop closure detection in SLAM by combining visual and spatial appearance". In: *Robotics and Autonomous Systems* 54.9 (2006), pp. 740–749.

[74] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang. "The apolloscape open dataset for autonomous driving and its application". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (2019), pp. 2702–2719.

[75] Z. Huang. "Extensions to the k-means algorithm for clustering large data sets with categorical values". In: *Data mining and knowledge discovery* 2.3 (1998), pp. 283–304.

[76] N. Jacobs, W. Burgin, N. Fridrich, A. Abrams, K. Miskell, B. H. Braswell, A. D. Richardson, and R. Pless. "The Global Network of Outdoor Webcams: Properties and Applications". In: *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL)*. 2009, pp. 111–120.

[77] 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*. 2012, pp. 774–787.

[78] H. Jegou, M. Douze, and C. Schmid. "Product quantization for nearest neighbor search". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.1 (2010), pp. 117–128.

[79] H. Jégou, M. Douze, C. Schmid, and P. Pérez. "Aggregating local descriptors into a compact image representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2010, pp. 3304–3311.

[80] H. Jégou, M. Douze, C. Schmid, and P. Pérez. "Aggregating local descriptors into a compact image representation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2010, pp. 3304–3311.

[81] H. Jégou and A. Zisserman. "Triangulation embedding and democratic aggregation for image search". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3310–3317.

[82] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, and E. Trulls. "Image matching across wide baselines: From paper to practice". In: *International Journal of Computer Vision* (2020), pp. 1–31.

[83] E. Johns and G.-Z. Yang. "Feature co-occurrence maps: Appearance-based localisation throughout the day". In: *IEEE International Conference on Robotics and Automation*. 2013, pp. 3212–3218.

[84] J. Johnson, M. Douze, and H. Jégou. "Billion-scale similarity search with GPUs". In: *arXiv preprint arXiv:1702.08734* (2017).

[85] A. Kendall and R. Cipolla. "Geometric loss functions for camera pose regression with deep learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5974–5983.

[86] A. Kendall and Y. Gal. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in Neural Information Processing Systems* (2017).

[87] A. Kendall, M. Grimes, and R. Cipolla. "PoseNet: A convolutional network for real-time 6-dof camera relocalization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2938–2946.

[88] J. Ko and D. Fox. "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models". In: *Autonomous Robots* 27.1 (2009), pp. 75–90.

[89] I. Kostavelis and A. Gasteratos. "Semantic mapping for mobile robotics tasks: A survey". In: *Robotics and Autonomous Systems* 66 (2015), pp. 86–103.

[90] S. Kosub. "A note on the triangle inequality for the jaccard distance". In: *Pattern Recognition Letters* 120 (2019), pp. 36–38.

[91] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g2o: A general framework for graph optimization". In: *IEEE International Conference on Robotics and Automation*. 2011, pp. 3607–3613.

[92] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník. "Artificial intelligence for long-term robot autonomy: A survey". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4023–4030.

[93] M. Labbe and F. Michaud. "Appearance-based loop closure detection for online large-scale and long-term operation". In: *IEEE Transactions on Robotics* 29.3 (2013), pp. 734–745.

[94] J.-F. Lalonde, A. A. Efros, and S. G. Narasimhan. "Webcam clip art: Appearance and illuminant transfer from time-lapse sequences". In: *ACM Transactions on Graphics* 28.5 (2009), pp. 1–10.

[95] Y. Latif, R. Garg, M. Milford, and I. Reid. "Addressing challenging place recognition tasks using generative adversarial networks". In: *IEEE International Conference on Robotics and Automation*. 2018, pp. 2349–2355.

[96] Y. Latif, G. Huang, J. J. Leonard, and J. Neira. "An Online Sparsity-Cognizant Loop-Closure Algorithm for Visual Navigation". In: *Robotics: Science and Systems*. 2014.

[97] H. Le, T. Hoang, and M. J. Milford. "BTEL: A binary tree encoding approach for visual localization". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4354–4361.

[98] V. Lepetit, F. Moreno-Noguer, and P. Fua. "EPnP: An accurate O(n) solution to the PnP problem". In: *International Journal of Computer Vision* 81.2 (2009), p. 155.

[99] S. Leutenegger, M. Chli, and R. Y. Siegwart. "BRISK: Binary robust invariant scalable keypoints". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2011, pp. 2548–2555.

[100] R. Lin, J. Xiao, and J. Fan. "Nextvlad: An efficient neural network to aggregate frame-level features for large-scale video classification". In: *European Conference on Computer Vision Workshops*. 2018.

[101] C. Linegar, W. Churchill, and P. Newman. "Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation". In: *IEEE International Conference on Robotics and Automation*. 2015, pp. 90–97.

[102] D. G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[103] D. G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[104] S. Lowry and H. Andreasson. "Lightweight, viewpoint-invariant visual place recognition in changing environments". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 957–964.

[105] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. "Visual place recognition: A survey". In: *IEEE Transactions on Robotics* 32.1 (2015), pp. 1–19.

[106] Z. Luo, T. Shen, L. Zhou, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan. "Contextdesc: Local descriptor augmentation with cross-modality context". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2527–2536.

[107] Z. Luo, T. Shen, L. Zhou, S. Zhu, R. Zhang, Y. Yao, T. Fang, and L. Quan. "Geodesc: Learning local descriptors by integrating geometry constraints". In: *European Conference on Computer Vision*. 2018, pp. 168–183.

[108] S. Lynen, B. Zeisl, D. Aiger, M. Bosse, J. Hesch, M. Pollefeys, R. Siegwart, and T. Sattler. "Large-scale, real-time visual–inertial localization revisited". In: *The International Journal of Robotics Research* 39.9 (2020), pp. 1061–1084.

[109] W. Maddern, M. Milford, and G. Wyeth. "CAT-SLAM: probabilistic localisation and mapping using a continuous appearance-based trajectory". In: *The International Journal of Robotics Research* 31.4 (2012), pp. 429–451.

[110] W. Maddern, G. Pascoe, M. Gadd, D. Barnes, B. Yeomans, and P. Newman. "Real-time Kinematic Ground Truth for the Oxford RobotCar Dataset". In: *arXiv preprint arXiv: 2002.10152* (2020).

[111] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. "1 year, 1000 km: The Oxford RobotCar dataset". In: *The International Journal of Robotics Research* 36.1 (2017), pp. 3–15.

[112] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman. "Averaging quaternions". In: *Journal of Guidance, Control, and Dynamics* 30.4 (2007), pp. 1193–1197.

[113] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks". In: *IEEE International Conference on Robotics and Automation*. 2017, pp. 4628–4635.

[114] C. McManus, B. Upcroft, and P. Newman. "Scene signatures: Localised and point-less features for localisation". In: *Robotics: Science and Systems* (2014), pp. 1–9.

[115] K. Mikolajczyk and C. Schmid. "A performance evaluation of local descriptors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.10 (2005), pp. 1615–1630.

[116] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. "A comparison of affine region detectors". In: *International Journal of Computer Vision* 65.1 (2005), pp. 43–72.

[117] M. Milford and G. Wyeth. "Persistent navigation and mapping using a biologically inspired SLAM system". In: *The International Journal of Robotics Research* 29.9 (2010), pp. 1131–1153.

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

[119] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. "Working hard to know your neighbor's margins: Local descriptor learning loss". In: *Advances in Neural Information Processing Systems* (2017).

[120] T. Morris, F. Dayoub, P. Corke, G. Wyeth, and B. Upcroft. "Multiple map hypotheses for planning and navigating in non-stationary environments". In: *IEEE International Conference on Robotics and Automation*. 2014, pp. 2765–2770.

[121] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale. "Summary maps for lifelong visual localization". In: *Journal of Field Robotics* 33.5 (2016), pp. 561–590.

[122] M. Muja and D. G. Lowe. "Scalable nearest neighbor algorithms for high dimensional data". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2227–2240.

[123] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[124] N. Murray and F. Perronnin. "Generalized max pooling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2473–2480.

[125] NASA. *Space Robotics Challenge*. https://www.nasa.gov/directorates/spacetech/centennial_challenges/space_robotics/index.html. Online; accessed 04 February 2021.

[126] T. Naseer, W. Burgard, and C. Stachniss. "Robust visual localization across seasons". In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 289–302.

[127] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. "Robust visual robot localization across seasons using network flows". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014.

[128] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder. "The mapillary vistas dataset for semantic understanding of street scenes". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 4990–4999.

[129] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. "DTAM: Dense tracking and mapping in real-time". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2011, pp. 2320–2327.

[130] A. Nguyen, J. Yosinski, and J. Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 427–436.

[131] K. Ni, A. Kannan, A. Criminisi, and J. Winn. "Epitomic location recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.12 (2009), pp. 2158–2167.

[132] M. Nowakowski, C. Joly, S. Dalibard, N. Garcia, and F. Moutarde. "Topological localization using Wi-Fi and vision merged into FABMAP framework". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2017, pp. 3339–3344.

[133] E. Pepperell, P. I. Corke, and M. J. Milford. "All-environment visual place recognition with SMART". In: *IEEE International Conference on Robotics and Automation*. 2014, pp. 1612–1618.

[134] H. Porav, W. Maddern, and P. Newman. "Adversarial training for adverse conditions: Robust metric localisation using appearance transfer". In: *IEEE International Conference on Robotics and Automation*. 2018, pp. 1011–1018.

[135] F. Radenović, G. Tolias, and O. Chum. "Fine-tuning CNN image retrieval with no human annotation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.7 (2018), pp. 1655–1668.

[136] J. Revaud, C. De Souza, M. Humenberger, and P. Weinzaepfel. "R2D2: Reliable and repeatable detector and descriptor". In: *Advances in Neural Information Processing Systems*. 2019, pp. 12405–12415.

[137] S. R. Richter, Z. Hayder, and V. Koltun. "Playing for benchmarks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2213–2222.

[138] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. "Playing for data: Ground truth from computer games". In: *European Conference on Computer Vision*. Springer. 2016, pp. 102–118.

[139] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping". In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2020. URL: https://github.com/MIT-SPARK/Kimera.

[140] E. Rosten, R. Porter, and T. Drummond. "Faster and better: A machine learning approach to corner detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.1 (2008), pp. 105–119.

[141] C. Rubino, A. Del Bue, and T.-J. Chin. "Practical Motion Segmentation for Urban Street View Scenes". In: *IEEE International Conference on Robotics and Automation*. 2018, pp. 1879–1886.

[142] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An efficient alternative to SIFT or SURF". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2011, pp. 2564–2571.

[143] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited, 2016.

[144] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou. "How should we evaluate supervised hashing?" In: *IEEE International Conference on Acoustics, Speech and Signal Processin*. 2017, pp. 1732–1736.

[145] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, et al. "Benchmarking 6DOF outdoor visual localization in changing conditions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8601–8610.

[146] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. "Understanding the limitations of cnn-based absolute camera pose regression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3302–3312.

[147] N. Savinov, A. Dosovitskiy, and V. Koltun. "Semi-parametric topological memory for navigation". In: *International Conference on Learning Representations* (2018).

[148] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys. "Quad-networks: unsupervised learning to rank for interest point detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1822–1830.

[149] G. Schindler, M. Brown, and R. Szeliski. "City-scale location recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–7.

[150] C. Schmid, R. Mohr, and C. Bauckhage. "Evaluation of interest point detectors". In: *International Journal of Computer Vision* 37.2 (2000), pp. 151–172.

[151] T. Schneider, M. T. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. "maplab: An Open Framework for Research in Visual-inertial Mapping and Localization". In: *IEEE Robotics and Automation Letters* (2018). DOI: 10.1109/LRA.2018.2800113.

[152] J. L. Schonberger and J.-M. Frahm. "Structure-from-motion revisited". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4104–4113.

[153] J. L. Schonberger, H. Hardmeier, T. Sattler, and M. Pollefeys. "Comparative evaluation of hand-crafted and learned local features". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1482–1491.

[154] S. Se, D. Lowe, and J. Little. "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks". In: *The International Journal of Robotics Research* 21.8 (2002), pp. 735–758.

[155] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 806–813.

[156] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. "Scene coordinate regression forests for camera relocalization in RGB-D images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2930–2937.

[157] 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.

[158] J. Sivic and A. Zisserman. "Video Google: A text retrieval approach to object matching in videos". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2003, p. 1470.

[159] N. Snavely, S. M. Seitz, and R. Szeliski. "Photo tourism: exploring photo collections in 3D". In: *ACM Siggraph 2006 Papers*. 2006, pp. 835–846.

[160] L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers. "GN-Net: The gauss-newton loss for multi-weather relocalization". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 890–897.

[161] A. Stylianou, A. Abrams, and R. Pless. "Characterizing feature matching performance over long time periods". In: *IEEE Winter Conference on Applications of Computer Vision*. 2015, pp. 892–898.

[162] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, et al. "The limits and potentials of deep learning for robotics". In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 405–420.

[163] N. Sünderhauf, P. Neubert, and P. Protzel. "Are we there yet? Challenging SeqSLAM on a 3000 km journey across all four seasons". In: *IEEE International Conference on Robotics and Automation, Workshop on Long-Term Autonomy*. 2013.

[164] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[165] Y. Tian, B. Fan, and F. Wu. "L2-net: Deep learning of discriminative patch descriptor in euclidean space". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 661–669.

[166] C. Toft, W. Maddern, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, T. Pajdla, et al. "Long-Term Visual Localization Revisited". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[167] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla. "24/7 place recognition by view synthesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1808–1817.

[168] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. "Training deep networks with synthetic data: Bridging the reality gap by domain randomization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 969–977.

[169] M. A. Uy and G. H. Lee. "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4470–4479.

[170] Y. Verdie, K. Yi, P. Fua, and V. Lepetit. "TILDE: A temporally invariant learned detector". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5279–5288.

[171] O. Vysotska and C. Stachniss. "Effective visual place recognition using multi-sequence maps". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1730–1736.

[172] O. Vysotska and C. Stachniss. "Lazy data association for image sequences matching under substantial appearance changes". In: *IEEE Robotics and Automation Letters* 1.1 (2015), pp. 213–220.

[173] O. Vysotska and C. Stachniss. "Relocalization under substantial appearance changes using hashing". In: *Proceedings of the IROS Workshop on Planning, Perception and Navigation for Intelligent Vehicles, Vancouver, BC, Canada*. Vol. 24. 2017.

[174] M. Wang and W. Deng. "Deep visual domain adaptation: A survey". In: *Neurocomputing* 312 (2018), pp. 135–153.

[175] P. Wang, R. Yang, B. Cao, W. Xu, and Y. Lin. "DeLS-3D: Deep localization and segmentation with a 3D semantic map". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5860–5869.

[176] F. Warburg, S. Hauberg, M. López-Antequera, P. Gargallo, Y. Kuang, and J. Civera. "Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2626–2635.

[177] D. Whitley. "A genetic algorithm tutorial". In: *Statistics and computing* 4.2 (1994), pp. 65–85.

[178] Wikipedia. *Nonholonomic system*. https://en.wikipedia.org/wiki/Nonholonomic_system. Online; accessed 17 January 2021.

[179] M. Xu, T. Fischer, N. Sünderhauf, and M. Milford. "Probabilistic appearance-invariant topometric localization with new place awareness". In: *IEEE Robotics and Automation Letters* (2021).

[180] M. Xu, N. Snderhauf, and M. Milford. "Probabilistic Visual Place Recognition for Hierarchical Localization". In: *IEEE Robotics and Automation Letters* (2020).

[181] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. "LIFT: Learned invariant feature transform". In: *European Conference on Computer Vision*. Springer. 2016, pp. 467–483.

[182] J. Yu, C. Zhu, J. Zhang, Q. Huang, and D. Tao. "Spatial pyramid-enhanced NetVLAD with weighted triplet loss for place recognition". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.2 (2019), pp. 661–674.

[183] K. Yu and T. Zhang. "Improved Local Coordinate Coding using Local Tangents". In: *International Conference on Machine Learning*. 2010.

[184] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. "Understanding deep learning requires rethinking generalization". In: *International Conference on Learning Representations* (2017).

[185] H. Zhou, T. Sattler, and D. W. Jacobs. "Evaluating local features for day-night matching". In: *European Conference on Computer Vision*. Springer. 2016, pp. 724–736.