FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

University of West Bohemia
Faculty of Applied Sciences

Department of Cybernetics

Semantic Segmentation in Long-term Visual Localization

Ing. Lukáš Bureš

DISSERTATION THESIS

submitted in partial fulfilment of the requirements
for the degree of
Doctor of Philosophy in the field of
Cybernetics

Supervisor: Prof. Ing. Luděk Müller, Ph.D.
Department of Cybernetics

Pilsen, 2022

FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Katedra kybernetiky

Sémantická segmentace v dlouhodobé vizuální lokalizaci

Ing. Lukáš Bureš

DISERTAČNÍ PRÁCE

k získání akademického titulu doktor
v oboru **Kybernetika**

Školitel: Prof. Ing. Luděk Müller, Ph.D.
Katedra kybernetiky

Plzeň, 2022

Supervisor:

Prof. Ing. Luděk Müller, Ph.D.
Department of Cybernetics
Faculty of Applied Sciences
University of West Bohemia
Univerzitní 8
306 14 Pilsen
Czech Republic

*I want to dedicate my thesis to my loved wife Kamila
and our family.*

Abstract

This thesis has five main goals. At first, it maps the datasets used for long-term visual localization and selects viable datasets for further evaluation. Next, one of the current state-of-the-art pipelines is selected and enhanced. Results with carefully fine-tuned methods' parameters accomplish better localization results. Furthermore, it shows that dynamic objects in an image are unnecessary for long-term visual localization because they do not contain any helpful information and can be ignored. The fourth goal in this thesis is to embed semantic segmentation information into the SuperPoint keypoint detector and descriptor by editing training data. Finally, the new state-of-the-art results on a selected dataset are achieved by applying a novel keypoint filtering approach based on semantic segmentation information. The significance of this work shows the importance of analyzing underlying image information in long-term visual localization and keypoint detection in general.

Keywords: Long-Term Visual Localization, Visual Features, Visual Keypoints, Keypoints Detectors, Keypoints Descriptors, Neural Networks, Computer Vision, Machine Learning

Abstrakt

Tato práce má pět hlavních cílů. Nejprve mapuje datové sady používané pro dlouhodobou vizuální lokalizaci a vybere vhodné datové sady pro další vyhodnocení. Dále je vybrán a vylepšen jeden ze současných state-of-the-art přístupů. Výsledky s pečlivě vyladěnými parametry vybrané metody dosahují lepších výsledků lokalizace. Dále je ukázáno, že dynamické objekty v obrázku jsou pro dlouhodobou vizuální lokalizaci zbytečné, protože neobsahují žádnou užitečnou informaci a lze je zcela odstranit. Čtvrtým cílem této práce je pokusit se vložit sémantickou informaci do detektoru a deskriptoru klíčových bodů SuperPoint úpravou trénovacích dat. Závěrem je dosaženo nových state-of-the-art výsledků na vybrané datové sadě aplikací nového přístupu filtrování klíčových bodů založeného na sémantické informaci. Význam této práce ukazuje důležitost analýzy obrazové informace v úloze dlouhodobé vizuální lokalizace a detekce klíčových bodů obecně.

Klíčová slova: Dlouhodobá vizuální lokalizace, Vizuální příznaky, Vizuální klíčové body, Detektory klíčových bodů, Deskriptory klíčových bodů, Neuronové sítě, Počítačové vidění, Strojové učení

Acknowledgements

First, I want to thank my family, especially my long-loved wife **Ing. Kamila Bureš Haller** and our family.

Next, thanks belong to my supervisor **Prof. Ing. Luděk Müller, Ph.D.** for all provided support, patience, and consultations he gave me. Furthermore, to the head of my supervisory department, **Prof. Ing. Josef Psutka, CSc.** for tolerance and financial support. I want to thank all of my colleagues, especially **Ing. Marek Hrúz, Ph.D.**, for the meaningful discussions while writing this thesis.

Furthermore, I want to thank Patrick Wenzel for validating my results on the 4Seasons dataset (test0 and test1 sequences specifically).

Other thanks, I would like to say to all people who supported and trusted me during this long process of my Ph.D. study and writing this thesis.

Declaration

The author hereby declares that he compiled this thesis independently, using only the listed resources and literature.

In Pilsen **Ing. Lukáš Bureš**

Table of Contents

Table of Contents **i**

List of Figures **vii**

List of Tables **xv**

Nomenclature **xix**

List of Abbreviations **xxi**

I Introduction **1**

1 Introduction **3**

1.1 Motivation and Applications 3

1.2 Dissertation Thesis Structure 4

2 Visual Features **7**

2.1 Visual Feature Definition 7

2.2 Brief History of Visual Features 8

2.2.1 Corners Detectors 8

2.2.2 Traditional Features Detectors and Descriptors 9

2.2.3 State-of-the-art Features Detectors and Descriptors 10

2.3 Requirements: Keypoints Detector, Descriptor, and Matcher 10

2.4 Problem Statement 12

3 Dissertation Goals **13**

3.1 Selecting Dataset for Evaluation 13

3.2 Enhanced Parameter Setting 13

3.3 Preprocessing for Keypoint Masking 14

3.4 Fusion of Training Information 14

3.5 Keypoint Filtering 14

II Methodology 17

4 State-of-the-art Methods 19

4.1 Network of Vector of Locally Aggregated Descriptors 19
4.1.1 Bag of Words 19
4.1.2 Vector of Locally Aggregated Descriptors 20
4.1.3 Network of Vector of Locally Aggregated Descriptors 21
4.2 Detect-and-Describe Network 22
4.2.1 Feature Description 23
4.2.2 Feature Detection 24
4.2.3 Training loss 25
4.2.4 Summary 26
4.3 Repeatable and Reliable Detector and Descriptor 26
4.3.1 R2D2 Architecture 27
4.4 Visual Localization Using Semantic Segmentation and Depth Prediction 28
4.4.1 Semantic Verification of Image Retrieval 29
4.4.2 Multiple Feature Extractors and Enhanced R2D2 Descriptor . 30
4.4.3 Depth Consistency Verification 30
4.4.4 Pose Estimation With Weighted-RANSAC 31
4.5 Deep Local Feature 31
4.5.1 Dense Localized Feature Extraction 32
4.5.2 Attention-based Keypoint Selection 33
4.6 Deep Local and Global Features 33
4.6.1 Model 34
4.6.2 Training 35
4.7 SuperPoint 37
4.7.1 SuperPoint Architecture 37
4.7.2 Synthetic Pretraining and MagicPoint 38
4.7.3 Homographic Adaptation 39
4.8 SuperGlue 39
4.8.1 Formulation 40
4.8.2 Attentional Graph Neural Network 41
4.8.3 Optimal Matching Layer 43
4.8.4 Loss 44
4.9 Hierarchical Multi-scale Attention for Semantic Segmentation 44
4.9.1 Hierarchical Multi-Scale Attention 45
4.9.2 Auto Labelling 47
4.10 Summary 47

5 Datasets **49**
5.1 CMU Seasons Dataset 49
5.1.1 Extended CMU Seasons Dataset 50
5.2 RobotCar Seasons Dataset 51
5.3 Cross-Seasons Correspondence Dataset 53
5.4 Symphony Seasons Dataset 54
5.5 NRK Train Dataset 55
5.6 Aachen Day-Night Dataset 56
5.6.1 Aachen v1.1 57
5.7 Google Landmarks Dataset v2 57
5.8 4Seasons 58
5.9 HPatches Dataset 59
5.10 Summary 60

III Contribution to the State-of-the-art **61**

6 Long-Term Visual Localization Analysis **65**
6.1 The Problem 65
6.2 Benchmark 67
6.2.1 Used Dataset 68
6.2.2 Evaluation 69
6.2.3 Data Submission Format 69
6.2.4 Ranking Method 70
6.3 Long-Term Visual Localization Pipeline 70
6.4 Development Environment 71
6.5 Discussion 72

7 Parameter Tuning **73**
7.1 Methods 73
7.1.1 Previous Results 73
7.2 Parameters Description 75
7.3 Hypotheses 76
7.3.1 Aachen Day Images 77
7.3.2 Aachen Night Images 78
7.3.3 Aachen Day-Night Images 78
7.4 Experiments and Results 79
7.4.1 Aachen Day Images 79
7.4.2 Aachen Night Images 80
7.4.3 Aachen Day-Night Images 80

7.5 Discussion . . . 81

8 Semantic Segmentation 83

8.1 Hypotheses . . . 83
8.2 Methods . . . 83
8.3 Parameters Description . . . 84
8.4 Introduction of Semantic Segmentation Groups of Classes . . . 85
8.5 Experiments and Results . . . 85
8.5.1 Aachen Night Images . . . 86
8.5.2 Aachen Day Images . . . 87
8.6 Discussion . . . 92

9 SuperPoint Training 93

9.1 Training . . . 93
9.2 Discussion . . . 96

10 Keypoint Filtering 99

10.1 Selected Datasets . . . 99
10.2 Parameters Settings . . . 100
10.3 Experiments Description . . . 100
10.4 Semantic Segmentation . . . 101
10.5 Keypoint Filtering . . . 103
10.6 Results . . . 104
10.6.1 4Seasons Training Sequence . . . 104
10.6.2 4Seasons Validation Sequence . . . 107
10.6.3 4Seasons Test0 Sequence . . . 110
10.6.4 4Seasons Test1 Sequence . . . 116
10.6.5 Aachen v1.1 Day Sequence . . . 122
10.7 Examining Statistical Significance of Results . . . 125
10.8 Discussion . . . 127

IV Conclusion 129

11 Conclusion 131

11.1 Thesis Summary . . . 131
11.2 Fulfillment of Dissertation Goals . . . 132
11.2.1 Selecting Dataset for Evaluation . . . 132
11.2.2 Enhanced Parameter Setting . . . 132
11.2.3 Preprocessing for Keypoint Masking . . . 133
11.2.4 Fusion of Training Information . . . 134

11.2.5 Keypoint Filtering 134
11.3 Future Work 136

A 4Seasons Training Sequence **137**

B 4Seasons Validation Sequence **143**

C Aachen v1.1 Day Sequence **149**

Bibliography **I**

List of Figures

4.1 The visualization of Bag of Words (BoW) pipeline (from left to right). 20

4.2 Visualization of VLAD's moments. 21

4.3 The figure shows the comparison between different approaches for feature detection and description. The left pipeline corresponds to a standard two-stage detect-then-describe approach. In contrast, D2-Net pipeline (right) uses a single CNN which extracts dense features that serve as both descriptors and detectors [1]. 23

4.4 D2-Net detect-and-describe network [1]. 26

4.5 Overview of network for jointly learning repeatable and reliable matches. 28

4.6 An overview diagram of the pipeline. 29

4.7 Overall architecture of image retrieval system, using DEep Local Feature (DELF) and attention-based keypoint selection [2]. On the left, it illustrates the pipeline for extraction and selection of DELF. On the right, it illustrates large-scale feature-based retrieval pipeline. . . 32

4.8 DELG model (left) jointly extracts deep local and global features. Global features are used in the first stage of a retrieval system, to efficiently select the most similar images (bottom). Local features can then be employed to re-rank top results (top-right), increasing precision of the system (for more details see [3]). 34

4.9 Training pipeline: the components highlighted in green are used solely during training. There are two classification losses: ArcFace for global feature learning and softmax for attention learning. In both cases, the classification objective is to distinguish different landmarks. The autoencoder (purple) is further trained with a reconstruction loss. The whole model is learned end-to-end [3]. 35

4.10 SuperPoint Decoders: both decoders operate on a shared and spatially reduced representation of the input. For fast and easy training, both decoders use non-learned upsampling to bring the representation back to $\mathbb{R}^{H\times W}$. 37

4.11 Feature matching with SuperGlue establishes pointwise correspondences from off-the-shelf local features. It acts as a middle-end between hand-crafted or learned frontend and backend [4]. 40

4.12 SuperGlue has two major components: the attentional graph neural network and the optimal matching layer. The attentional graph neural network uses a keypoint encoder to map keypoint positions $\boldsymbol{p}$ and their visual descriptors $\boldsymbol{d}$ into a single vector, and then uses alternating self- and cross-attention layers (repeated L times) to create more powerful representations f. The optimal matching layer creates an M by N score matrix, augments it with dustbins, then finds the optimal partial assignment using the Sinkhorn algorithm (for T iterations) [4]. 41

4.13 Left: shows the architecture from [5] where the attention for each scale is learned explicitly. Right: shows hierarchical attention architecture. Right top: training pipeline where the network learns to predict attention between adjacent scale pairs. Right bottom: displays hierarchical structure to combine multiple scales of predictions. Lower scale attention determines the contribution of the next higher scale. The image was taken from [6]. 46

5.1 Example of CMU Seasons dataset. The figure shows two images representing the same scene from the slice number 24. Both images are captured under different seasonal conditions. 49

5.2 In this comparison, there is the same suburban scene as a part of the extended CMU Seasons dataset. The left image shows the location during spring. The right image reflects changes in the natural environment during the summer. 51

5.3 Example of RobotCar Seasons dataset. The figure demonstrates two pictures in one row showing the same scene. The left images show the scene in winter and the right images were captured in summer. All of the images were taken by using a camera mounted on the left side of the car. 52

5.4 The cross-seasons correspondence dataset reflects weather and season changes during one year of images capturing. Both images show environment changes for the same scene in the urban location in summer and fall. 53

5.5 Two images of the same scene which were captured during different year seasons and showing features from an unstructured and natural environment. 54

5.6 Left: the sky over the Nordlandsbanen. Right: Hellum T. and Carlsen J. from NRK mount the forward-facing camera before the winter recording along the Nordlandsbanen. Photos from NRK. 55
5.7 The Aachen Day-Night dataset contains images taken by hand-held cameras in the old city of Aachen, Germany. The left images demonstrate the same scene in the summer, and the right one represents the same square during the winter season. Both images were taken during the daytime. 57
5.8 GLDv2 has been completely created by the worldwide Wikimedia Commons community. All images are freely licensed and can be freely reproduced in publications. These two images represent the same scene of the castle Kašperk, Czech Republic. The left image was taken in the summer season, and the right one was captured during winter by community users. 58
5.9 An example of 4Seasons dataset images (cam0 left, cam1 right). . . . 59
5.10 The HPatches dataset operates with images with extreme illumination changes. Both images show the same scene of Brooklyn city, NY, the USA, taken from one of the 116 image sequences where illumination change is evident. 59

7.1 Two examples of Aachen Night images (1024 pixels longer side) taken under poor lighting conditions and with a lot of noise. 78

8.1 Left: original image from Aachen Night dataset. Right: failed semantic segmentation where the people (red) in the left bottom corner are missing, vegetation (green) is poorly segmented specifically in the top right corner, and a wall (purple) is in the top left corner where the sky should be segmented instead. 86
8.2 Left: an original Aachen Day image. Right: semantic segmentation in the color palette of the Cityscape dataset. 87
8.3 Left: original Aachen Day image. Right: masked **human** h group represents person and rider Cityscape classes. 88
8.4 Left: original Aachen Day image. Right: masked **sky** s group is represented only by sky Cityscape class. 88
8.5 Left: original Aachen Day image. Right: masked **vehicle** v group aggregates car, truck, bus, caravan, trailer, train, motorcycle, bicycle, and license plate Cityscape classes. 89
8.6 Left: original Aachen Day image. Right: masked **nature** n group consists of vegetation and terrain Cityscape classes. 89

8.7 Two different examples (rows) of the final masking of input Aachen Day images. Left: original image. Right: masked image with used all four groups of classes (sky, human, vehicle, and nature). 90

9.1 Example of Synthetic Shapes dataset: triangles, quadrilaterals, lines, cubes, checkerboards, and stars each shape with ground truth corner locations (image was taken from [7]). 94
9.2 Two examples of MS-COCO dataset. The original image (left) and its segmentation (right) segmented with HRNet-OCR method from Section 4.9. 96

10.1 Top: an example from 4Seasons dataset. Left: binary mask for vehicle group of classes. Right: binary mask for human group of classes (it is empty because no human is presented in the image). 102
10.2 Top: an example from Aachen dataset. Left: binary mask for vehicle group of classes. Right: binary mask for human group of classes. . . . 102
10.3 4Seasons, training sequence, run 1: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 105
10.4 4Seasons, training sequence, run 1: the results for 0.10m localization threshold for experiments with and without dynamic keypoints. . . . 105
10.5 4Seasons, training sequence, run 1: the results for 0.20m localization threshold for experiments with and without dynamic keypoints. . . . 106
10.6 4Seasons, training sequence, run 1: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. . . . 106
10.7 4Seasons, validation sequence, run 1: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 108
10.8 4Seasons, validation sequence, run 1: the results for 0.10m localization threshold for experiments with and without dynamic keypoints. . 108
10.9 4Seasons, validation sequence, run 1: the results for 0.20m localization threshold for experiments with and without dynamic keypoints. . 109
10.104Seasons, validation sequence, run 1: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. . 109
10.114Seasons, test0 sequence, Experiment 1: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 112
10.124Seasons, test0 sequence, Experiment 1: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.112
10.134Seasons, test0 sequence, Experiment 1: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.113
10.144Seasons, test0 sequence, Experiment 1: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.113

10.15 4Seasons, test0 sequence, Experiment 2: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 114
10.16 4Seasons, test0 sequence, Experiment 2: the results for 0.10m localization threshold for experiments with and without dynamic keypoints. 114
10.17 4Seasons, test0 sequence, Experiment 2: the results for 0.20m localization threshold for experiments with and without dynamic keypoints. 115
10.18 4Seasons, test0 sequence, Experiment 2: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. 115
10.19 4Seasons, test1 sequence, Experiment 1: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 118
10.20 4Seasons, test1 sequence, Experiment 1: the results for 0.10m localization threshold for experiments with and without dynamic keypoints. 118
10.21 4Seasons, test1 sequence, Experiment 1: the results for 0.20m localization threshold for experiments with and without dynamic keypoints. 119
10.22 4Seasons, test1 sequence, Experiment 1: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. 119
10.23 4Seasons, test1 sequence, Experiment 2: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 120
10.24 4Seasons, test1 sequence, Experiment 2: the results for 0.10m localization thresholds for experiments with and without dynamic keypoints. 120
10.25 4Seasons, test1 sequence, Experiment 2: the results for 0.20m localization thresholds for experiments with and without dynamic keypoints. 121
10.26 4Seasons, test1 sequence, Experiment 2: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. 121
10.27 Aachen v1.1, day query sequence, run 1: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 123
10.28 Aachen v1.1, day query sequence, run 1: the results for 0.25m and 2° localization threshold for experiments with and without dynamic keypoints. 123
10.29 Aachen v1.1, day query sequence, run 1: the results for 0.50m and 5° localization threshold for experiments with and without dynamic keypoints. 124
10.30 Aachen v1.1, day query sequence, run 1: the results for 5.00m and 10° localization threshold for experiments with and without dynamic keypoints. 124

A.1 4Seasons, training sequence, run 2: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 137

A.2 4Seasons, training sequence, run 2: the results for 0.10m localization threshold for experiments with and without dynamic keypoints. . . . 138
A.3 4Seasons, training sequence, run 2: the results for 0.20m localization threshold for experiments with and without dynamic keypoints. . . . 138
A.4 4Seasons, training sequence, run 2: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. . . . 139
A.5 4Seasons, training sequence, run 3: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 139
A.6 4Seasons, training sequence, run 3: the results for 0.10m localization threshold for experiments with and without dynamic keypoints. . . . 140
A.7 4Seasons, training sequence, run 3: the results for 0.20m localization threshold for experiments with and without dynamic keypoints. . . . 140
A.8 4Seasons, training sequence, run 3: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. . . . 141

B.1 4Seasons, validation sequence, run 2: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 143
B.2 4Seasons, validation sequence, run 2: the results for 0.10m localization threshold for experiments with and without dynamic keypoints. . 144
B.3 4Seasons, validation sequence, run 2: the results for 0.20m localization threshold for experiments with and without dynamic keypoints. . 144
B.4 4Seasons, validation sequence, run 2: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. . 145
B.5 4Seasons, validation sequence, run 3: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 145
B.6 4Seasons, validation sequence, run 3: the results for 0.10m localization threshold for experiments with and without dynamic keypoints. . 146
B.7 4Seasons, validation sequence, run 3: the results for 0.20m localization threshold for experiments with and without dynamic keypoints. . 146
B.8 4Seasons, validation sequence, run 3: the results for 0.50m localization threshold for experiments with and without dynamic keypoints. . 147

C.1 Aachen v1.1, day query sequence, run 2: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 149
C.2 Aachen v1.1, day query sequence, run 2: the results for 0.25m and 2° localization threshold for experiments with and without dynamic keypoints. 150
C.3 Aachen v1.1, day query sequence, run 2: the results for 0.50m and 5° localization thresholds for experiments with and without dynamic keypoints. 150

C.4 Aachen v1.1, day query sequence, run 2: the results for 5.00m and 10° localization threshold for experiments with and without dynamic keypoints. . . . 151
C.5 Aachen v1.1, day query sequence, run 3: the results for all localization thresholds for experiments with and without dynamic keypoints. . . . 151
C.6 Aachen v1.1, day query sequence, run 3: the results for 0.25m and 2° localization thresholds for experiments with and without dynamic keypoints. . . . 152
C.7 Aachen v1.1, day query sequence, run 3: the results for 0.50m and 5° localization thresholds for experiments with and without dynamic keypoints. . . . 152
C.8 Aachen v1.1, day query sequence, run 3: the results for 5.00m and 10° localization threshold for experiments with and without dynamic keypoints. . . . 153

List of Tables

7.1 Published Original HLoc results in the long-term visual localization benchmark on Aachen dataset. 74

7.2 The table shows the comparison of the Original HLoc method with a few publicly released retests. Results are ordered from the best (first row) to the least performing method. 74

7.3 The comparison of the Original HLoc and its top 4 competitors. Results are ordered from the best (first row) to the least performing method. 74

7.4 Preciously selected parameters helped to outperform previous state-of-the-art results of the Original HLoc in two most precise localization thresholds for Aachen Day dataset. Results are ranked by Schulze method from Section 6.2.4 - the first row is the best. 79

7.5 Preciously selected parameters helped to outperform previous state-of-the-art results of the Original HLoc in two most precise localization thresholds for Aachen Night dataset. Results are ranked by Schulze method from Section 6.2.4 - the first row is the best. 80

7.6 Comparsion of the Original HLoc results in the long-term visual localization benchmark on Day-Night Aachen dataset compared with the Original HLoc (100 iterations) and the Combined tuned HLoc results. 81

8.1 The used characters in the Semantic Segmentation Groups of Classes column represents combinations of used groups of classes for masking input images. More specifically human h, vehicle v, sky s, and nature n groups. The percentages represent how many images were successfully localized within a threshold from the dataset. 91

8.2 The comparison of Tuned HLoc, Masked h HLoc, and Original HLoc precision. 92

9.1 Performance of Synthetic Shapes detector: MagicPoint and Original MagicPoint [7] models outperform classical detectors in detecting corners of simple geometric shapes and are robust to added noise. . . 94

9.2 The repeatability results of trained MagicPoint detector on HPatches dataset, 57 Illumination Scenes (IS) and 59 Viewport Scenes (VS). . . 95

10.1 Percentage of static and dynamic keypoints occurrence for Aachen v1.1 day query sequence (only for human h and vehicle v group of classes). 103

10.2 Percentage of static and dynamic keypoints occurrence for 4Seasons sequences (only for human h and vehicle v group of classes). 103

10.3 The table shows the comparsion of the Original HLoc method and the Filtered dynamic KPs HLoc method (both with the same parameters setting). The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The results are reported for standard precision thresholds 0.10m / 0.20m / 0.50m. 104

10.4 The table shows the comparsion of the Original HLoc method and the Filtered dynamic KPs HLoc method (both with the same parameters setting). The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The results are reported for standard precision thresholds 0.10m / 0.20m / 0.50m. 107

10.5 All publicly known state-of-the-art results for the 4Seasons test0 sequence were collected. Furthermore, achieved results are presented here. Experiment 2 outperformed all publicly known state-of-the-art results for all localization thresholds. The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and for 100% static keypoints with 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The standard localization thresholds are 0.10m / 0.20m / 0.50m. 111

10.6 All publicly known state-of-the-art results for the 4Seasons test1 sequence were collected. Furthermore, achieved results are presented here. Experiment 2 outperformed all publicly known state-of-the-art results for the two most accurate localization thresholds. The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and for 100% static keypoints with 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The standard localization thresholds are 0.10m / 0.20m / 0.50m. 117
10.7 The table shows the comparsion of the Original HLoc method and the Filtered dynamic KPs HLoc method (both with the same parameters setting). The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The results are reported for standard precision thresholds 0.25m 2° / 0.50m 5° / 5.00m 10°. 122
10.8 The considered results for examining statistical significance. The standard localization accuracy thresholds are 0.10m / 0.20m / 0.50m. 125
10.9 The considered results for examining statistical significance. The standard localization thresholds are 0.25m 2° / 0.50m 5° / 5.00m 10°. 126
10.10 The feature comparison of all used image sequences. 128

Nomenclature

KAZE “is a Japanese word that means wind. In nature, the wind is defined as the flow of air on a large scale, and normally this flow is ruled by nonlinear processes. In this way, we make the analogy with nonlinear diffusion processes in the image domain.” [8] xxi, 9, 20

List of Abbreviations

AGAST Adaptive and Generic Corner Detection Based on the Accelerated Segment Test 9
AKAZE Accelerated-KAZE 9
AMP Automatic Mixed Precision 84
ANN Artificial Neural Network 19, 22, 47, 134
AOC Adaptive Ordinal Cost 31
ASSP Atrous Spatial Pyramid Pooling 44
AST Accelerated Segment Test 9

BoW Bag of Words vii, 10, 19, 20
BRIEF Binary Robust Independent Elementary Features xxii, 9
BRISK Binary Robust Invariant Scalable Keypoints 9

CNN Convolutional Neural Network 10, 12, 22, 23, 25, 26, 32, 34
CPU Central Processing Unit 72
CVPR Conference on Computer Vision and Pattern Recognition 70

D2-Net Detect-and-Describe Network vii, 10, 12, 22, 23, 25, 26, 67, 71, 111, 117
DCV Depth Consistency Verification 28–30
DELF DEep Local Feature vii, 10, 29, 31–33, 37, 55, 71
DELG DEep Local and Global features vii, 10, 33–35, 37
DNN Deep Neural Network 29
DoF Degrees of Freedom 28, 50, 65
DoG Difference of Gaussian 9, 24

ECCV European Conference on Computer Vision 67, 70
EDV Estimated Depth Value 30, 31

FAST Features from Accelerated Segment Test xxii, 9, 94, 95
FCN Fully Convolutional Network 32
FLANN Fast Library for Approximate Nearest Neighbors 66
FREAK Fast Retina Keypoint 9

GLDv2 Google Landmarks Dataset v2 ix, 57, 58

GPS Global Positioning System 51, 54, 56

GPU Graphics Processing Unit 39, 43, 72, 84

HDF Hierarchical Data Format 71

HLoc Hierarchical Localization xv–xvii, 73–76, 78–81, 83–85, 91, 92, 100, 101, 104, 107, 110, 111, 116, 117, 122, 125, 126, 131–133, 135

HOG Histogram of Oriented Gradients 9

IMU Inertial Measurement Unit 54

INS Inertial Navigation System 51

KDKPs Keep Dynamic Keypoints 125, 126

LATCH Learned Arrangements of Three Patch Codes 9

LBP Local Binary Patterns 9

LIDAR Light Detection And Ranging 51, 52, 66

LIFT Learned Invariant Feature Transform 33

MAP Mean Average Precision 94

MLP Multilayer Perceptron 41

MSD Maximal Self-Dissimilarities 9

MSER Maximally Stable Extremal Regions 9

NetVLAD Network of Vector of Locally Aggregated Descriptors 10, 19, 21, 22, 47, 55, 70, 71, 73, 75, 76, 79, 80, 85, 100, 136

NMS Non-Maximum Suppression 73, 75, 76, 79, 80, 86, 95, 100

NRK Norwegian Broadcasting Corporation ix, 55

NVM N-View Match 70, 71

ORB Oriented FAST and Rotated BRIEF 9

PCA Principal Components Analysis 33, 37

PDV Predicted Depth Value 30, 31

PnP Perspective-n-Point 28–31, 111, 117

PTZ Pan-Tilt-Zoom 54

R2D2 Repeatable and Reliable Detector and Descriptor 10, 12, 26, 27, 30, 71, 111, 117

RAM Random Access Memory 72

RANSAC RANdom SAmple Consensus 28–31, 100, 111, 117

RDKPs Remove Dynamic Keypoints 125, 126

RootSIFT RootSIFT 56

RTK-GNSS Real-Time Kinematic positioning - Global Navigation Satellite System 58, 59

SCC Semantic Consistency Check 28–30

SCW Semantic Consistency Weight 28, 29, 31

SfM Structure from Motion 23, 28, 30, 39, 66, 70, 71

SIFT Scale-Invariant Feature Transform xxii, 9, 20, 24, 33, 40, 56, 71

SLAM Simultaneous Localization and Mapping 39, 40, 49, 54, 60, 68

SURF Speeded-Up Robust Features 9

SUSAN Smallest Univalue Segment Assimilating Nucleus 9

SUV Sport Utility Vehicle 49

VLAD Vector of Locally Aggregated Descriptors vii, 10, 19–22

VRAM Video Random Access Memory 72

Part I

Introduction

Chapter 1

Introduction

In this chapter, the motivation is briefly discussed, a few applications of keypoint detectors and descriptors are mentioned, and the dissertation thesis structure is introduced.

1.1 Motivation and Applications

The ability to effectively represent local visual information is the key to a vast range of computer vision applications. The range of these applications may contain image alignment, which requires local image descriptors to be accurately matched between different views of the same scene, image classification, and image retrieval where massive descriptor collections are frequently scanned to locate the ones most relevant to those of a query image.

At the core of the problem, there is a challenge of extracting local representations at locations of keypoints. Keypoints are typically distributed sparsely in the image in a discriminative and invariant manner to various image transformations. Additional requirements, often critical, are connected with an efficiency of a representation in terms of the computational costs required to produce it, the space required to store it, and the time required to search for matching descriptors in large descriptor repositories.

Over the past two decades, several distinct approaches for designing these detectors and descriptors have appeared. It started with corners, moved to hand-crafted keypoints, and finally to learned visual features.

Corners detectors and hand-crafted keypoints detectors are mainly focused on invariance. Those method approaches are invariant due to rotation, scale, brightness, contrast, illumination, noise, occlusion, viewpoint, blur, and many others. These general tasks can sometimes be limited when real-world applications come.

The application of keypoint detectors and descriptors or visual features, in general, can be found in a task where it is necessary to find corresponding points across multiple images that can be captured at different times, different year seasons, or during different light conditions. This process is a critical step in many actual computer vision problems. The most important computer vision problems where keypoints plays an important role are:

- Object detection and recognition: detecting car plates, pedestrians, traffic signs, and traffic lights.
- Autonomous robot navigation: map-generation with structure from motion algorithms.
- Photo stitching: creating panoramic photos, generating a real estate indoor 3D model for a better property listing.
- Stereo matching and stereo rectification.
- Augmented reality 3D object positioning, refining, and position updating.
- Large scale and large database image registration and retrieval.
- Keypoint filtering based on semantic information or any other information such as depth information, instance segmentation, or objects.
- There are many more use cases like object tracking, security, surveillance, depth 3D reconstruction, and elders' monitoring.

1.2 Dissertation Thesis Structure

This dissertation thesis is logically organized into eleven chapters as follows:

1. The **Introduction** in Chapter 1, and there is the motivation described for the selected dissertation topic. It follows with a few examples of possible applications and the importance of the selected research topic. Then the structure of the thesis is introduced.
2. **Visual Features** are presented in Chapter 2, where all the related terms are briefly introduced. A broader description of the long-term visual localization is described here as well.
3. **Dissertation Goals** in Chapter 3 contain the list of the main goals of this dissertation thesis.
4. **State-of-the-art Detectors and Descriptors** in Chapter 4 present current state-of-the-art methods and approaches are primarily based on neural networks architectures.

5. **Datasets** are involved in Chapter 5. There is presented a set of publicly available datasets which are used for evaluation and benchmarking state-of-the-art methods. Datasets reflect year seasons, weather, natural environment, objects, illumination, and other appearance changes.

6. **Long-Term Visual Localization Analysis** in Chapter 6 defines the selected problem and introduces a benchmark. Next, there is the experiment pipeline and development environment described.

7. **Parameter Tuning Experiments** in Chapter 7 contains the author's original contribution to the field of long-term visual localization. The research focuses on improving current state-of-the-art methods by precisely fine-tuning parameters.

8. **Experiments with Semantic Segmentation** in Chapter 8 contains the author's contribution to the state-of-the-art. This chapter is focused on original research and experiments based on semantic segmentation masking of input images for keypoint filtering.

9. **SuperPoint Training Experiments** from Chapter 9 describe training SuperPoint keypoint detector and descriptor. The training is focused on providing semantic segmentation information and embedding it into the learned SuperPoint model.

10. **Keypoint Filtering** in Chapter 10 showcases an additional author's contribution to the state-of-the-art. It enhances long-term visual localization accuracy by filtering dynamic keypoints with the help of semantic segmentation on selected datasets.

11. The **Conclusion** is the last Chapter 11. It summarizes this thesis and evaluates the results and contributions of this work. It also involves a proposal for possible directions for future work.

Chapter 2

Visual Features

Visual features represent the primary topic of this dissertation thesis, and this section will introduce the definition, brief history, requirements, and problem statement.

2.1 Visual Feature Definition

The terms **corner**, **interest point**, **point of interest**, **keypoint**, **visual feature** are widely used in literature but the exact definition is missing or is not unified among the computer vision scientists. The terms mentioned above will be described and defined regarding the need to interpret the thesis.

Corner

A corner is the simplest and most interpretable visual feature. It can be defined as the intersection of two edges. The corner can also be defined as a point for which there are two dominant and different edge directions in a local neighborhood of the point.

Interest Point

An interest point (or point of interest) is an image point with a well-defined position and can be robustly detected. It implies that an interest point can be a corner. Nevertheless, it can also be an isolated point of local intensity maximum or minimum, end of a line, or a point on a curve where the curvature is locally maximal.

Keypoint

A keypoint can be defined as a point of interest with a semantic meaning. It can represent meaningful information for objects in an image, e.g., the right eye corner for face detectors or the left knee joint for body tracking systems. The keypoint is usually coupled with 2D or 3D coordinates, and additional information can also be attached.

Visual Feature

A visual feature is in the literature mostly referred to as a compact numerical representation (typically in vector form) of image content that enables efficient comparison (e.g., feature matching) between pairs of content items. It usually refers to the representation itself.

Summary

The terms **corner**, **interest point**, **point of interest**, **keypoint**, and **visual feature** are used mostly interchangeably in literature and scientific papers (unless otherwise stated). This leads to confusing and inconsistent naming across whole computer vision area and all available literature.

2.2 Brief History of Visual Features

Here are briefly mentioned historical, traditional, and state-of-the-art methods for corners, keypoints, and features visual detecting and describing. The methods are ordered historically to present a coherent overview.

2.2.1 Corners Detectors

Historically, early algorithms were focused on finding corners by performing edge detection and then analyzing the edges to find rapid changes in direction. In computer vision, the first important corner detector was the Moravec corner detector from 1977 [9, 10]. Moravec defined the concept of points of interest as distinct regions in images. Moravec used the corner detector for his research involving navigation of the Stanford Cart through a clustered environment.

The Harris and Stephens corner detector from 1988 [11] is the next essential corner detector. The authors wanted to help researchers interpret a robot's environment based on image sequences. The corner detector focuses on a low-level processing step that matches corresponding points in consecutive image frames. Nevertheless, Harris and Stephens concentrated on tracking corners and edges between frames.

Another example of a corner detector is Good Feature to Track from Shi and Tomasi from 1994 [12]. They modified the Harris corner detector, changing the scoring function and getting better results than the original Harris corner detector.

Many other corner detectors have been introduced since then. But two more which are worth to mention are Smallest Univalue Segment Assimilating Nucleus (SUSAN) [13] and Trajkovic and Hedley corner detector [14] from the year 1997 and 1998 respectively.

2.2.2 Traditional Features Detectors and Descriptors

One of the major method published in 2004 by Lowe was Scale-Invariant Feature Transform (SIFT) [15]. It has been developed for object description and visual detection based on keypoints [16]. The important idea of scale-space detection was introduced in SIFT. The following method developed by Bay in 2006 is speed up SIFT and it is called Speeded-Up Robust Features (SURF) [17]. The SURF focuses on constructing a computationally fast and stable descriptor that could run in real-time.

Historically, scientists looked for a way how to improve keypoint detecting speed to meet real-time applications needs. The Features from Accelerated Segment Test (FAST) detector was introduced by Rosten and Drummond in 2005 [18, 19]. FAST is faster than the DoG detectors but not invariant to scale changes. In the year 2010, Mair presents Adaptive and Generic Corner Detection Based on the Accelerated Segment Test (AGAST), which is based on Accelerated Segment Test (AST) (enhanced version of FAST). In the same year 2010, it was introduced Binary Robust Independent Elementary Features (BRIEF) by Calonder in [20]. The BRIEF provides a shortcut to the binary strings directly without finding descriptors.

The next binary descriptor is Oriented FAST and Rotated BRIEF (ORB) that was proposed by Rublee in the year 2011 [21]. ORB was developed as a good alternative to SIFT and SURF. Another binary method is Binary Robust Invariant Scalable Keypoints (BRISK) by Leutenegger from 2011 [22]. BRISK uses a circular symmetric region shape arranged in four concentric rings for reaching better scale invariance.

Many other detector and descriptor methods arised during the time. Let's mention a few of them such as Fast Retina Keypoint (FREAK) [23], KAZE [8], Accelerated-KAZE (AKAZE) [24], and Learned Arrangements of Three Patch Codes (LATCH) [25].

Different approaches for global image describing can be found in Local Binary Patterns (LBP) [26], Maximally Stable Extremal Regions (MSER) [27], Histogram of Oriented Gradients (HOG) [28], and Maximal Self-Dissimilarities (MSD) [29] methods. Detailed information about many keypoints detectors and descriptors methods can be found in other educational materials[1] of the author of this thesis.

[1] http://home.zcu.cz/~lbures/

2.2.3 State-of-the-art Features Detectors and Descriptors

Network of Vector of Locally Aggregated Descriptors (NetVLAD) was introduced in 2016 by Arandjelovic in [30] and it is one of state-of-the-art keypoint descriptor method used for image classification and image retrieval. It is based on Vector of Locally Aggregated Descriptors (VLAD) [31] and Bag of Words (BoW) [32] ideas. NetVLAD descriptor was used for image-based place recognition on the bucolic environment across seasons in [33].

Detect-and-Describe Network (D2-Net) was firstly introduced in [1] by Dusmanu in the year 2019. D2-Net is a detect-and-describe network that uses single CNN to find and describe keypoints simultaneously. The detection phase is postponed to a later stage, and obtained keypoints are more stable than their traditional counterparts based on the early detection of low-level structures. The Repeatable and Reliable Detector and Descriptor (R2D2) is another state-of-the-art method with detect-and-describe approach. It was firstly presented in 2019 by Revaud in [34]. The R2D2 allows simultaneously output sparse, repeatable, and reliable keypoints.

DEep Local Feature (DELF) was developed by Noh in 2017 [2], and it was mainly created for large-scale image retrieval. DELF is learned with weak supervision in the form of image-level labels only, coupled with a semantic feature selection mechanism. The DEep Local and Global features (DELG) extends DELF and was presented by Cao in 2020 [3]. DELG combines generalized mean pooling for global features and attentive selection for local features.

Finally, in 2018, DeTone published SuperPoint [7] that is a self-supervised framework for training interest point detectors and descriptors suitable for a large number of multiple-view geometry problems. For matching two sets of local features, Sarlin developed SuperGlue in 2020 [4]. The SuperGlue is a neural network that matches two local features sets by jointly finding correspondences and rejecting non-matchable points. It learns matching from existing local features using a neural network architecture.

2.3 Requirements: Keypoints Detector, Descriptor, and Matcher

Keypoints detectors, descriptors, and matchers are needed to specify a set of needs that each system should fulfill as well as possible. Every step has a purpose in the pipeline, but it detects-describe-match keypoints in most scenarios. The detection and description steps can be tied together in some state-of-the-art methods and systems, and it is represented, e.g., by one detect-and-describe CNN. Next will be specified requirements, which should each step meet.

Detector

The requirements for a keypoint detector can be defined in the following manner. It is demanded to fulfill multiple criteria for obtaining the best possible results. One requirement is that as much as possible true keypoints and as least as possible false keypoints should be detected. This condition is application-dependent, and the trade-off between detection speed and the number of detected keypoints should be considered. Detected keypoints should be well localized to prevent mismatching. A detector should have a reasonable repeatability rate, implying good stability of detected keypoints. In most demanding tasks, it is necessary to detect keypoints robustly due to visual noise. Nowadays, real-world applications are very computationally demanding, and keypoint detector should aim to be very computationally efficient in a production environment.

Descriptor

Once the keypoint is detected at a specific location, scale, orientation, then the local descriptor can be constructed based on these parameters around each keypoint in an invariant manner. The invariance is task-dependent. The descriptor should be aligned with the keypoint orientation and proportional to its detected scale. The following descriptor requirement is to describe keypoint as unique as possible, even in the case of multiple object instance occurrence in the same scene, e.g., windows, fence plank. The descriptor should take as least storage as possible but still maintain its uniqueness. The current visual feature representation is vector representation, and it can be floating-point or binary descriptors, which directly influence the storage demands [35].

Matcher

Image matching or feature matching is establishing correspondences between two images of the same scene or object. A common approach to image matching consists of detecting a set of keypoints from images and representing the local neighborhood around each keypoint by a feature descriptor. A simple similarity measure finds the Euclidean norm between the two feature descriptors. This measure performs well as the correct matches need to have the closest match significantly closer than the closest incorrect match to achieve reliability.

Establishing feature correspondences by finding the nearest neighbor in the descriptor space is very computationally demanding, and a matcher should address this. The next difficult challenge is to discard ambiguous matches and matches that are not correct in the other image. Therefore, a more precise way is needed to discard the features that do not have suitable matches.

Summary

This section mentioned general keypoint detector, descriptor, and matcher requirements. Depending on the use case where methods are applied, there can be many more requirements.

A few state-of-the-art methods (e.g., Detect-and-Describe Network (D2-Net) Section 4.2 and Repeatable and Reliable Detector and Descriptor (R2D2) Section 4.3) combine detecting and describing phases into one step. It is called detect-and-describe approach and it is represented by a single CNN.

2.4 Problem Statement

The problem statement should define the ideal and final state when a problem is fully solved. Furthermore, an ideal solution for the long-term visual localization can be presented as follows: for every query image, a system should return a precise camera position and camera rotation. The precise localization would be achieved during all year seasons and the day and night.

Current state-of-the-art results are far from ideal, and methods can perform poorly in bad weather conditions. Many researchers, startups, and global companies try to get the best possible solution for real-world applications.

The solutions can be implemented in many real-life applications like autonomous cars that can influence everyone's life.

Chapter 3

Dissertation Goals

By evaluating the current state-of-the-art methods and visual features used for long-term visual localization can be stated that there is still enough space for improvement.

This dissertation thesis has five main goals. Firstly, it maps the datasets used for long-term visual localization and selects viable datasets for further evaluation. Next, it selects one of the current state-of-the-art methods or approaches and enhances it. Then, it shows that parts of an image are unnecessary for long-term visual localization because they do not contain any helpful information and can be removed. Furthermore, this thesis tries to fuse semantic segmentation into a selected keypoint detector. Finally, the author achieves state-of-the-art results on a selected dataset by applying a developed keypoint filtering approach.

3.1 Selecting Dataset for Evaluation

The problem of finding a dataset that can be used for evaluation long-term visual localization is still recent. Many different datasets were released and used for competitions.

This goal seeks to find a dataset and an evaluation service that can be used for method comparison. The output should describe datasets, select one or more datasets for further evaluation, and an evaluation service or approach. Creating a new dataset or evaluation service is not part of this work.

3.2 Enhanced Parameter Setting

After selecting a benchmarking dataset and evaluation service from the previous step, the current state-of-the-art methods will be used as a baseline. Carefully fine-tuning parameters are supposed to achieve better results for the selected benchmarking dataset.

New and improved state-of-the-art results on the selected dataset should be achieved. Selecting a state-of-the-art pipeline for long-term visual localization with carefully fine-tuning selected methods parameters should improve current results. Developing a new keypoint detector, descriptor, matcher, and long-term visual localization pipeline is not part of this work.

3.3 Preprocessing for Keypoint Masking

In recent research [36, 37], additional information was used for improving localization results. The additional information is not presented by default in a dataset, and it can be semantic segmentation, instance segmentation, or estimated depth. It shows that image preprocessing is still a proper way to contribute to the state-of-the-art.

This goal reveals that some image parts are unnecessary for long-term visual localization. Furthermore, they do not bring any valuable information and can be masked or filtered out. After removing unnecessary image parts, the overall visual localization pipeline should perform at least without modification. Developing a new semantic segmentation, instance segmentation, and depth estimation method is not part of this work.

3.4 Fusion of Training Information

Since most of the current state-of-the-art keypoint detectors are based on neural network models thus, training data always plays an important role. A fusion of training information can boost performance (e.g., not all keypoints will need to be detected or described).

The goal leads to experiments with the current implementations of keypoint detection methods and modifying the input training data in a way that could help to embed, e.g., semantic segmentation information into the current models. It may not be successful without modifying network architectures. The development or modification of a keypoint detector or detector model is not in the scope of this goal.

3.5 Keypoint Filtering

The problem of general keypoint detectors is that they detect all keypoints regardless of how beneficial the keypoints are. Some keypoints do not add any meaningful information for solving the task of long-term visual localization. That means that they will never match, or if they match, it can be the wrong match. They represent an unwanted noise that is added to the localization pipeline. This challenging problem needs to be addressed.

The last goal aspires to find an approach how to remove keypoints that are not valuable and helpful (e.g., keypoints on a dynamic object like a car and pedestrian). The new method/system/approach should be introduced. It may achieve better long-term visual localization results than results obtained by a selected state-of-the-art method for a selected dataset.

Part II

Methodology

Chapter 4

State-of-the-art Methods

This chapter describes state-of-the-art keypoint detectors, descriptors, matchers, and other related methods. In many cases, the state-of-the-art methods are based on end-to-end ANNs training with a fine-tuned optimization and loss. Some of the described methods combine detection and description phases for obtaining better results.

4.1 Network of Vector of Locally Aggregated Descriptors

In general, Network of Vector of Locally Aggregated Descriptors (NetVLAD) is considered to be a state-of-the-art descriptor for solving image classification. To understand how NetVLAD was developed and used, it is necessary to introduce two methods that preceded its development. The first method is Bag of Words (BoW). The second one is referred as Vector of Locally Aggregated Descriptors (VLAD) and is based on BoW. All three methods are described in detail below to show the gradual development.

4.1.1 Bag of Words

Bag of Words (BoW) is also known as Bag of Visual Words. It is a method for image classification [32]. The method treats image features as words if the features consist of keypoints and descriptors. The keypoints are considered to be essential image points. By using the keypoints and descriptors, the vocabularies are created. Consequently, every image can be represented as a frequency histogram of features in the image. The frequency histogram helps find similar images or predict the category of the image.

The building of the BoW includes detecting features, extracting descriptors from each image in the dataset, and building a visual dictionary. The detecting and extracting is executed using feature extractor algorithms such as SIFT or KAZE.

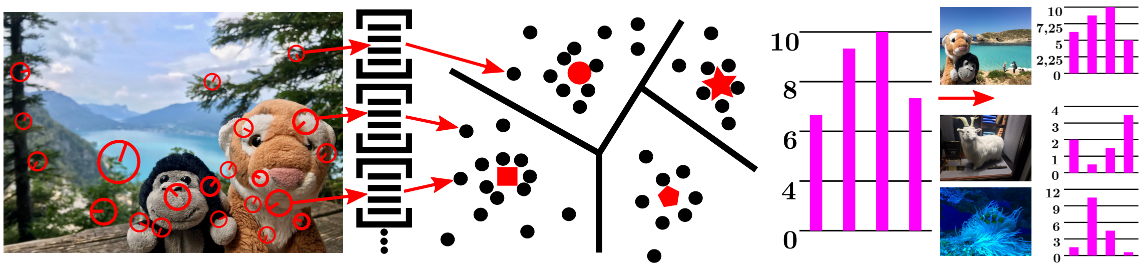

Figure 4.1: The visualization of Bag of Words (BoW) pipeline (from left to right).

Thus, the clusters are made from the descriptors by using K-Means, DBSCAN, or some similar clustering algorithm. The center of each cluster is then used as the visual dictionary's vocabulary. In the last step, the frequency histogram is created from the vocabularies and the frequency of the image's vocabularies. Furthermore, these histograms represent the Bag of Words (BoW). The entire process is visually described in Figure 4.1.

4.1.2 Vector of Locally Aggregated Descriptors

Vector of Locally Aggregated Descriptors (VLAD) is supposed to be an extension of BoW concept. In praxis, it means that the regions are extracted and described by using descriptors. Then, each descriptor is assigned to the closest cluster of a vocabulary. Residual of each descriptor is accumulated. The residual is represented as a vector of differences between descriptors and cluster centers [31]. Moreover, the residuals are concatenated into a single-dimensional descriptor.

Adding the difference of each descriptor leads to the more discriminative property in a feature vector. And it is the first advantage of VLAD compared to the BoW. This first-order statistic provides better discrimination for a classifier.

Mathematically, the process starts just like in the case of BoW. Firstly, a codebook from the descriptors from a training dataset is trained, as $C = \{c_1, c_2, \cdots, c_k\}$ where k is a number of "visual words" usually obtained by K-means clustering. Then, each local descriptor x is associated to its nearest visual word $c_i = \mathrm{NN}(x)$. Consequently, for each visual word c_i the differences $x - c_i$ of the vectors x assigned to c_i are accumulated, see Figure 4.2. It represents the distribution of the vectors with respect to the center. Each local descriptor is d-dimensional and there are N descriptors. The VLAD vector for each image is represented by $v_{i,j}$ where $i = 1, \cdots, k$ and $j = 1, \cdots, d$. Thus, a component of v is obtained as a sum over all the image descriptors as follows

$$v_{i,j} = \sum_{x \text{ such that } \mathrm{NN}(x)=c_i} (x_j - c_{i,j}) \tag{4.1}$$

which can be rewritten as

$$v_{i,j} = \sum_{n=1}^{N} a_i(\boldsymbol{x}_n)(x_{n,j} - c_{i,j}) \tag{4.2}$$

where $a_i(\boldsymbol{x}_n)$ denotes the membership of the descriptor $\boldsymbol{x}_n$ to i-th visual word, i.e. it is 1 if cluster c_i is the closest cluster to descriptor x_n and 0 otherwise.

The vector $\boldsymbol{v}$ is consequently normalized with its L2-norm as

$$\boldsymbol{v} := \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}. \tag{4.3}$$

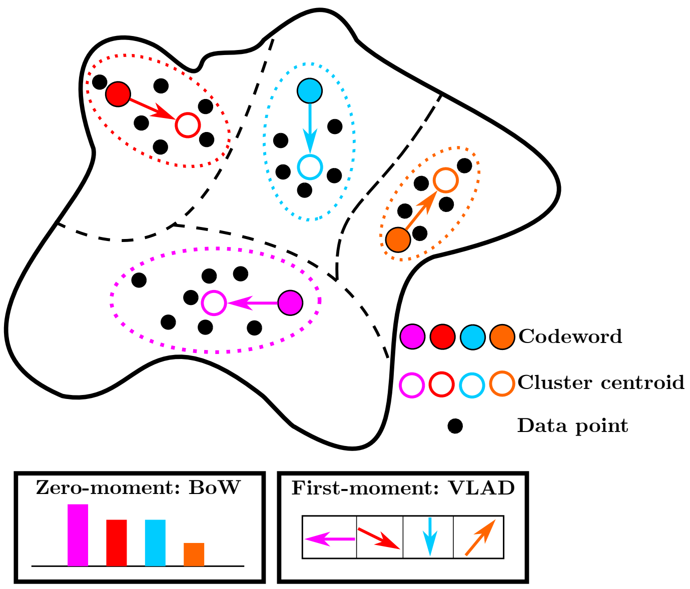

Figure 4.2: Visualization of VLAD's moments.

There are some extensions of VLAD using various normalization options. In [38] there are proposed several normalization techniques, including intra-normalization and power normalization along with a spatial extension which is called MultiVLAD. This extension proves the benefits of recording multiple VLADs for an image and shows that retrieval performance is improved for small objects.

4.1.3 Network of Vector of Locally Aggregated Descriptors

Network of Vector of Locally Aggregated Descriptors (NetVLAD) is a new generalized VLAD layer for image representation commonly used in image retrieval.

The layer is readily pluggable into any ANN architecture and amenable to training via backpropagation [30]. The layer takes as input N D-dimensional local image descriptors, K cluster centres as VLAD parameter, and outputs VLAD image representation V. For constructing the layer, it is required that a layer's operation is differentiable to all its parameters and the input.

Since VLAD shows discontinuities whose source is the hard assignment $a_i(\boldsymbol{x}_n)$ from Equation 4.2, it is necessary to make the operation differentiable and replace it with soft assignment of descriptors to multiple clusters

$$\bar{a}_i\left(\boldsymbol{x}_n\right) = \frac{\exp^{-\alpha\|\boldsymbol{x}_n - \boldsymbol{c}_i\|^2}}{\sum_{i'} \exp^{-\alpha\|\boldsymbol{x}_n - \boldsymbol{c}_{i'}\|^2}}, \tag{4.4}$$

where α is a parameter (positive constant) that controls the decay of the response with the magnitude of the distance. By expanding the squares, it results in

$$\bar{a}_i\left(x_n\right) = \frac{\exp^{\boldsymbol{w}_i^\top \boldsymbol{x}_n + b_i}}{\sum_{i'} \exp^{\boldsymbol{w}_{i'}^\top \boldsymbol{x}_n + b_{i'}}} \tag{4.5}$$

where vector $\boldsymbol{w}_i = 2\alpha\boldsymbol{c}_i$ and scalar $b_i = -\alpha\|\boldsymbol{c}_i\|^2$.

By plugging the soft-assignment in Equation 4.5 into the VLAD descriptor in Equation 4.2 it leads to

$$v_{i,j} = \sum_{n=1}^{N} \frac{\exp^{\boldsymbol{w}_i^\top \boldsymbol{x}_n + b_i}}{\sum_{i'} \exp^{\boldsymbol{w}_{i'}^\top \boldsymbol{x}_n + b_{i'}}} \left(x_{n,j} - c_{i,j}\right) \tag{4.6}$$

where $\{\boldsymbol{w}_i\}$, $\{b_i\}$, and $\{\boldsymbol{c}_i\}$ are three independent sets of trainable parameters for each cluster compared to one set for original VLAD. This ability helps NetVLAD to be more flexible. All parameters of NetVLAD are learned for the specific task in an end-to-end manner.

This descriptor was used together with the CMU-Seasons dataset and the Symphony Lake dataset for tasks of image-based place recognition on the bucolic environment across seasons in [33].

4.2 Detect-and-Describe Network

The method was firstly introduced in [1] and gets its name from Detect-and-Describe Network (D2-Net) approach. It addresses the problem of finding reliable pixel-level correspondences under difficult image conditions. D2-Net solves this problem by an approach where a single CNN plays the main role. It is simultaneously a dense feature descriptor and a feature detector. The detection is postponed to a later stage, and obtained keypoints are more stable than their traditional counterparts based on the early detection of low-level structures. The D2-Net model is trained by using pixel correspondences extracted from readily available large-scale SfM reconstructions, without any further annotations.

Left part of Figure 4.3 illustrates the common pipeline of using hand-crafted detectors [11, 17, 15, 39, 40] and descriptors [17, 15, 20, 21, 22]. The methods where either the descriptor [41, 42, 43] or detector [44, 45], or both detector and descriptor [46, 47] are replaced with a learned alternative exist.

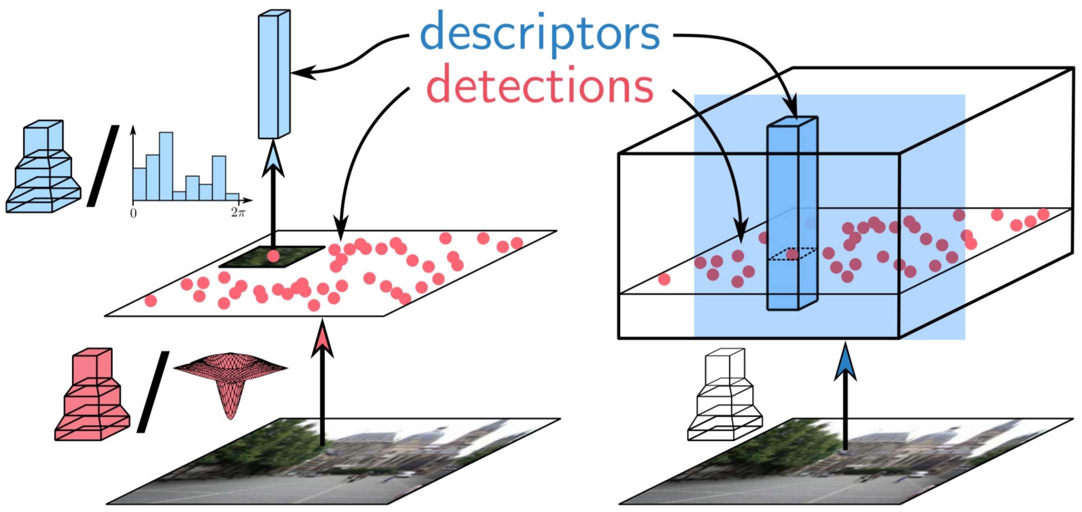

Figure 4.3: The figure shows the comparison between different approaches for feature detection and description. The left pipeline corresponds to a standard two-stage detect-then-describe approach. In contrast, D2-Net pipeline (right) uses a single CNN which extracts dense features that serve as both descriptors and detectors [1].

The feature detector often considers only small image regions [46] for speeding up the detecting process and typically focuses on low-level structures such as corners [11] or blobs [15]. The descriptor then captures higher-level information in a larger patch around the keypoint.

D2-Net proposes a single branch describe-and-detect approach to sparse feature extraction, as shown in Figure 4.3 (right). This approach can detect keypoints belonging to higher-level structures and locally unique descriptors.

4.2.1 Feature Description

At first, for obtaining 3D tensor F is needed to apply a CNN $\mathcal{F}$ on the input image I. The 3D tensor F is defined as $F = \mathcal{F}(I), F \in \mathbb{R}^{h \times w \times n}$, where $h \times w$ is the spatial resolution of the feature maps and n is the number of channels.

Similarly to other works [2, 48, 49] 3D tensor F is defined as a dense set of descriptor vectors $\boldsymbol{d}$

$$\boldsymbol{d}_{ij} = F_{ij}, \boldsymbol{d} \in \mathbb{R}^n, \tag{4.7}$$

with $i = 1, \cdots, h$ and $j = 1, \cdots, w$.

The descriptor vectors can be compared between images to establish correspondences using the Euclidean distance. The L2-normalization is applied to the descriptors prior to compare them

$$\boldsymbol{d} := \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|}. \tag{4.8}$$

During the training stage, descriptors are adjusted such that the same points in the scene produce similar descriptors.

4.2.2 Feature Detection

A different interpretation of the 3D tensor F [50] is as a collection of 2D responses D

$$D^k = F_{::k}, D^k \in \mathbb{R}^{h \times w}, \tag{4.9}$$

where $k = 1, \cdots, n$. In this representation, the feature extraction function $\mathcal{F}$ can be thought of as n different feature detector function $\mathcal{D}^k$, each producing a 2D response map D^k. The detection response maps are analogous to DoG response maps obtained by SIFT. The raw scores are post-processed, and only a subset of locations is selected as the output keypoints.

Hard feature detection

It exists multiple detection maps D^k $(k = 1, \cdots, n)$ and a detection can take place on any of them. The requirement for a point detection (i, j) is that detection D^k_{ij} is a local maximum in D^k, with $k = \underset{t}{\operatorname{argmax}}\, D^t_{ij}$. This corresponds, for each pixel (i, j), to select the most preeminent detector $\mathcal{D}^k$ (channel selection) and verify whether there is a local-maximum at position (i, j) on that particular detector's response map D^k.

Soft feature detection

During the training process, there is the hard-feature detection stage softened to be amenable for backpropagation.

First, soft local-maximum score is defined as

$$\alpha^k_{ij} = \frac{\exp\left(D^k_{ij}\right)}{\sum_{(i',j') \in \mathcal{N}(i,j)} \exp\left(D^k_{i'j'}\right)}, \tag{4.10}$$

where $\mathcal{N}(i, j)$ is set of 9 neighbors of pixel (i, j), including pixel itself.

Then the soft channel selection (that emulates channel-wise non-maximum suppression) is defined as follows

$$\beta_{ij}^{k} = \frac{D_{ij}^{k}}{\max\limits_{t} D_{ij}^{k}}. \tag{4.11}$$

Considering both criteria, a single score map is defined

$$\gamma_{ij} = \max_{k} \left(\alpha_{ij}^{k} \beta_{ij}^{k}\right), \tag{4.12}$$

where k are all feature maps.

Finally, the soft detection score s_{ij} at pixel (i, j) is defined as an image-level normalization

$$s_{ij} = \frac{\gamma_{ij}}{\sum_{(i',j')} \gamma_{i',j'}}. \tag{4.13}$$

Multiscale Detection

CNN descriptors have a certain degree of scale invariance, but they tend to fail in cases with significant viewpoint changes. To obtain more robust features, D2-Net proposes image pyramids [51] which are used only for the test stage.

The input image I has an image pyramid I^{ρ} that contains 3 different resolutions $\rho = 0.5, 1, 2$ for extracting feature maps F^{ρ}. Then, the larger image structures are propagated from the lower resolution feature maps to the higher resolution ones, in the following way

$$\tilde{F}^{\rho} = F^{\rho} + \sum_{\gamma < \rho} F^{\gamma}. \tag{4.14}$$

F^{γ} are resized to the resolution of F^{ρ} using bilinear interpolation.

4.2.3 Training loss

The loss $\mathcal{L}$ jointly optimizes the detection and description objectives. D2-Net extends triplet margin ranking loss, which has been successfully used for descriptor learning [41, 52], also account for the detection stage. D2-Net loss that has been used for CNN model training is defined as

$$\mathcal{L}\left(I_1, I_2\right) = \sum_{c \in \mathcal{C}} \frac{s_c^{(1)} s_c^{(2)}}{\sum_{q \in \mathcal{C}} s_q^{(1)} s_q^{(2)}} m\left(p\left(c\right), n\left(c\right)\right), \tag{4.15}$$

where $s_c^{(1)}$ and $s_c^{(2)}$ are soft detections (from Equation 4.13) at points A and B in I_1 and I_2. $\mathcal{C}$ is the set of all correspondences between I_1 and I_2. The $m\left(c\right)$ is a triplet margin ranking loss with $p(c)$ positive and $n(c)$ negative descriptor distance.

The loss produces a weighted average of the margin terms m overall matches based on their detection scores. The most distinctive correspondences will get higher relative scores and vice-versa. Correspondences with higher relative scores are encouraged to have similar descriptors distinctive from the rest.

4.2.4 Summary

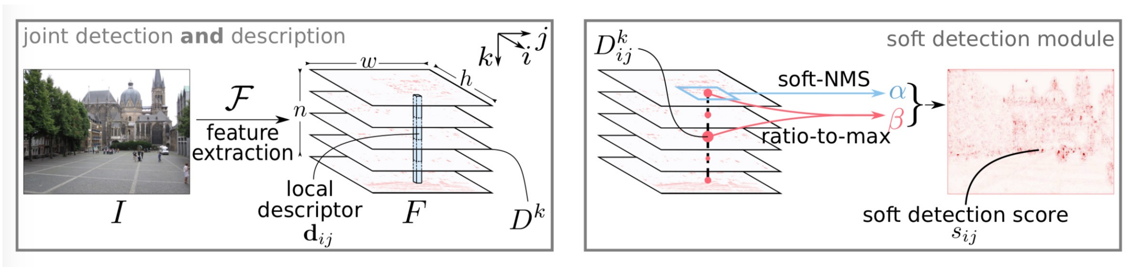

Figure 4.4: D2-Net detect-and-describe network [1].

For network overview see Figure 4.4. In D2-Net feature extraction CNN $\mathcal{F}$ is used to extract feature maps that play a dual role:

1. Local descriptors $\boldsymbol{d}_{ij}$ are simply obtained by traversing all the n feature maps D^k at a spatial position (i, j).

2. Detections are obtained by performing a non-local-maximum suppression on a feature map followed by a non-maximum suppression across each descriptor. During training, keypoint detection scores s_{ij} are computed from a soft local-maximum score α and a ratio-to-maximum score per descriptor β.

4.3 Repeatable and Reliable Detector and Descriptor

The Repeatable and Reliable Detector and Descriptor (R2D2): since descriptors should be learned only in regions for which matching can be performed with high confidence, this should be obtained by learning keypoint detection and description together with a predictor of the local descriptor discriminations. This process prevents working with ambiguous areas and leads to reliable keypoint detections and descriptions [34].

Since a robust feature detector enables robust visual localization, structure-based methods perform well, and the critical part is feature extraction and matching. Compared to classical methods working on the detect-then-describe principle, the

R2D2 works on the detect-and-describe approach. It means keypoints are detected and described at once. It allows simultaneously output sparse, repeatable, and reliable keypoints that outperform state-of-the-art detectors and descriptors on the HPatches dataset [53].

The repeatability is represented by image locations that are invariant to usual image transformations. Further, reliability is ensured by good image locations (discriminative and robust) for matching purposes.

4.3.1 R2D2 Architecture

Since the R2D2 focus on predicting a set of sparse locations of an input image that are repeatable and reliable for local feature matching, it is necessary to make an explicit distinction between repeatability and reliability. In order to keep this rule, it is required to predict separately two complementary aspects.

Therefore, a fully-convolutional network is trained and there are predicted 3 outputs of image with size $H \times W$. The first output is 3D tensor $\boldsymbol{X} \in \mathbb{R}^{H \times W \times D}$ corresponding to a set of dense D-dimensional, one per pixel. And the second one is a heatmap $\boldsymbol{S} \in [0, 1]^{H \times W}$. This heatmap serves to provide sparse and repeatable keypoint locations. The sparsity is ensured by extracting keypoints at locations corresponding to local maxima in $\boldsymbol{S}$, once the heatmap is trained to contain strong and repeatable local maxima. The third and last output is referred to a reliability map $\boldsymbol{R} \in [0, 1]^{H \times W}$ defining the estimated reliability of descriptor $\boldsymbol{X}_{ij}$ at each pixel (i, j) if $i = 1, \cdots, W$ and $j = 1, \cdots, H$.

The entire architecture is shown in Figure 4.5. However, the backbone is inspired by L2-Net. The last 8×8 convolutional layer is replaced by three 2×2 convolutional layers. The replacement reduces the number of weights by a factor of 5 for a similar or slightly better accuracy. And the 128-dimensional output tensor works as an input to:

- L2-normalization layer to obtain descriptors $\boldsymbol{X}$,
- elementwise square operation followed by a 1×1 convolutional layer and softmax function to obtain the reliability confidence value $\boldsymbol{R}$ of each descriptor,
- the same operations to obtain the repeatability map $\boldsymbol{S}$.

This learning-based feature extraction method detects and describes keypoints in images. The method learns both keypoint repeatability and confidence for keypoint reliability from relevant training data. The network is trained with self-supervision using a mixture of synthetic (images with known transformations) and real data (point correspondences). The R2D2 works perfectly for visual localization and can also obtain good results compared to state-of-the-art methods.

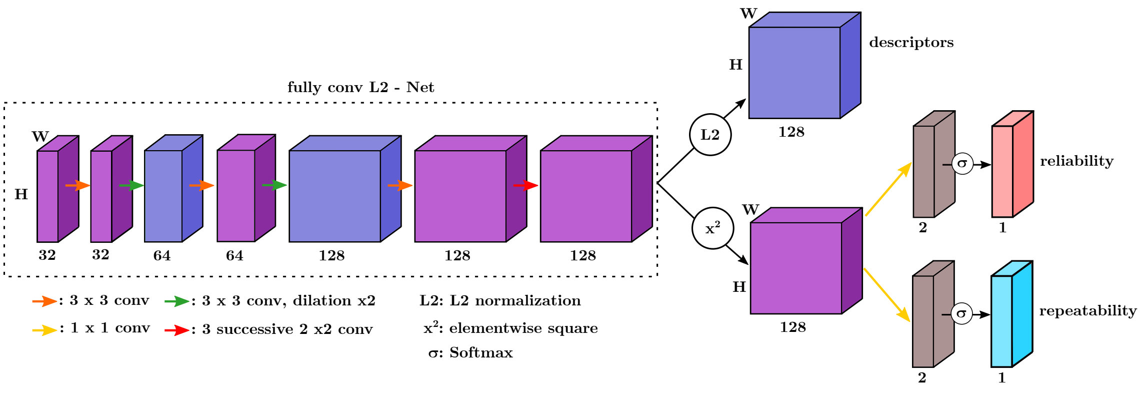

Figure 4.5: Overview of network for jointly learning repeatable and reliable matches.

4.4 Visual Localization Using Semantic Segmentation and Depth Prediction

Visual localization is one of the main challenges for numerous computer vision applications, such as augmented reality, intelligent robotics, or long-term visual navigation of autonomous driving [54, 55].

A core task for the applications is to find the precise location of any query image with a map of 3D points reconstructed by SfM with database images [56, 57, 58]. The map is typically used to describe the position of landmarks [59, 60] that are pre-collected from the point features extracted from the database images.

During localization, the points in the 2D query image and the points with the 3D world coordinate in the map are determined, and finally, the six DoF pose of the query image is recovered with Perspective-n-Point (PnP) [61].

This method [36] extends common visual localization pipeline with Semantic Consistency Check (SCC), Depth Consistency Verification (DCV), and Weighted-RANdom SAmple Consensus (RANSAC).

The localization pipeline is shown in Figure 4.6 and is based on a standard retrieval-based framework [62]. It consists of three main modules as follows.

1. **Map construction:** From captured database images standard SfM algorithm constructs a 3D model that is represented by a large number of 3D interest points. The 2D-3D keypoint matching relations produced during SfM are recorded, semantic segmentation is estimated for every database image, and semantic labels are associated with the 3D points.

2. **Image retrieval:** A coarse search is performed by matching the query with the database images using global descriptors computed by a deep neural network [2]. This process is efficient given that there are far fewer database images than the 3D points in the SfM model. This problem was handled by calculating homography matrices to reject outliers and by Semantic Consistency

Weight (SCW) for reranking the retrieval results with a verification step.

3. **Feature matching and pose estimation:** 2D interest points are extracted and matched, 2D-3D query-database matching relations are built for PnP pose estimation. Multiple learning-based feature extractors are used for more feature points, and SCC and DCV are performed for semantic and depth consistency verification. All matching relations are weighted with consistency scores for bias sampling during the weighted-RANSAC and PnP.

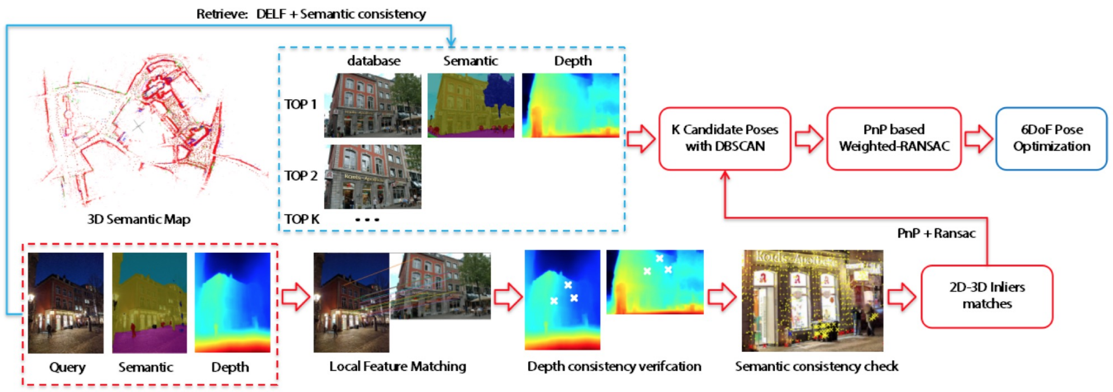
Retrieve: DELF + Semantic consistency
database
Semantic
Depth
TOP 1
TOP 2
TOP K
3D Semantic Map
K Candidate Poses with DBSCAN
PnP based Weighted-RANSAC
6DoF Pose Optimization
PnP + Ransac
2D-3D Inliers matches
Query
Semantic
Depth
Local Feature Matching
Depth consistency verifcation
Semantic consistency check

Figure 4.6: An overview diagram of the pipeline.

4.4.1 Semantic Verification of Image Retrieval

Learning-based approaches can achieve good performance in the image retrieval task, see [2, 30, 63, 64, 65]. The DELF from Section 4.5 was used for obtaining a predefined number of candidate images. However, a significant amount of incorrect results is generated. This problem is handled by the SCW and a clustering process.

Current semantic segmentation approaches are mainly based on DNNs, such as PSPNET [66], BiSeNet [67], and DeepLab v3+ [68] which was used. For every retrieved image, the homography matrix from the query image is computed by RANSAC, producing several inliers S_c. The number of matches with identical semantic label S_f. SCW is defined by

$$S_R = \frac{\alpha_1 S_c + \alpha_2 S_f}{\alpha_1 + \alpha_2}, \tag{4.16}$$

where α_1 and α_2 are scoring weight scalars. SCW is then calculated for all query-retrieval pairs, and only those with high SCW scores are proceeded further. The inaccurate semantic prediction leads to a number of incorrect candidates which are filtered out by a standard clustering technique.

Semantic consistency eliminates the inconsistent matches with the SCC. The semantic labels are compared for an output matched pair of features, and pairs with different labels are discarded. Features points on dynamic objects (pedestrians and vehicles) are also removed. SCC is applied before RANSAC.

4.4.2 Multiple Feature Extractors and Enhanced R2D2 Descriptor

Learning-based feature extractors are trained on large datasets and can automatically discover feature extraction representation that is the most suited to the data. SuperPoint [7] from Section 4.7 and R2D2 [34] from Section 4.3 are among the most recent approaches that achieve impressive performance. Experiments show [36] that a combination of features by both detectors results in a significant increase in the number of matched pairs and better pose estimation accuracy.

The R2D2 confidence of a matching result is closely related to the similarity between the descriptor score of the two features. Thus, R2D2 descriptor incorporates the detection confidence score by

$$d' = d \times s, \tag{4.17}$$

where d is the original descriptor, s is the descriptor score and d' is the new descriptor.

4.4.3 Depth Consistency Verification

Recent monocular depth prediction also enables a strong depth constraint on feature matching for accurate visual localization. DCV effectively identifies fallacious matched features associated with low depth consistency.

SfM produces a cloud of 3D points with a set of database images. After feature matching is completed and correct database images are retrieved, the correspondence between query and 3D points is built. The depth estimate of one of the feature points is found by projecting the corresponding 3D point to the image plane

$$P_{\text{img}} = K\left(RP_{\text{wrd}} + T\right), \tag{4.18}$$

where P_{wrd} is the coordinate of 3D point, P_{img} is the image coordinates, K is the intrinsic matrix, R is the estimated rotation matrix and T is the estimated translation matrix with PnP. With known x and y coordinates the scale of P_{img} can be determined hence its z coordinate, or the Estimated Depth Value (EDV). This allows to check the depth value against the one obtained from a deep network [69] or Predicted Depth Value (PDV).

The scale of the PDV is often unknown, it cannot be compared directly with

EDV. A solution is to compare their ordinal value but the depth values of different keypoints are not evenly spaced. This issue is overcome by the Adaptive Ordinal Cost (AOC) which is defined as

$$C = |D_i - D_j|, \tag{4.19}$$

where C is the AOC, D is the PDV, i and j are ordinal values in the EDVs and PDVs respectively. After AOC is estimated for all feature points, they are divided by their mean to cope with the unknown scale.

4.4.4 Pose Estimation With Weighted-RANSAC

A standard RANSAC PnP computes a hypothetical pose M with a subset of input matches, and all the other samples are tested by computing reprojection error e_p with the pose. A sample p is accepted as an inlier if the error is lower than a preset threshold e_t, shown by the following indicator function

$$T\left(p, M\right) = \begin{cases} 1, & e_p < e_t \\ 0, & \text{otherwise.} \end{cases} \tag{4.20}$$

The process is repeated for a random samples until enough inliers are found. The semantic information is leverage by the weighted-RANSAC scheme, where the threshold is reduced according to semantic consistency, defined by the new indicator function

$$w_\mu T\left(p, M\right) = \begin{cases} 1 - \left(\frac{e_p}{\mu e_t}\right)^2, & |e_p| < \mu e_t \\ 0, & \text{otherwise,} \end{cases} \tag{4.21}$$

where w_μ is weight μ and μ is the reduction ratio calculated by normalizing SCW from equation 4.16.

4.5 Deep Local Feature

DEep Local Feature (DELF) is a local feature descriptor created specifically for large-scale image retrieval as a fundamental task in computer vision. Since DELF is learned with weak supervision, it uses image-level labels only, and it is coupled with a mechanism for semantic feature selection, which shares most network layers with the descriptor. This framework can be used for image retrieval as a drop-in replacement for other keypoint detectors and descriptors, enabling more accurate feature matching and geometric verification. DELF achieves excellent performance when combined with global descriptors as well [2].

The feature descriptor DELF is coupled with the attention model. It is possible to use the CNN architecture and generate feature scores using very little extra computation. This process allows the extraction of both local descriptors and keypoints via one forward pass over the network, which is very convenient.

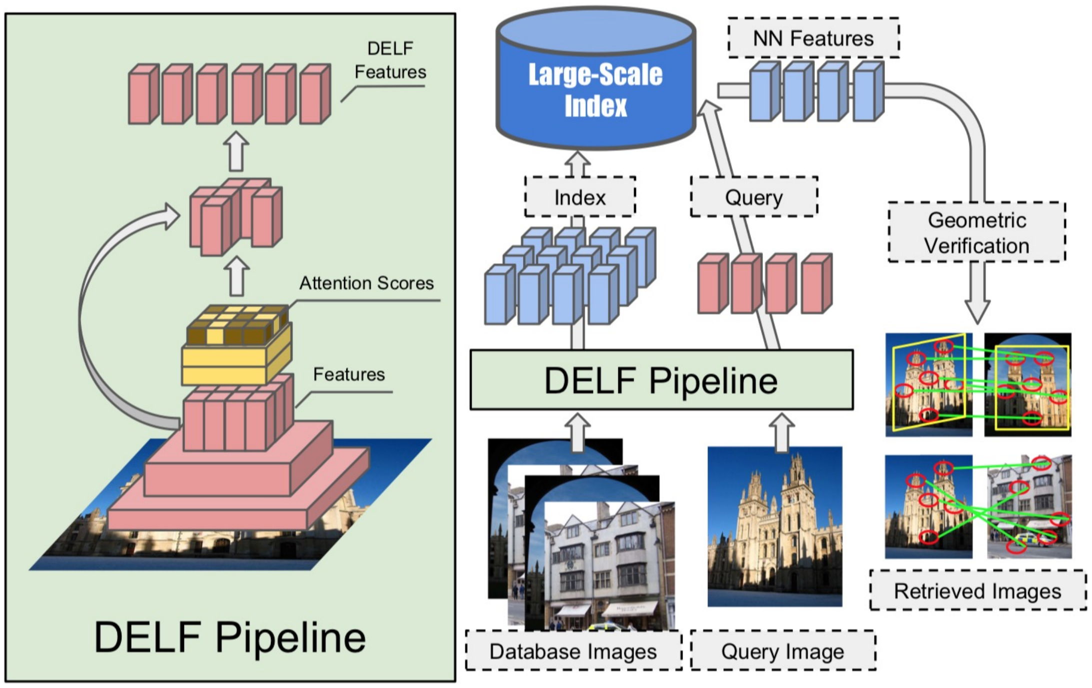

Figure 4.7: Overall architecture of image retrieval system, using DEep Local Feature (DELF) and attention-based keypoint selection [2]. On the left, it illustrates the pipeline for extraction and selection of DELF. On the right, it illustrates large-scale feature-based retrieval pipeline.

4.5.1 Dense Localized Feature Extraction

At first, the dense features are extracted from an image by applying the Fully Convolutional Network (FCN) that is designed according to the ResNet50 model [70]. Thus, an image pyramid is built direction up, and the FCN is applied for each level independently to handle scale changes. The obtained feature maps represent a dense grid of local descriptors. Features are used to being localized based on their receptive fields, whose pixel coordinates of the center serve as the feature location. Consequently, features that describe image regions of different sizes are obtained using the image pyramid.

The ResNet50 model is trained on ImageNet [71] as a baseline. Thus, it is possible to improve local descriptors and improve the discriminativeness. A landmark recognition application is used to annotate datasets of landmark images, and it is trained the network with a standard cross-entropy loss for image classification. The input images are then center cropped to produce square images and rescaled to 250 $\times$ 250. Random 224 $\times$ 224 crops can be thus used for training. It leads to learning representations of local descriptors.

4.5.2 Attention-based Keypoint Selection

The approach considers an adequate selection of a subset of the features. It does not operate using densely extracted features directly for image retrieval. A substantial part of these features would not be relevant to a recognition task, leading to distracting the retrieval process. Therefore, a keypoint selection is very significant for retrieval systems' accuracy and computational efficiency.

A landmark classifier is trained to measure relevance scores for local feature descriptors explicitly. A weighted sum pools the features while the attention network predicts the weights. The training procedure is similar to the one described in Section 4.5.1 including the loss function and datasets. An embedding for the whole input image is generated, and the image is used to train a softmax-based landmark classifier.

Compared to classical approaches including using of techniques such as SIFT and LIFT [46] where keypoints are first detected and later described, the using of DELF allows keypoint selection to come after descriptor extraction.

It is required to reduce the dimensionality of selected features to obtain improved retrieval accuracy. The selected features are L2-normalized, which leads to reducing the dimensionality to 40 by PCA. Consequently, the features are once again L2-normalized.

The pipeline illustrated on the right side in Figure 4.7 can be used for image retrieval tasks. DELF for database images are indexed offline. The index supports querying by retrieving nearest neighbor features to rank database images based on geometrically verified matches.

4.6 Deep Local and Global Features

DEep Local and Global features (DELG) was introduced in [3] and extends DELF from Section 4.5. DELG 's model combines generalized mean pooling for global features and careful selection for local features. The entire network can be learned end-to-end by balancing the gradient flow between two heads. It requires only image-level labels. DELG introduces an autoencoder-based dimensionality reduction technique for local features. It is integrated into the model for improving training efficiency and matching performance.

4.6.1 Model

DELG model is illustrated in Figure 4.8. It leverages hierarchical representations from CNNs [72] to represent the different types of features. Global features can be associated with deep layers representing high-level cues. Local features are more suitable to intermediate layers that encode localized information.

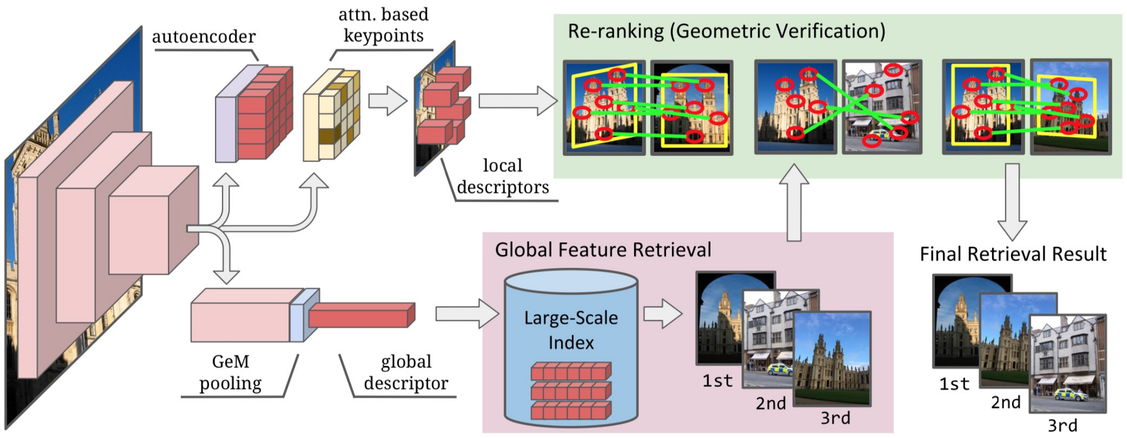

Figure 4.8: DELG model (left) jointly extracts deep local and global features. Global features are used in the first stage of a retrieval system, to efficiently select the most similar images (bottom). Local features can then be employed to re-rank top results (top-right), increasing precision of the system (for more details see [3]).

A CNN backbone is applied to obtain two feature maps $S \in \mathbb{R}^{H_S \times W_S \times C_S} and D \in \mathbb{R}^{H_D \times W_D \times C_D}$, representing shallower and deeper activations respectively, where H, W, and C correspond to the height, width and number of channels.

Generalized mean pooling [73] is used for aggregation deep activations into a global feature. Aggregated representation of global feature is whitening which is represented with a fully-connected layer $F \in \mathbb{R}^{C_F \times C_D}$, with learned bias $b_F \in \mathbb{R}^{C_F}$, similar to [74]. These two components produce a global feature $g \in \mathbb{R}^{C_F}$ that summarizes the discriminative contents of the whole image

$$g = F \times \left(\frac{1}{H_D W_D} \sum_{h,w} d_{h,w}^p \right)^{\frac{1}{p}} + b_F, \tag{4.22}$$

where p denotes the generalized mean power parameter, and the exponentiation $d_{h,w}^p$ is applied element wise.

For local features is important to select only the relevant regions for matching. It is achieved by using an attention module M [2]. The attention module predicts which extracted local features are discriminative for the objects of interest. This is performed as $\mathcal{A} = M(S)$, where M is a small convolutional network and $\mathcal{A} \in \mathbb{R}^{H_S \times W_S}$ denotes the attention score map associated with the features from S.

The extracted local feature at position h, w is represented with a local descriptor $l_{h,w} \in \mathcal{L}$ and its corresponding keypoint detection score $a_{h,w} \in \mathcal{A}$. Their locations in the input image are set to centers of corresponding receptive field [75].

The global and local descriptors are L2-normalized into $\hat{g}$ and $\hat{l}_{h,w}$.

4.6.2 Training

The model is trained using only image-level labels (see Figure 4.9). Patch-level supervision for training local features is not required. DELG discovers discriminative features by learning, distinguishing the different classes given by image-level labels. This weakly-supervised local feature setting is essential to control the gradient flow between learning the global and local features.

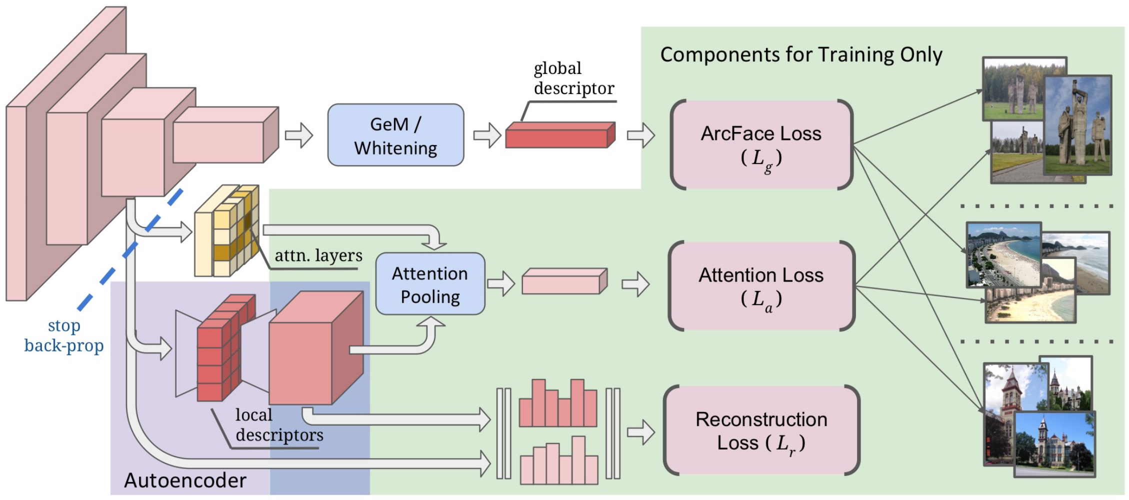

Figure 4.9: Training pipeline: the components highlighted in green are used solely during training. There are two classification losses: ArcFace for global feature learning and softmax for attention learning. In both cases, the classification objective is to distinguish different landmarks. The autoencoder (purple) is further trained with a reconstruction loss. The whole model is learned end-to-end [3].

Global Features

For global feature learning, DELG uses loss function with L2-normalized classifier weights $\hat{W}$, followed by scaled softmax normalization and cross-entropy loss [76]. Additionally, the ArcFace margin is used [77]. Concretely, given $\hat{g}$, it first computes the cosine similarity against $\hat{W}$, adjusted by the ArcFace margin. The ArcFace-adjusted cosine similarity can be expressed as

$$AF(u,c) = \begin{cases} \cos(\arccos(u) + m), & \text{if } c = 1 \\ u, & \text{if } c = 0 \end{cases} \tag{4.23}$$

where u is the cosine similarity, m is the ArcFace margin and c is a binary value indicating if this is the ground-truth class. The cross-entropy loss, computed using softmax normalization is

$$L_g\left(\hat{g}, y\right) = -\log\left(\frac{\exp\left(\gamma \times AF\left(\hat{w}_k^\top \hat{g}, 1\right)\right)}{\sum_n \exp\left(\gamma \times AF\left(\hat{w}_n^\top \hat{g}, y_n\right)\right)}\right), \tag{4.24}$$

where γ is a learnable scalar, $\hat{w}_i$ refers to the L2-normalized classifier weights for class n, y is the one-hot label vector and k is the index of the ground-truth class ($y_k = 1$).

Local Features

Two losses are used for training the local features. First, a mean-squared error regression loss that measures how well the autoencoder can reconstruct S. Denote $S' = T'\left(\mathcal{L}\right)$ (the local descriptors are obtained as $\mathcal{L} = T\left(S\right)$, where $\mathcal{L} \in \mathbb{R}^{H_S \times W_S \times C_S}$, where H, W, and C correspond to the height, width and number of channels in each case) as the reconstructed version of S, with same dimensions, where T' is a 1×1 convolutional layer. The loss is

$$L_r\left(S', S\right) = \frac{1}{H_S W_S C_S} \sum_{h,w} \|s'_{h,w} - s_{h,w}\|^2. \tag{4.25}$$

Second loss is a cross-entropy classification loss that incentivizes the attention module to select discriminative local features. It is done by pooling the reconstructed features S' with attention weights $a_{h,w}$

$$a' = \sum_{h,w} a_{h,w} s'_{h,w}. \tag{4.26}$$

Then using softmax-cross-entropy loss

$$L_a\left(a', k\right) = -\log\left(\frac{\exp\left(v_k^\top a' + b_k\right)}{\sum_n \exp\left(v_n^\top a' + b_n\right)}\right), \tag{4.27}$$

where v_n , b_n refer to the classifier weights and biases for class n and k is the index of the ground-truth class. The total loss is $L_g + \lambda L_r + \beta L_a$.

Controlling Gradients

The gradient back-propagation is stopped from L_r and L_a to the network backbone. It means that the network backbone is optimized solely based on L_g and will produce the desired hierarchical feature representation.

DELG and DELF Comparison

Comparation of DELG's training cost to DELF's: DELF require one additional run for attention learning, followed by a PCA computation step. DELG's advantage is that the attention and autoencoder layers are already adapting to the network backbone while it is training. For DELF, this process only happens after the backbone is fully trained.

4.7 SuperPoint

The SuperPoint is a self-supervised framework for training interest point detectors and descriptors that are suitable for many multiple-view geometry problems in computer vision [7]. It is based on a fully-convolutional model that operates on full-sized images and jointly computes pixel-level interest point locations and associated descriptors in one forward pass. It introduced Homographic Adaptation for boosting interest point detection repeatability. The SuperPoint keypoints and descriptors show excellent performance on the HPatches dataset.

4.7.1 SuperPoint Architecture

SuperPoint is designed as a fully-convolutional neural network architecture that operates on a full-sized image and produces interest point detections accompanied by fixed-length descriptors in a single forward pass (see Figure 4.10).

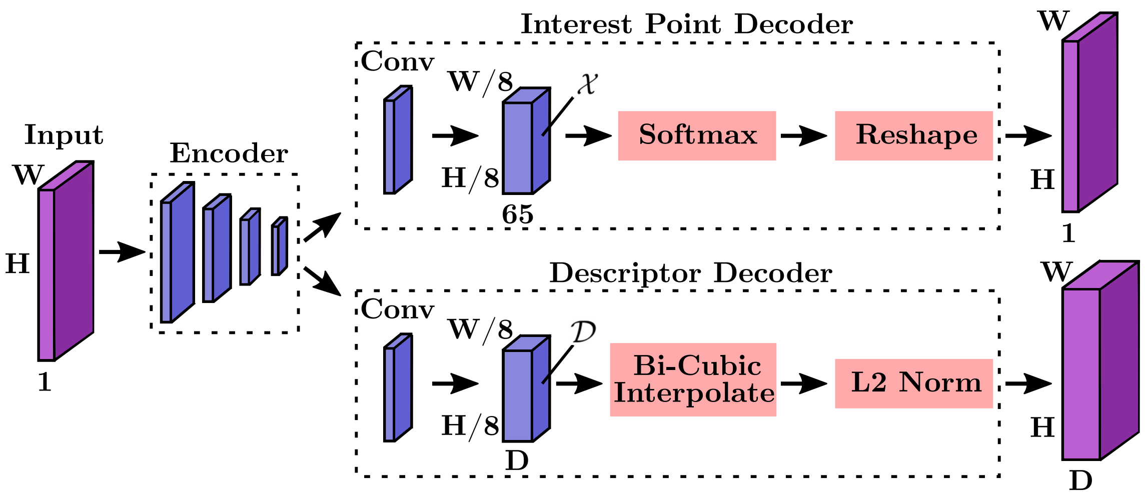

Figure 4.10: SuperPoint Decoders: both decoders operate on a shared and spatially reduced representation of the input. For fast and easy training, both decoders use non-learned upsampling to bring the representation back to $\mathbb{R}^{H\times W}$.

The architecture is split into two "heads" after the encoder part. Each "head" learns task-specific weights – one for interest point detection and the other one for interest point description.

Most of the network's parameters are shared between these two tasks. That is the difference from traditional systems. The detection and description tasks are handled separately, and the description task needs keypoints locations for its calculation.

Interest Point Decoder

In interest point detection decoder "head" each pixel of output corresponds to a probability of "point-ness" for that pixel in the input. The standard network design for dense prediction involves an encoder-decoder pair. The spatial resolution is decreased via pooling or stridden convolution and then upsampled back to full resolution via deconvolution operations, such as SegNet [78]. Upsampling layers add a large amount of computation and introduce unwanted checkerboard artifacts [79]. Thus, SuperPoint designs the interest point detection "head" with an explicit decoder[1] to reduce the computation of the model.

Descriptor Decoder

The descriptor "head" computes $\mathcal{D} \in \mathbb{R}^{H_c \times W_c \times D}$ and outputs a tensor with size $\mathbb{R}^{H \times W \times D}$. To output a dense map of L2-normalized fixed-length descriptors, SuperPoint first outputs a semi-dense grid of descriptors (e.g., one every 8 pixels). Learning descriptors semi-densely rather than densely reduces training memory and speeds up the training process. The decoder then performs the bicubic interpolation of the descriptor and then L2-normalizes the activations to be unit length (see Figure 4.10).

Loss Function

The final loss is the sum of two intermediate losses: one for the interest point detector, $\mathcal{L}_p$, and one for the descriptor, $\mathcal{L}_d$. SuperPoint originally uses pairs of synthetically warped images which have both

1. pseudo-ground truth interest point locations and
2. the ground truth correspondence from a randomly generated homography H which relates the two images.

It allows optimizing the two losses simultaneously, given a pair of images.

4.7.2 Synthetic Pretraining and MagicPoint

Because of the lack of an extensive database of interest point labeled images. For training, a large-scale synthetic dataset called Synthetic Shapes was created. It

[1]This decoder has no parameters, and is known as "sub-pixel convolution" or "depth to space" in TensorFlow or "pixel shuffle" in PyTorch.

consists of simplified 2D geometric data rendering of quadrilaterals, triangles, lines, and ellipses. Then homographic warps for image augmentation were applied.

The data is generated on the fly, and there are no duplicates. This training data is used to train an interest point detector, represented by the convolution network.

The network is trained with the detector pathway of the SuperPoint architecture (ignoring the descriptor “head”) and trained on Synthetic Shapes (see Figure 4.10). The resulting model is called MagicPoint and is used for creating pseudo-ground-truth labeled keypoints from real-world images.

4.7.3 Homographic Adaptation

Homographies give exact or almost exact image-to-image transformations for camera motion with only rotation around the camera center, scenes with large distances to objects, and planar scenes.

The Homographic Adaptation process can be described as follows

1. it takes an unlabeled image and base MagicPoint detector as the input
2. then, is randomly sampled 100 homographic matrices for transformation, and the input image is wrapped
3. next, MagicPoint keypoints’ responses (heatmaps) are calculated, and the image is unwrapped
4. finally, aggregated heatmap is obtained, and interest point superset can be used for the training purposes.

The Homographic Adaptation technique was used at training time to improve the base MagicPoint architecture’s generalization ability on real-world images. The process can be repeated to continually self-supervise and improve the interest point detector.

4.8 SuperGlue

SuperGlue [4] is a neural network that matches two sets of local features by jointly finding correspondences and rejecting non-matchable points. Assignments are estimated by solving a differentiable optimal transport problem, whose costs are predicted by a graph neural network.

It introduces a flexible context aggregation mechanism based on attention, enabling SuperGlue to reason about the underlying 3D scene and feature assignments jointly. SuperGlue performs matching in real-time on a modern GPU and can be integrated into modern SfM or SLAM systems.

SuperGlue presents a new way of thinking about the feature matching problem. Instead of trying to learn better features followed by simple heuristic matching (e.g., brute force matching), it learns matching from existing local features using a neural network architecture.

In the context of SLAM, which typically [80] decomposes the problem into the visual feature extraction front-end and the bundle adjustment or pose estimation back-end, the SuperGlue network lies in the middle (see Figure 4.11).

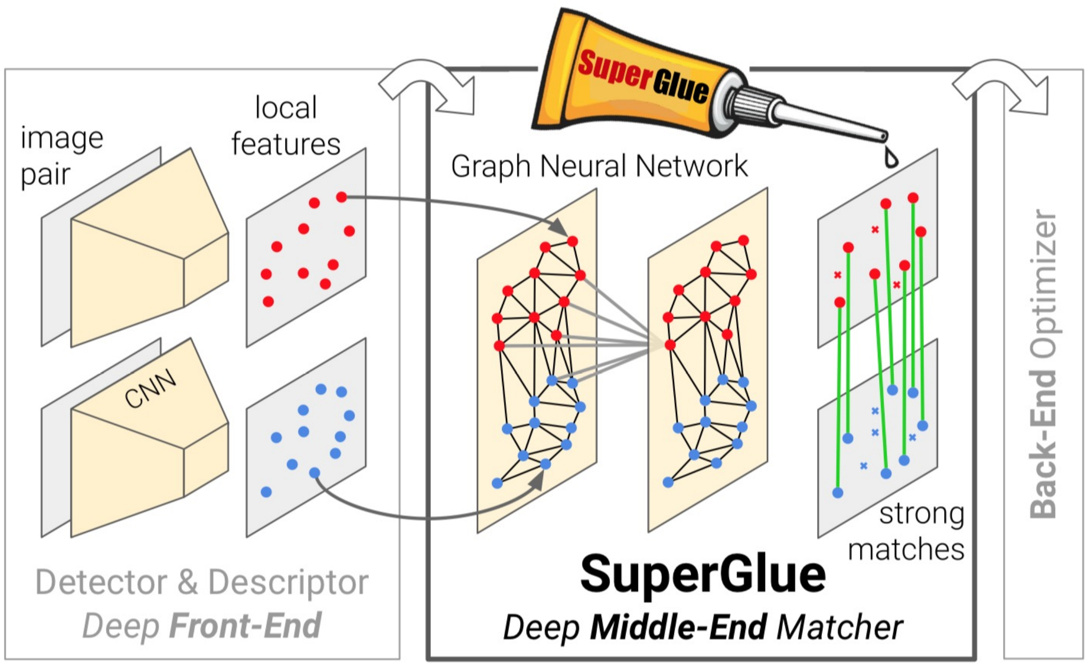

Figure 4.11: Feature matching with SuperGlue establishes pointwise correspondences from off-the-shelf local features. It acts as a middle-end between hand-crafted or learned frontend and backend [4].

4.8.1 Formulation

Consider two images A and B, each with a set of keypoint positions $\boldsymbol{p}$ and associated visual descriptors $\boldsymbol{d}$ – jointly $(\boldsymbol{p}, \boldsymbol{d})$ as the local features. Positions consist of x and y image coordinates and a detection confidence $c, \boldsymbol{p}_i \coloneqq (x, y, c)_i$. Visual descriptors $\boldsymbol{d}_i \in \mathbb{R}^D$ are descriptors like SIFT or SuperPoint (see Section 4.7). Images A and B have M and N local features, indexed by $\mathcal{A} \coloneqq \{1, \cdots, M\}$ and $\mathcal{B} \coloneqq \{1, \cdots, N\}$, respectively.

Partial Assignment

Correspondences across images must adhere to certain physical constraints

1. a keypoint can have at most a single correspondence in the other image and

2. some keypoints will be unmatched due to occlusion and failure of the detector.

Based on these two constraints: correspondences derive from a partial assignment between the two sets of keypoints. Each possible correspondence need a confidence value. The partial soft assignment matrix $P \in [0,1]^{M \times N}$ is defined as follows

$$P\mathbf{1}_N \leq \mathbf{1}_M \text{ and } P^\top \mathbf{1}_M \leq \mathbf{1}_N. \tag{4.28}$$

4.8.2 Attentional Graph Neural Network

Humans look back-and-forth when matching a given ambiguous keypoint in two images. They sift through tentative matching keypoints, examine each, and look for contextual cues that help disambiguate the true match from other self-similarities [81]. It is an iterative process that can focus its attention on specific locations.

The first block of the SuperGlue is an Attentional Graph Neural Network (see Figure 4.12). Given initial local features, it computes matching descriptors $\boldsymbol{f}_i \in \mathbb{R}^D$ by letting the features communicate with each other.

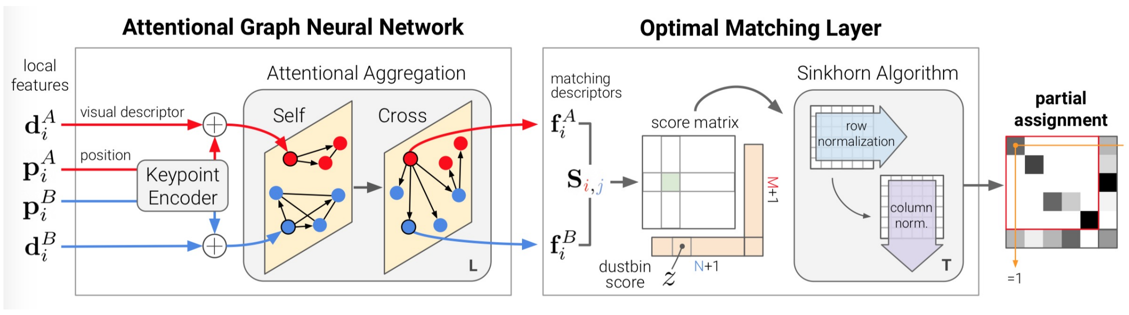

Figure 4.12: SuperGlue has two major components: the attentional graph neural network and the optimal matching layer. The attentional graph neural network uses a keypoint encoder to map keypoint positions $\boldsymbol{p}$ and their visual descriptors $\boldsymbol{d}$ into a single vector, and then uses alternating self- and cross-attention layers (repeated L times) to create more powerful representations f. The optimal matching layer creates an M by N score matrix, augments it with dustbins, then finds the optimal partial assignment using the Sinkhorn algorithm (for T iterations) [4].

Keypoint Encoder

The representation ${}^{(0)}\boldsymbol{x}_i$ for each keypoint i combines its visual appearance and location. The keypoint position is embedded into a high-dimensional vector with a Multilayer Perceptron (MLP) as

$${}^{(0)}\boldsymbol{x}_i = \boldsymbol{d}_i + \mathrm{MLP}_{\mathrm{enc}}\left(\boldsymbol{p}_i\right). \tag{4.29}$$

The encoder enables the graph network to reason about appearance and position jointly.

Multiplex Graph Neural Network

SuperGlue includes a single graph with keypoints of both images (nodes). The graph has two types of undirected edges. It is called a multiplex graph – see [82], and [83] for more details.

1. Intra-image edges, or self edges, $\mathcal{E}_{\text{self}}$, connect keypoints i to all other keypoints within the same image.

2. Inter-image edges, or cross edges, $\mathcal{E}_{\text{cross}}$, connect keypoints i to all keypoints in the other image.

The graph uses message passing formulation [84, 85] to propagate information to both types of edges. The resulting multiplex graph neural network starts with a high-dimensional state for each node and computes at each layer an updated representation by simultaneously aggregating messages across all given edges for all nodes.

Attentional Aggregation

An attention mechanism performs the aggregation and computes the message $\boldsymbol{m}_{\mathcal{E}\to i}$. Self edges are based on self-attention [86] and cross edges are based on cross-attention. A representation of i for the query q_i, retrieves the values v_j of some elements based on their attributes, the keys k_j. The message is computed as a weighted average of the values

$$\boldsymbol{m}_{\mathcal{E}\to i} = \sum_{j:(i,j)\in\mathcal{E}} \alpha_{ij}\boldsymbol{v}_j, \tag{4.30}$$

where the attention weight α_{ij} is the softmax over the key-query similarities $\alpha_{ij} = \text{softmax}_j\left(\boldsymbol{q}_i^\top \boldsymbol{k}_j\right)$.

The key, query, and value are computed as linear projections of deep features of the graph neural network. A query keypoint i is in the image Q and all source keypoints are in image S, $(Q, S) \in \{A, B\}^2$, then

$$\boldsymbol{q}_i = W_1^{(l)}\boldsymbol{x}_i^Q + \boldsymbol{b}_1 \tag{4.31}$$

and

$$\begin{bmatrix}\boldsymbol{k}_j\\ \boldsymbol{v}_j\end{bmatrix} = \begin{bmatrix}W_2\\ W_3\end{bmatrix}{}^{(l)}\boldsymbol{x}_j^S + \begin{bmatrix}\boldsymbol{b}_2\\ \boldsymbol{b}_3\end{bmatrix}. \tag{4.32}$$

Each layer l has its projection parameters, and it is learned and shared for all keypoints of both images.

4.8.3 Optimal Matching Layer

The second block (see Figure 4.12) is the optimal matching layer, which produces a partial assignment matrix. The assignment P can be obtained by computing a score matrix $S \in \mathbb{R}^{M \times N}$ for all possible matches and maximizing the total score $\sum_{i,j} S_{i,j} P_{i,j}$ under the constraints in Equation 4.28. This is equivalent to solving a linear assignment problem.

Score Prediction

The pairwise score is defined as the similarity of matching descriptors:

$$S_{i,j} = \left\langle \boldsymbol{f}_i^A, \boldsymbol{f}_j^B \right\rangle, \forall (i,j) \in \mathcal{A} \times \mathcal{B}, \tag{4.33}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. The matching descriptors are not normalized, and their magnitude can change per feature and during training to reflect the prediction confidence.

Occlusion and Visibility

To suppress some keypoints, each set is augmented with a dustbin, and unmatched keypoints are explicitly assigned to it. The SuperPoint has also used dustbins (see Section 4.7) to account for image cells that might not have a detection.

The score S is augmented to the score $\bar{S}$ by appending a new row and column, the point-to-bin and bin-to-bin scores, filled with a single learnable parameter

$$\bar{S}_{i,N+1} = \bar{S}_{M+1,j} = \bar{S}_{M+1,N+1} = z \in \mathbb{R}. \tag{4.34}$$

The keypoints from A will be assigned to a single keypoint in B or the dustbin. Each dustbin has as many matches as keypoints in the other set: N, M for dustbins in A, B respectively.

Sinkhorn Algorithm

The solution optimization problem corresponds with optimal transport [87]. Its entropy-regularized formulation naturally results in the desired soft assignment and can be efficiently solved on GPU with the Sinkhorn algorithm that is differentiable (for more information read [88, 89]).

4.8.4 Loss

SuperGlue is trained in a supervised manner from ground truth matches $\mathcal{M} = \{(i,j)\} \subset \mathcal{A} \times \mathcal{B}$. Unmatched keypoints are labeled $\mathcal{I} \subseteq \mathcal{A}$ and $\mathcal{J} \subseteq \mathcal{B}$. Given these labels, the negative log-likelihood is minimized by the assignment $\bar{P}$

$$\text{Loss} = -\sum_{(i,j)\in\mathcal{M}} \log \bar{P}_{i,j} - \sum_{i\in\mathcal{I}} \log \bar{P}_{i,N+1} - \sum_{j\in\mathcal{J}} \log \bar{P}_{M+1,j}. \tag{4.35}$$

This loss simultaneously maximizes the precision and the recall of the matching.

4.9 Hierarchical Multi-scale Attention for Semantic Segmentation

The following method described here is from the source [6]. Whenever it is referred to the HRNet-OCR in this work, this modified method is intended, and not the original one from the source [90] if not stated explicitly otherwise.

The semantic segmentation task labels all pixels within an image as belonging to one of N classes. Specific predictions are best handled at a lower resolution (a large structure that requires more global contexts), and others are better handled at a higher resolution (the fine details, e.g., edges of objects or thin structures).

The multi-scale inference is a common practice to address this trade-off. The predictions (done at a range of scales) are combined with averaging or max pooling. Using averaging to combine multiple scales generally improves results, but it suffers from the problem of combining the best predictions with poorer ones.

For example, the best prediction for a given pixel comes from the $2\times$ scale, and a much worse prediction comes from the $0.5\times$ scale. Then, averaging will combine these predictions, resulting in sub-par output. On the other hand, Max-pooling selects only one of N scales to use for a given pixel, while the optimal answer may be a weighted combination across the different scales of predictions.

Multi-scale context methods: use a neural network with a low output stride for state-of-the-art semantic segmentation. It allows the networks to resolve precise detail better, but it also affects shrinking the receptive field. This reduction in the receptive field can cause networks to have difficulty predicting large objects in a scene. Pyramid pooling can counteract the shrunken receptive field by assembling multi-scale context. PSPNet [66] uses a spatial pyramid pooling module that assembles features at multiple scales using the features obtained from the final layer of the network using a sequence of pooling and convolution operations. DeepLab [68] uses Atrous Spatial Pyramid Pooling (ASSP), which employs atrous convolutions with different dilation levels, thus creating a denser feature compared to PSPNet. More recently, ZigZagNet [91], and ACNet [92] leverage intermediate features instead of just the features from the final layer of the network to create the multi-scale context.

Relational context methods: use pyramid pooling techniques attend to fixed, square context regions because pooling and dilation are typically employed symmetrically. Furthermore, such techniques tend to be static and not learned. However, relational context methods build context by attending to the relationship between pixels and are not bound to square regions. The learned nature of relational context methods allows context to be built based on image composition. OCRNet [90], DANET [93], CFNet [94], and OCNet [95] use such relationships to build better context.

Multi-scale inference: in both relation and multi-scale context methods use multi-scale evaluation to achieve the best results [96, 68, 97, 90]. Two common approaches combine network predictions at multiple scales: average and max pooling. Average pooling involves equally weighting output from different scales, which may be sub-optimal. The methods mentioned above share the trait that the network and attention heads are trained with a fixed set of scales. Only those scales may be used at run-time, else the network must be retrained.

Auto-labelling: the most recent semantic segmentation work for the Cityscapes dataset has utilized the 20000 coarsely labeled images as-is for the training of state-of-the-art models [95, 98]. However, a significant amount of each coarse image is unlabelled due to the coarseness of the labels. For achieving state-of-the-art results on the Cityscapes dataset, the method adopts an auto-labeling strategy (motivated by [99]). The dense labels are generated for the coarse images in Cityscapes. Most image classification auto-labeling work uses continuous or soft labels, but this method generates hard thresholded labels (for storage efficiency and training speed). With soft labels, a teacher network provides a continuous probability for each of N classes for each pixel of an image, whereas for hard labels, a threshold is used to pick a single top-class per pixel. Similar to [100, 101] it generates hard and dense labels for the coarse Cityscapes images. The method performs a single iteration of the teacher model's full training with the default provided coarse and fine labeled images. After this joint training, it auto-labels the coarse images, which are then substituted in teacher training recipe to obtain state-of-the-art test results. Using pseudo-generated hard labels with hierarchical attention helps obtain state-of-the-art results on the Cityscapes dataset.

4.9.1 Hierarchical Multi-Scale Attention

The attention mechanism is conceptually similar to [5]. Where a dense mask is learned for each scale separately. The final decision is made based on these multi-scale predictions. The predictions are combined by performing pixel-wise multiplication between masks with the predictions. Followed by pixel-wise summation among the different scales to obtain the final results. The schema can be seen in Figure 4.13.

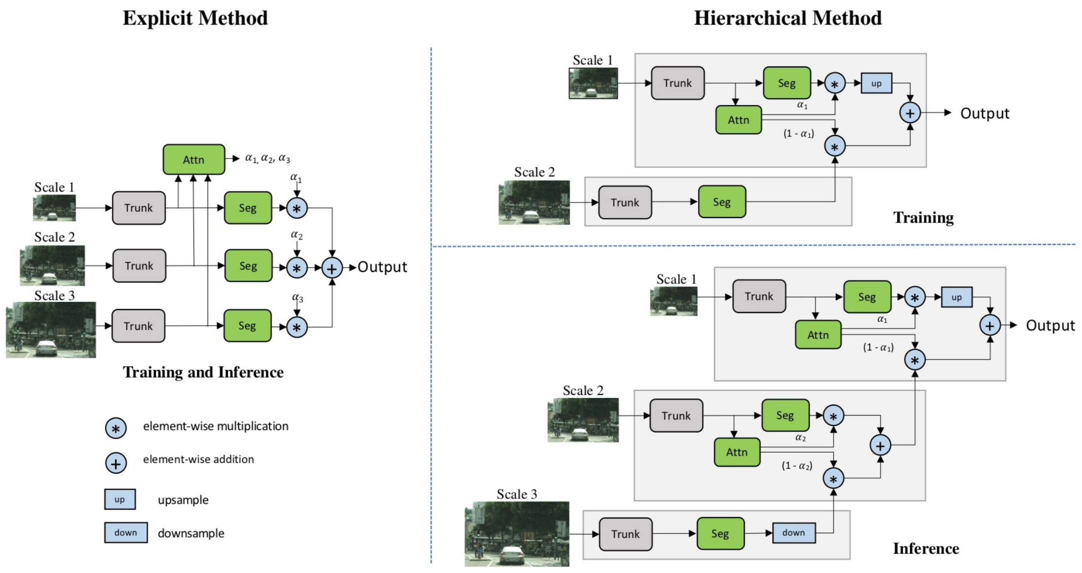

Figure 4.13: Left: shows the architecture from [5] where the attention for each scale is learned explicitly. Right: shows hierarchical attention architecture. Right top: training pipeline where the network learns to predict attention between adjacent scale pairs. Right bottom: displays hierarchical structure to combine multiple scales of predictions. Lower scale attention determines the contribution of the next higher scale. The image was taken from [6].

The method from [5] is referred to as explicit. The presented hierarchical method learns a relative attention mask between adjected scales. Instead of learning all attention masks for each of a fixed set of scales.

Only adjacent scale pairs are trained. A given set of image features from a single lower scale predicts a dense pixel-wise relative attention between the two image scales (see Figure 4.13). For obtaining the pair of scaled images, it takes a single input image and scales it down by a factor of 2 (original image with a $1\times$ and $0.5\times$ scaled input). Any scale-down ratio can be selected.

It is essential to note that the network is a rescaled version of the original training images. Because it uses image scale augmentation in the training process, this allows predicting relative attention for a range of image scales. It is possible to use the learned attention to hierarchically combine the N scales of predictions.

The authors of the method give precedence to lower scales. Furthermore, upsample images up to higher scales. The idea of higher scales has a more global context, and it can be chosen where predictions need to be refined by higher scale predictions.

This hierarchical method has two main advantages:

1. it can be flexibly selected the scales and add newly selected scales to previously trained model (it is not limited to a specific scales used during the model training) and
2. the hierarchical structure allows to improve the training efficiency (compared to explicit training method).

4.9.2 Auto Labelling

The method is inspired by recent work [102, 99] on auto-labeling for classification tasks. It uses auto-labeling for the Cityscape dataset to improve labeling quality. The Cityscape dataset contains 20000 coarsely labeled images and 3500 finely labeled images. The label quality of the coarse images is very modest and contains a large number of unlabelled pixels. The Auto-labeling approach improves label quality.

The soft auto-labeling technique is commonly used. It would need to store many labeling data (e.g., 3.2 TB for the Cityscape dataset), but it would slow the training process considerably.

The method uses the hard labeling strategy, where for a given pixel, the top class prediction of the teacher network is selected. The labels are based on teacher network output probability and are thresholded. Predictions that exceed the threshold are considered as correct labels. The threshold is set to 0.9.

4.10 Summary

Chapter 4 describes selected and relevant state-of-the-art methods that can be used in the task of long-term visual localization. It contains methods for image retrieval, ANNs based keypoint detectors and descriptors. It also mentions the current popular detect-and-describe approach of keypoints. The chapter explains methods like NetVLAD, SuperPoint, SuperGlue, and HRNet-OCR used in the following experiments.

Chapter 5

Datasets

This chapter is focused on the datasets that reflect year seasons, weather, natural environment, objects, and illumination changes. Mostly, these datasets were captured overall year seasons and over more than one year. They provide an outstanding collection of high-quality images and videos for research purposes. Most of the mentioned datasets are used for the state-of-the-art benchmarks in various Computer Vision tasks, e.g., long-term visual localization, keypoint matching, SLAM, semantic segmentation, and instance segmentation.

5.1 CMU Seasons Dataset

The CMU Seasons dataset is based on a subset of the CMU visual localization dataset, which consists of 100000 images. All of the images were taken in sequences throughout one year in Pittsburgh, PA, USA [103].

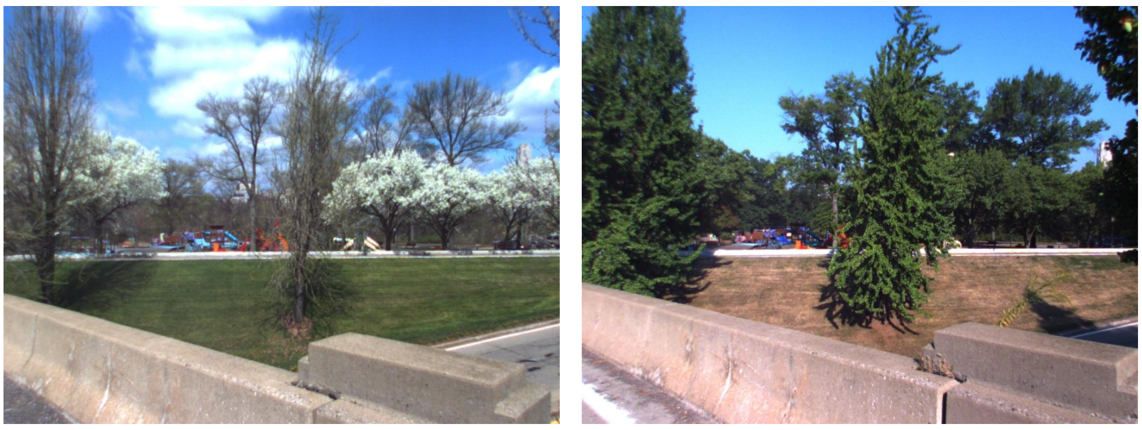
Figure 5.1: Example of CMU Seasons dataset. The figure shows two images representing the same scene from the slice number 24. Both images are captured under different seasonal conditions.

Two cameras were mounted on a rooftop of an autonomous Sport Utility Vehicle (SUV) car at 45 degrees forward/left and forward/right angles for creating the original dataset. Those cameras recorded 16 traversals on an 8.5 km long route in urban and suburban locations throughout different seasons, weather, and light conditions. One traversal is used as a reference scene representation, and the rest are used for the query. There are 7159 database images, 75335 query images, and 187 query sequences.

The CMU Seasons dataset operates with images taken under one reference condition: sunny weather and no tree foliage. Those images were captured at 17 specific locations referred to as slices. Slices 2-8 show urban scenes, slices 9-10 and 17 demonstrate suburban scenes, and slices 18-22 and 24-25 display park scenes. The example of two images showing the same scenes but taken under different seasons conditions from slice 24 is shown in Figure 5.1.

The dataset offers a reference 3D model reconstructed using Structure-from-Motion regarding the single reference condition. The mentioned 3D model delimits a set of six Degrees of Freedom (DoF) reference poses for the database images. The query images represent 17 specific locations and show them under different conditions. The reference six DoF poses have not been released yet.

The dataset was primarily used to measure the impact of changing conditions on camera pose estimation accuracy by using the state-of-the-art methods [104].

In the real world, the CMU dataset was used to solve a critical problem in precision agriculture, which was autonomous crop monitoring at high spatial and temporal resolution. The method was built upon the 4D reconstruction approach to crop monitoring that operates with a spatiotemporal model of dynamic scenes [105]. It is helpful for agriculture applications these days.

5.1.1 Extended CMU Seasons Dataset

The extended CMU Seasons dataset is a larger version of the CMU Seasons dataset, described in section 5.1, which is based on the CMU visual localization dataset. This dataset contains almost 40% more images.

10338 reference images were taken on the same day and in favorable conditions. For this sequence of images, the camera poses are well known. The test set with 56613 query images includes a wide range of all-weather and season conditions such as sunny, rainy, and snowy days involving spring, winter, and the season and weather conditions. Only half of all camera poses are known for this test sequence with all conditions.

The extended dataset covers different kinds of landscapes. It is possible to find urban, suburban, and park-like areas dominated by vegetation on both roadsides. The example can be seen in Figure 5.2.

The dataset has better quality than the deprecated CMU Seasons dataset, and

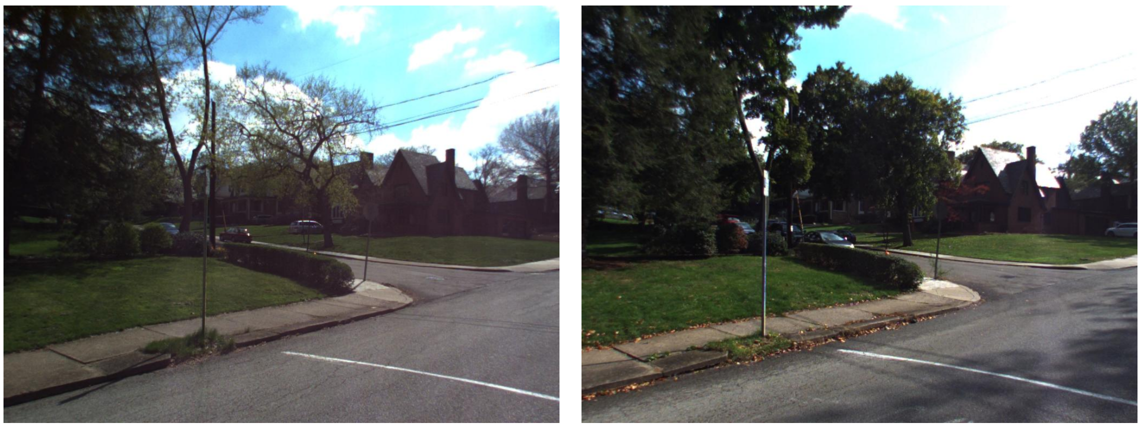

Figure 5.2: In this comparison, there is the same suburban scene as a part of the extended CMU Seasons dataset. The left image shows the location during spring. The right image reflects changes in the natural environment during the summer.

it is considered to be more practical and often used for new challenges in the future.

This dataset was tested in the approach of feature point sparse-to-dense hypercolumn matching, suitable for robust and accurate outdoor visual localization in long-term scenarios [106].

5.2 RobotCar Seasons Dataset

The RobotCar Seasons dataset is a subset of the RobotCar dataset consisting of more than 20 million images. These images were captured by 6 cameras mounted on an autonomous car and by using Light Detection And Ranging (LIDAR), Global Positioning System (GPS), and Inertial Navigation System (INS) ground truth. The data was precisely collected every two weeks for over a year throughout all year seasons, day and night, and different weather conditions including heavy rain, snow, dynamic objects, seasonal effects, and constructions.

Collecting of the data led to more than 100 traversals captured on the same 10km long route in central Oxford, UK [107]. This RobotCar Seasons dataset includes 26121 reference images and 11934 query images. The examples of images showing the same scene and captured in summer and winter are in Figure 5.3. All images are stored as lossless compressed PNG files in unrectified 8-bit raw Bayer format.

This dataset is significant for researching long-term localization and mapping for autonomous vehicles in the real-world in the fast dynamic urban environment.

There are many research areas where the dataset has been used, for example, experience-based classification for pedestrian detection or camera-only localization in challenging outdoor environments, where changes in lighting, weather, and season cause traditional localization systems to fail [108]. This dataset was employed for obtaining a dense Deja-Vu feature representation that can be used to perform

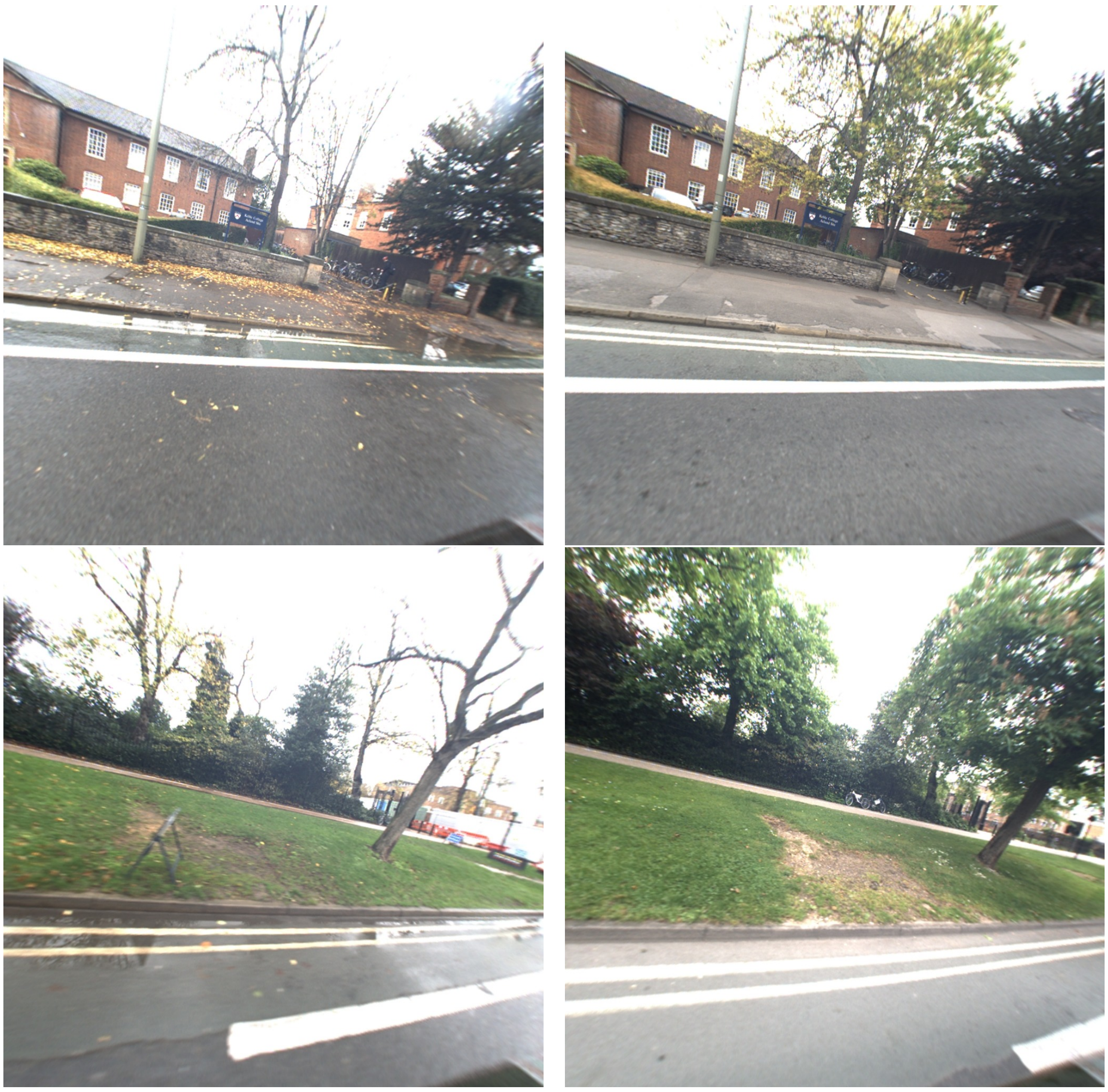
Figure 5.3: Example of RobotCar Seasons dataset. The figure demonstrates two pictures in one row showing the same scene. The left images show the scene in winter and the right images were captured in summer. All of the images were taken by using a camera mounted on the left side of the car.

localization, sparse matching, or image retrieval, regardless of the current seasonal or temporal appearance [109].

Another real example where the LIDAR part of the RobotCar Seasons dataset was applied is a generation of accurate static maps that are encumbered by the presence of ephemeral objects such as vehicles, pedestrians, or bicycles [110]. The LIDAR part of the dataset was also used for approaches to autonomous vehicle localization and navigation typically involve 3D LIDAR scanners and a static, curated 3D map, both of which are expensive to acquire and maintain [111].

The RobotCar-Seasons v2 dataset can be seen in literature or benchmarks. It consists of the same database and query images as the original RobotCar-Seasons dataset but uses a different test-train split.

5.3 Cross-Seasons Correspondence Dataset

The cross-seasons correspondence dataset consists of samples containing two nearby images and a set of 2D-2D point correspondences between those two images. These images were taken during the entire year in urban and suburban locations in the USA and UK [112]. Since the images reflect different year seasons and weather conditions, the correspondences are built using geometric 3D consistency between the two points. The reason is the geometry is used to being more stable and valuable compared to, e.g., photometric information across the different seasons and weather conditions.

The cross-seasons correspondence dataset is based on two existing datasets. The first one is the CMU visual localization dataset from Section 5.1 which includes 28766 image pairs. The second one is the RobotCar dataset from Section 5.2 with 6511 image pairs. An example of two images demonstrating the same scene over two year seasons is in Figure 5.4.

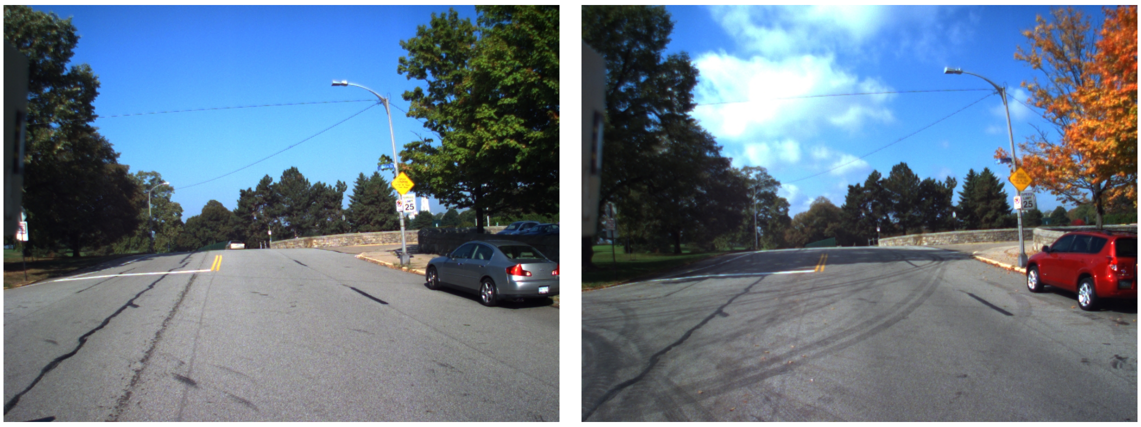
Figure 5.4: The cross-seasons correspondence dataset reflects weather and season changes during one year of images capturing. Both images show environment changes for the same scene in the urban location in summer and fall.

In order to create the final dataset, there are used four following steps. The first step includes calculating the camera position for all images in a standard coordinate system. The second step creates a robust 3D point cloud of each condition's surrounding geometry and traversal. The 3D point clouds are matched across all conditions in the next step. Thanks to the first step, all clouds have the same coordinate system. It is very convenient because the matching can be executed by using the position of the 3D points. The camera's position is known for every image, and therefore, in the last step, the pixel positions for the 2D-2D correspondences can be calculated.

5.4 Symphony Seasons Dataset

The symphony seasons dataset represents a dataset created on the 1.3 km shore of the Lake Symphony in Metz, France. All images were taken using a Pan-Tilt-Zoom (PTZ) camera mounted on a crewless surface vehicle designed in the style of a pontoon-boat. The boat kept a constant distance from the shore while taking the pictures on its way. The boat captured the shore every ten days for about three years.

The entire dataset has 5031232 images from 121 visual surveys. It is possible to observe many changes in weather conditions, year seasons, illumination, viewpoint, and water reflectivity in many comparisons survey-to-survey [113]. However, the boat tried to keep a constant distance from the shore. It has to deal with other issues, e.g., different water levels and spots caused by pollen and insects on the camera. The boat could not be sent on the way when the lake was completely frozen.

The 600 GB dataset is divided into two sets. The first one contains 4 GB of data with full surveys. The second set includes 200 MB of data with sub-sampled surveys. All of the surveys include GPS, Inertial Measurement Unit (IMU), and compass data that is synchronized to the 704 × 480 @ 10 fps color images.

This dataset consists of images from the natural landscape that provides a better 3D structure than the urban environment. The example of these images can be seen in Figure 5.5. Therefore, it is helpful for Simultaneous Localization and Mapping (SLAM) community and helps develop and improve precision agriculture, environment monitoring, and many others using existing localization methods.

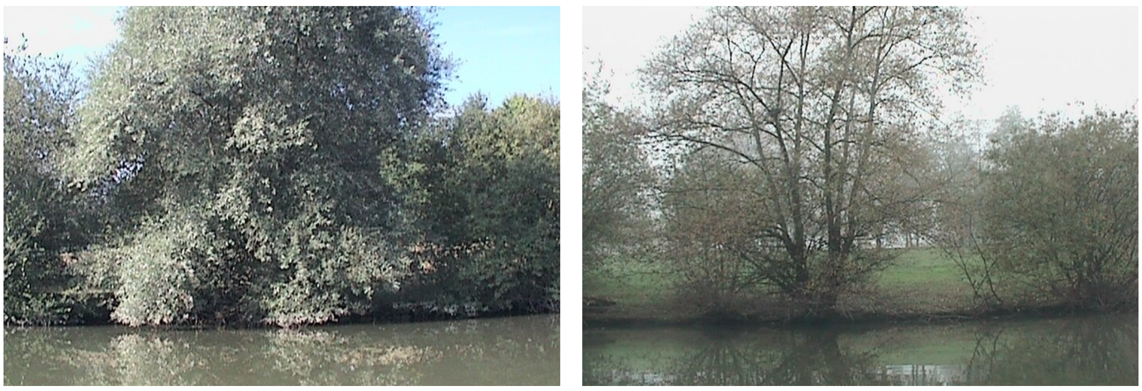
Figure 5.5: Two images of the same scene which were captured during different year seasons and showing features from an unstructured and natural environment.

The symphony seasons dataset and CMU Seasons dataset previously mentioned in Section 5.1 were used to reach the state-of-the-art image retrieval performance [33] for a global image description computed from its semantic and topological information. It is built from the wavelet transforms of the image semantic edges. Matching two images is then equivalent to matching their semantic edge transforms. This method is also applicable to urban scenes on which it is on the same level as the current baselines Network of Vector of Locally Aggregated Descriptors (NetVLAD) and DEep Local Feature (DELF) see Sections 4.1 and 4.5 respectively.

5.5 NRK Train Dataset

The Norwegian Broadcasting Corporation (NRK) (in original: Norsk Rikskringkasting) has made Norway's northernmost railway linking Trondheim and Bodø. It was recorded four times, once for every season. The length of this ride is around ten hours[1].

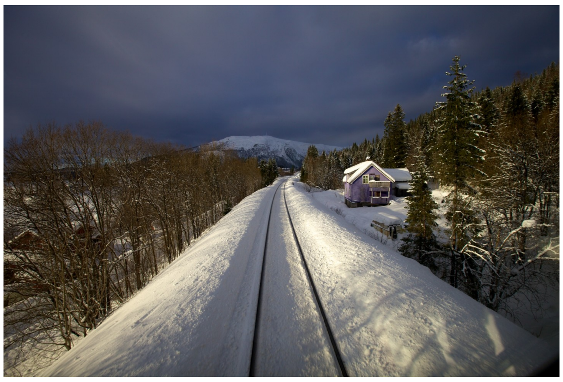

Figure 5.6: Left: the sky over the Nordlandsbanen. Right: Hellum T. and Carlsen J. from NRK mount the forward-facing camera before the winter recording along the Nordlandsbanen. Photos from NRK.

The Nordland Railway ("Nordlandsbanen" in Norwegian) celebrated its 50th anniversary in the year 2013 and has created this dataset. The railway is 728 kilometers long, and passes through spectacular scenery 5.6, varying from the fjord area around Trondheim in the south, through valleys, over mountains, and along fjords before crossing the Arctic Circle at Saltfjellet and descending to the coastal city of Bodø.

The journey was recorded once for every season (Spring, Summer, Fall, and Winter) to show the different weather conditions and natural seasonal changes over almost 3000 km.

Videos were used to develop robust place recognition algorithms in changing environments such as the SeqSLAM algorithm [114].

The recording was done using a SonyXDcam, placed in the front of the NSB Di 4 locomotives used on Nordlandsbanen. Since NSB Di 4 locomotive is an unsta-

[1]The original article can be found here: https://nrkbeta.no/2013/01/15/nordlandsbanen-minute-by-minute-season-by-season/

ble camera platform with many vibrations, the image stabilizing lens from Canon (HJ15ex8.5B KRSE-V) was used (see in Figure 5.6).

Even though the Norwegian State Railway managed to keep the schedule on each of the recorded trips, the four journeys differed in position at any given time. The train will, for example, not enter a station at the exact second in both June and March and cannot keep the same speed at all times. Video had to be synchronized to make the train appear to be at the same place at the same time in every video stream.

The GPS position of the train was logged continuously on each trip. The authors decided to use the Summer clip as the master, and the video set was synchronized due to the Summer clip with GPS coordination.

5.6 Aachen Day-Night Dataset

The Aachen Day-Night dataset [115] is a subset of the original Aachen dataset [116] which contains 1.5M 3D points and 369 query images. This Aachen Day-Night dataset consists of 4328 reference images and 922 query images. There are 824 images captured in the daytime and 98 ones taken in the nighttime.

The entire dataset was captured in the old inner city of Aachen, Germany, by hand-held cameras and mobile phones over about two years. All query images were captured during the nighttime and daytime and only using mobile phone cameras. Therefore, this dataset is suitable for solving localization problems using mobile devices, e.g., Augmented or Mixed Reality. The 98 nighttime query images were taken using software HDR to obtain high-quality and well-illuminated images.

The collection of images was publicly released, and it is not intended for commercial use and should be applied primarily for research purposes.

The Aachen Day-Night dataset (see example in Figure 5.7) was used to prove that the proposed Structure from motion using dense convolutional neural network features with the keypoint relocalization outperforms a state-of-the-art Structure from motion (COLMAP[2] [56], [117], and [118] using RootSIFT (RootSIFT)) by a large margin [119].

This dataset was also used to learn keypoint detection and description together with a predictor of the local descriptor discriminativeness. The detection-and-description approach simultaneously outputs sparse, repeatable, and reliable keypoints that outperform state-of-the-art detectors and descriptors on the HPatches dataset and the Aachen Day-Night localization benchmark [34].

[2] https://colmap.github.io/

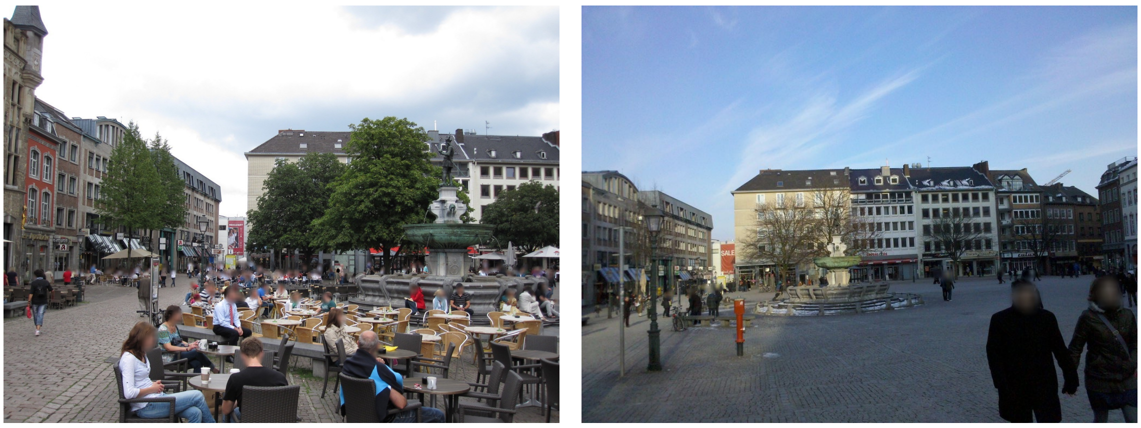

Figure 5.7: The Aachen Day-Night dataset contains images taken by hand-held cameras in the old city of Aachen, Germany. The left images demonstrate the same scene in the summer, and the right one represents the same square during the winter season. Both images were taken during the daytime.

5.6.1 Aachen v1.1

In the year 2020, the Aachen Day-Night v1.1 has been released [120]. It is an extension of the Aachen Day-Night dataset that contains additional nighttime query images. There are 2369 new reference images (6697 in total), and the query images were extended to a total of 1015 images (93 more images were added in the nighttime image set, the dataset contains 191 nighttime images in total, none query images in the daytime have been added).

The reference poses for the nighttime query images in the original Aachen Day-Night dataset were updated to more accurate poses, and the error thresholds were updated to the same as the ones for daytime.

5.7 Google Landmarks Dataset v2

Google Landmarks Dataset v2 (GLDv2) represents a new benchmark for large-scale, fine-grained instance recognition and image retrieval in the domain of human-made and natural landmarks [121], see the example in Figure 5.8.

This dataset contains over 5M images and 200k distinct instance labels. The dataset includes 4M labeled training images for the instance recognition task and 762k index images for the image retrieval task. The test set includes over 118k query images with ground truth annotations for the retrieval and recognition tasks. The dataset is sourced from Wikimedia Commons, the most extensive crowdsourced collection of landmark photos globally. One of this dataset's most significant benefits is its highly long-tailed class distribution, a large fraction of out-of-domain test photos, and sizeable intra-class variability.

The GLDv2 focuses on the task of recognizing landmarks. A benefit seems to be the recognition labels used for training image descriptors or pretraining approaches for related domains where less data is available. Also, this dataset is suitable for transfer learning by applying learned descriptors on independent datasets.

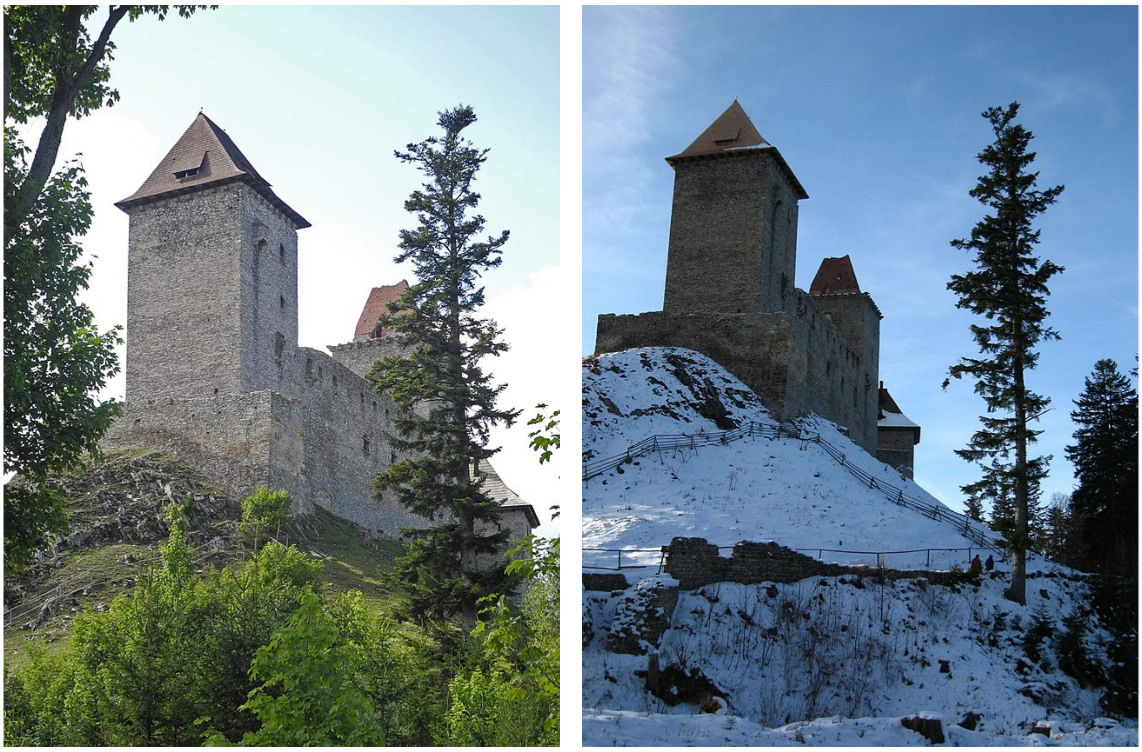

Figure 5.8: GLDv2 has been completely created by the worldwide Wikimedia Commons community. All images are freely licensed and can be freely reproduced in publications. These two images represent the same scene of the castle Kašperk, Czech Republic. The left image was taken in the summer season, and the right one was captured during winter by community users.

The original Google Landmarks Dataset v1 contains about 2.3M images from 30k landmarks that are unstable because the users who uploaded the images can delete them. On the other hand, GLDv2 stores only images with licenses allowing free reproduction and indefinite detention.

5.8 4Seasons

The 4Seasons dataset [122] covers seasonal and challenging perceptual conditions. The data was collected in different scenarios and under a wide variety of weather conditions and illuminations. The dataset contains more than 350 km of recordings in nine different environments (e.g., multilevel parking garage, urban environment, countryside, and highway). It provides globally consistent reference poses with up-to centimeter accuracy obtained from the fusion of direct stereo visual-inertial odometry with Real-Time Kinematic positioning - Global Navigation Satellite System (RTK-GNSS).

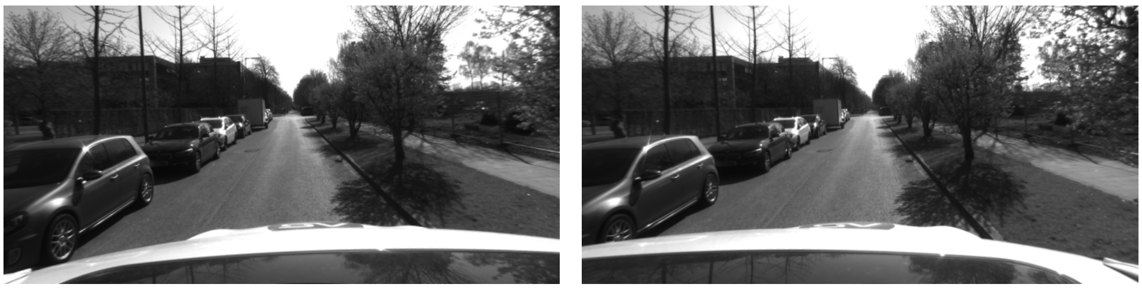

Figure 5.9: An example of 4Seasons dataset images (cam0 left, cam1 right).

The 4Seasons dataset contains recordings from a stereo-inertial camera system coupled with a high-end RTK-GNSS receiver for global positioning. The dataset consists of 30 individual sequences. Five sequences are used for visual localization (reference, training, validation, test0, and test1).

5.9 HPatches Dataset

The HPatches dataset represents a public benchmark for local descriptors [53]. This dataset is based on a patch and removes ambiguities plaguing existing image-based benchmarks. Since it considers many different scenes and visual types, the HPatches dataset can improve limited data and task diversity present in other datasets.

The entire dataset contains 116 image sequences while 57 of them contain illumination / photometric viewport transformations and 59 viewpoint/geometric illumination transformations. Each sequence comprises one reference image and five target images under different illumination and/or a different viewpoint (see Figure 5.10). The reference image is related to each target image by a ground truth homography.

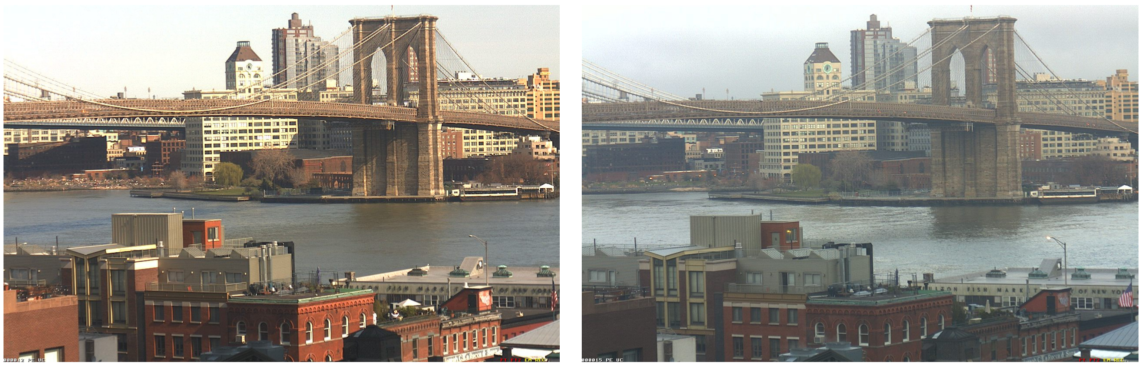

Figure 5.10: The HPatches dataset operates with images with extreme illumination changes. Both images show the same scene of Brooklyn city, NY, the USA, taken from one of the 116 image sequences where illumination change is evident.

5.10 Summary

In this section, there are presented the most well-known datasets used by scientists nowadays. Datasets can be used for tasks like SLAM, image retrieval, visual localization, long-term visual localization, 3D camera pose estimation, keypoints detection, keypoints description, and keypoints matching. The described datasets are used for benchmarking purposes of state-of-the-art methods as well.

Part III

Contribution to the State-of-the-art

Introduction

This part of the thesis and the research are carried out solely by the author of this dissertation thesis. Unless otherwise specified, for example, by citing the source or referencing the source code. All used source code for the carried experiment is publicly available on the author's GitHub page[1]. This Part III introduces the contribution to the field of long-term visual localization. The contribution follows the goals from Chapter 3.

The first Chapter 6 describes the long-term visual localization problem and discusses the used dataset to validate results on the specific benchmark. Afterward, the evaluation method and ranking method are explained in detail. Finally, the development environment is briefly mentioned.

Chapter 7 consists of experiments focused on enhancing the performance of the state-of-the-art methods used in the problem of long-term visual localization on a selected benchmarking dataset.

Next, Chapter 8 focuses on image preprocessing and using semantic segmentation. A class grouping based on semantic segmentation is introduced and used in long-term visual localization. It also shows an image's area that does not bring any valuable information and can be removed without worsening localization results.

Chapter 9 presents two possible approaches for training SuperPoint keypoint detector and descriptor with a priority on embedding semantic information into SuperPoint's model.

Last Chapter 10 introduces the author's new state-of-the-art results on selected datasets and their standard benchmarks. Further, it compares the author's obtained results with other methods.

[1]https://github.com/LukasBures/dissertation

Chapter 6

Long-Term Visual Localization Analysis

6.1 The Problem

The problem of long-term visual localization can be described as follows: estimating the six DoF camera pose from which a given image was taken relative to a reference scene representation. Visual localization is crucial for augmented, mixed, virtual reality, robotics, and self-driving cars.

To evaluate visual localization over time, a benchmark dataset is needed. It aims to evaluate six DoF pose estimation accuracy over significant appearance variations caused by changes in seasonal (spring, summer, fall, and winter) and during different illumination (day/night, dawn/sunset) conditions.

Long-term visual localization has multiple subtasks in the pipeline where the improvements to state-of-the-art can be made. The following open problems and areas can be improved and are mentioned below:

- Detecting keypoints is usually the first step in the pipeline. End-to-end trained neural networks replaced the traditional approaches in which corners and similar points were considered keypoints. During the training process, there are multiple ways where the neural network can be improved, e.g., training data (grayscale or color, preprocessing and postprocessing), the structure of the neural network (deep or shallow), training procedure itself (training from scratch or adapting weights), optimizing criterium (function).

- The next part is a keypoint descriptor, which is used later for matching purposes. The keypoint descriptor was, in traditional methods, represented by a feature vector describing a local window around the detected keypoint. The current describing approaches can be replaced by a neural network trained end-to-end. However, the ultimate goal is to get the most general descriptor unbiased to the given image.

- The third area of improvement can be made in keypoint matcher. Originally, keypoint matchers like brute force or nearest neighbor (FLANN) were used. In the current state-of-the-art, matchers are used in the form of neural networks that outperforms traditional matchers with a significant margin. The matcher has to be robust and match the right keypoints even if there are multiple instances of the same objects, e.g., tiles, windows, and fence.

- Before the matching phase starts, coarse matching is mainly used to speed the process. The coarse matching helps find the n most visually similar images and order them by a similarity value. It helps tremendously because the matching stage does not need to exhaustively match every possible query-dataset image combination. Improving the global descriptor to find the most similar image to a given query image can help enhance long-term visual localization overall.

- The SfM is also part of the pipeline. It can be improved as follow: the number of keypoint co-occurrences (by improving coarse matching), the robustness of co-occurrences (with detecting the exactly same keypoints in different views), use a different type of data (LIDAR) if the dataset or particular problem allows it.

- Carefully fine-tuning parameters of a method can improve the results of a given method. A selected method can be unoptimized for the specific task or dataset. Specific parameter tuning can enhance its performance even beyond state-of-the-art results.

- Before the pipeline even starts detecting keypoints, a preprocessing (masking, segmenting, thresholding, resolution, or illumination transformations) can be applied to enhance input images. A similar approach can be applied to post-processing images or keypoint descriptors (dimension reduction, transformations, concatenating with other features) before the data goes to the next stage of the long-term visual localization pipeline.

This dissertation thesis focuses on **fine-tuning parameters** of a method, **preprocessing input data**, and **using semantic segmentation for removing keypoints on dynamic objects**.

The mentioned open problems and areas prove the problem itself contains many possibilities and is still relevant to the research and industry community. With the current state of handheld and wearable devices, the problems are going to be even more relevant in applications of Augmented Reality and Mixed Reality.

6.2 Benchmark

For evaluation results, a benchmark is needed. A benchmark can play a role in the validity and honesty of results and prevent not scientifically correct results from being considered a new state-of-the-art.

A standardized dataset can be utilized as a benchmark for a specific task. See Section 5 for more information about various benchmarking datasets. A dataset can evaluate multiple tasks, e.g., semantic segmentation, visual localization, and instance segmentation. A standardized dataset can be published in a scientific paper where objectivity and correctness are sufficient. The open-source or science access/license for working with data is needed for validating mentioned.

Since European Conference on Computer Vision (ECCV) conference is one of the significant conferences in the computer vision community, it is possible to assume the benchmark used during ECCV 2020 Workshop[1] with focus on the Long-Term Visual Localization under Changing Conditions Workshop fulfill the needs for benchmark objectivity and validity.

The submission page[2] for the benchmark allows to submit results to various benchmarking datasets that are commonly used in top-tier computer vision conferences for the problem of long-term visual localization. Benchmarking datasets which can be evaluated are: Aachen Day-Night and Aachen Day-Night v1.1 datasets (see Section 5.6), CMU Seasons dataset Localization and Extended CMU Seasons datasets (see Section 5.1), RobotCar Seasons and RobotCar Seasons v2 datasets (see Section 5.2), Symphony Seasons dataset (see Section 5.4), InLoc dataset [123, 48], and SILDa Weather and Time of Day dataset [124].

Among many rules[3] and recommendations for benchmark and challenge submission, there are a few which is worth mentioning here:

- Combinations of existing methods, e.g., using SuperPoint features in a localization approach implemented in COLMAP[4], a state-of-the-art feature matching algorithm in combination with local features such as SuperPoints or D2-Net features, or exchanging the components of existing algorithms to boost performance.

- Submissions show that existing methods can outperform methods with results published on the benchmarks, e.g., **carefully tuning parameters** or using a different training dataset.

[1] https://sites.google.com/view/ltvl2020/challenges
[2] https://www.visuallocalization.net
[3] https://sites.google.com/view/ltvl2020/challenges
[4] https://colmap.github.io

- **Combining existing methods with preprocessing or postprocessing approaches**, e.g., using histogram equalization on the input images, building better 3D models (for example, through model compression or the use of dense 3D models), or integrating an existing localization algorithm into a (visual) Odometry / SLAM system.
- Using matches obtained by an existing method for multi-image localization.
- Showing that existing methods work well on challenges, even though the community believes they do not work.

The recommendations of **carefully tuning parameters** and **preprocessing approaches** are further expanded in this thesis.

6.2.1 Used Dataset

In this thesis, the dataset chosen for further evaluation is the Aachen Day-Night dataset[5] (without its v1.1 extension, see Section 5.6). The choice was made based on current assumptions and facts:

- It is the standardized and well-known dataset in the computer vision community. It is known for long-term visual localization and other visual keypoint-related tasks.
- It has the most populated submission table[6] which allows the best and the most comprehensive comparison with other methods on the unified benchmark.
- The most populated submission table may show that the Aachen Day-Night dataset is: one of the most popular in the computer vision community, or it is the first listed dataset on that benchmark webpage.
- The original Aachen Day-Night dataset was released in 2012 [116] and the current benchmarking dataset was released in 2018 [115]. One of the most well-known and used datasets for the long-term visual localization and visual keypoint task.
- Due to its age, many researchers and research teams have experienced using this dataset, which is intuitive.

The Aachen Day-Night dataset consists of 4328 reference images. Together with their corresponding ground truth poses and 922 query images (824 daytime and 98 nighttime). A triangulated 3D model is provided and is used by structure-based

[5] https://data.ciirc.cvut.cz/public/projects/2020VisualLocalization/Aachen-Day-Night/
[6] https://www.visuallocalization.net/benchmark/

localization approaches. The reference poses for the query images are unknown to ensure fairness and comparability, and the benchmark provides an evaluation service to measure pose accuracy.

6.2.2 Evaluation

The pose accuracy is evaluated for each submitted method in each dataset and challenge. The evaluation service follows [115] evaluation approach. It defines a set of thresholds on the position and orientation errors of the estimated pose. Each (X meters, Y degrees) threshold reports the percentage of query images localized within X meters and Y degrees of the ground truth location.

More specifically for the selected Aachen Day-Night dataset, the camera pose estimations thresholds are defined for all conditions (day/night) as follows:

1. the most precise threshold is set to 0.25m and 2°,
2. the follows threshold is set to 0.50m and 5°,
3. and the least precise threshold is set to 5.00m and 10°.

These thresholds define how precise the camera pose estimation is. For example, if a pose estimation fits into the most precise threshold, it is also included in the other two. However, when a pose estimation fits only under the least precise threshold, it is not included in the most precise threshold nor middle one.

6.2.3 Data Submission Format

The evaluation service[7] allows automatic validation (after a standard registration) of submitted pose estimations and calculates percentages (see Section 6.2) and puts a submitted method in the submission table (see Section 6.2.2).

The results have to be submitted as a text file using the following file format:

- Each query image corresponds to one line.
- The line should store result as: $name.jpg\ q_w\ q_x\ q_y\ q_z\ t_x\ t_y\ t_z$.
- The *name* corresponds to the image's file name without directory names.
- $q_w\ q_x\ q_y\ q_z$ represent the rotation from world to camera coordinates as a unit quaternion (versor).
- $t_x\ t_y\ t_z$ are the camera translation (not the camera position).

[7] https://www.visuallocalization.net/

- The file naming convention is: *Aachen_eval_yourmethodname.txt* where *yourmethodname* is a unique method name.

An example of one line in submission file can look like: $IMG\_20140520\_182846.jpg$ 0.004300400000000 -0.009711270000000 0.087104600000000 0.996143000000000 724.380999999999972 23.511800000000008 180.934000000000026.

The evaluation service expect that the coordinate system in which the camera pose is expressed is the N-View Match (NVM) coordinate system.

6.2.4 Ranking Method

For ranking the methods it is used the same method as for the Robust Vision Challenge CVPR 2018[8], Robust Vision Challenge CVPR 2020[9], and in ECCV 2020 Workshop[10].

For the dataset, the submitted results are ranked based on the percentages. The rankings are computed by using the Schulze Proportional Ranking method [125].

The Schulze Proportional Ranking method is based on a pairwise comparison of results. If the results of a method are not available for a dataset, the comparison will assume that it performs worse than a method for which the results are available.

6.3 Long-Term Visual Localization Pipeline

In this section, it is described the long-term visual localization pipeline which allows creating results that can be submitted in the benchmark[11].

The code for long-term visual localization[12] and for benchmark[13] itself can be found on Github.

Specifically, hierarchical localization uses the advantage of coarse to fine localization. The most similar images to a query image are found using global descriptors, e.g., NetVLAD. This approach eliminates a huge computational demand which would be needed for calculating keypoints matches directly to all keypoints in the SfM map.

For further experiments the standardized pipeline (for Aachen dataset from Section 6.2.1) idea was taken from the paper [126] and the base code from Github[14] and can be described as follows:

[8]http://www.robustvision.net/rvc2018.php
[9]http://www.robustvision.net
[10]https://www.visuallocalization.net/workshop/eccv/2020/
[11]https://www.visuallocalization.net/
[12]https://github.com/cvg/Hierarchical-Localization
[13]https://github.com/tsattler/visuallocalizationbenchmark
[14]https://github.com/cvg/Hierarchical-Localization

1. **Local Feature Extracting:** at first, the local features for the database and query images are extracted and stored in a Hierarchical Data Format (HDF) file i.e. in H5 file. Many keypoints detectors and descriptors D2-Net (Section 4.2), R2D2 (Section 4.3), DELF (Section 4.5), and SuperPoint (Section 4.7) are nowadays popular and give state-of-the-art results in many applications.

2. **SfM Reconstruction:** next, the pairs for the SfM reconstruction are generated in the pipeline. Instead of matching all database images exhaustively, the existing SIFT model [115] is used for obtaining a priori information about which image pairs are the most co-visible. At first, SIFT model is converted from NVM to the COLMAP format. Then, a co-visibility search is performed by selecting the top 20 most co-visible neighbors for each image.

3. **Matching Database Images:** creates matches that are utilized for building the SfM model in the next step. For matching can be used, e.g., Nearest Neighbor or SuperGlue (see Section 4.8) matchers. All matches are then stored.

4. **Triangulate SfM Model:** a new SfM model calculated from the given poses is in this step. It triangulates the sparse 3D point cloud given by the matches, and the reference poses are stored in the SIFT COLMAP model.

5. **Match Query Images:** for precomputing coarse-to-fine approach, which allows decreasing computational costs dramatically. For every query image, a list of n the most similar images is retrieved from the database. This is done by a image retrieval method e.g. NetVLAD (see Section 4.1). These pairs are also used for localization in the next step. The matching itself is performed by a matcher, e.g., Nearest Neighbor or SuperGlue (see Section 4.8).

6. **Localization:** the last computational step performs hierarchical localization using the precomputed retrieval and matches. Output is stored in required format and contains the estimated camera query poses (see Section 6.2.3).

7. **Submission:** the final result file with camera pose estimations is in the defined format (see Section 6.2.3). The file has to be manually submitted (after standard registration or login process) into the submission service that evaluates (see Section 6.2.2) and ranks (see Section 6.2.4) the results.

The described pipeline can be improved in every mentioned step. The ideas for improvements are generally described and discussed in Section 6.1.

6.4 Development Environment

For all experiments in this thesis, Ubuntu 16.04.7 LTS with Python 3.6.12, GNU bash 4.3.48, and Machine Learning toolboxes Torch 1.7.0+cu101 plus Tensorflow 1.15.4 and 2.2.0 were used. CUDA® V10.1.243 was used for GPU acceleration. Many other Python packages and useful libraries for machine learning, computer vision, and data manipulation were used e.g. OpenCV, NumPy, h5py, argparse, etc.

For all calculations and experiments, author used a desktop PC with Intel® Core™ i7-5820K CPU @3.30 GHz, 96 GB RAM, and for GPU optimized code were used Nvidia GeForce® GTX 1080 Ti (11 GB VRAM) along with Nvidia GeForce® GTX 1080 (8 GB VRAM).

6.5 Discussion

This Chapter described the problem of long-term visual localization. Then selected benchmark, used dataset, and evaluation procedure were described as well. Finally, the development pipeline and the approach for the solution were discussed. The development environment was mentioned, too.

In the problem description in Section 6.1 and in the pipeline description in Section 6.3, there were mentioned methods, approaches, and ways where a contribution to state-of-the-art can be made. Specifically, two approaches were selected for further expansion, namely tuning method parameters and data preprocessing.

The following chapters introduce a unique contribution extending the current state-of-the-art (if not stated differently, e.g., by citation or link).

Chapter 7

Parameter Tuning

In this chapter, the selected methods are presented, and the selection is discussed. Then, previous results and unsuccessful replication from other scientists are shown, and the baseline is set. The next important sections describe the selected parameters of the methods and hypotheses. Finally, the new state-of-the-art results of the original experiments are presented, and the discussion is carried out.

7.1 Methods

Based on selected benchmark[1] in Section 6.2 and dataset (see Section 5.6 and 6.2.1) picking the top state-of-the-art performing approach as a baseline seems to be obvious. The top performing method on Aachen benchmarking dataset is Hierarchical Localization (HLoc) pipeline [126] with Github source code[2]. It uses NetVLAD (see Section 4.1), SuperPoint (see Section 4.7), SuperGlue (see Section 4.8), and COLMAP[3]. The specific parameters and achieved results are mentioned further in this section. The pipeline described in Section 6.3 was used.

7.1.1 Previous Results

The best results published by the HLoc method [126] that includes the NetVLAD retrieval for coarse matching (pretrained on Pittsburgh 30k dataset [127], retrieved the top 50 most similar images), SuperPoint features [7] (4096 keypoints, Non-Maximum Suppression (NMS) radius 3 pixels, and scaled images to 1024 pixels at longer side), and SuperGlue matcher [4] with pretrained outdoor model. Whenever it is referred to the Original HLoc it means these settings. The results can be seen in Table 7.1 where the numbers represent images percentages (see Section 6.2.2)

[1] https://www.visuallocalization.net/
[2] https://github.com/cvg/Hierarchical-Localization
[3] https://colmap.github.io/

within the limits for day/night conditions (0.25m, 2°) / (0.50m, 5°) / (5.00m, 10°). In this format, every other result is presented.

Method	**Day** [%]	**Night** [%]
Original HLoc [126]	89.6 / 95.4 / 98.8	86.7 / 93.9 / 100.0

Table 7.1: Published Original HLoc results in the long-term visual localization benchmark on Aachen dataset.

Other scientists and experts tried to reproduce the results of HLoc, and a few publicly released retests[4] were published, but all of them performed worse than the original published results, see Table 7.2. It may look like a problem with result replications or with the proper setting of the method.

Method	**Day** [%]	**Night** [%]
Original HLoc [126]	**89.6** / 95.4 / 98.8	**86.7** / **93.9** / **100.0**
Retest 1	89.3 / **96.0** / 98.8	84.7 / **93.9** / **100.0**
Retest 2	89.3 / 95.9 / 98.8	85.7 / 92.9 / **100.0**
Retest 3	88.8 / 95.4 / **99.0**	85.7 / **93.9** / **100.0**
Retest 4	89.3 / **96.0** / 98.5	84.7 / 91.8 / **100.0**

Table 7.2: The table shows the comparison of the Original HLoc method with a few publicly released retests. Results are ordered from the best (first row) to the least performing method.

Next, it is worth to mention other publicly published method competitors and their results, see Table 7.3.

Method	**Day** [%]	**Night** [%]
Original HLoc [126]	**89.6** / 95.4 / 98.8	**86.7** / **93.9** / **100.0**
KAPTURE-R2D2-FUSION[5] [128]	89.4 / **96.4** / **99.2**	84.7 / 92.9 / 98.0
NetVLAD+DISP[6]	88.8 / 95.1 / 98.2	82.7 / 90.8 / 96.9
ONavi [36]	85.7 / 93.7 / 98.9	81.6 / 91.8 / **100.0**
KAPTURE-R2D2-APGeM[7] [128]	88.7 / 95.8 / 98.8	81.6 / 88.8 / 96.9

Table 7.3: The comparison of the Original HLoc and its top 4 competitors. Results are ordered from the best (first row) to the least performing method.

The Original HLoc method outperforms all competitors and clearly wins in performance (see Section 6.2.4 for more information about ranking method).

[4]https://www.visuallocalization.net/benchmark/
[5]https://github.com/naver/kapture
[6]A paper is not available. Authors published results with note: "DISP was trained self-supervised on Oxford Bumblebee XB3 front camera image sequences from: 06 | 14 | 19 May, 11 | 14 | 21 Nov, 05 Dec 2014, 27 Feb, 29 May, 24 Jul, 1 Sep 2015"
[7]https://github.com/naver/kapture

7.2 Parameters Description

In this section, parameters of methods used in the Original HLoc pipeline are described. The input parameters can be fine-tuned for better results in the selected benchmark and the Aachen dataset.

The methods considered for input parameters fine-tuning from the Original HLoc pipeline are NetVLAD, SuperPoint, and SuperGlue. Their input parameters are described in the following text.

The first method is NetVLAD (see Section 4.1) utilized in coarse-to-fine approach for image retrieval n most similar images to a query image, where thus only one considered parameter is:

- **Top n most similar images:** the n parameter can dramatically improve or worsen matching time. With a low n, there are not many coarse matching pairs, and it improves processing speed and can decrease overall precision (with more possibilities for mismatching). The n is usually set to 20-50 most similar images.

Next SuperPoint (see Section 4.7) has parameters as follows:

- **Descriptor dimension:** the convolutional neural network of SuperPoint was trained to generate a keypoint descriptor of length 256.
- **NMS radius:** removes nearby points and by default is set to 4 pixels.
- **Keypoint threshold:** is by default set to 0.005. It represents the SuperPoint detector confidence threshold [7] that limits the scores of keypoints (the scores range between 0 and 1, higher is better).
- **Maximum number of keypoints:** is generally lower than all detected keypoints in an image (-1 returns all keypoints). If the maximum number of keypoints is lower than the number of keypoints detected in an image, only k top keypoints are returned (ordered by the keypoint score). The k parameter varies from experiment to experiment, but mostly between 1024-4096 detected keypoints per image.
- **Remove border:** is a parameter that defines the thickness of the border area in which detected keypoints are ignored and removed. It is for discarding keypoints that could not be detected precisely (due to missing information out of the image). By default, it is set to 4 pixels for every border.

Two more parameters are used for image preprocessing:

- **Maximum resize:** allows resizing of the longer side of the image to a specific value. Most of the Aachen dataset images are 1024 pixels wide (the longer side), but some of them are 1600 pixels wide. Images can be resized to arbitrary size but mentioned two sizes are most reasonable.
- **Transform to grayscale:** is a parameter that is mainly tied up to a used method that determines input data dimension (RGB or grayscale). In the case of SuperPoint, this is set to *True* to transform input images to grayscale images.

And for SuperGlue keypoint matcher (see Section 4.8), the following parameters can be modified:

- **Descriptor dimension:** is the same as the output dimension from a keypoint detector. For the SuperPoint, it is set to 256.
- **Weights:** are pre-trained neural network weights. Two sets of weights were released [4]. The first one was released for indoor and the second for outdoor environments.
- **Iterations of Sinkhorn algorithm:** T defines how many iterations are used for finding the optimal partial assignment (see Section 4.8.2) [89, 88, 87]. T is usually set to 20-200 iterations.
- **Match threshold:** returns the matches with the confidence score higher than a selected threshold. By default the threshold is set to 0.2 (see Section 4.8.1).

The described parameters can widely change the system's overall performance, and thus further fine-tuning can bring better results. The parameters are fine-tuned based on the hypotheses described in the following section.

7.3 Hypotheses

In this section, there are described hypotheses that should lead to better performance and results of the Original HLoc for Aachen Day-Night dataset in the selected benchmark[8].

If it is not stated otherwise in Sections 7.3 and 7.4 the parameters stayed identical to the Original HLoc. The unchanged parameters are NetVLAD (the 50 most similar images), NMS radius (3 pixels), border removing (4 pixels), transforming to grayscale (*True*), descriptor dimension (256), used pretrained weights of SuperGlue (outdoor), and match threshold (0.2).

[8]https://www.visuallocalization.net

Next, it is discussed how changing the rest of the parameters (number of detected keypoints, resize, keypoint threshold, and number of iterations of Sinkhorn algorithm) could lead to better performance for Aachen Day, Night, and Day-Night images.

7.3.1 Aachen Day Images

At first, a kind of resolution normalization could get better results because Aachen Day images have two lengths of longer side 1024 and 1600 pixels. Upsizing smaller images to 1600 pixels on the longer side could detect more robust keypoints.

Lowering the number of detected keypoints by SuperPoint (instead of the original 4096) to a lower number (e.g., 2048 keypoints) could remove less viable keypoints. It returns only the top-performing keypoints based on the score.

The keypoint threshold is set by default to 0.005, and it is used to filter out poorly performed keypoints (based on the scores). Increasing the keypoint threshold twice may filter out more keypoints and affect the overall results.

And finally, increasing the iterations of the Sinkhorn algorithm could find a better solution and help to match keypoints more precisely.

For mentioned hypotheses, experiments are performed, and relevant research carried out to achieve better results on the selected dataset (see Section 7.4).

7.3.2 Aachen Night Images

The previous hypotheses from Subsection 7.3.1 are valid for Aachen Night images and are taken into consideration in the experiments.

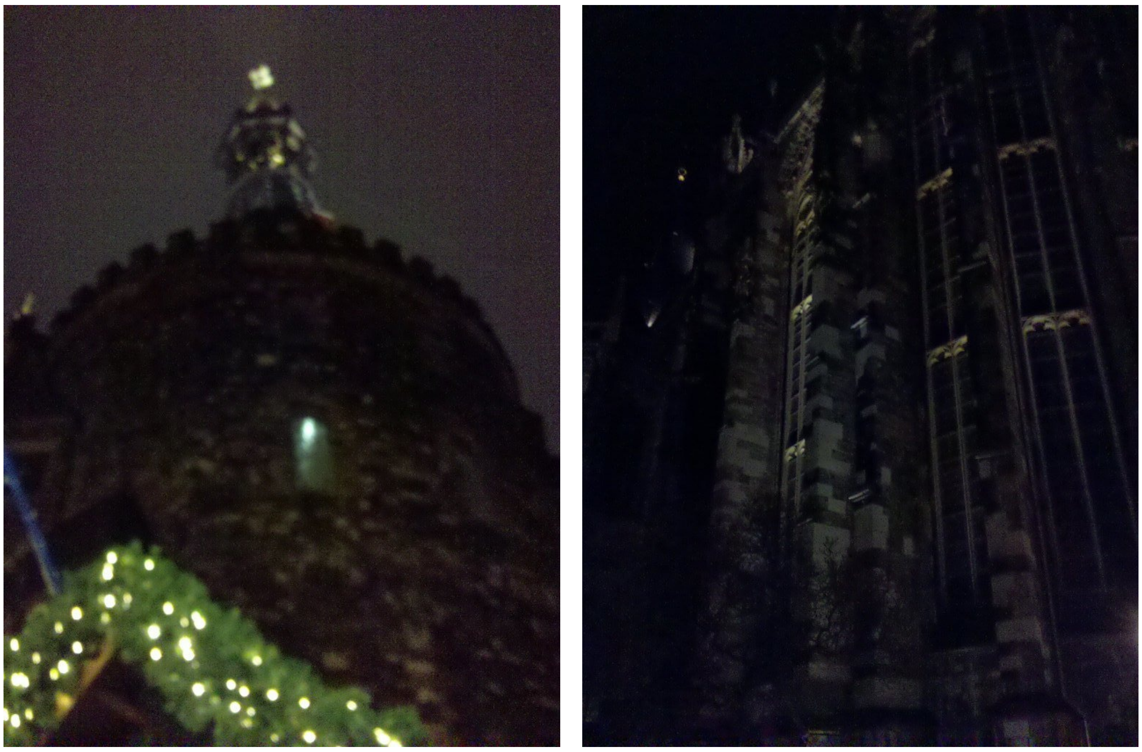
Figure 7.1: Two examples of Aachen Night images (1024 pixels longer side) taken under poor lighting conditions and with a lot of noise.

The only different thing for consideration is resizing original Aachen Night images. Downsizing the images (with a longer side of 1600 pixels) would decrease the resolution and remove beneficial visual information for keypoint detection. On the other hand, upsizing the images (longer side 1024 pixels) could decrease the localization performance. Those images were taken under poor light conditions and even with digital zoom and contain a lot of noise (see Image 7.1).

7.3.3 Aachen Day-Night Images

If hypotheses from Subsections 7.3.1 and 7.3.2 are confirmed then the combination of Day and Night results should outperform previous top state-of-the-art methods and more specifically the Original HLoc.

Next, using the parameters from the Original HLoc should lead to similar results presented in [126] for the Aachen Day-Night dataset. An increased number of iterations of the Sinkhorn algorithm should find a better optimum and help the overall system perform better in the long-term visual localization task. Even if it would take much longer to compute, it is worth experimenting with it because the benchmark itself is not used in real-time.

7.4 Experiments and Results

In this section, the selected results that performed well and outperformed state-of-the-art results are presented. However, many other tests and experiments were performed, they are not presented here because they performed poorly or similarly to the competitors. More than the 50 experiments were executed for every section (Day, Night, and Day-Night images). The calculation of one experiment took about 12-36 hours to complete based on selected parameters (for computation was used SW and HW from Section 6.4).

The results that outperform all current state-of-the-art approaches on the Aachen dataset for Day and Night are presented. For baseline, it was chosen the Original HLoc pipeline with NetVLAD, SuperPoint, and SuperGlue described in Section 6.3 and fine-tuned results are reported in the following subsections.

7.4.1 Aachen Day Images

Based on previous hypotheses from Section 7.3.1 the parameters were carefully set to: NetVLAD: uses the 50 most similar images. Image preprocessing: grayscale *True*, resize to 1600 pixels. SuperPoint: 2048 maximum keypoints, keypoint threshold 0.01, NMS radius 3 pixels. SuperGlue: outdoor weights, 100 Sinkhorn iterations. The results are presented in Table 7.4.

Method	**Day** [%]
Tuned HLoc	**89.7** / **95.8** / 98.7
Original HLoc [126]	89.6 / 95.4 / **98.8**

Table 7.4: Preciously selected parameters helped to outperform previous state-of-the-art results of the Original HLoc in two most precise localization thresholds for Aachen Day dataset. Results are ranked by Schulze method from Section 6.2.4 - the first row is the best.

Table 7.4 shows that the selected parameters helped to outperform previous state-of-the-art methods in the two most precise camera localization thresholds. It outperforms the most precise threshold by 0.1% and the next by 0,4%. For the last, the least precise localization threshold, the fine-tuned parameters perform 0.1% worse. It can be caused by estimated camera poses for the images that were refined and thus, meet stricter thresholds.

Based on the results it can be presumed that camera pose estimations for some images were refined (meets stricter thresholds). Since ground truth camera poses estimations are unknown, and the evaluation service provides no more information about camera localization. However, no more new images were localized successfully to pass the least precise localization threshold.

7.4.2 Aachen Night Images

Based on previous hypotheses from Section 7.3.2 the parameters were carefully set to: NetVLAD: uses the 50 most similar images. Image preprocessing: grayscale *True*, without any resize. SuperPoint: 2048 maximum keypoints, keypoint threshold 0.01, NMS radius 3 pixels. SuperGlue: outdoor weights, 100 Sinkhorn iterations. The main difference between Day and Night setting is that no resize was performed for Night images. The results are shown in Table 7.5.

Method	**Night** [%]
Tuned HLoc	**87.8 / 94.9 / 100.0**
Original HLoc [126]	86.7 / 93.9 / **100.0**

Table 7.5: Preciously selected parameters helped to outperform previous state-of-the-art results of the Original HLoc in two most precise localization thresholds for Aachen Night dataset. Results are ranked by Schulze method from Section 6.2.4 - the first row is the best.

Table 7.5 clearly shows that the new state-of-the-art results were achieved on Aachen Night dataset. The improvement was made in the most precise two thresholds (the last one was already at 100% and can not be improved). The previous state-of-the-art results were surpassed by 1.0% in the first and by 1.1% in the second threshold. Achieved results show that Aachen Night images without resizing can improve localization accuracy.

7.4.3 Aachen Day-Night Images

The results from Subsections 7.4.1 and 7.4.2 were combined and named the Combined tuned HLoc in Table 7.6.

One more experiment is worth mentioning here. This experiment has similar parameters setting as in the Original HLoc (NetVLAD with the 50 most similar images, 4096 keypoints, NMS radius 3 pixels, with grayscale preprocessing and image resizing to 1024 pixels on the longer side) but it has 100 iterations of Sinkhorn algorithm. This experiment shows that it is possible to replicate the results of the Original HLoc and slightly surpass it. Results are introduced in Table 7.6 labeled as the Original HLoc, 100 iterations.

The results from Table 7.6 show that the Combined tuned HLoc outperforms the Original HLoc for Day-Night Aachen dataset. Next, it shows the possibility to

Method	**Day** [%]	**Night** [%]
Combined tuned HLoc	**89.7** / **95.8** / 98.7	**87.8** / **94.9** / **100.0**
Original HLoc (100 iterations)	89.6 / 95.6 / **98.8**	86.7 / 93.9 / **100.0**
Original HLoc [126]	89.6 / 95.4 / **98.8**	86.7 / 93.9 / **100.0**

Table 7.6: Comparsion of the Original HLoc results in the long-term visual localization benchmark on Day-Night Aachen dataset compared with the Original HLoc (100 iterations) and the Combined tuned HLoc results.

enhance the Original HLoc by increasing iterations of the Sinkhorn algorithm (by 0.2% for Day images in the second-best threshold). Finally, it shows the possibility of reproducing the Original HLoc results.

The only one difference between the settings from experiments in Sections 7.4.1 and 7.4.2 is that Aachen Day images were resized to 1600 pixels width (upsampled) and Aachen Night images stayed unchanged.

7.5 Discussion

This chapter mentioned used methods and previous results for Aachen Day-Night dataset in Section 7.1, description of parameters in used methods in Section 7.2, hyphotheses for experiments with parameters tuning in Section 7.3, and new state-of-the-art results were presented in Section 7.4.

The Table 7.2 shows that other researches were not able to replicate the Original HLoc results published in [126] for Aachen Day-Night dataset. And Table 7.3 shows performance of the 4 top competitors published in the benchmark.

In this thesis, the problem with replicating the state-of-the-art results for long-term visual localization for the Aachen Day-Night dataset was successfully overcome, and results were replicated and improved in Section 7.4. The number of iterations of the Sinkhorn algorithm was increased to 100 iterations, and it enhanced results by 0.2% for Aachen Day images in the second-best threshold (see Table 7.6). The results for Aachen Night images stayed unchanged.

The results for Aachen Day (Table 7.4) and Night (Table 7.5) images were combined in the final Table 7.6. The Combined tuned HLoc outperforms the Original HLoc and other methods in Day and Night images separately and also in combined Day-Night images.

This chapter shows the possibility of improving state-of-the-art results on the specific dataset by carefully fine-tuning of the selected method's parameters. This approach is even encouraged and explicitly mentioned in the benchmark rules (see Section 6.2).

Chapter 8

Semantic Segmentation

This chapter describes hypotheses of semantic segmentation experiments and the used methods. Then, parameters of the selected semantic segmentation method are described, and groups of semantic segmentation classes are defined. This is followed by the presentation of the results of the performed experiments. Finally, the results are discussed.

8.1 Hypotheses

The main hypothesis that should be prove by experiments is that an image from the selected Aachen dataset contains an area of the image that does not help in the task of long-term visual localization. The area can have associated semantic information from semantic segmentation.

In the case that the area is removed (masked out and replaced by black color), the selected HLoc pipeline should perform at least as well as the Original HLoc or better. If this area is found, it can be stated that the area does not improve the overall results. Apriori, as the area, moving and dynamic objects in the image could be selected. The area of dynamic objects could be obtained by a semantic segmentation method. The dynamic object could have a high-level interpretation[1] from semantic segmentation classes. Then, groups of classes could be introduced to aggregate number of classes into a lower number. Furthermore, groups of classes should be used to prove the primary hypothesis.

8.2 Methods

For long-term visual localization HLoc pipeline (Section 6.3) with Aachen dataset

[1]For example, car, bus, and truck classes represent vehicles (the group of classes) as high-level interpretations.

(Section 5.6), evaluation (Section 6.2.2), and ranking method (Section 6.2.4) were used.

For semantic segmentation HRNet-OCR one of the current state-of-the-art methods was used. HRNet-OCR is described in Section 4.9. The performance of HRNet-OCR gives top results[2] on the Cityscape dataset[3] which is similar to the selected Aachen dataset (both are urban environments with similar objects classes like a human, vehicle, and vegetation). The semantic segmentation HRNet-OCR method was selected for further experiments based on this assumption. The implementation of HRNet-OCR was taken from the official Nvidia GitHub[4].

8.3 Parameters Description

The parameters of methods used in HLoc pipeline were described in Section 7.2.

In this section, only the parameters used for semantic segmentation HRNet-OCR method are described. The individual setting for the performed experiments is statted below:

- The Cityscape **dataset** was used. It means that the semantic segmentations are in the color palette of the Cityscape dataset.
- **cv** stands for cross-validation split. It is used in the Cityscape dataset. However, for the used Aachen dataset, this parameter was set to 0, which means no split nor validation was performed because the Aachen Day dataset does not have ground-truth semantic segmentations and labels.
- **syncbn** is synchronized batch normalization and was set to *true* value since the code was run on 2 GPUs (see Section 6.4).
- **apex**[5] was used with *true* value. Apex is the utility for streamline mixed precision and distributed training in Pytorch.
- **fp16** was set to *true* value. It enables Nvidia Apex Automatic Mixed Precision (AMP).
- **bs_val** is batch size for validation and was set to 1. The implementation needs this parameter, but it was never used because no validation is performed for the Aachen Day dataset.
- **eval** parameter allows to evaluate an folder. It was set to $folder$.

[2] https://paperswithcode.com/sota/semantic-segmentation-on-cityscapes
[3] https://www.cityscapes-dataset.com
[4] https://github.com/NVIDIA/semantic-segmentation
[5] https://github.com/NVIDIA/apex

- **eval_folder** contains path to the folder Aachen Day images.
- **dump_assets** *true* indicates to save all other assets.
- **dump_all_images** *true* indicates to save all images.
- **n_scales** were used default values 0.5, 1.0, and 2.0. It sets image scale levels on which multi-scale attention for semantic segmentation is performed.
- **snapshot** is the path to a pretrained weight snapshot. It was used from GitHub[6].
- **arch** defines used ocrnet.HRNet_Mscale architecture.
- **result_dir** is the folder path where the results are stored.

The rest parameters were set to default values.

8.4 Introduction of Semantic Segmentation Groups of Classes

Multiple different classes can be segmented in semantic segmentation, e.g., human, bicycle, or train. The Cityscape dataset uses the following classes that were merged into four groups. **Nature** n group consists of vegetation and terrain classes. Next, **sky** s group is represented only by sky class. **Human** h group is represented by person and rider classes. Furthermore, **vehicle** v group aggregates car, truck, bus, caravan, trailer, train, motorcycle, bicycle, and license plate classes.

The Cityscape dataset has a few more classes that were omitted because they are rigid or belong to void classes. These classes are: unlabeled, ego vehicle, rectification border, out of the rectangle of interest, static, dynamic, and ground class. Some flat and construction classes are present: road, sidewalk, parking, rail track, building, wall, fence, guard rail, bridge, and tunnel class. And some objects are: pole, pole group, traffic light, and traffic sign class.

8.5 Experiments and Results

The used parameters for HLoc (for parameters description see Section 7.2) were: the NetVLAD retrieval for coarse matching (pretrained on Pittsburgh 30k dataset [127], retrieved the top 50 most similar images), SuperPoint features [7] (4096 keypoints,

[6]Go to https://github.com/NVIDIA/semantic-segmentation and see Download Weights section.

keypoint threshold 0.02, NMS radius 3 pixels, and scaled images to 1600 pixels at longer side) and SuperGlue matcher [4] (outdoor weights, 100 Sinkhorn iterations).

For semantic segmentation experiments, HRNet-OCR method with parameters described in Section 8.3 was used.

8.5.1 Aachen Night Images

Several experiments were performed with the Aachen Night dataset, but the current state-of-the-art semantic segmentation methods can not handle the night scenery well. The example of poor semantic segmentation can be seen in Figure 8.1. Possible improvements and a general idea of handling Aachen Night images are described in Section 11.3. Due to semantic segmentation methods' bad performance at night, only Aachen Day images were considered in the next experiments.

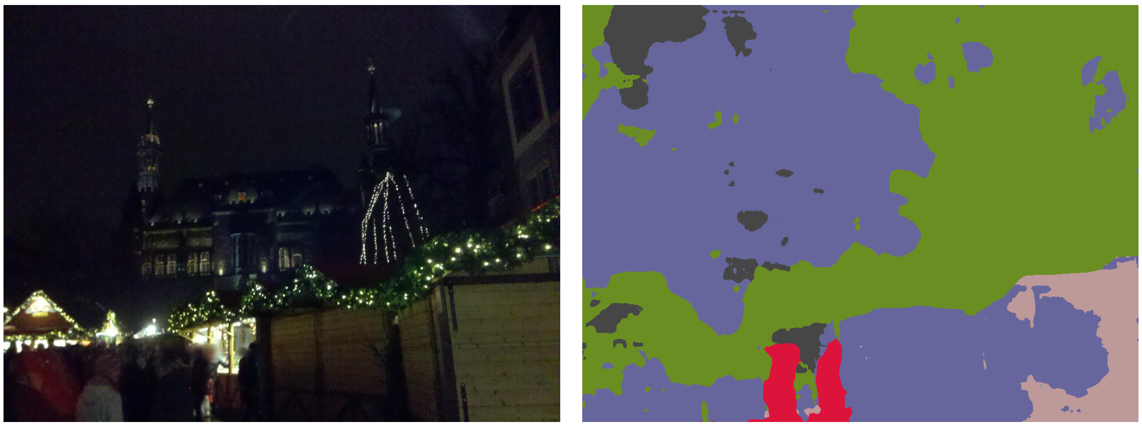

Figure 8.1: Left: original image from Aachen Night dataset. Right: failed semantic segmentation where the people (red) in the left bottom corner are missing, vegetation (green) is poorly segmented specifically in the top right corner, and a wall (purple) is in the top left corner where the sky should be segmented instead.

8.5.2 Aachen Day Images

Based on hypotheses from Section 8.1 and used methods described at the beginning of Section 8.5 the following experiments were carried.

At first, the semantic segmentations for all Aachen Day images were obtained. An example of semantic segmentation can be seen in Figure 8.2 (right image). The output of segmentation is in the color palette of the Cityscape dataset. For example, people are red, the sky is blue, and trees are green.

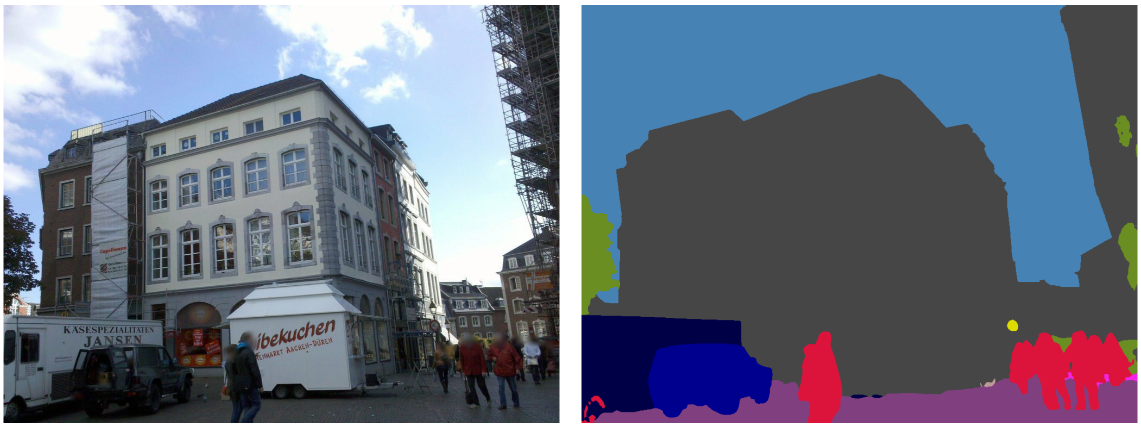

Figure 8.2: Left: an original Aachen Day image. Right: semantic segmentation in the color palette of the Cityscape dataset.

The segmented image contains all semantic classes. The four groups of semantic classes were extracted accordingly to description in Section 8.4. The segmented classes in every group were unified and one binary mask was created. The mask was used for masking the original image. The examples of masking results for specific group of classes can be seen in Figures 8.3, 8.4, 8.5, and 8.6.

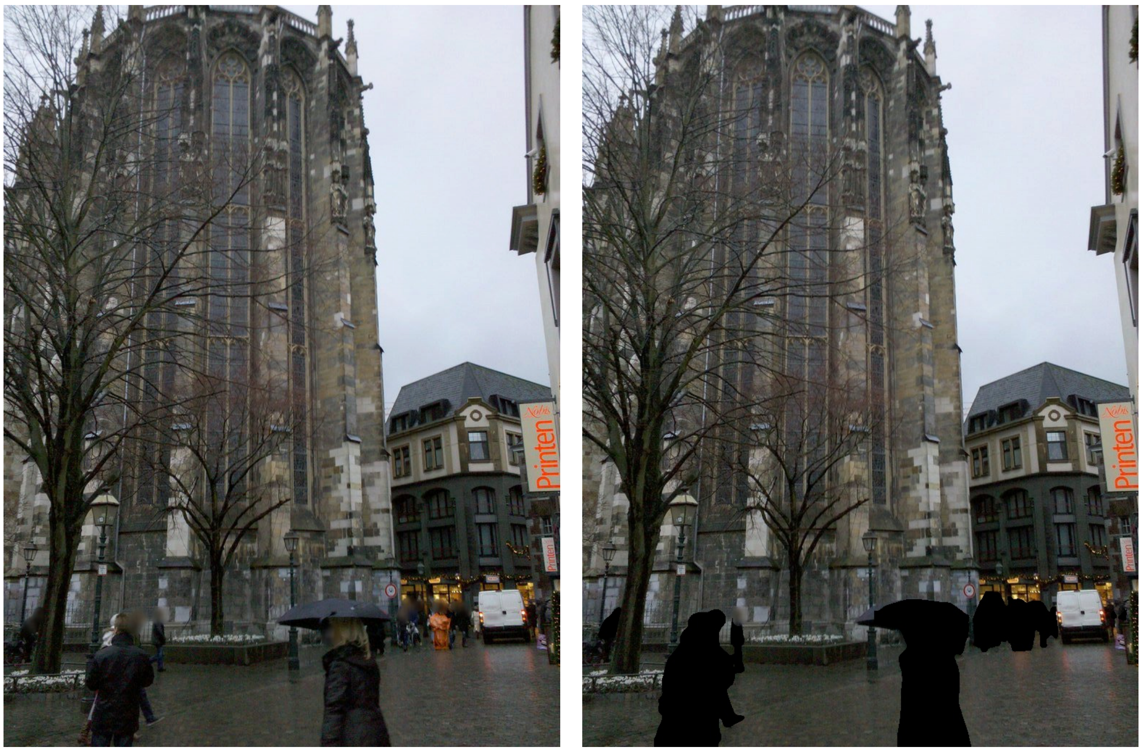

Figure 8.3: Left: original Aachen Day image. Right: masked **human** h group represents person and rider Cityscape classes.

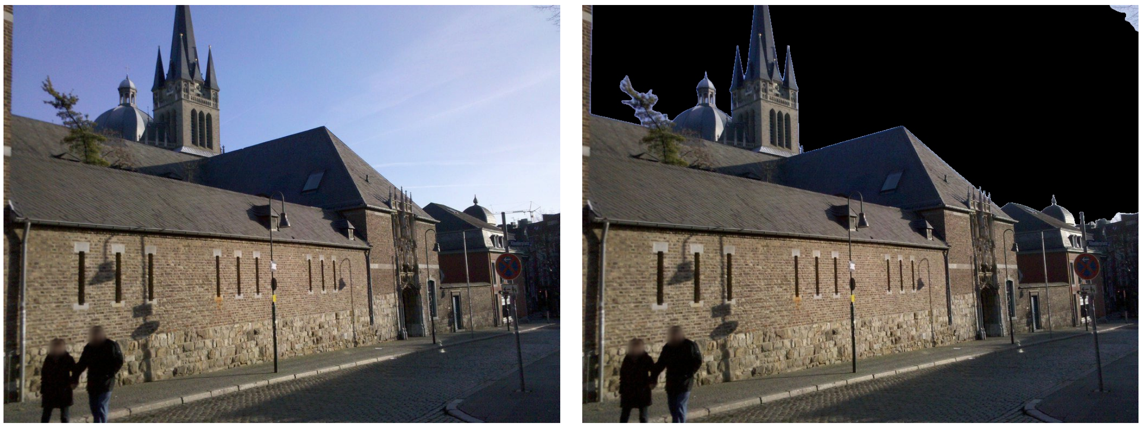

Figure 8.4: Left: original Aachen Day image. Right: masked **sky** s group is represented only by sky Cityscape class.

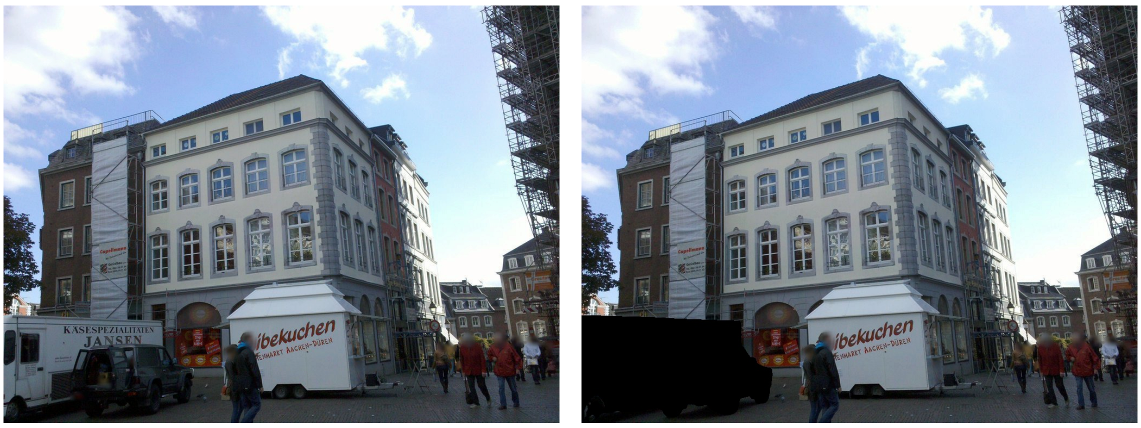

Figure 8.5: Left: original Aachen Day image. Right: masked **vehicle** v group aggregates car, truck, bus, caravan, trailer, train, motorcycle, bicycle, and license plate Cityscape classes.

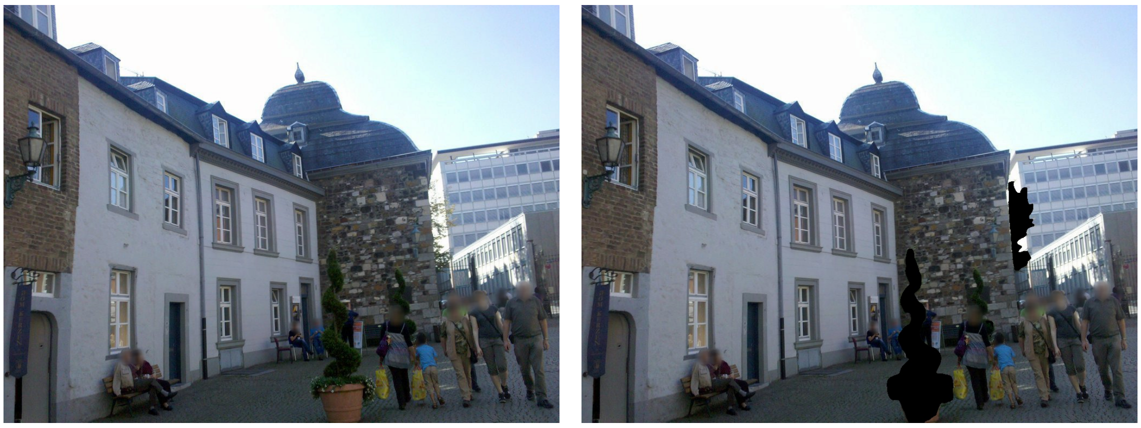

Figure 8.6: Left: original Aachen Day image. Right: masked **nature** n group consists of vegetation and terrain Cityscape classes.

The example of using all groups of classes for masking original Aachen Day image can be seen in the following Figure 8.7.

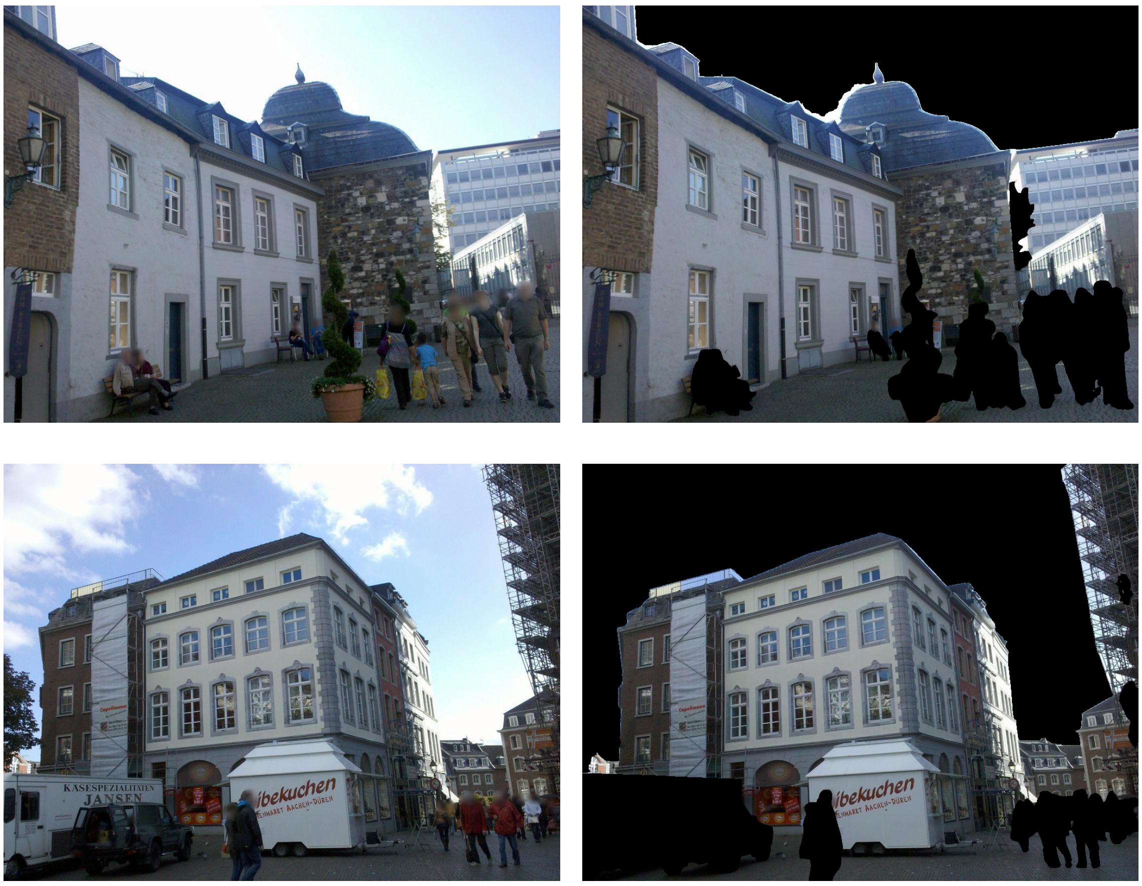

Figure 8.7: Two different examples (rows) of the final masking of input Aachen Day images. Left: original image. Right: masked image with used all four groups of classes (sky, human, vehicle, and nature).

All combinations of groups of classes were created and all experiments were carried out. The results can be seen in Table 8.1. The percentages represent how many images were successfully localized within a threshold from the dataset. Every experiment was rerun five times and averaged to eliminate randomness. The top performed result is with the human h group of classes and the second best one is with the vehicle v group of classes. Thus, the hypothesis from Section 8.1 was valid and confirmed.

Semantic Segmentation Groups of Classes	**Average Day [%]**
h	**89.6** / 95.5 / **98.8**
v	89.4 / 95.4 / **98.8**
hsv	89.3 / **95.8** / **98.8**
hs	89.3 / 95.4 / **98.8**
ns	89.2 / 94.7 / 98.5
hv	89.1 / 95.4 / **98.8**
s	89.0 / 95.4 / **98.8**
sv	89.0 / 95.4 / **98.8**
hns	88.6 / 94.5 / 98.5
n	88.5 / 94.9 / 98.5
hnv	88.5 / 94.7 / 98.5
$hnsv$	88.5 / 94.7 / 98.5
hn	88.5 / 94.4 / 98.5
nv	88.3 / 94.8 / 98.5
nsv	88.3 / 94.5 / 98.5

Table 8.1: The used characters in the Semantic Segmentation Groups of Classes column represents combinations of used groups of classes for masking input images. More specifically human h, vehicle v, sky s, and nature n groups. The percentages represent how many images were successfully localized within a threshold from the dataset.

On the other hand, the experiments that contain nature n group of classes performed poorly and are at the bottom of Table 8.1. It may be caused by the fact that segmented and masked trees look different in every year season. During Fall, Winter, and Spring seasons, it is possible to see through the deciduous trees (where buildings can be located) and use the visual information that is impossible to see during the Summer season. However, the semantic segmentation method labels the branches of deciduous trees and the buildings behind them as the tree class.

The final comparison of obtained results is in Table 8.2. Regarding to the results it can be stated that the experiment with semantic segmentation where group of classes h (human) was used (named: Masked h HLoc) performs superior to the current state-of-the-art method the Original HLoc. It outperforms the Original HLoc by 0.1% in the second best localization threshold on Aachen Day dataset and it matches results in other localization thresholds. The Tuned HLoc method from Chapter 7 outperforms both the Masked h HLoc and the Original HLoc methods on Aachen Day dataset.

Method	**Day** [%]
Tuned HLoc	**89.7** / **95.8** / 98.7
Masked *h* HLoc	89.6 / 95.5 / **98.8**
Original HLoc [126]	89.6 / 95.4 / **98.8**

Table 8.2: The comparison of Tuned HLoc, Masked *h* HLoc, and Original HLoc precision.

8.6 Discussion

The approach of using semantic segmentation in presented way helped to achieve comparable state-of-the-art results and outperformed the Original HLoc by 0.1% in the second best localization threshold on the Aachen Day dataset.

More importantly, the ideas and hypotheses from Section 8.1 were confirmed by performed experiments. They confirm the existence of an image's area that can be masked out and removed (more specifically, replaced by black pixels). And the overall results of the long-term visual localization pipeline are not worsened and provide similar or better results (see Tables 8.1 and 8.2).

Thus, it can be clearly stated that the area of the image that is represented by the human group of Cityscapes classes (person and rider) can be removed. It does not provide valuable information for long-term visual localization tasks because it contains dynamic objects and only provides unmatchable keypoints. Therefore, the detected keypoints on the *h* group of classes can only worsen the solution's precision. If they are matched, it is undoubtedly a wrong match.

Chapter 9

SuperPoint Training

From previous Chapters 7 and 8 it is obvious that a better approach for adding semantic segmentation information into the long-term visual localization task could be beneficial.

This Chapter describes the pipeline and experiments of training custom SuperPoint. The main aim of the experiments is trying to train the SuperPoint to detect only keypoints on static objects. Such a SuperPoint should be well generalized using the semantic information contained in the image. It will help to overcome the two-staged process (detect all keypoints and then use only static keypoints) that will need to be applied instead (see Chapter 10).

9.1 Training

The original training code released by Rémi Pautrat and Paul-Edouard Sarlin on GitHub[1] was used and adapted for the SuperPoint training process.

The training procedure follows the outline from the original paper [7] and was briefly described in Section 4.7.

Training MagicPoint on Synthetic Shapes

First step of training SuperPoint is to train a base detector called MagicPoint. A large-scale dataset of synthetic data is needed for the MagicPoint training. The dataset is called Synthetic Shapes according to [7]. Synthetic Shapes contains simple 2D objects such as quadrilaterals, triangles, lines, and ellipses (see Figure 9.1). This generated dataset removes label ambiguity because it contains easily detected Y, L, and T-junctions for robust keypoint detection.

The disadvantage of using only synthetically generated data is that it does not contain all real-world shapes and conditions of all potential keypoints. MagicPoint

[1] https://github.com/rpautrat/SuperPoint

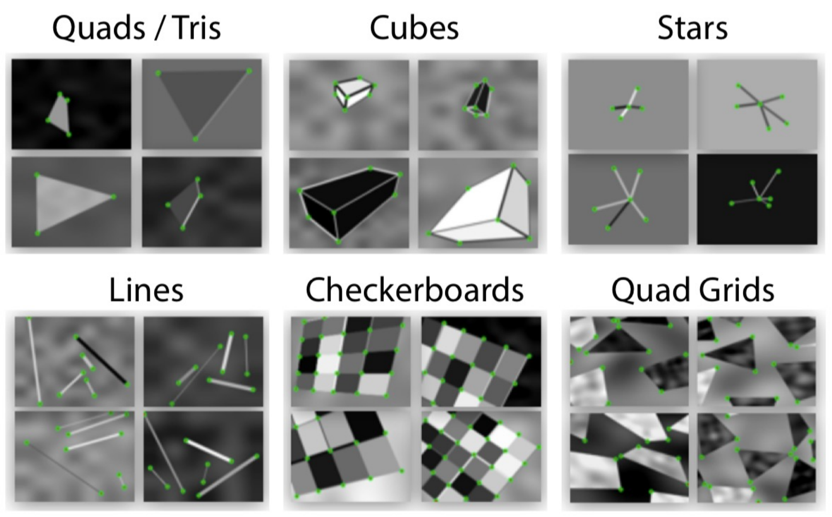

Figure 9.1: Example of Synthetic Shapes dataset: triangles, quadrilaterals, lines, cubes, checkerboards, and stars each shape with ground truth corner locations (image was taken from [7]).

keypoint detector performs well on synthetically generated images but only reasonably well for real-world images. It is the trade-off between not having that extensive manually labeled real-world dataset with ground truth keypoints and not having the perfect real-world keypoint detection performance. As stated before, the MagicPoint keypoint detector performs reasonable well and can be used for the next step of training SuperPoint.

The detector performance of Mean Average Precision (MAP) on 1000 held-out images of the Synthetic Shapes dataset can be seen in Table 9.1.

		MagicPoint	Original MagicPoint [7]	FAST	Harris	Shi
MAP	no noise	0.975	**0.979**	0.405	0.678	0.686
MAP	noise	0.968	**0.971**	0.061	0.213	0.157

Table 9.1: Performance of Synthetic Shapes detector: MagicPoint and Original MagicPoint [7] models outperform classical detectors in detecting corners of simple geometric shapes and are robust to added noise.

Exporting Detections on MS-COCO

The MS-COCO dataset [129] is labeled with trained base MagicPoint detector. It means that a trained MagicPoint keypoint detector performs detection on the MS-COCO dataset. The detected keypoints are used as labels for the MS-COCO dataset for further training.

Training MagicPoint on MS-COCO

This step is similar to the first step of training MagicPoint on the Synthetic Shapes dataset. However, there is only one difference: instead of using the Synthetic Shapes dataset for training, the labeled MS-COCO dataset from the previous step is used. This step leads to a better generalization of trained MagicPoint.

This step and the previous one can be repeated multiple times for better MagicPoint detector performance (however, it is not required and usually does not help). For further experiments, only one iteration of training MagicPoint was performed.

Evaluating the Repeatability on HPatches

At this point the repeatability on HPatches dataset from Section 5.9 is evaluated. The results of the trained MagicPoint detector are similar to the Original MagicPoint detector reported in [7]. Thus, it can be stated that the trained MagicPoint detector is comparable good and successful (see Table 9.2).

	IS, NMS=4	IS, NMS=8	VS, NMS=4	VS, NMS=8
SuperPoint[7]	**0.652**	**0.631**	0.503	**0.484**
Original MagicPoint[7]	0.575	0.507	0.322	0.260
MagicPoint	0.564	0.501	0.314	0.252
FAST	0.575	0.472	0.503	0.404
Harris	0.620	0.533	**0.556**	0.461
Shi	0.606	0.511	0.552	0.453
Random	0.101	0.103	0.100	0.104

Table 9.2: The repeatability results of trained MagicPoint detector on HPatches dataset, 57 Illumination Scenes (IS) and 59 Viewport Scenes (VS).

Training SuperPoint on MS-COCO

The first tried approach on how to incorporate semantic information about underlying static/dynamic visual objects into current SuperPoint architecture is to mask the area of the particular dynamic object with black pixels and use these images for training the SuperPoint. Figure 9.2 shows two examples of segmentations. For SuperPoint training (masked out / blacked), two groups of classes were used: human and vehicle. It can be seen in Figure 9.2 on the right side. Red areas represent human class, and blue areas represent vehicle class.

This approach did not lead to the expected behavior and did not help with detecting only the keypoints on static objects. Due to the number of training images in the MS-COCO dataset and the generalization ability of the SuperPoint, this training approach failed. It means it detects all keypoints but does not respect underlying visual semantic information.

The second tried approach is to eliminate the keypoints detected on underlying dynamic objects during the SuperPoint training by modifying pseudo-ground truth

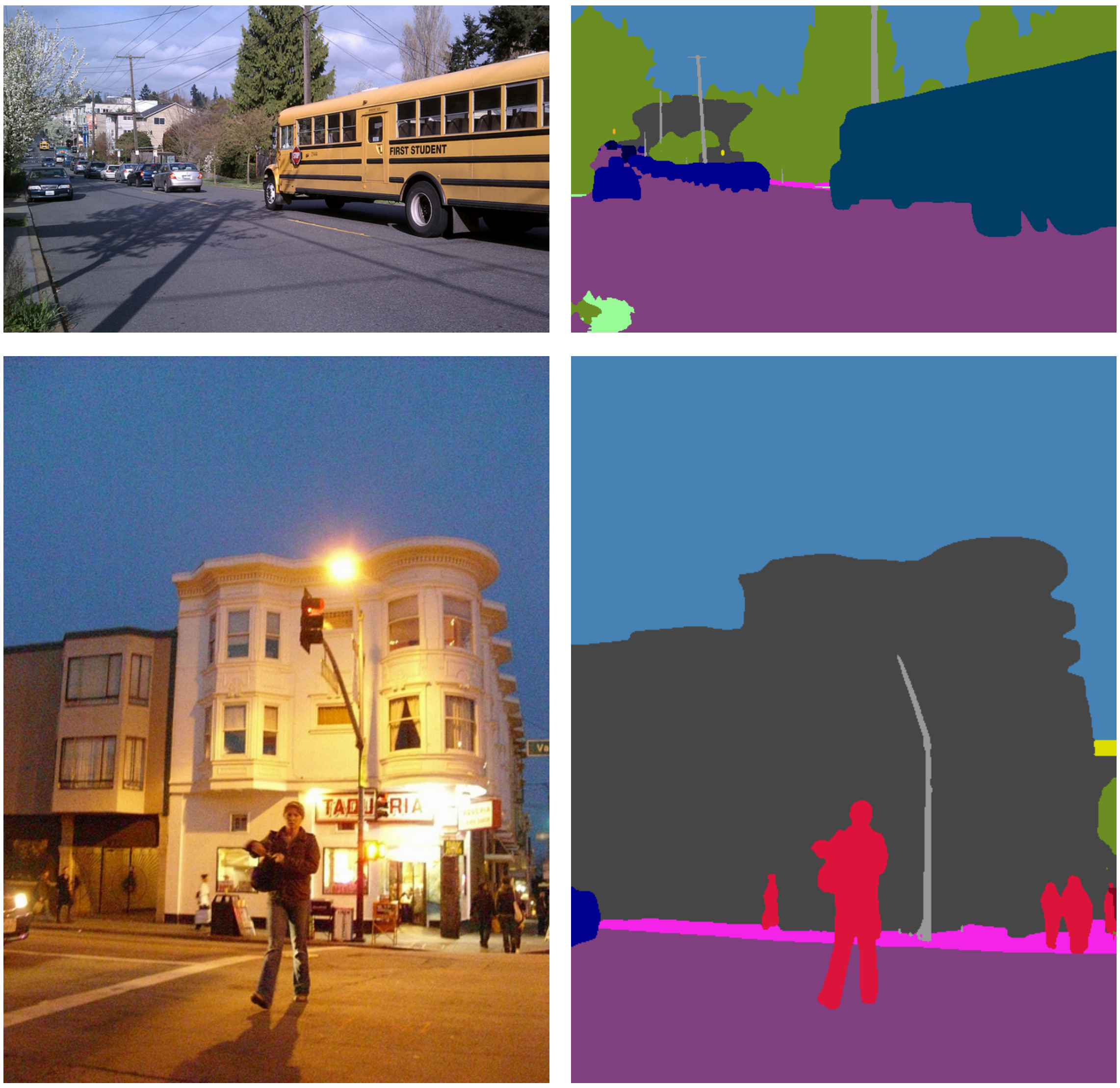
Figure 9.2: Two examples of MS-COCO dataset. The original image (left) and its segmentation (right) segmented with HRNet-OCR method from Section 4.9.

keypoints generated by the MagicPoint detector. The keypoints detected on dynamic objects are removed from the set of detected keypoints. The results were similar to the first approach. The SuperPoint keypoint detector and descriptor learned to detect keypoints generally and ignored the added semantic information about dynamic objects. The added semantic information here means filtered/removed dynamic keypoints from pseudo-ground truth data. The dynamic keypoint can be defined as a keypoint detected on an object's area labeled as dynamic by semantic segmentation such as a car or human.

9.2 Discussion

Even though both methods yielded a relatively good SuperPoint model (examined by raw looking at sets of detected keypoints), unfortunately, none of the two mentioned training approaches were successful. One of the reasons is that the MS-COCO dataset contains many common objects from the real world, where humans and vehicles are not highly presented objects. Thus, the generalization ability of SuperPoint outweighed added information from semantic segmentation, and SuperPoint was not able to learn this added information.

An improvement for the two mentioned approaches can be to use only the training images where selected dynamic objects are widely presented. Nevertheless, the trained SuperPoint keypoint detector and descriptor can detect all keypoints and not respect the set semantic constraints. The modification of SuperPoint's neural network architecture would be needed for achieving a keypoint selectivity based on semantic segmentation, but it was not in the scope of this experiment.

As was discussed at the beginning of this Chapter, the two-step approach (detect all keypoints and then use only the static ones) can be the solution for the outlined problem and brings the boost of visual localization accuracy. More experiments based on this idea are discussed in Chapter 10.

Chapter 10

Keypoint Filtering

From the previous chapters and performed experiments, a different approach to keypoint filtering needs to be applied. This chapter discusses selected datasets, and an enhanced process of semantic segmentation keypoint filtering is described. The achieved results are presented. Finally, the discussion is carried out along with a comparison table for selected datasets.

10.1 Selected Datasets

Two datasets: Aachen v1.1 (Section 5.6 more specifically Section 5.6.1) and 4Seasons (Section 5.8) were selected for exemeriments in this chapter. The main dataset selecting criteria were:

- dataset has to contain an urban environment,
- dataset has to contain human and vehicle groups of classes from semantic segmentation,
- dataset has to be easy to evaluate - localization ground truth data has to be available, or an automatic submission mechanism has to be working,
- dataset has to be used for benchmarks and challenges for easy comparing with other methods and results, and
- dataset has to be well known in the community of scientists.

The standard query set of images was selected for Aachen v1.1, and it was evaluated on the automatic online validation system[1]. The database sequence was used for building the reference localization model. For evaluation, only the day images were considered because the selected HRNet-OCR semantic segmentation method fails for the night images or poorly segments the night images.

[1]https://www.visuallocalization.net/

Five sequences for the 4Seasons dataset are available for visual localization (reference, training, validation, test0, and test1). The reference sequence was used for building the reference localization model. The training and validation sequences contain ground truth positions, and the local evaluation can be performed. The training sequence can be taken for evaluation because none of the used models was trained with the 4Seasons training sequence. SuperPoint [7] keypoint detector and descriptor was trained from MS-COCO dataset [129], SuperGlue [4] keypoint matcher was trained from MegaDepth dataset [69], and HRNet-OCR [6] semantic segmentation model was trained from Cityscape dataset [130].

10.2 Parameters Settings

The HLoc pipeline described in Section 6.3 with following parameters (if not explicitly stated otherwise) was used for all further experiments. SuperPoint and SuperGlue were used for 4Seasons and Aachen v1.1 datasets with these settings: NMS radius = 3 pixels, maximum number of keypoints = 1024 keypoints, maximum resize size (size of longer side of image) = 1024 pixels, transform to grayscale = True, SuperGlue weights = outdoor, iterations of Sinkhorn algorithm = 5 iterations (for speed up matching).

Moreover, the following specific parameters were used for Aachen v1.1 dataset: retrieval method NetVLAD with top n most similar images = 50 images. The number of localization pairs was set to 10 images for the 4Seasons dataset.

All other parameters were kept the same for Aachen v1.1 and 4Seasons datasets and their values were set to the default values described in Section 7.2.

10.3 Experiments Description

The HLoc pipeline[2] was used as the baseline and it was heavily edited and enhanced to be able to run the following experiments.

The experiments' leading hypothesis is that the overall long-term visual localization should be improved and more accurate by removing dynamic keypoints. There is a meager chance that the dynamic object will be captured at the same place in two different image sequences and times because only long-term visual localization is examined. Thus, dynamic keypoints bring only noise into the matching phase, and even if the dynamic keypoint is matched, there is a high chance that it is a wrong match. These miss-matches add only noise into the final localization (if they survive miss-matches elimination phases like geometric verification and RANSAC).

[2] https://github.com/cvg/Hierarchical-Localization

The process of fulfilling the outlined hypothesis can be described as follows: for each image $I(x, y)$ in the given dataset D, semantic segmentation ss is performed for each group of classes $gc \in \{h, v\}$, where h stands for human and v for vehicle group of classes. Then, it is stored the binary semantic masks $M_{gc\in\{h,v\}} = ss(gc, I(x, y))$, where $M_{gc\in\{h,v\}}(x, y)$ for a pixel (x, y) is 1 for dynamic objects and 0 for static objects (see Section 10.4). All keypoints $kp_{i=\{1,\ \cdots,\ 1024\}}$ for $\forall I(x, y) \in D$ are detected. Next, each keypoint is labeled as dynamic or static based on its location (x, y) and all underlying binary semantic masks' values $\forall I\ (x, y) \in D$ and each its keypoint kp_i:

$$\text{if} \sum_{gc\in\{h,v\}} kp_i M_{gc}(x, y) = 0 \begin{cases} \text{then } kp_i \text{ is static,} \\ \text{else } kp_i, \text{ is dynamic,} \end{cases} \tag{10.1}$$

where $i = \{1,\ \cdots,\ 1024\}$ is a number of keypoints in an image.

The robustness of the proposed methods is needed to be examined. The robust method needs to perform even with a lower amount of keypoints. Thus all keypoints are detected and then labeled accordingly to Equation 10.1. The number of static keypoints is lowered from 100% by 5% as low as only 5% keypoints remains. The number of dynamic keypoints is unchanged for original methods, but it is set to 0% for newly proposed methods. That means all dynamic keypoints are removed.

For example, if 95% of all detected static keypoints must remain, 5% of static keypoints are randomly selected and removed, then the 95% of remaining static keypoints are utilized for original and newly proposed methods. The random selection is different for each experiment, method, and run (but the implementation of randomness uses the same seed for the random number generator).

The next step is to iteratively decrease the number of static keypoints together with enabling or disabling dynamic keypoint filtering and execute the whole HLoc pipeline for each experiment and method.

10.4 Semantic Segmentation

The HRNet-OCR semantic segmentations were precalculated for each dataset from Section 10.1 and for each image from all available image sequences. Then, the binary masks were extracted and stored for human h and vehicle v group of classes separately in one h5 file. These binary masks are used for keypoint filtering later in Section 10.5.

The examples of binary masks for vehicle and human group of classes can be seen in Figures 10.1 and 10.2.

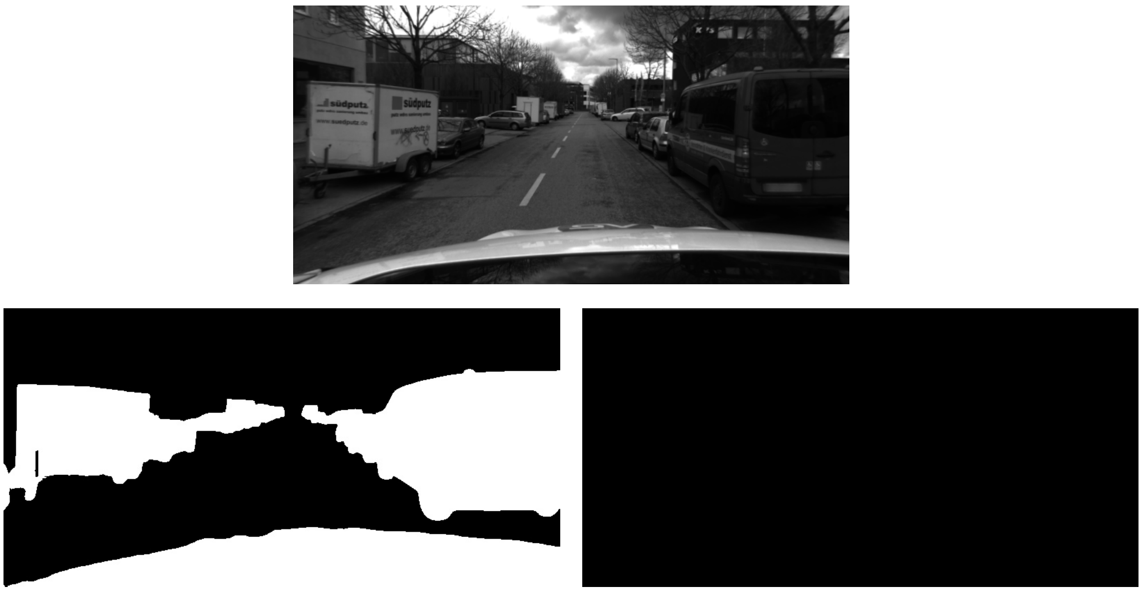

Figure 10.1: Top: an example from 4Seasons dataset. Left: binary mask for vehicle group of classes. Right: binary mask for human group of classes (it is empty because no human is presented in the image).

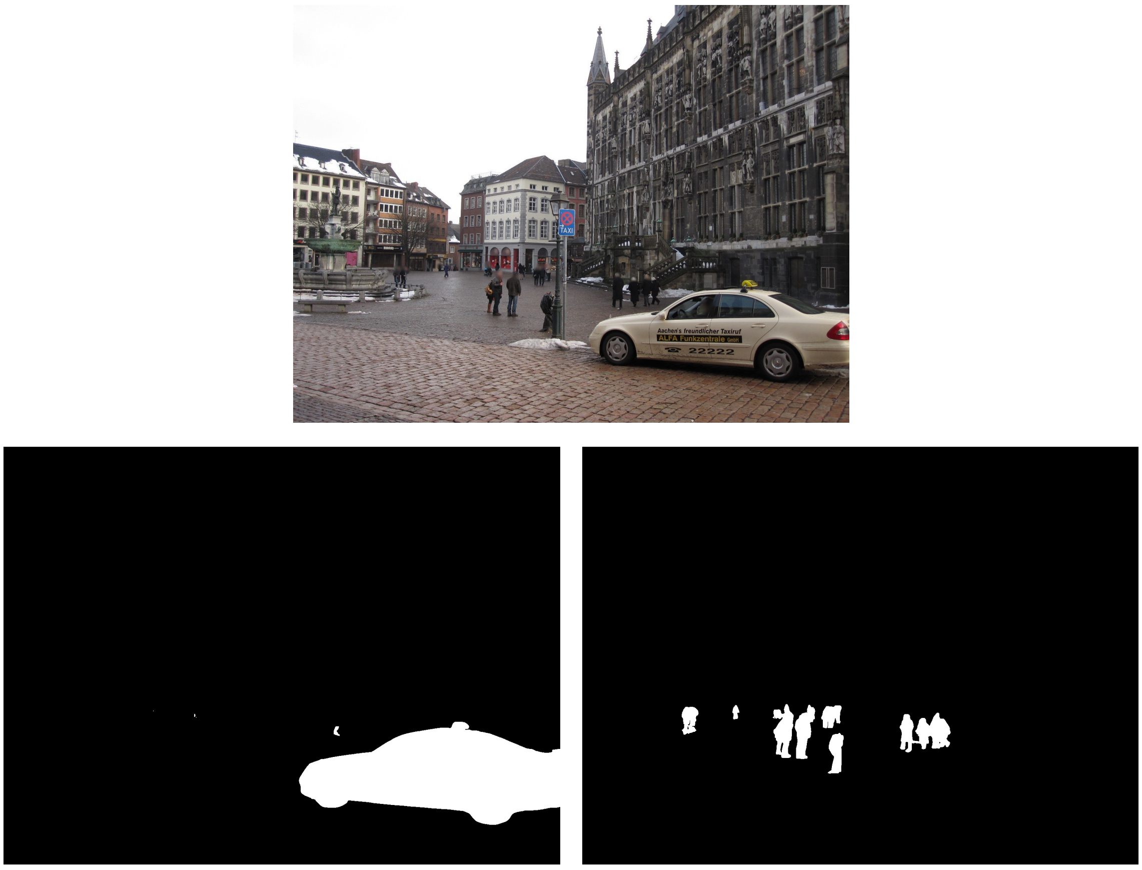

Figure 10.2: Top: an example from Aachen dataset. Left: binary mask for vehicle group of classes. Right: binary mask for human group of classes.

10.5 Keypoint Filtering

For each dataset the keypoint filtering is applied only on query images (reference sequence is unchanged). The keypoint filtering approach was described in Section 10.3 and keypoint labeling was specifically introduced in Equation 10.1.

In Tables 10.1 and 10.2, there is the distribution of dynamic and static keypoints for used dataset sequences. All sequences utilized only human h and vehicle v group of semantic classes. Even though the limit of the maximum number of keypoints per image was set to 1024 (see parameter setting in Section 10.2), the lower number of keypoints can be found in some images. It is caused by the fact that the image does not have a rich visual appearance, and the SuperPoint method can not detect enough keypoints.

Dataset	Aachen v1.1, query sequence, day images
% static keypoints	94.06%
% dynamic keypoints	5.94%

Table 10.1: Percentage of static and dynamic keypoints occurrence for Aachen v1.1 day query sequence (only for human h and vehicle v group of classes).

4Seasons dataset sequence	Training	Validation	Test0	Test1
% static keypoints	89.97%	77.72%	90.26%	**90.36%**
% dynamic keypoints	10.03%	**22.28%**	9.74%	9.64%

Table 10.2: Percentage of static and dynamic keypoints occurrence for 4Seasons sequences (only for human h and vehicle v group of classes).

10.6 Results

All achieved results are presented in figures and tables in this section. If it is not stated otherwise, the parameter setting from Section 10.2 was used for easy result comparison.

10.6.1 4Seasons Training Sequence

The three runs (repetitions) with the same parameters setting of the experiment were performed. Only the run 1 is presented here (run 2 and run 3 can be seen in Appendix A). All runs performed similarly, and thus general conclusions can be made. For the 4Seasons training sequence, the ground truth localizations are publicly available. The obtained results were verified locally[3] due to ground truth localizations.

The achieved results by the Filtered dynamic KPs HLoc method surpassed the Original HLoc method and achieved better results on 4Seasons training sequence for all localization thresholds. All results can be seen in Table 10.3 and Figure 10.3 (with detailed Figures 10.4, 10.5, and 10.6).

Method	**Training sequence** [%]
Original HLoc	88.3 / 97.7 / 98.9
Filtered dynamic KPs HLoc	**92.4** / **99.6** / **100.0**

Table 10.3: The table shows the comparsion of the Original HLoc method and the Filtered dynamic KPs HLoc method (both with the same parameters setting). The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The results are reported for standard precision thresholds 0.10m / 0.20m / 0.50m.

[3]Locally means that everyone can verify the results on their computers, and no third party is needed.

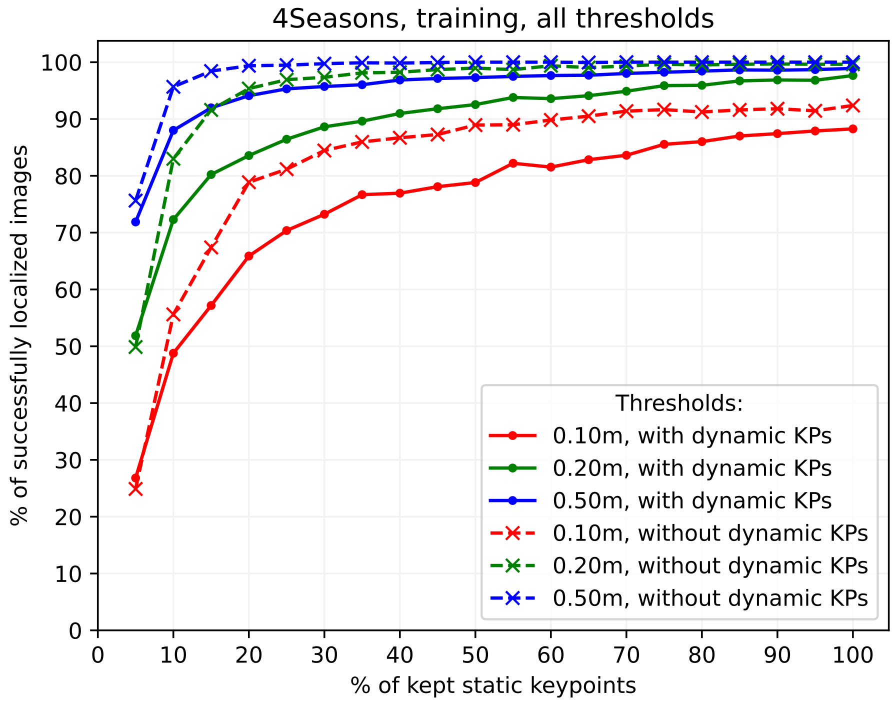

Figure 10.3: 4Seasons, training sequence, run 1: the results for all localization thresholds for experiments with and without dynamic keypoints.

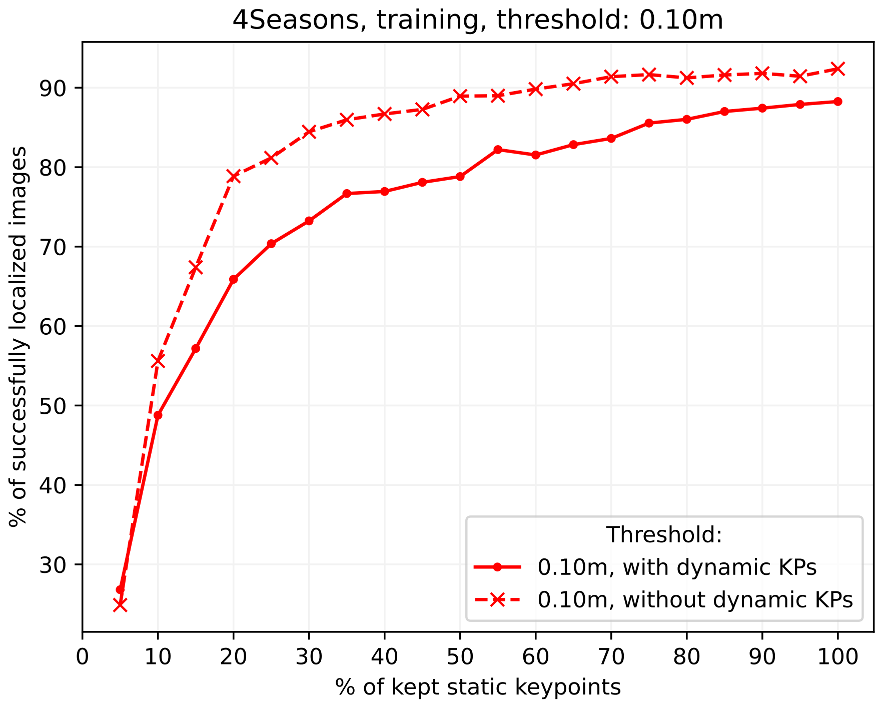

Figure 10.4: 4Seasons, training sequence, run 1: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

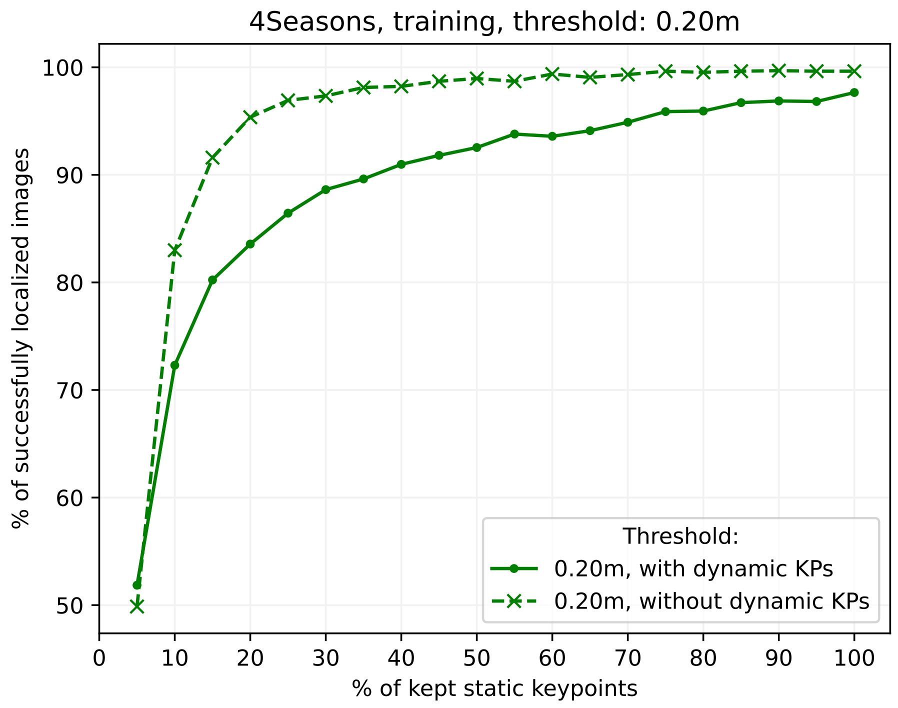

Figure 10.5: 4Seasons, training sequence, run 1: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

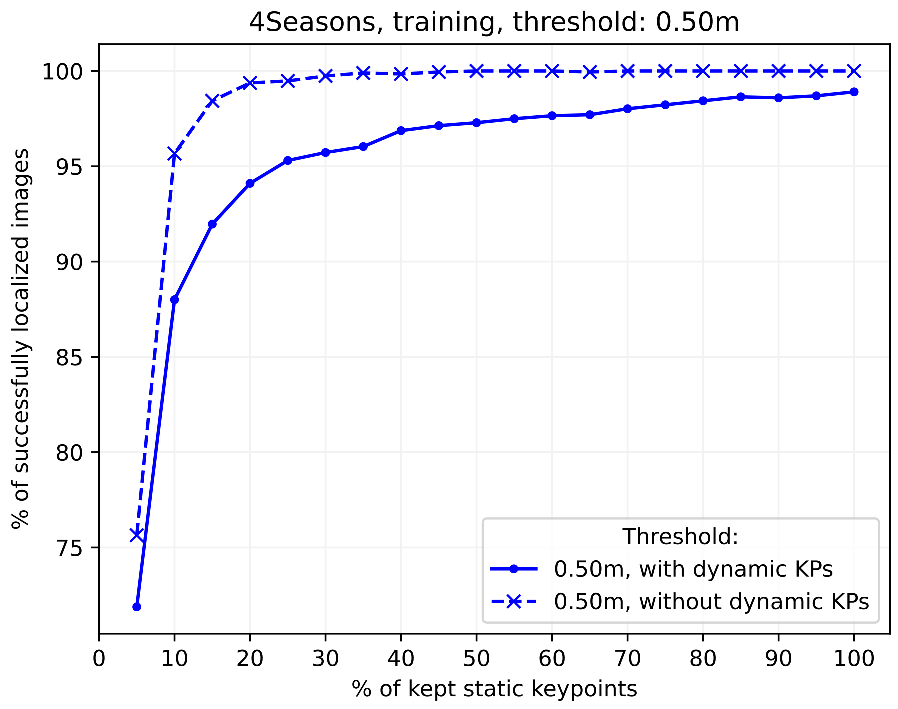

Figure 10.6: 4Seasons, training sequence, run 1: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

10.6.2 4Seasons Validation Sequence

This section describes the results of the 4Seasons validation sequence and is similar to the previous Section 10.6.1. The three runs (repetitions) with the same parameters setting of the experiment were performed. Only the run 1 is presented here (run 2 and run 3 can be seen in Appendix B). All runs performed similarly, and thus general conclusions can be made. For the 4Seasons validation sequence, the ground truth localizations are publicly available. The obtained results were verified locally[4] due to ground truth localizations.

The achieved results by the Filtered dynamic KPs HLoc method surpassed the Original HLoc method and achieved better results on 4Seasons validation sequence for all localization thresholds. All results can be seen in Table 10.4 and Figure 10.7 (with detailed Figures 10.8, 10.9, and 10.10).

Method	**Validation sequence [%]**
Original HLoc	81.8 / 96.2 / 99.7
Filtered dynamic KPs HLoc	**89.4** / **98.3** / **100.0**

Table 10.4: The table shows the comparsion of the Original HLoc method and the Filtered dynamic KPs HLoc method (both with the same parameters setting). The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The results are reported for standard precision thresholds 0.10m / 0.20m / 0.50m.

[4]Locally means that everyone can verify the results on their computers, and no third party is needed.

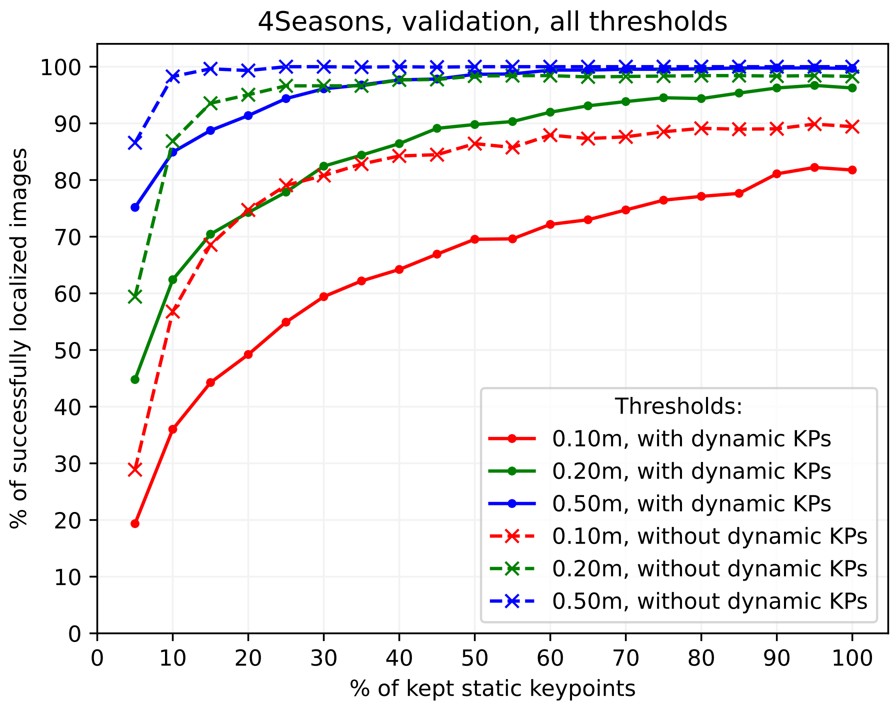

Figure 10.7: 4Seasons, validation sequence, run 1: the results for all localization thresholds for experiments with and without dynamic keypoints.

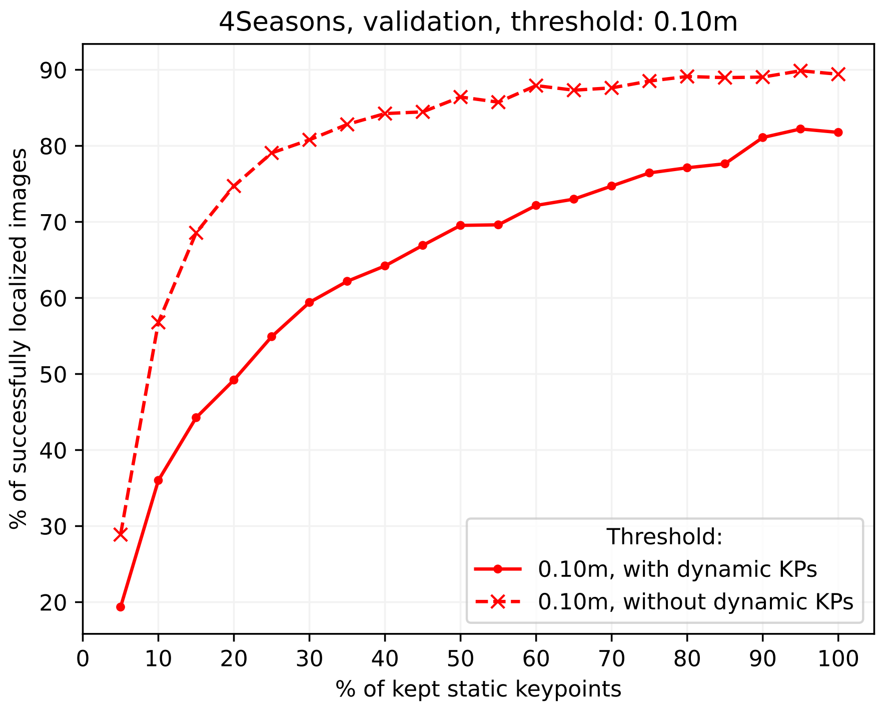

Figure 10.8: 4Seasons, validation sequence, run 1: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

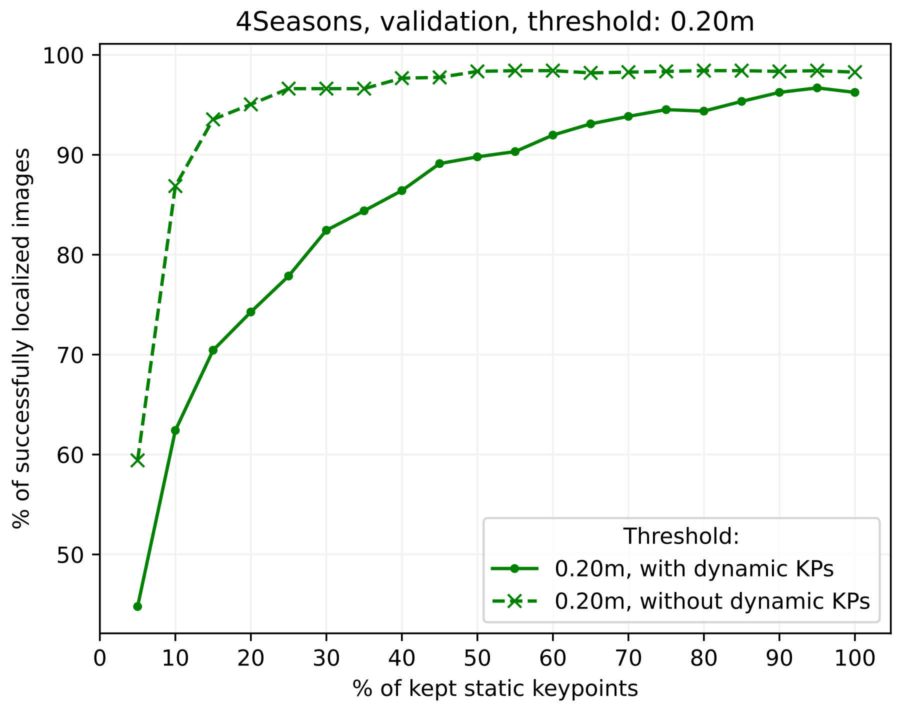

Figure 10.9: 4Seasons, validation sequence, run 1: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

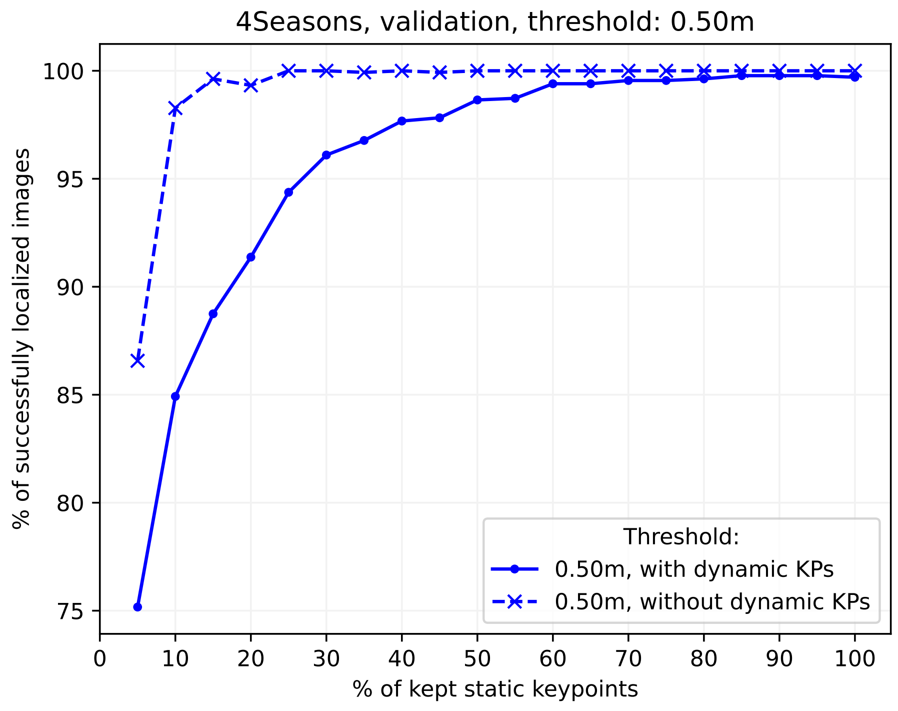

Figure 10.10: 4Seasons, validation sequence, run 1: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

10.6.3 4Seasons Test0 Sequence

Two experiments were performed for the 4Seasons test0 sequence, each with one run. The first Experiment 1 with the parameters setting from Section 10.2 was computed. The second Experiment 2 with the following parameters setting: the maximum number of keypoints = 4096 keypoints, maximum resize size (size of the longer side of image) = 1600 pixels, iterations of Sinkhorn algorithm = 50 iterations (slow but more robust). All other parameters settings were kept unchanged and the same as described in Section 10.2.

For the 4Seasons test0 sequence, the ground truth localizations are not publicly available, and the obtained results were sent for verification to Patrick Wenzel[5] (one of the 4Seasons dataset creators).

The Filtered dynamic KPs HLoc method surpassed the Original HLoc method and all other publicly known methods. The results are shown in Table 10.5 and Figure 10.11 (with detailed Figures 10.12, 10.13, and 10.14) for the Experiment 1 and Figure 10.15 (with detailed Figures 10.16, 10.17, and 10.18) for the Experiment 2. It achieved the new state-of-the-art results on the 4Seasons test0 sequence with the Experiment 2. Even in the Experiment 1 (with worse parameters setting) the Filtered dynamic KPs HLoc method performed better for 0.20m and 0.50m localization thresholds compared to the Experiment 2, Original HLoc method.

Experiment 2 performed better on most levels of kept static keypoint percentages. The localization precision difference between the Experiment 1 and Experiment 2 is mainly caused due to the higher number of the Sinkhorn algorithm iterations in the SuperGlue keypoint matcher.

[5]Patrick Wenzel, wenzel@cs.tum.edu, Technical University of Munich, Computer Vision Group, https://vision.cs.tum.edu, Boltzmannstraße 3, 85748 Garching, Germany.

Method	Test0 sequence [%]
Results from MLAD ECCV 2020[6]	
SuperPoint/SuperGlue + PnP + RANSAC	21.2 / 33.9 / 60.0
R2D2 + PnP + RANSAC	21.5 / 33.1 / 53.0
D2-Net + PnP + RANSAC	12.5 / 29.3 / 56.7
SuperPoint + PnP + RANSAC	15.5 / 27.5 / 47.5
HLoc + SuperGlue	92.3 / 97.8 / 99.2
isrf_N8_10k_o2s	22.3 / 34.1 / 57.4
Results from HLoc GitHub[7]	
HLoc + SuperGlue	91.8 / 97.7 / 99.2
Achieved results	
Experiment 1, original HLoc	89.0 / 96.1 / 99.0
Experiment 1, filtered dynamic KPs HLoc	92.3 / 98.5 / 99.7
Experiment 2, original HLoc	92.5 / 97.8 / 99.2
Experiment 2, filtered dynamic KPs HLoc	**93.8** / **98.7** / **99.8**

Table 10.5: All publicly known state-of-the-art results for the 4Seasons test0 sequence were collected. Furthermore, achieved results are presented here. Experiment 2 outperformed all publicly known state-of-the-art results for all localization thresholds. The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and for 100% static keypoints with 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The standard localization thresholds are 0.10m / 0.20m / 0.50m.

[6] https://sites.google.com/view/mlad-eccv2020/challenge

[7] https://github.com/cvg/Hierarchical-Localization/tree/master/hloc/pipelines/4Seasons

Experiment 1

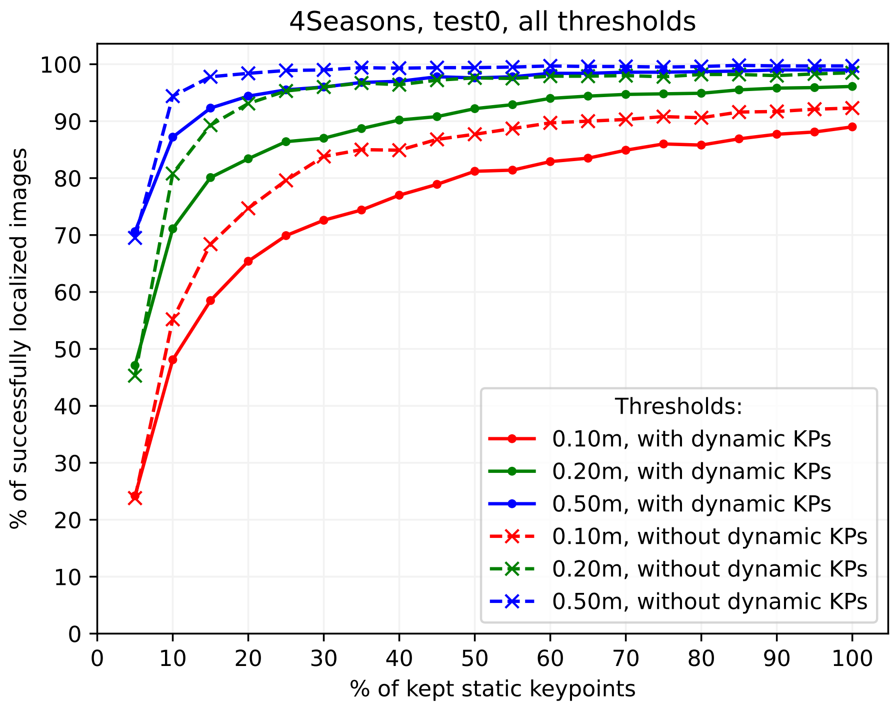

Figure 10.11: 4Seasons, test0 sequence, Experiment 1: the results for all localization thresholds for experiments with and without dynamic keypoints.

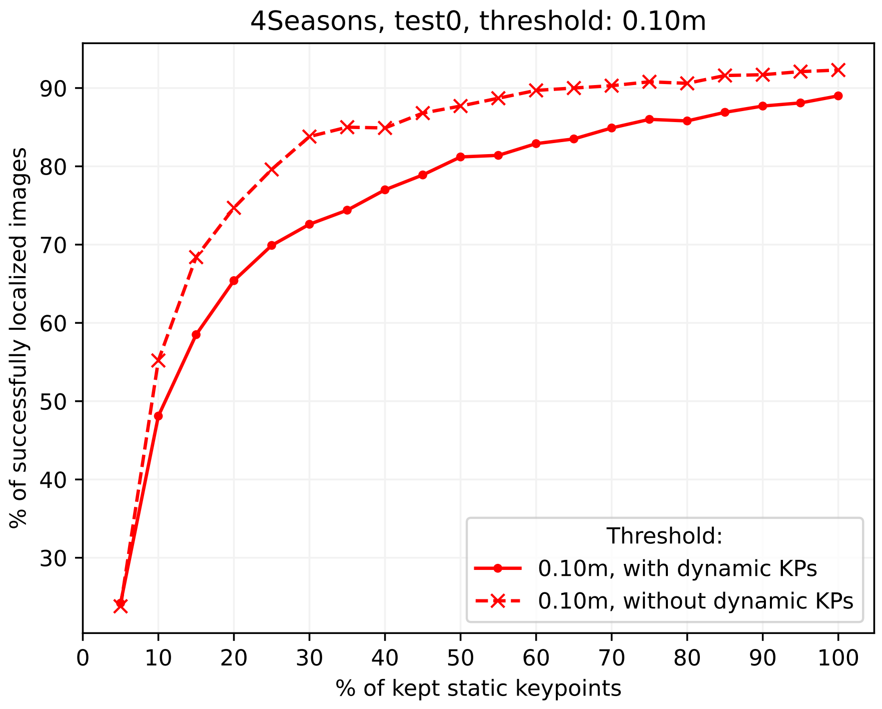

Figure 10.12: 4Seasons, test0 sequence, Experiment 1: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

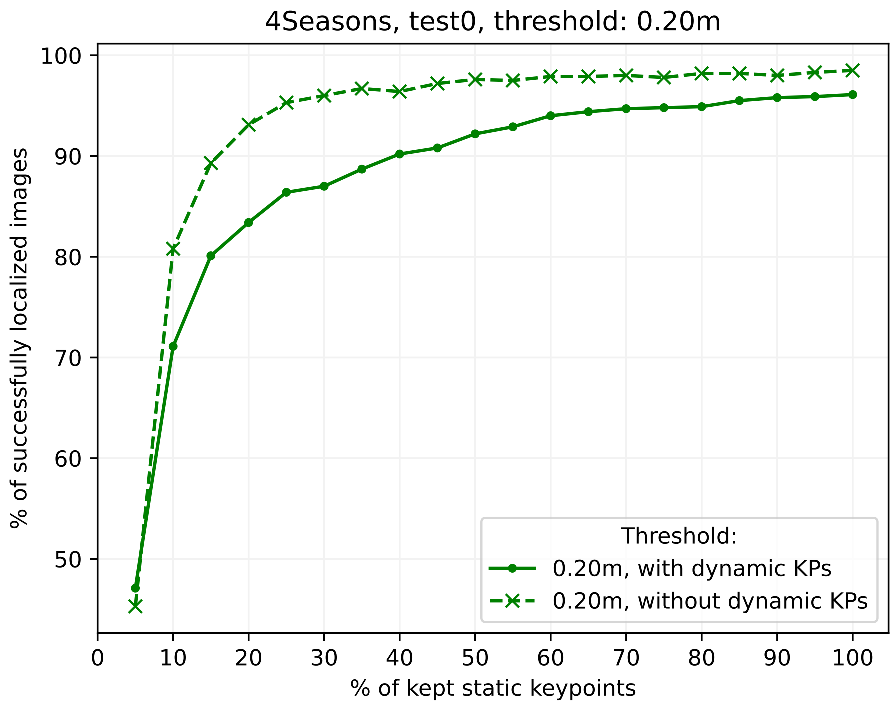

Figure 10.13: 4Seasons, test0 sequence, Experiment 1: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

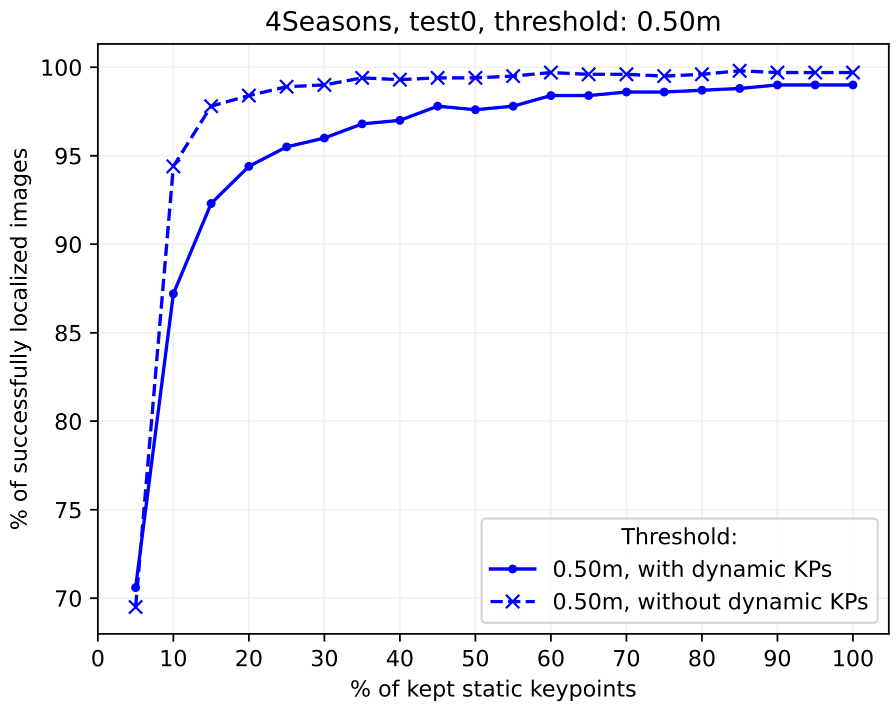

Figure 10.14: 4Seasons, test0 sequence, Experiment 1: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

Experiment 2

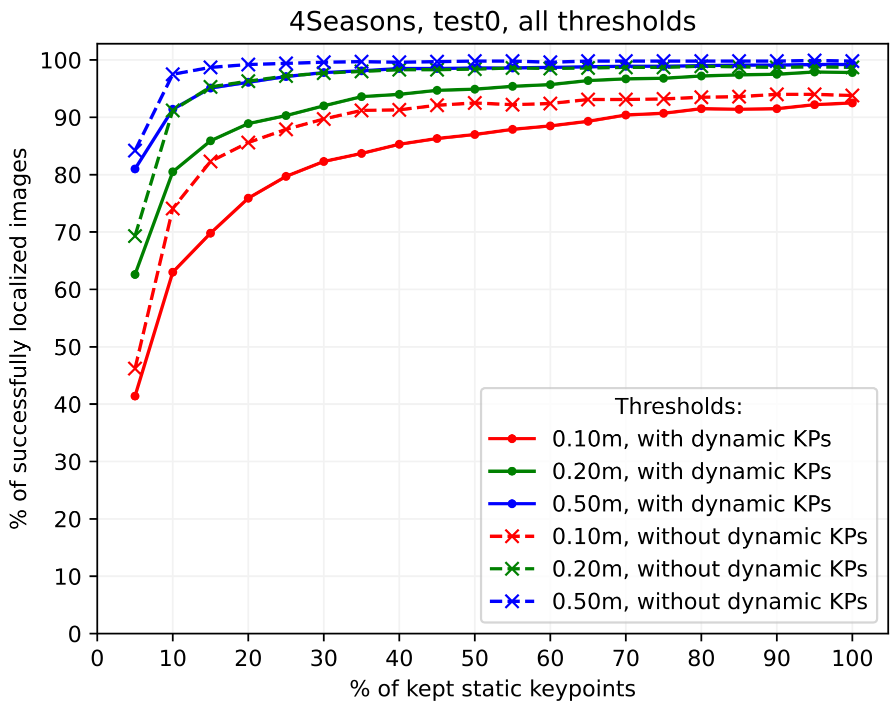

Figure 10.15: 4Seasons, test0 sequence, Experiment 2: the results for all localization thresholds for experiments with and without dynamic keypoints.

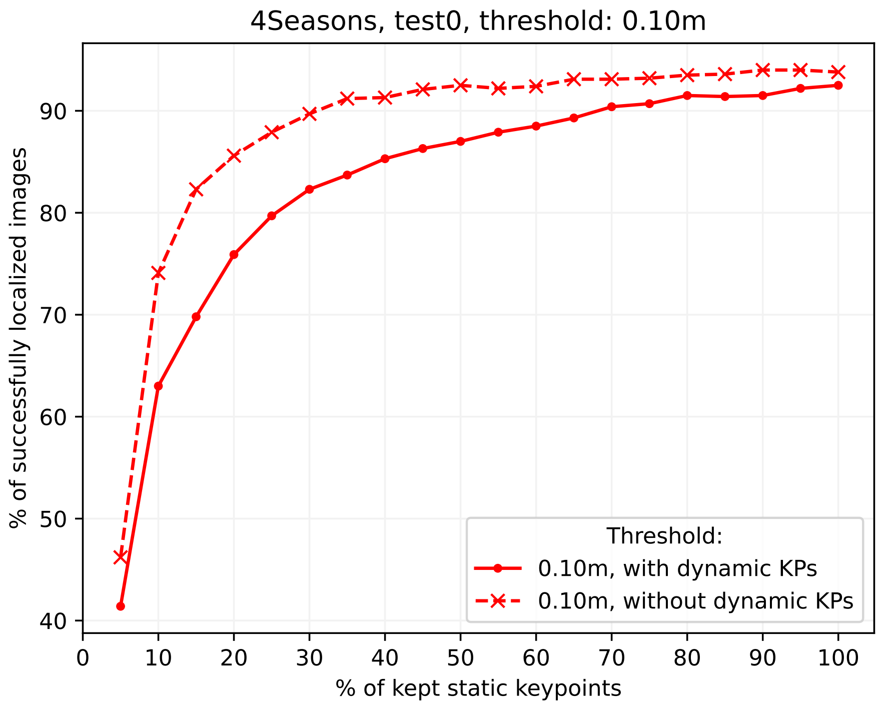

Figure 10.16: 4Seasons, test0 sequence, Experiment 2: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

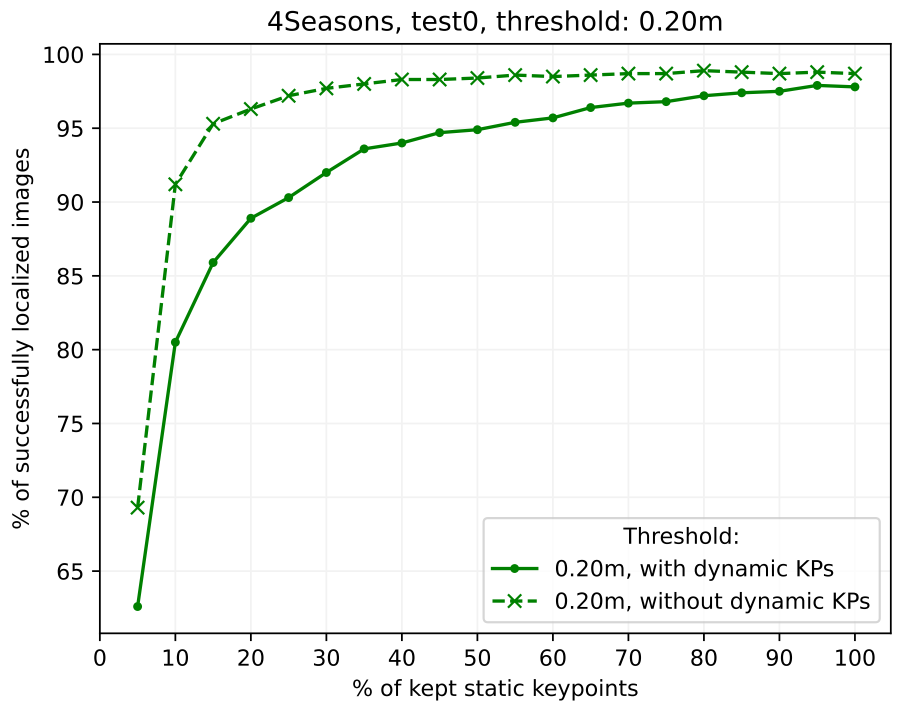

Figure 10.17: 4Seasons, test0 sequence, Experiment 2: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

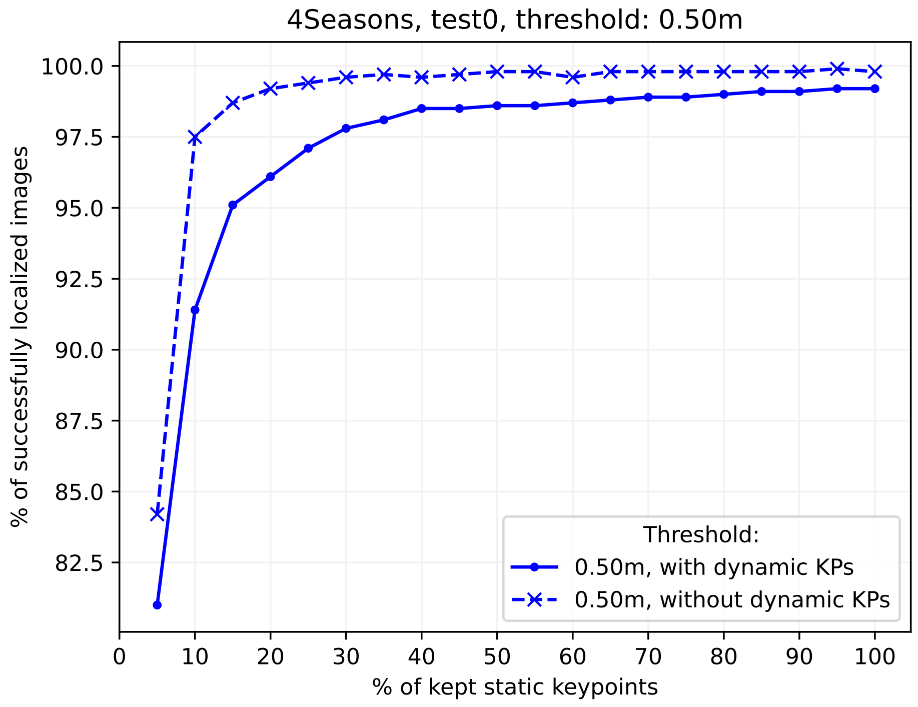

Figure 10.18: 4Seasons, test0 sequence, Experiment 2: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

10.6.4 4Seasons Test1 Sequence

This section describes the results of the 4Seasons test1 sequence and is similar to the previous Section 10.6.3. Two experiments were performed for the 4Seasons test1 sequence, each with one run. The first Experiment 1 with the parameters setting from Section 10.2 was computed. The second Experiment 2 with the following parameters setting: the maximum number of keypoints = 4096 keypoints, maximum resize size (size of the longer side of image) = 1600 pixels, iterations of the Sinkhorn algorithm = 50 iterations (slow but more robust). All other parameters settings were kept unchanged and the same as described in Section 10.2.

For the 4Seasons test1 sequence, the ground truth localizations are not publicly available, and the obtained results were sent for verification to Patrick Wenzel[8] (one of the 4Seasons dataset creators).

The Filtered dynamic KPs HLoc method surpassed the Original HLoc method and all publicly known methods. The results are shown in Table 10.6 and Figure 10.19 (with detailed Figures 10.20, 10.21, and 10.22) for the Experiment 1 and Figure 10.23 (with detailed Figures 10.24, 10.25, and 10.26) for the Experiment 2. It achieved the new state-of-the-art results on the 4Seasons test1 sequence with the Experiment 2.

Experiment 2 performed better on most levels of kept static keypoint percentages. The localization precision difference between Experiment 1 and Experiment 2 is mainly due to the higher number of the Sinkhorn algorithm iterations in the SuperGlue keypoint matcher.

[8]Patrick Wenzel, wenzel@cs.tum.edu, Technical University of Munich, Computer Vision Group, https://vision.cs.tum.edu, Boltzmannstraße 3, 85748 Garching, Germany.

Method	Test1 sequence [%]
Results from MLAD ECCV 2020[9]	
SuperPoint/SuperGlue + PnP + RANSAC	12.4 / 26.5 / 54.4
R2D2 + PnP + RANSAC	12.3 / 23.7 / 42.0
D2-Net + PnP + RANSAC	7.5 / 21.4 / 47.7
SuperPoint + PnP + RANSAC	9.0 / 19.4 / 36.4
HLoc + SuperGlue	67.6 / 92.9 / 98.7
isrf_N8_10k_o2s	13.2 / 26.0 / 47.8
Results from HLoc GitHub[10]	
HLoc + SuperGlue	67.3 / 93.5 / 98.7
Achieved results	
Experiment 1, original HLoc	63.5 / 89.5 / 98.4
Experiment 1, filtered dynamic KPs HLoc	65.5 / 92.5 / 98.5
Experiment 2, original HLoc	67.9 / 94.0 / **98.9**
Experiment 2, filtered dynamic KPs HLoc	**69.8** / **94.6** / 98.8

Table 10.6: All publicly known state-of-the-art results for the 4Seasons test1 sequence were collected. Furthermore, achieved results are presented here. Experiment 2 outperformed all publicly known state-of-the-art results for the two most accurate localization thresholds. The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and for 100% static keypoints with 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The standard localization thresholds are 0.10m / 0.20m / 0.50m.

[9] https://sites.google.com/view/mlad-eccv2020/challenge

[10] https://github.com/cvg/Hierarchical-Localization/tree/master/hloc/pipelines/4Seasons

Experiment 1

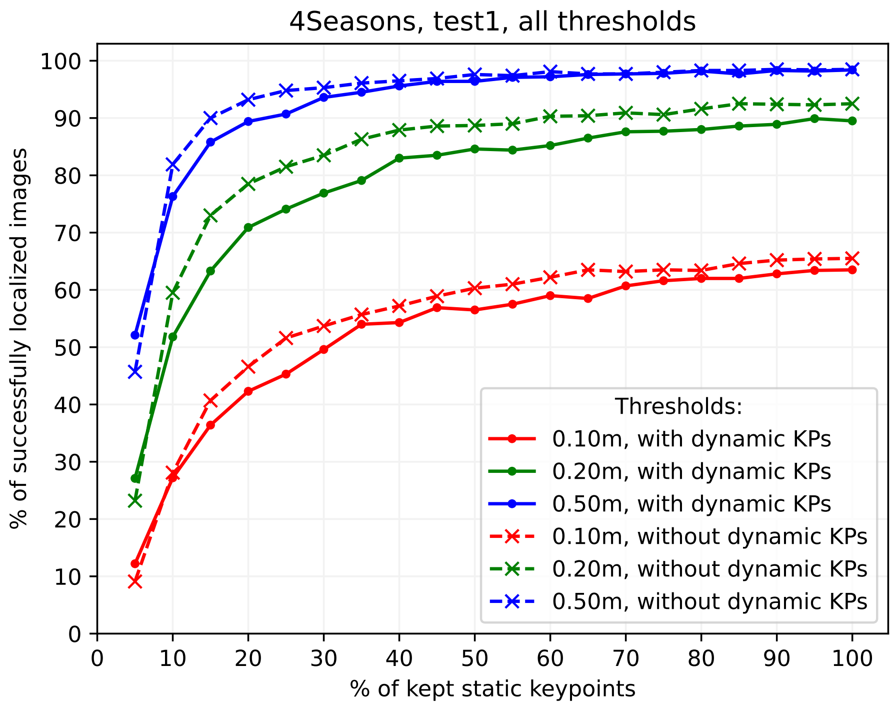

Figure 10.19: 4Seasons, test1 sequence, Experiment 1: the results for all localization thresholds for experiments with and without dynamic keypoints.

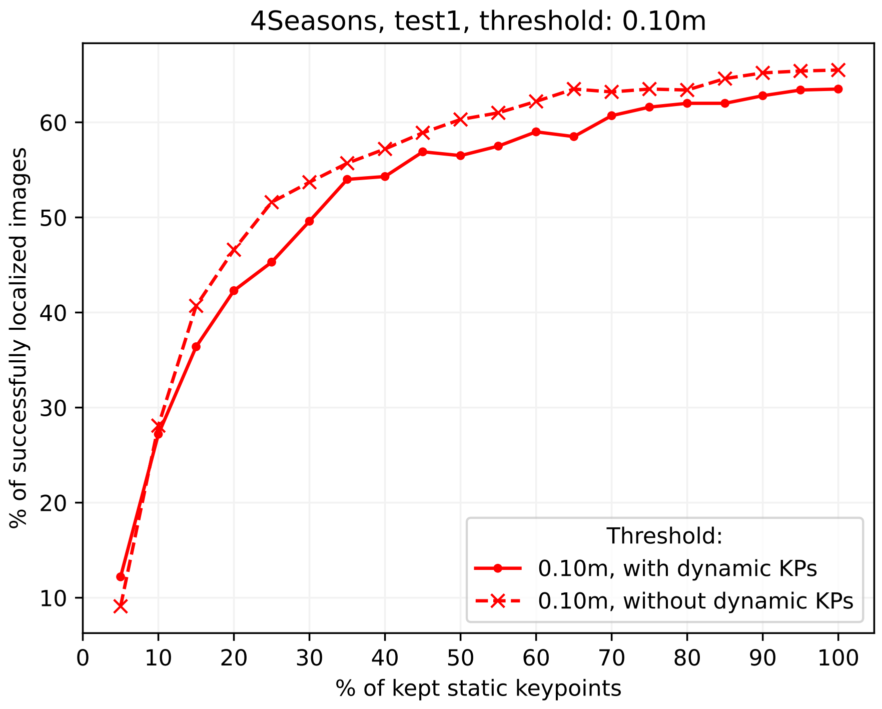

Figure 10.20: 4Seasons, test1 sequence, Experiment 1: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

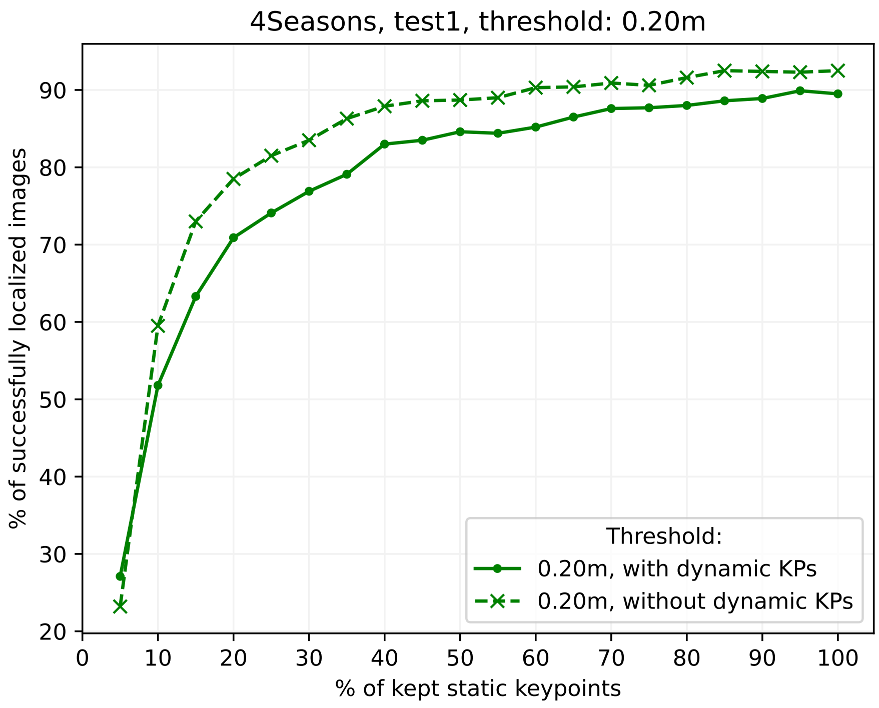

Figure 10.21: 4Seasons, test1 sequence, Experiment 1: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

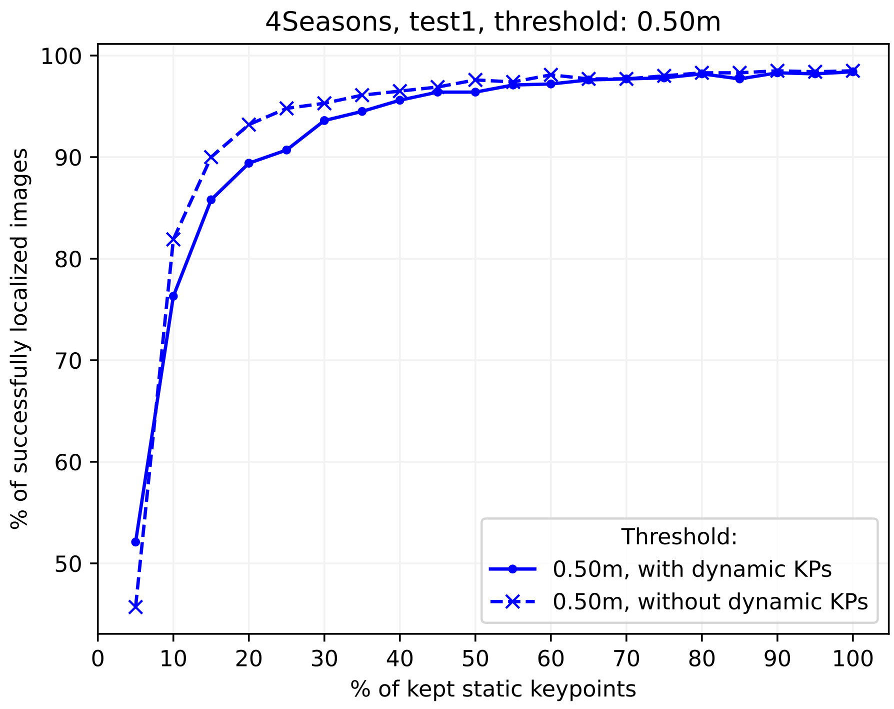

Figure 10.22: 4Seasons, test1 sequence, Experiment 1: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

Experiment 2

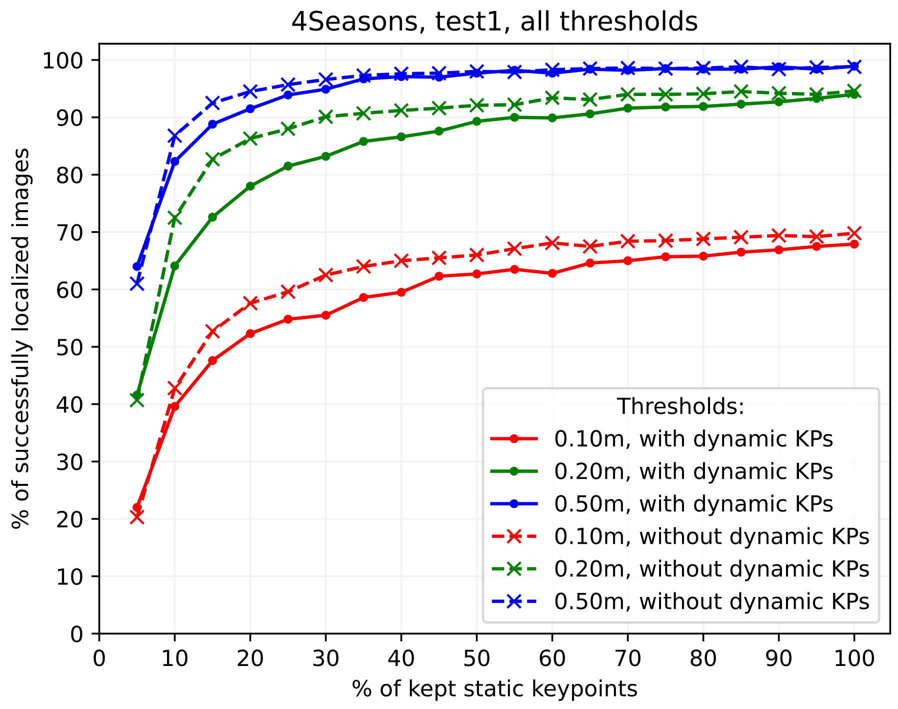

Figure 10.23: 4Seasons, test1 sequence, Experiment 2: the results for all localization thresholds for experiments with and without dynamic keypoints.

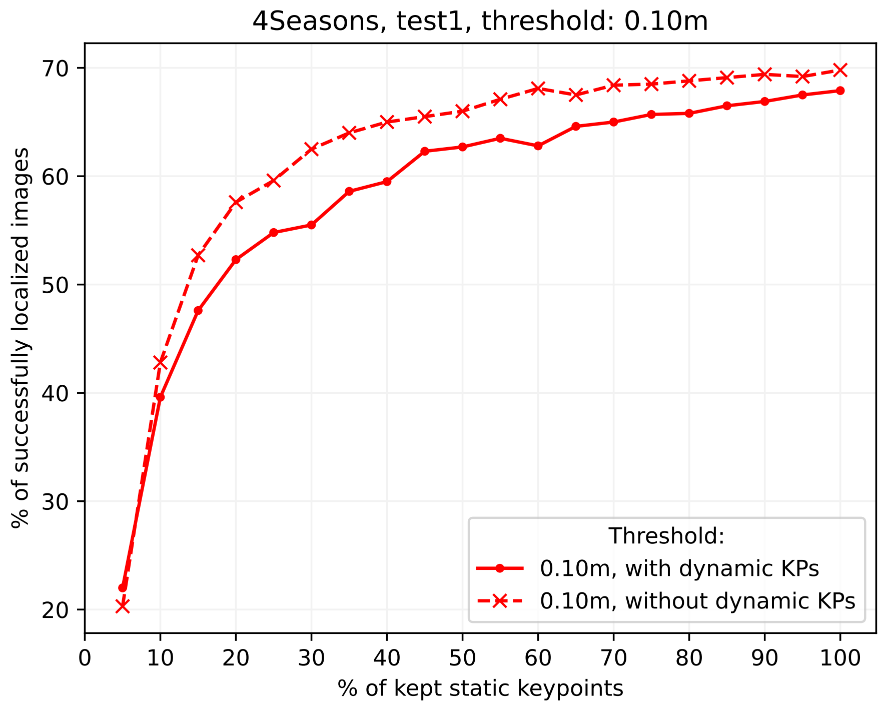

Figure 10.24: 4Seasons, test1 sequence, Experiment 2: the results for 0.10m localization thresholds for experiments with and without dynamic keypoints.

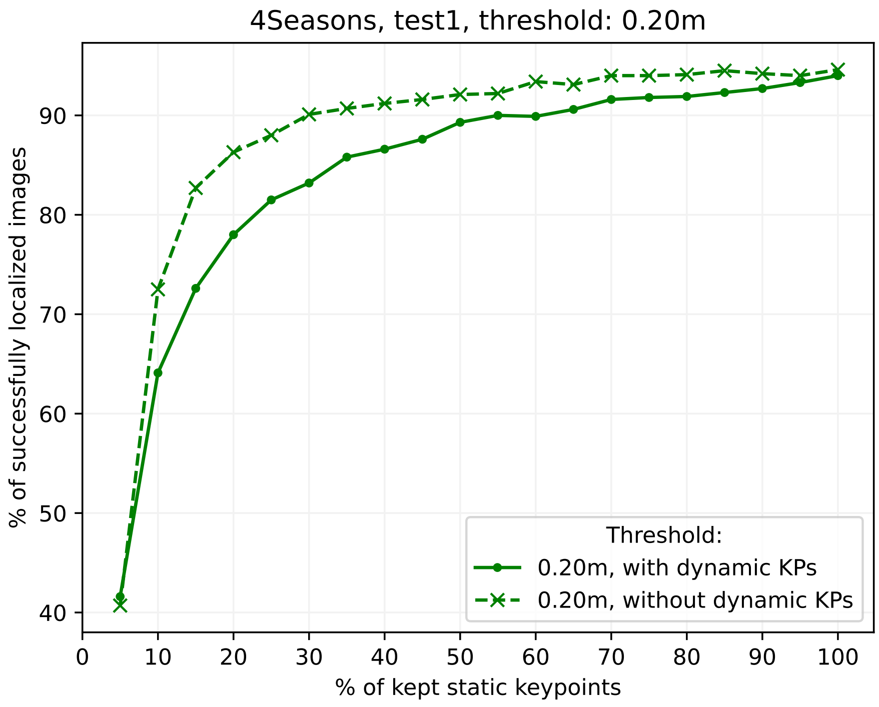

Figure 10.25: 4Seasons, test1 sequence, Experiment 2: the results for 0.20m localization thresholds for experiments with and without dynamic keypoints.

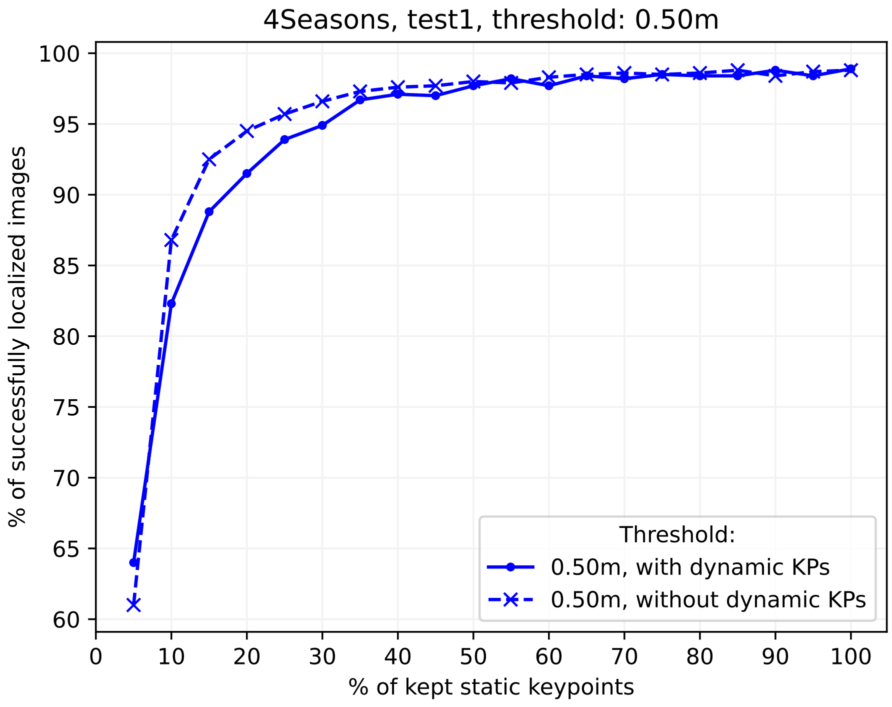

Figure 10.26: 4Seasons, test1 sequence, Experiment 2: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

10.6.5 Aachen v1.1 Day Sequence

The three runs (repetitions) with the same parameters setting of the experiment were performed. Only the run 1 is presented here (run 2 and run 3 can be seen in Appendix C). Only day images were taken into consideration based on Section 10.1. For the Aachen v1.1 query day sequence, the ground truth localizations are not publicly available, and the results were obtained from automatic online verification[11] like in the previous chapters.

The Filtered dynamic KPs HLoc method performs similarly to the Original HLoc method. The results are shown in Table 10.7 and Figure 10.27 (with detailed Figures 10.28, 10.29, and 10.30). The Filtered dynamic KPs HLoc method performs better than the Original HLoc method for run 1 when only results for 100% static keypoints with 100% of dynamic keypoints and for 100% static keypoints with 0% of dynamic keypoints are considered. It can not be stated that the Filtered dynamic KPs HLoc method decreases localization precision.

Method	**Day images [%]**
Original HLoc	86.8 / 94.3 / **98.3**
Filtered dynamic KPs HLoc	**87.0** / **94.5** / **98.3**

Table 10.7: The table shows the comparsion of the Original HLoc method and the Filtered dynamic KPs HLoc method (both with the same parameters setting). The table contains results only for 100% static keypoints with 100% of dynamic keypoints (the Original HLoc method) and 0% of dynamic keypoints (the Filtered dynamic KPs HLoc method). The results are reported for standard precision thresholds 0.25m 2° / 0.50m 5° / 5.00m 10°.

[11] https://www.visuallocalization.net

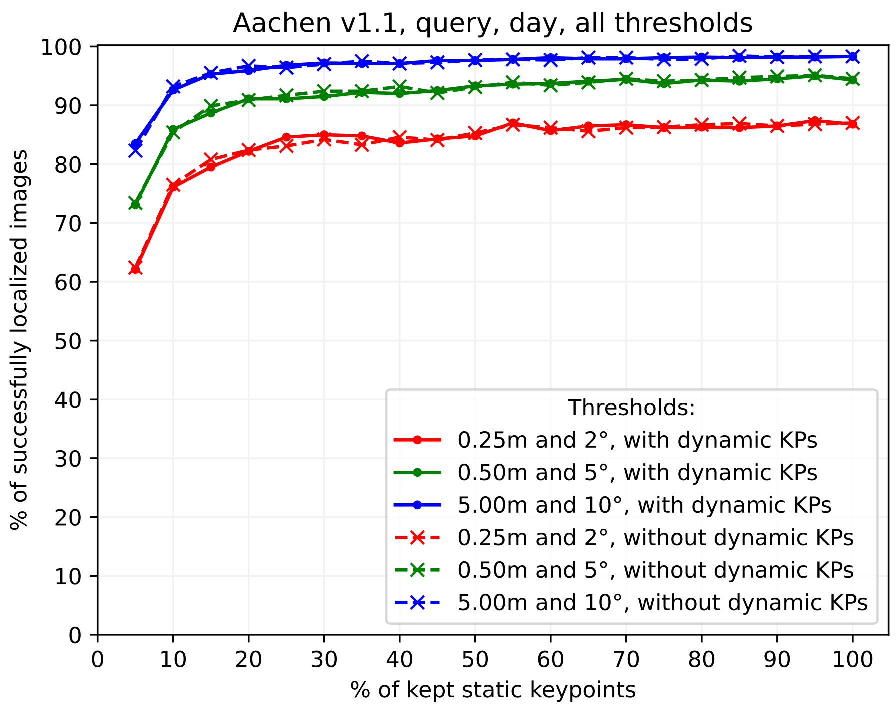

Figure 10.27: Aachen v1.1, day query sequence, run 1: the results for all localization thresholds for experiments with and without dynamic keypoints.

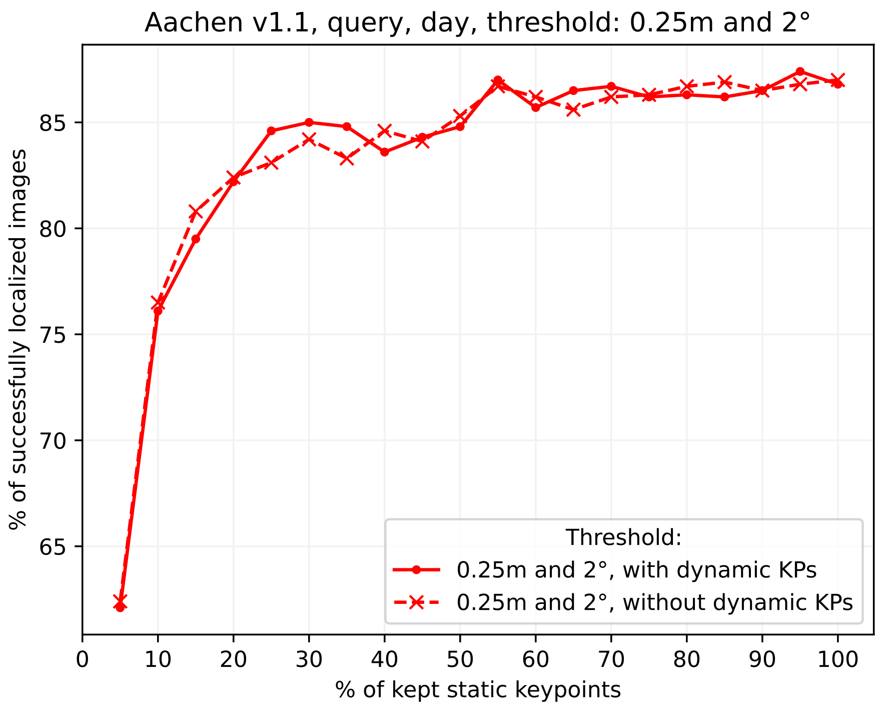

Figure 10.28: Aachen v1.1, day query sequence, run 1: the results for 0.25m and 2° localization threshold for experiments with and without dynamic keypoints.

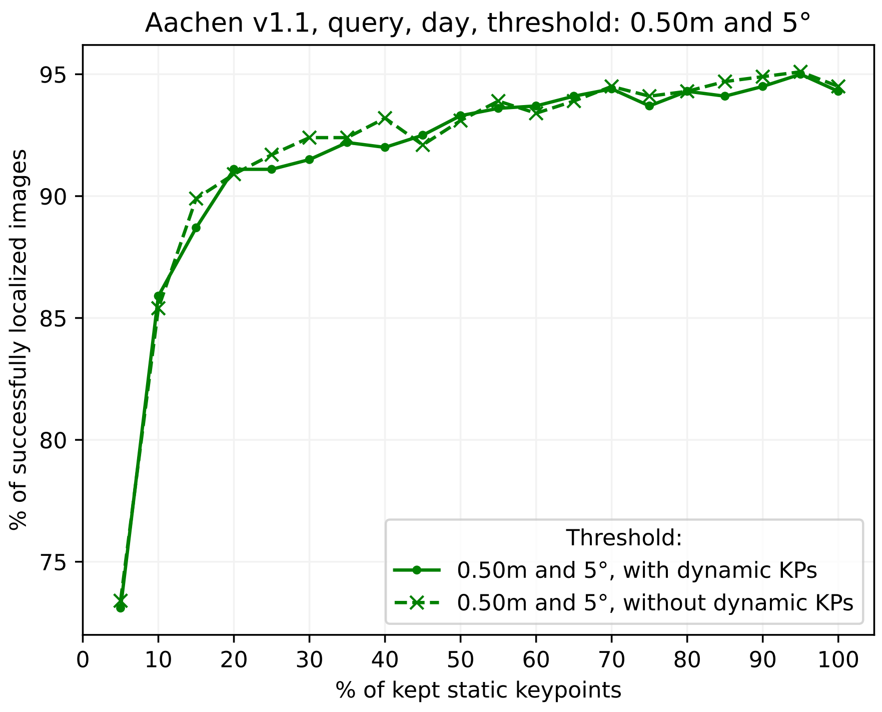

Figure 10.29: Aachen v1.1, day query sequence, run 1: the results for 0.50m and 5° localization threshold for experiments with and without dynamic keypoints.

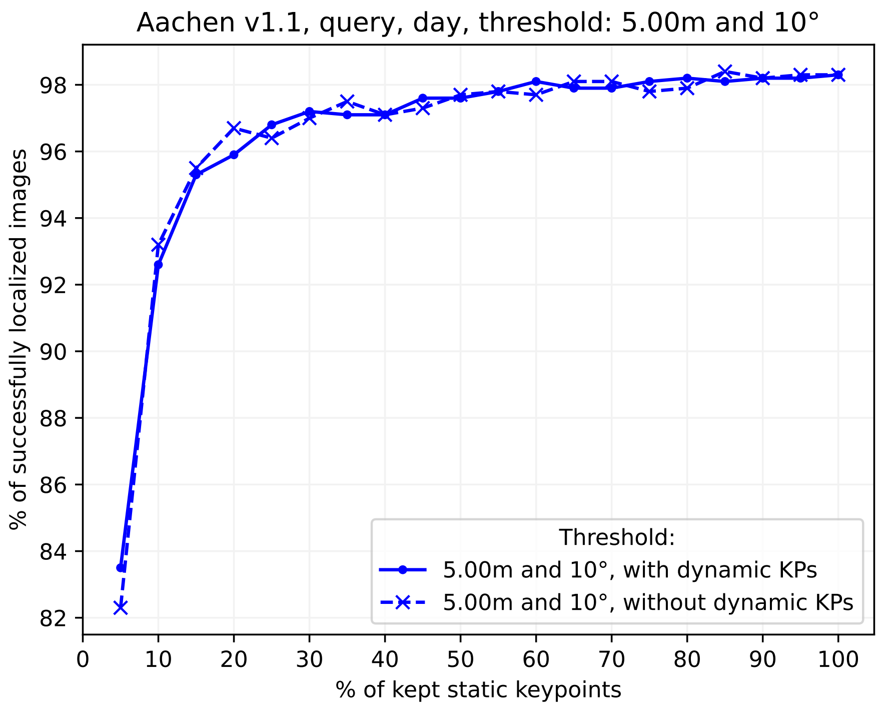

Figure 10.30: Aachen v1.1, day query sequence, run 1: the results for 5.00m and 10° localization threshold for experiments with and without dynamic keypoints.

10.7 Examining Statistical Significance of Results

4Seasons

The improvement over the current state-of-the-art reported results is statistically significant (p-value = 0.016). Due to the non-normal distribution of data, the one-sided Wilcoxon signed-rank test was chosen to test the null hypothesis H_0 that the novel approach Remove Dynamic Keypoints (RDKPs) (the Experiment 2, filtered dynamic KPs HLoc method - see Table 10.8) based on semantic information is not better than the current state-of-the-art method Keep Dynamic Keypoints (KDKPs)[12] (the HLoc + SuperGlue method - see Table 10.8). The test data has a sample size of 6 (see Table 10.8) and is constructed as pairs of RDKPs and KDKPs localization accuracy percentages for all three accuracy thresholds 0.10m, 0.20m, and 0.50m, for both tests test0 and test1.

The KDKPs[13] setup yields significantly better result (p-value = 0.031) than the setup reported on official GitHub repository[14] (the HLoc + SuperGlue method - see Table 10.8), the KDKPs setup was also compared to the RDKPs approach (using a one-sided Wilcoxon signed-rank) and gained the same result, i.e. RDKPs works statistically significantly better than best KDKPs setup with p-value of 0.031.

4Seasons	**Test0 sequence [%]**	**Test1 sequence [%]**
HLoc + SuperGlue (KDKPs)	91.8 / 97.7 / 99.2	67.3 / 93.5 / 98.7
Experiment 2, original HLoc (KDKPs)	92.5 / 97.8 / 99.2	67.9 / 94.0 / **98.9**
Experiment 2, filtered dynamic KPs HLoc (RDKPs)	**93.8** / **98.7** / **99.8**	**69.8** / **94.6** / 98.8

Table 10.8: The considered results for examining statistical significance. The standard localization accuracy thresholds are 0.10m / 0.20m / 0.50m.

[12] KDKPs means that no keypoints are filtered out: 100% of static and 100% dynamic keypoints are taken into account. RDKPs means that all dynamic keypoints are filtered out: 100% of static and 0% dynamic keypoints are considered, in Tables 10.5 and 10.6 named as "Experiment 2, filtered dynamic KPs HLoc".

[13] In Tables 10.5, 10.6, and 10.8 named as "Experiment 2, original HLoc".

[14] https://github.com/cvg/Hierarchical-Localization/tree/master/hloc/pipelines/4Seasons

Aachen v1.1

The localization accuracy percentage values obtained by RDKPs and KDKPs are not statistically significantly different. The performed Friedman test for 3 repetitions and 3 condition thresholds with the H_0 that RDKPs did not differ from KDKPs gave the p-value of 0.68. The test data has the sample size of 9 and was constructed as pairs of RDKPs and KDKPs localization accuracy percentages for three localization thresholds 0.25m 2°, 0.50m 5°, 5.00m 10° and for 3 repetitions (run 1, run 2, and run 3).

Aachen v1.1	**Run 1 [%]**	**Run 2 [%]**	**Run 3 [%]**
KDKPs[15]	86.8 / 94.3 / **98.3**	**87.3** / **94.5** / 98.3	**88.0** / 94.7 / 98.3
RDKPs[16]	**87.0** / **94.5** / **98.3**	86.4 / **94.5** / **98.4**	86.8 / **94.9** / **98.5**

Table 10.9: The considered results for examining statistical significance. The standard localization thresholds are 0.25m 2° / 0.50m 5° / 5.00m 10°.

[15]Original HLoc run 1 results were taken from Table 10.7 and other runs from Appendix C.

[16]Filtered dynamic KPs HLoc run 1 results were taken from Table 10.7 and other runs from Appendix C.

10.8 Discussion

Chapter 10 presented novel state-of-the-art results on selected datasets that show the importance of an additional interpretation of underlying information for detected keypoints. More specifically, the semantic information about the static and dynamic objects helps to improve long-term visual localization accuracy.

The newly achieved state-of-the-art results surpassed all previously published results at almost all percentage levels of kept static keypoints for the 4Seasons test0 and test1 sequences. The results performed better in 10-30% kept static keypoints, which increased the number of correctly localized camera poses. The overall long-term visual localization pipeline is more noise resistant, where the noise is presented in the form of dynamic keypoints.

Regarding the statistical significance analysis in Section 10.7 it can be stated that newly achieved results for the 4Seasons test0 and test1 sequences are statistically significant with p-value $= 0.016$ over the current state-of-the-art reported results. The achieved results for the Aachen v1.1 query day sequence are not statistically significantly different from the current state-of-the-art reported results (the p-value $= 0.68$ of H_0 that their medians are different).

In Table 10.10, there are features for each selected and analyzed image query. From those features, the main differences between the 4Seasons and Aachen v1.1 sequences are: Aachen 1.1 query day sequence contains fewer dynamic keypoints, and the keypoints are mainly located on the human semantic group of classes. The 4Seasons sequences contain up to approximately six times more images and approximately four times more dynamic keypoints located mainly on the vehicle semantic group of classes. Finally, maximum KPs count utilization[17] shows that 4Seasons sequences are less visually rich because the maximum KPs count utilization is lower (up to 27.08 % lower when the 4Seasons test1 and Aachen v1.1 query day sequences are compared - see Table 10.10). Another major difference is that Aachen v1.1 is evaluated not just for camera position accuracy but also for camera rotation accuracy.

[17]Maximum KPs count utilization is defined as follows: (Count of KPs (total) / Count of images * 1024) * 100. Where 1024 is the maximum number of keypoints per image. It describes the utilization of the total allowed keypoints. It can be interpreted as how hard the specific image sequence was for SuperPoint to detect the maximum allowed number of keypoints.

Feature	**Aachen v1.1 query day**	**4Seasons training**	**4Seasons validation**	**4Seasons test0**	**4Seasons test1**
Count of images	824	3,156	1,994	**5,024**	3,622
Count of KPs (total)	841,845	2,845,442	1,799,317	**4,492,232**	2,696,178
Count of KPs (static)	791,860	2,560,160	1,398,449	**4,054,606**	2,436,298
Count of KPs (dynamic)	49,985	285,282	400,868	**437,626**	259,880
Average count of KPs / image (static)	**961**	811	701	807	673
Average count of KPs / image (dynamic)	61	90	**201**	87	72
Maximum KPs count utilization	**99.77%**	88.05%	88.12%	87.32%	72.69%
% of KPs (static)	**94.06%**	89.97%	77.72%	90.26%	90.36%
% of KPs (dynamic)	5,94%	10.03%	**22.28%**	9.74%	9.64%
Major dynamic group of classes	**human**	vehicle	vehicle	vehicle	vehicle
Environment	urban	urban	urban	urban	urban
Threshold types	**position & rotation**	position	position	position	position

Table 10.10: The feature comparison of all used image sequences.

Part IV

Conclusion

Chapter 11

Conclusion

The thesis summary is introduced in this last chapter, followed by the thesis evaluation and discussion. The future work is outlined, and a few possible improvements are briefly mentioned at the end.

11.1 Thesis Summary

At first, in Methodology Part II, the rigorous discovery of historical keypoint detectors and descriptors was carried out. Then, related methods and state-of-the-art methods were explained, and suitable datasets for the task of long-term visual localization were introduced.

Part III consists of the contribution to the state-of-the-art. Here the thesis presents new state-of-the-art results in long-term visual localization on selected benchmarking Aachen Day-Night dataset. The newly achieved results were obtained by carefully fine-tuning parameters of HLoc pipeline. The achieved results outperform all other methods and approaches.

Part III introduces semantic segmentation for masking Aachen day images by grouping segmentation classes into four dynamic groups of classes. The experiments show that it can achieve comparable good results to the top state-of-the-art pipeline by removing the image's unnecessary area. Thus, it supports the idea that dynamic objects generate only unmatchable keypoints and, therefore, can be removed.

Part III also contains the new approach of dynamic keypoints filtering based on semantic segmentation information. This novel approach outperforms the current state-of-the-art results for the 4Seasons dataset (all sequences) and matches the state-of-the-art results for the Aachen v1.1 dataset (day images).

The goals, results, and contributions of this work are further discussed in the following section.

11.2 Fulfillment of Dissertation Goals

This section is dedicated to the evaluation of the goals from Chapter 3. The definition of the goals is repeated and followed by the evaluations and discussions of achieved results.

11.2.1 Selecting Dataset for Evaluation

The problem of choosing a suitable dataset that can be utilized to evaluate long-term visual localization is still recent. Many different datasets were released and used for competitions.

This goal aims to find a dataset and evaluation method that can be used for method comparison. The output should describe datasets, select one dataset for further evaluation, and an evaluation method or approach. Creating a new dataset or evaluation method is not part of this work.

The datasets that can be used for long-term visual localization and many other Computer Vision tasks were described in Chapter 5. The Aachen Day-Night dataset was selected for the performed experiments. The selection of Aachen Day-Night dataset was discussed in Section 6.2.1. The benchmark choice, the evaluation method, and the ranking method were explained in Section 6.2. Thus, this goal is **completed**.

11.2.2 Enhanced Parameter Setting

After selecting a benchmark dataset and evaluation method from the previous step, the current state-of-the-art methods were used as the baseline. By carefully fine-tuning parameters, better results for the selected benchmark dataset were achieved.

New and improved state-of-the-art results on the selected dataset were achieved. Carefully fine-tuning of selected methods' parameters improved long-term visual localization accuracy. Developing a new keypoint detector, descriptor, matcher, and long-term visual localization pipeline is not part of this work.

The Aachen Day-Night dataset was selected based on the previous goal. HLoc described in Section 6.3 was selected as the state-of-the-art pipeline. In Chapter 7, there were discussed previously achieved results on the Aachen Day-Night dataset with HLoc pipeline and performance of other state-of-the-art methods and approaches. This chapter also describes the importance of replicability of the results. In this work, the results of the Original HLoc were recreated, and even the parameters for HLoc pipeline were optimized for the selected dataset. The Tuned HLoc pipeline with tuned parameters yielded new state-of-the-art results on the Aachen Day-Night dataset. It outperforms the Original HLoc by 0.1% in the most accurate localization threshold and by 0.4% in the second most accurate localization threshold for Aachen Day images. For Aachen Night images, the Tuned HLoc setting outperformed the Original HLoc by 1.1% in the most precise localization threshold and by 1.0% in the second most precise localization threshold. Detailed results are given in Section 7.4.3. Based on newly achieved state-of-the-art results on the selected Aachen Day-Night dataset, this goal can be considered to be **fulfilled**.

11.2.3 Preprocessing for Keypoint Masking

Recently, the additional information was used, and it is not presented by default in a dataset (e.g., semantic segmentation, instance segmentation, estimated depth) for improving the localization results. It shows that image preprocessing is still a proper way to contribute to the state-of-the-art.

This last goal aims to show that some parts of the image are unnecessary for long-term visual localization, and they do not bring any valuable information and can be masked or filtered out. After removing unnecessary image parts, the overall visual localization pipeline should perform comparably to the original pipeline without modification. Developing a new semantic segmentation, instance segmentation, and depth estimation method is not part of this work.

The experiments performed in Chapter 8 use HRNet-OCR method for semantic segmentation. The HRNet-OCR is one of the top-performing state-of-the-art semantic segmentation methods. It performs best on the Cityscape dataset. The Cityscape dataset is similar to the selected Aachen Day-Night dataset (both were recorded urban environments). The experiments used only Aachen Day images because semantic segmentation methods perform poorly for Aachen Night images and night images in general. The HRNet-OCR method returns semantic segmentation of Aachen Day images in the color layout of the Cityscape dataset. There are multiple unnecessary classes in the Cityscape dataset, and thus the aggregation of segmented classes was introduced. The segmented classes were aggregated into four groups of classes. More specifically, it was into the sky, vehicle, human, and nature group of classes (described in Section 8.4). The Original HLoc pipeline was used with masked Aachen Day images, and the experiments for all combinations of the group of classes were performed. The results from the Section 8.5 show that the Masked h (human group of classes = person and driver) HLoc performs similar to the Original HLoc and even it outperforms it in the second-best localization threshold by 0.1%. In general, it can be stated that an image area does not bring any valuable information for long-term visual localization tasks because it is a dynamic object and generates only unmatchable keypoints. This goal can be considered **completed**.

11.2.4 Fusion of Training Information

The training data characteristics are essential for the current state-of-the-art methods. A keypoints detection method can perform very well and detect general keypoints without knowing any other underlying information (e.g., semantic information of underlying object). This underlying information can help understand the importance of a particular keypoint and may help to rank/filter/suppress specific keypoints for a given task.

The goal here was to try an approach for adding underlying semantic information about the underlying object to an existing state-of-the-art keypoint detector model. As was stated in Section 3.4 a trial of fusion of new information into an existing keypoint detection model would be unsuccessful without modifying the model architecture. However, a modification of the architecture of a method is not in the scope of this goal.

Two approaches of modification of training data for SuperPoint state-of-the-art keypoint detector and descriptor method were tested in Chapter 9. The first approach of incorporating the semantic information on the underlying static/dynamic objects was using the selected HRNet-OCR semantic segmentation method to remove human and vehicle groups of dynamic classes. The removal, in this case, means masking out / blacking the image (setting the underlying pixels to black color). The second approach was eliminating detected dynamic keypoints of the MagicPoint method and not using them in the training procedure of SuperPoint. The pipeline of training MagicPoint and SuperPoint was replicated, and the SuperPoint keypoint detector and descriptor was successfully trained. However, unfortunately, none of the two mentioned methods was successful. This concern was already outlined in the goal section itself. It is caused mainly because the two selected groups of dynamic classes are not very well included in the MS-COCO training dataset. The SuperPoint's ANN architecture was not created to be able to handle additional information about the underlying object. The SuperPoint was built to provide robust and general detection and description ability, and in this, it performs very well. Two slightly different approaches of fusing semantic segmentation information into the training process of selected state-of-the-art SuperPoint method were examined, and therefore this goal can be marked as **completed**.

11.2.5 Keypoint Filtering

The Computer Vision problem of long-term visual localization is still relevant and very popular in the scientific community. Many current approaches and methods try to solve a part of the problem or the whole pipeline of long-term visual localization. One significant part is to detect robust keypoints for matching images. The problem of general keypoint detectors is that they detect all keypoints regardless of how beneficial the keypoints are. Some keypoints do not add any meaningful information for solving the task of long-term visual localization. That means that they will never match, or if they match, it can be the wrong match. They represent an unwanted noise added to the localization pipeline. This challenging problem needs to be addressed.

The last goal aims to find an approach to removing keypoints that are not valuable and helpful for visual localization. The detected keypoints on dynamic objects labeled by the semantic segmentation were selected. The goal was to introduce a new approach and achieve new state-of-the-art results in long-term visual localization for the selected dataset.

Chapter 10 presented the approach for filtering keypoints in which keypoints are marked as dynamic removed. This approach achieved the new state-of-the-art results on the selected 4Seasons dataset. The benchmark sequence test0 performed 93.8% / 98.7% / 99.8% and sequence test1 achieved 69.8% / 94.6% / 98.8% for standard accuracy thresholds 0.10m / 0.20m / 0.50m. The percentages represent how many images were successfully localized within a threshold from the dataset. On 4Seasons training and validation sequences the novel approach also outperformed current methods with 89.4% / 98.3% / 100.0% and 92.4% / 99.6% / 100.0% results for the same accuracy thresholds. The 4Seasons ground truth localizations are publicly available for local validation for training and testing sequences. Nevertheless, for 4Seasons test0 and test1 sequences, ground truth localizations are not publicly known. The 4Seasons' creators were contacted, and testing sequences were validated by one of the 4Seasons' creators. From this point of view, it can be stated that the results are unbiased. The achieved results perform better than the HLoc baseline and the current state-of-the-art for most of kept static keypoints levels for all localization accuracy thresholds. Regarding the statistical significance analysis in Section 10.7, it can be stated that the newly achieved results for 4Seasons test0 and test1 sequences give statistically significantly better results with p-value $= 0.016$ over the current state-of-the-art reported results. The Aachen v1.1 query day sequence results are not statistically significantly different from the current state-of-the-art reported results of $H_0 = 0.68$. This last goal can be marked as **successfully achieved and completed** due to the original research and newly reached state-of-the-art results.

11.3 Future Work

As it was described in this thesis, there are many open problems in the domain of keypoint detecting, describing, and long-term visual localization. Many applications used in the real world will appear soon. These applications will aim to make the world better, and they will need to have the best state-of-the-art methods that will have to work in a real-world environment. In future work, multiple areas where an improvement could be made can be explored. In this section, a few ideas are briefly mentioned.

For more robust analysis of achieved results, it is needed to perform tests on other long-term visual localization benchmark datasets such as RobotCar Seasons, CMU Seasons, and others mentioned in Section 5.

The semantic segmentation approach can be used not only for masking purposes but also for filtration purposes [36]. Then, semantic segmentation implementation can obtain a more robust solution. A weighting or voting could be used for getting smoother edges of objects.

Next idea for results improvement can be to use depth consistency checking and filtering [36]. The Monocular approach can handle this task [69].

The used methods (SuperPoint, SuperGlue, and NetVLAD) can be improved by modifying neural network structure or by retraining on different training datasets.

Using noise reduction methods could improve the appearance of the Aachen Night images and overall results.

Some of Aachen Day's images are blurry, and using a sharpening method could improve them.

Experiments with a transformation of Aachen Night images to a day-like appearance [131, 132, 133] could help current pipelines to achieve better localization results.

Extending the SuperPoint model by the ability to understand the semantic segmentation under detected keypoints will reduce the number of detected dynamic keypoints, which therefore can again improve the overall localization precision.

The number of detected keypoints in the 4Seasons test1 is lower than the allowed threshold of 1024 keypoints for the performed experiments. Furthermore, thus adjusting a keypoint detecting method for generating more robust keypoints may lead to more matches and more precise localization of some images from the sequence.

Appendix A

4Seasons Training Sequence

In this appendix are the 2nd and 3rd runs of presented pipeline from Chapter 10 and their respective results and figures.

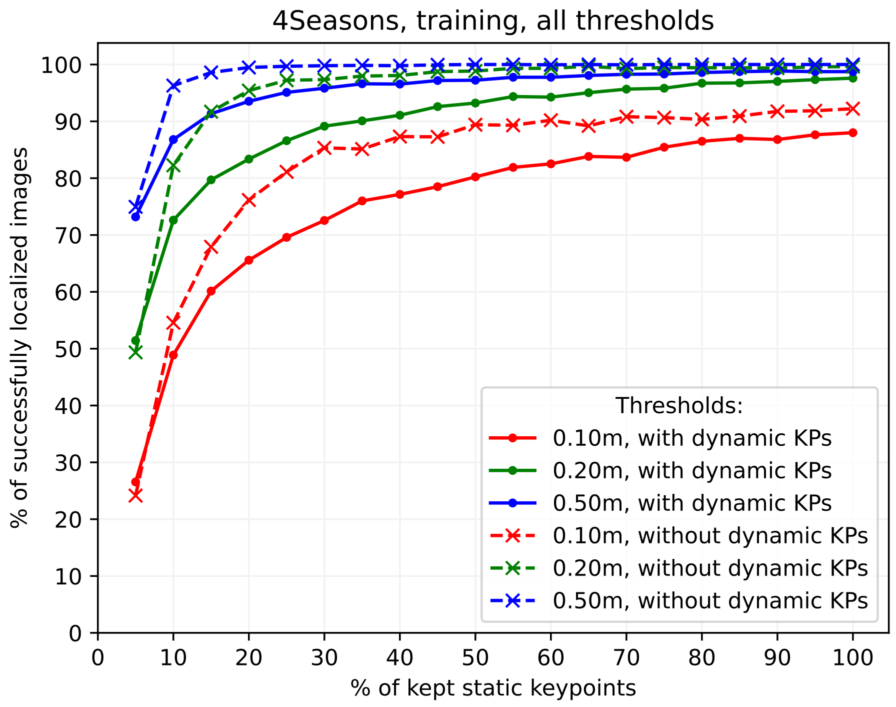

Figure A.1: 4Seasons, training sequence, run 2: the results for all localization thresholds for experiments with and without dynamic keypoints.

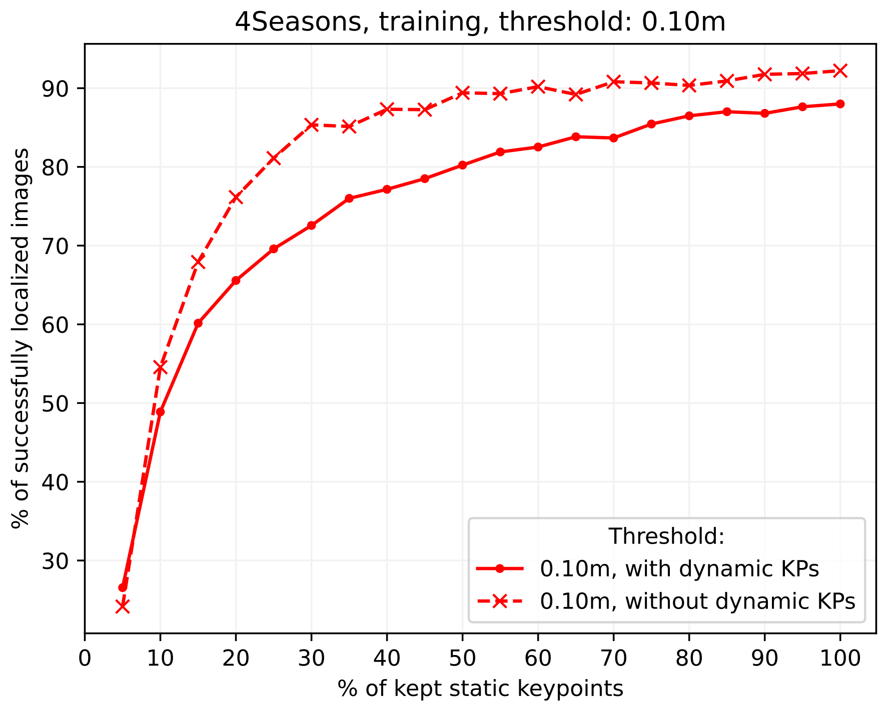

Figure A.2: 4Seasons, training sequence, run 2: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

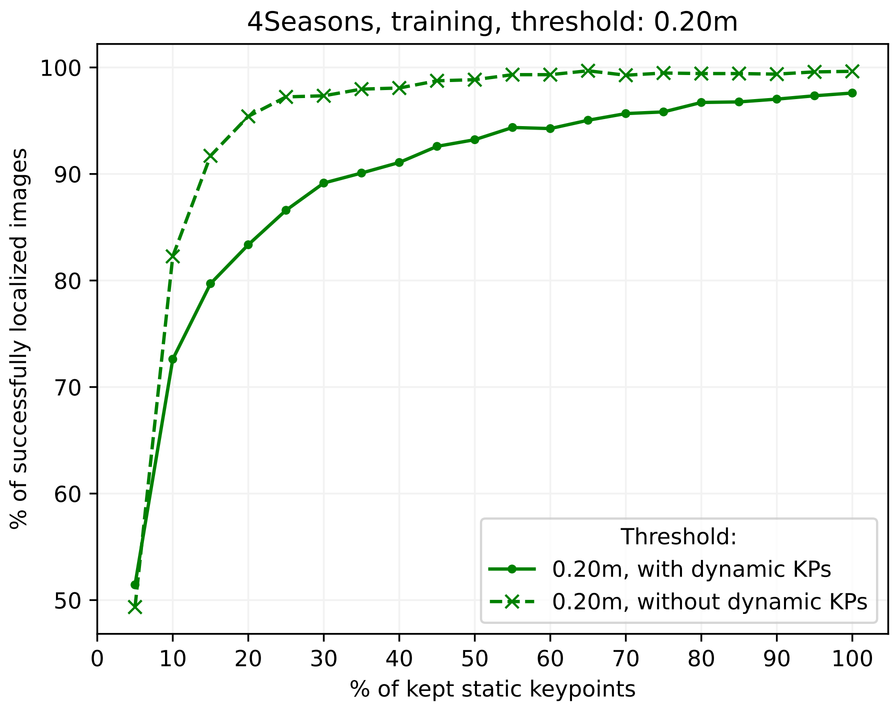

Figure A.3: 4Seasons, training sequence, run 2: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

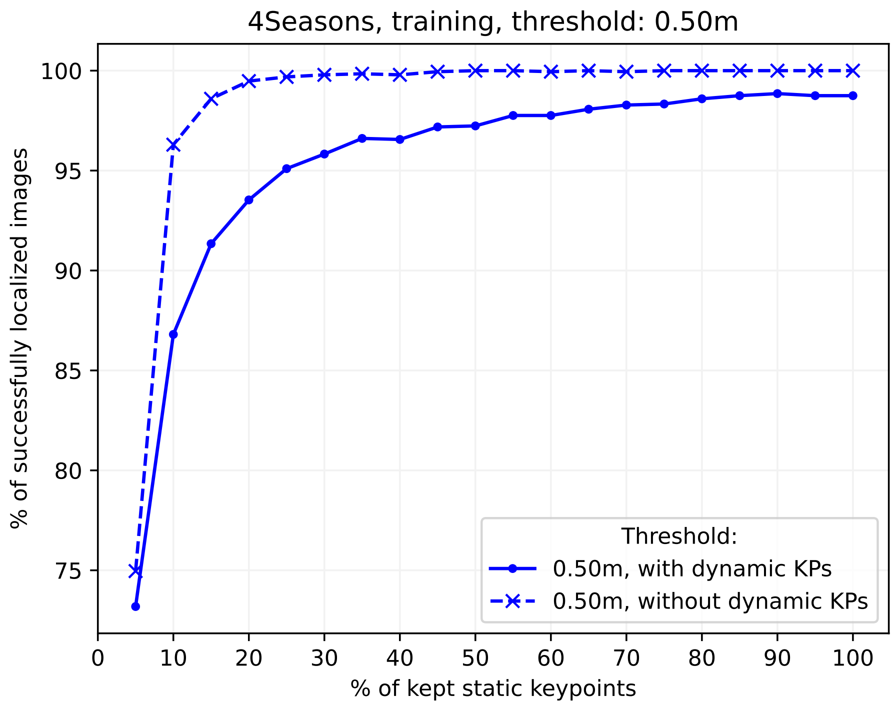

Figure A.4: 4Seasons, training sequence, run 2: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

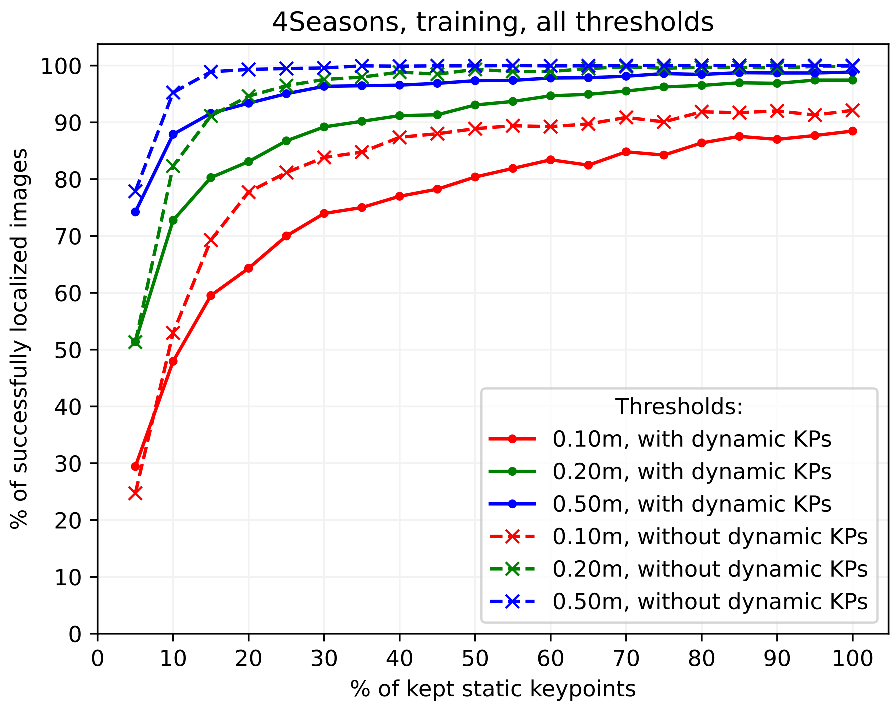

Figure A.5: 4Seasons, training sequence, run 3: the results for all localization thresholds for experiments with and without dynamic keypoints.

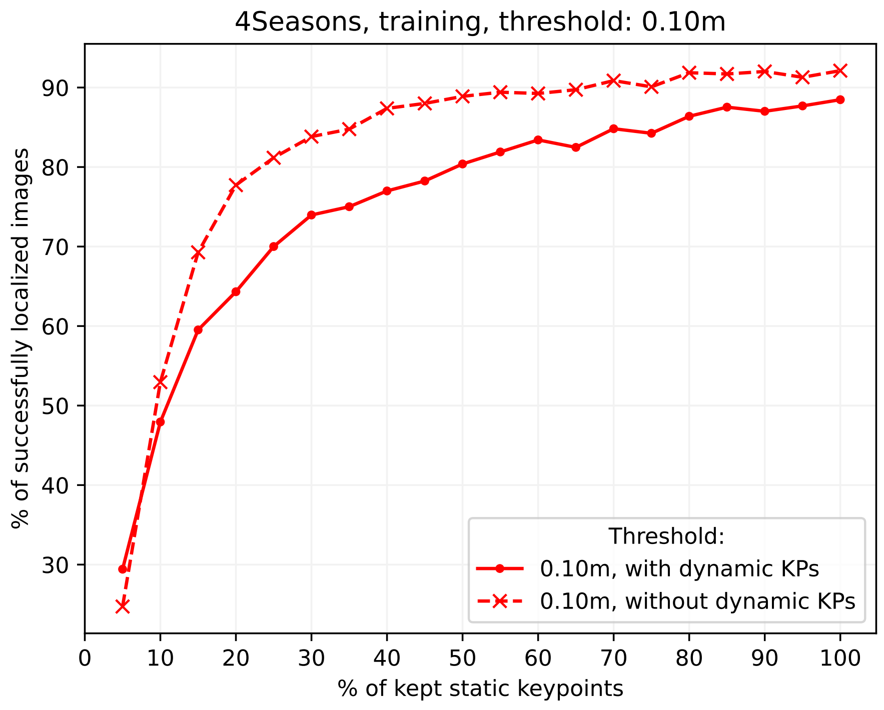

Figure A.6: 4Seasons, training sequence, run 3: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

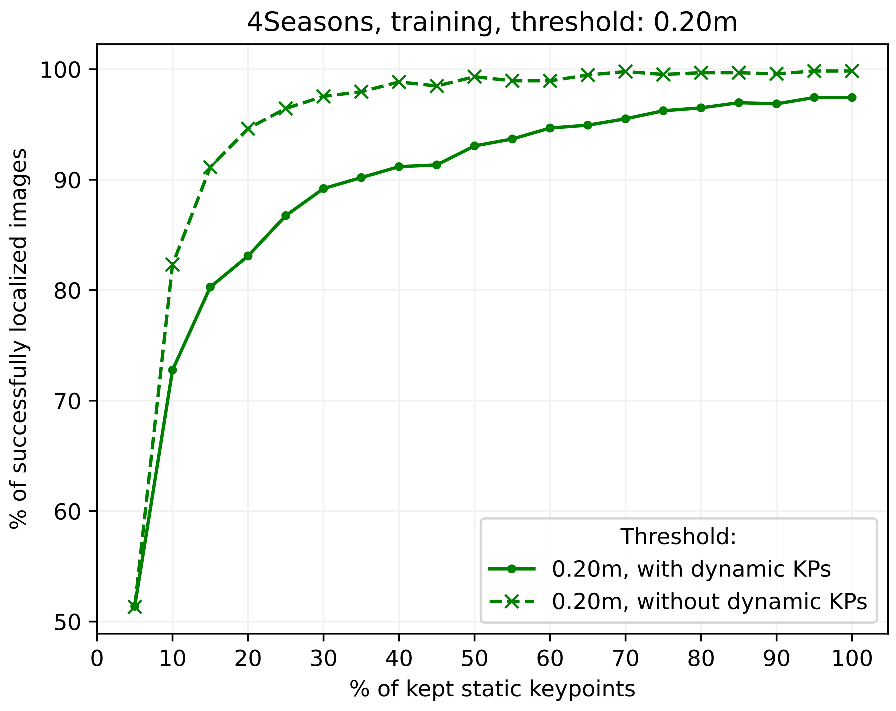

Figure A.7: 4Seasons, training sequence, run 3: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

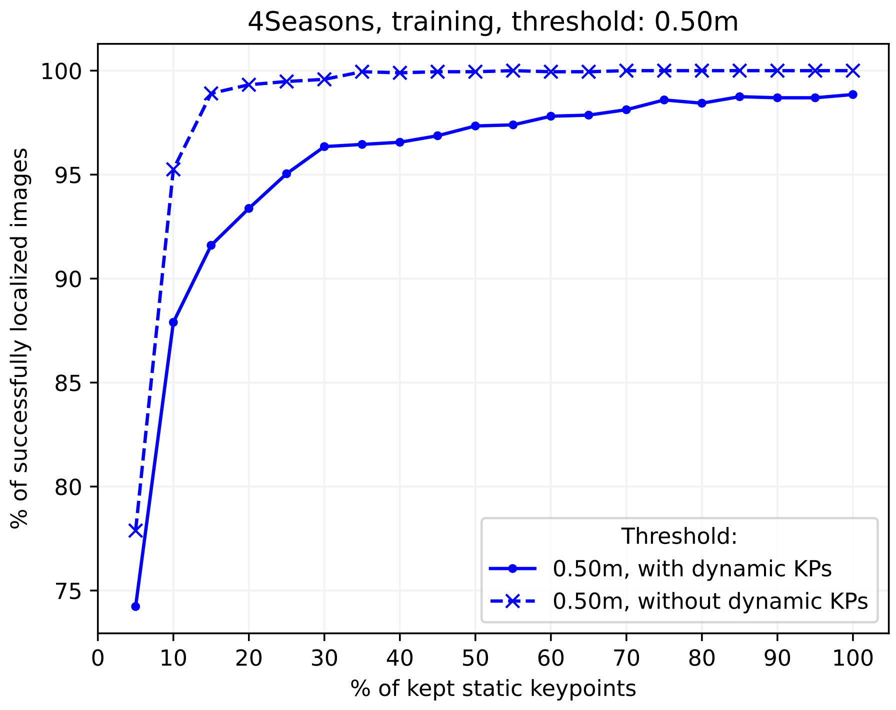

Figure A.8: 4Seasons, training sequence, run 3: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

Appendix B

4Seasons Validation Sequence

In this appendix are the 2nd and 3rd runs of presented pipeline from Chapter 10 and their respective results and figures.

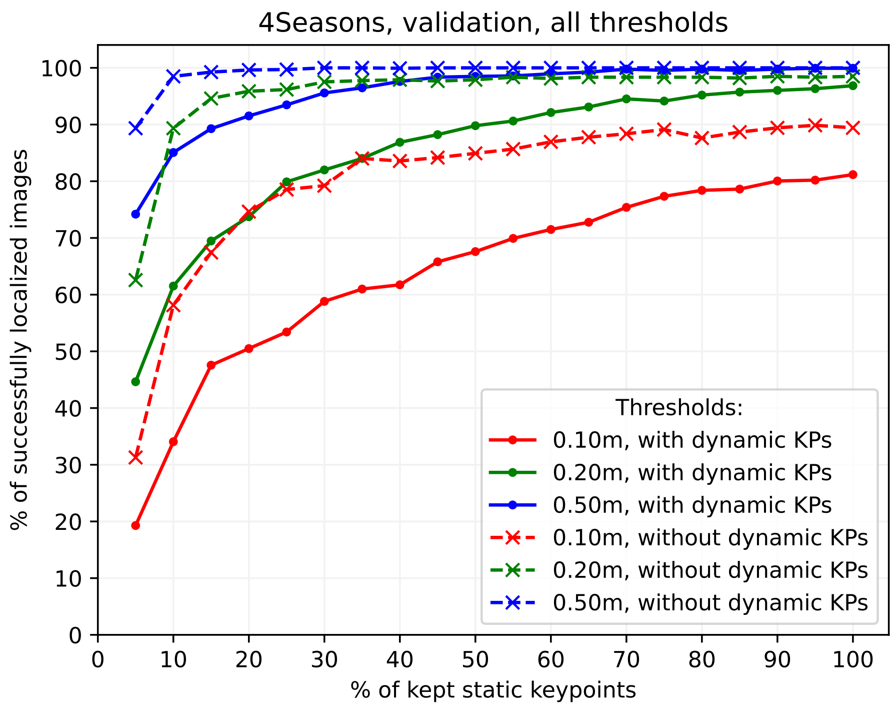

Figure B.1: 4Seasons, validation sequence, run 2: the results for all localization thresholds for experiments with and without dynamic keypoints.

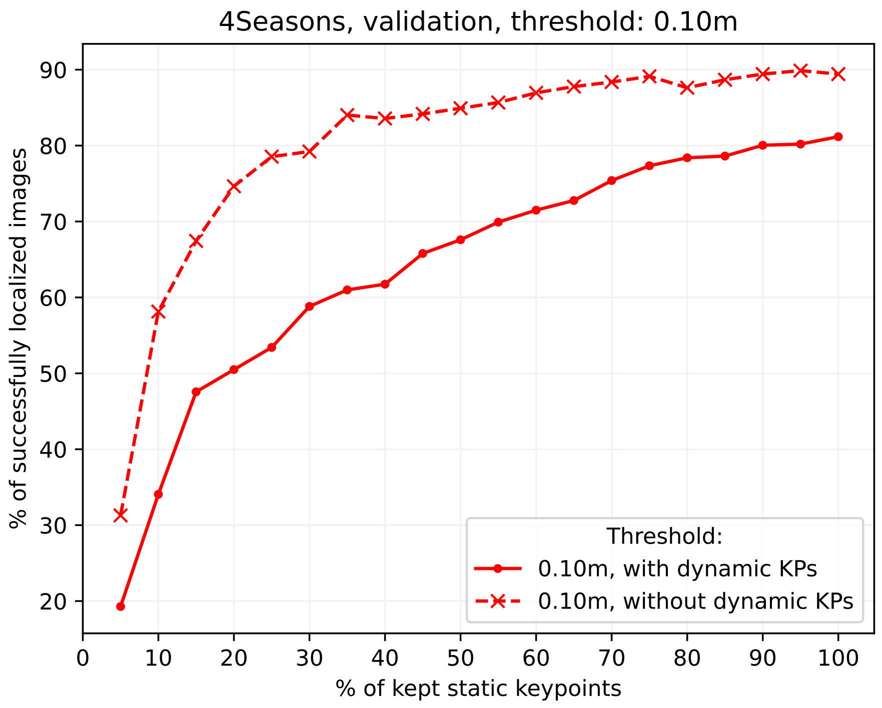

Figure B.2: 4Seasons, validation sequence, run 2: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

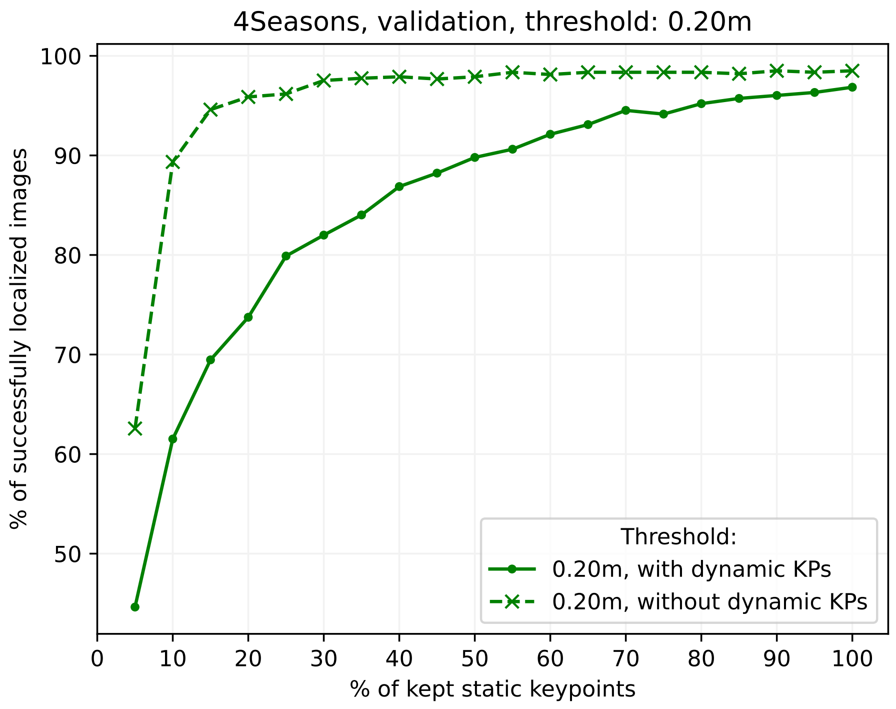

Figure B.3: 4Seasons, validation sequence, run 2: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

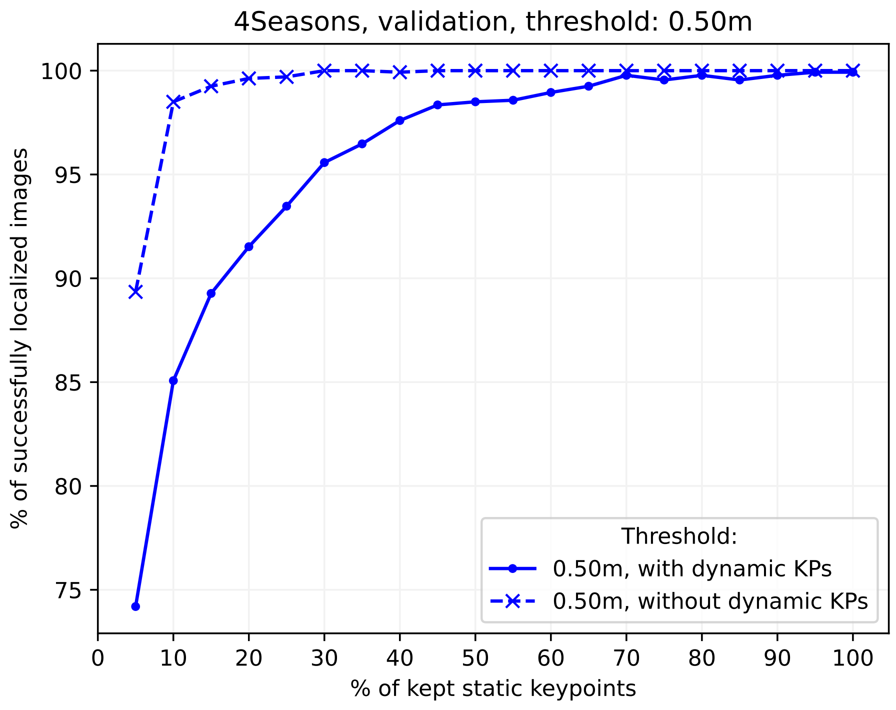

Figure B.4: 4Seasons, validation sequence, run 2: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

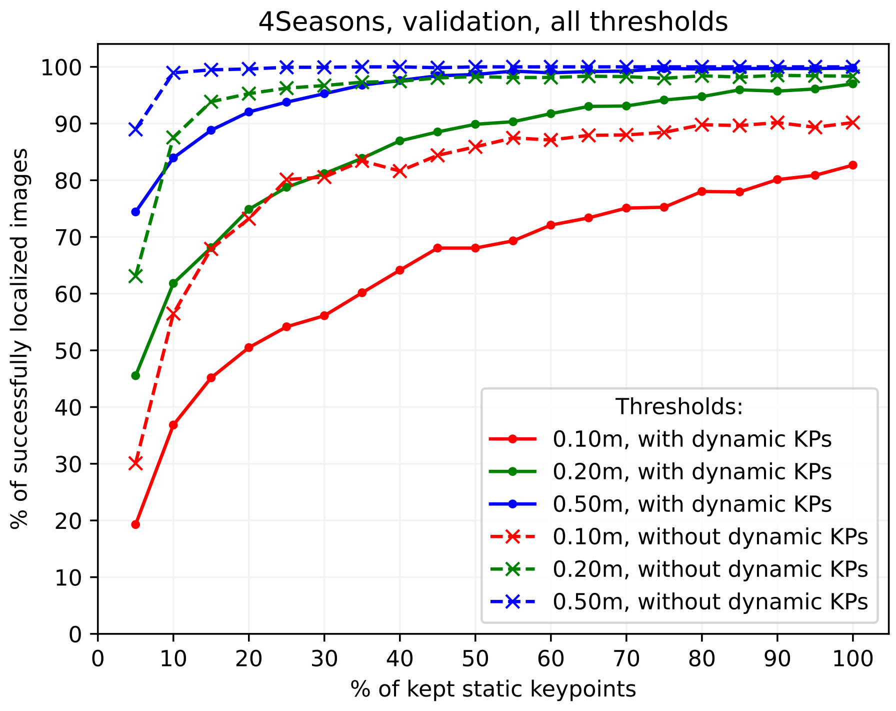

Figure B.5: 4Seasons, validation sequence, run 3: the results for all localization thresholds for experiments with and without dynamic keypoints.

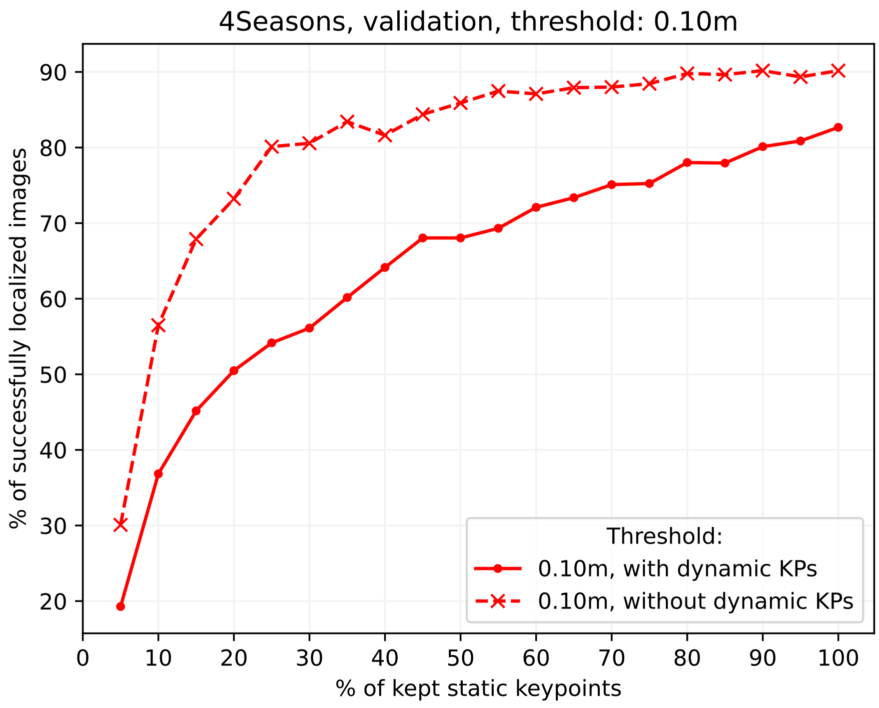

Figure B.6: 4Seasons, validation sequence, run 3: the results for 0.10m localization threshold for experiments with and without dynamic keypoints.

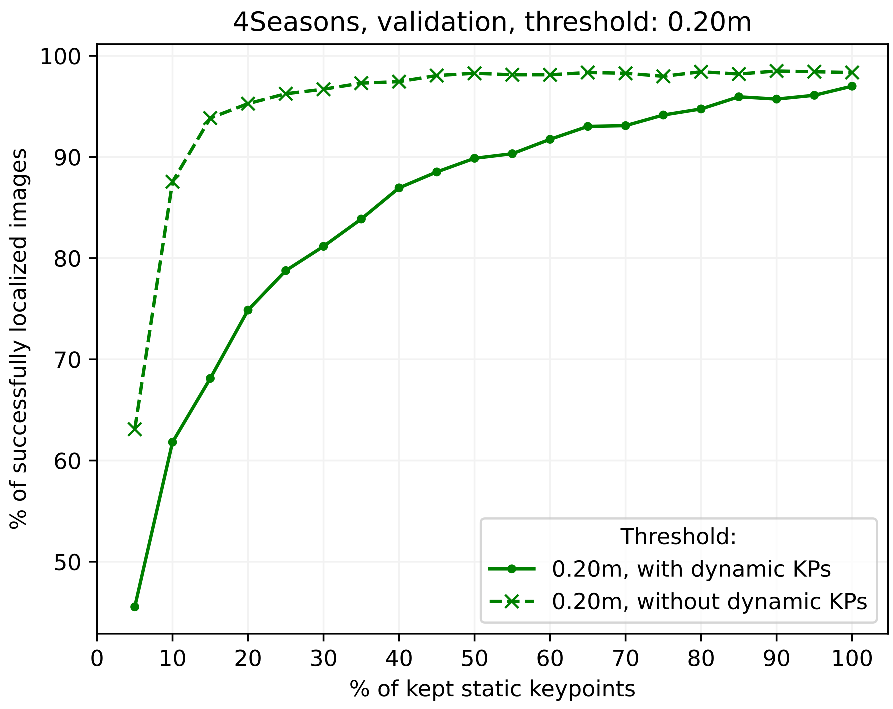

Figure B.7: 4Seasons, validation sequence, run 3: the results for 0.20m localization threshold for experiments with and without dynamic keypoints.

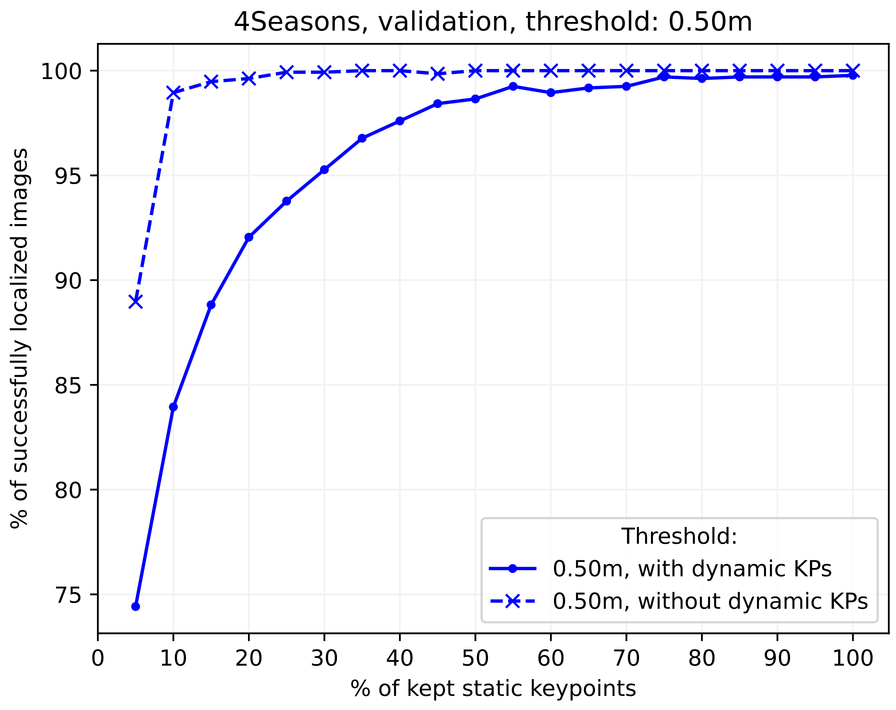

Figure B.8: 4Seasons, validation sequence, run 3: the results for 0.50m localization threshold for experiments with and without dynamic keypoints.

Appendix C

Aachen v1.1 Day Sequence

In this appendix are the 2nd and 3rd runs of presented pipeline from Chapter 10 and their respective results and figures.

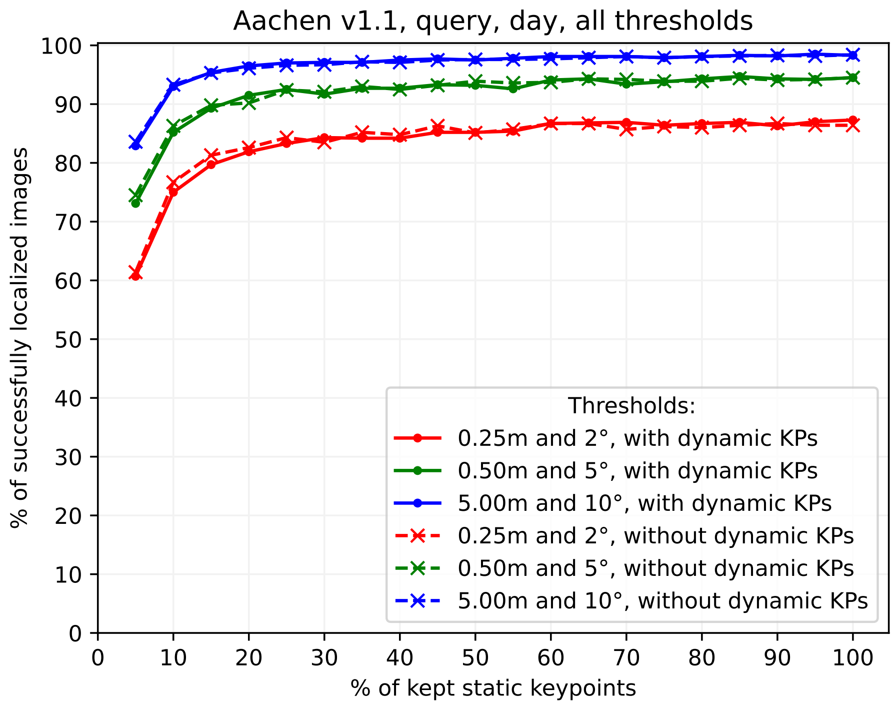

Figure C.1: Aachen v1.1, day query sequence, run 2: the results for all localization thresholds for experiments with and without dynamic keypoints.

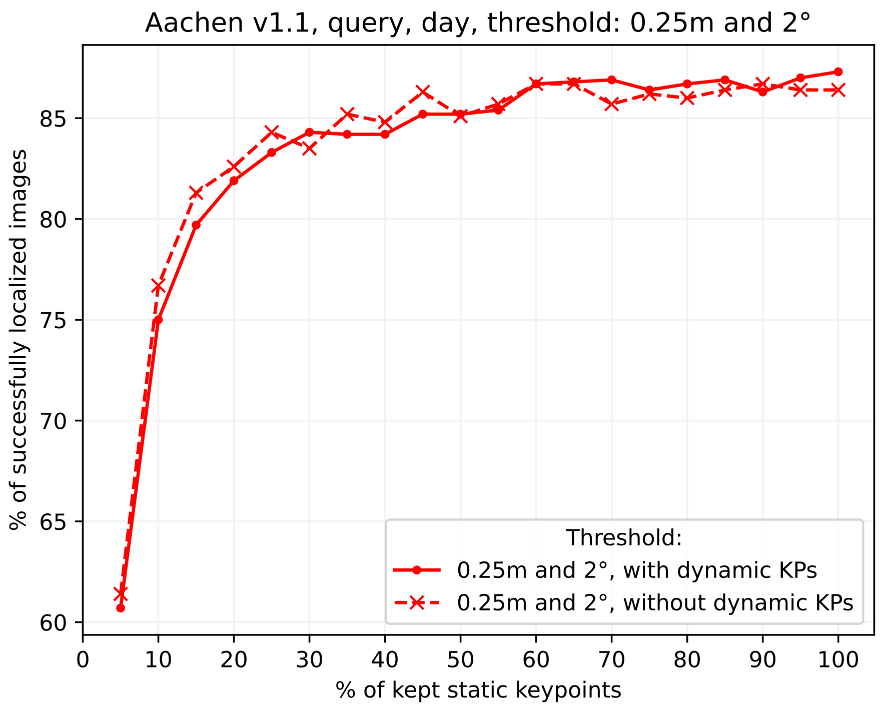

Figure C.2: Aachen v1.1, day query sequence, run 2: the results for 0.25m and 2° localization threshold for experiments with and without dynamic keypoints.

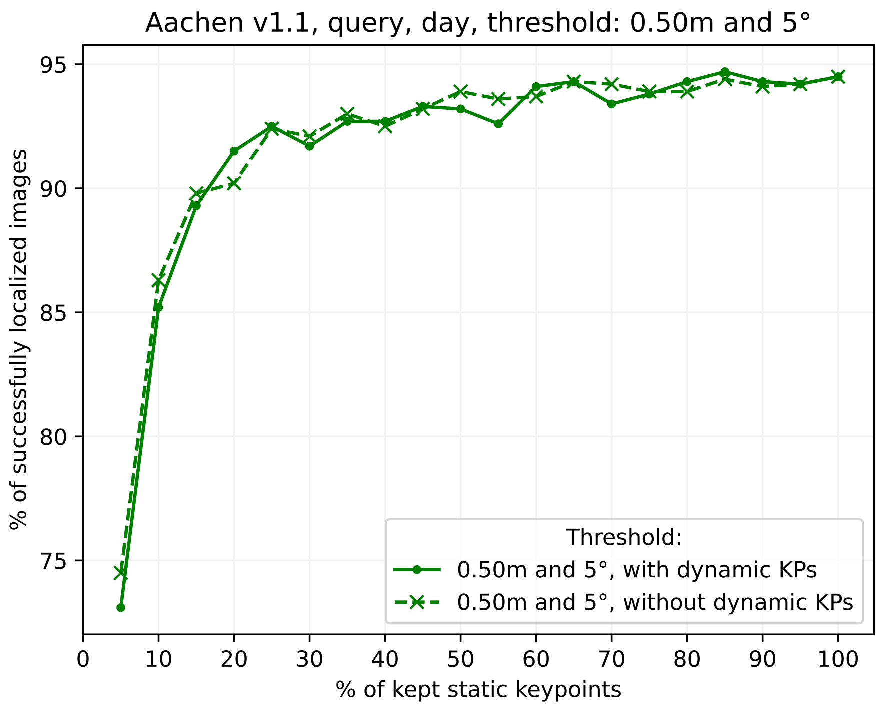

Figure C.3: Aachen v1.1, day query sequence, run 2: the results for 0.50m and 5° localization thresholds for experiments with and without dynamic keypoints.

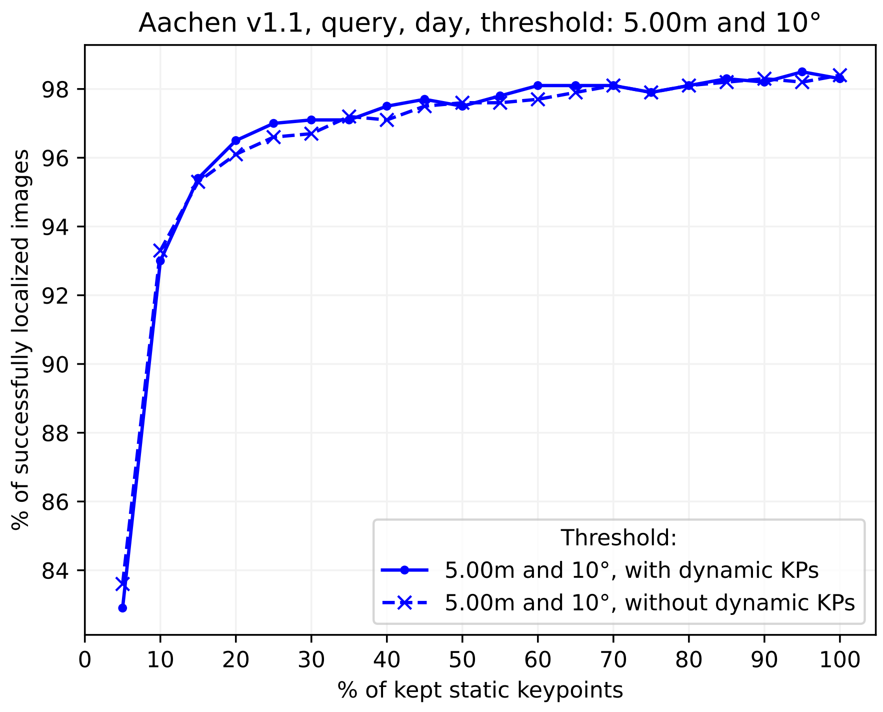

Figure C.4: Aachen v1.1, day query sequence, run 2: the results for 5.00m and 10° localization threshold for experiments with and without dynamic keypoints.

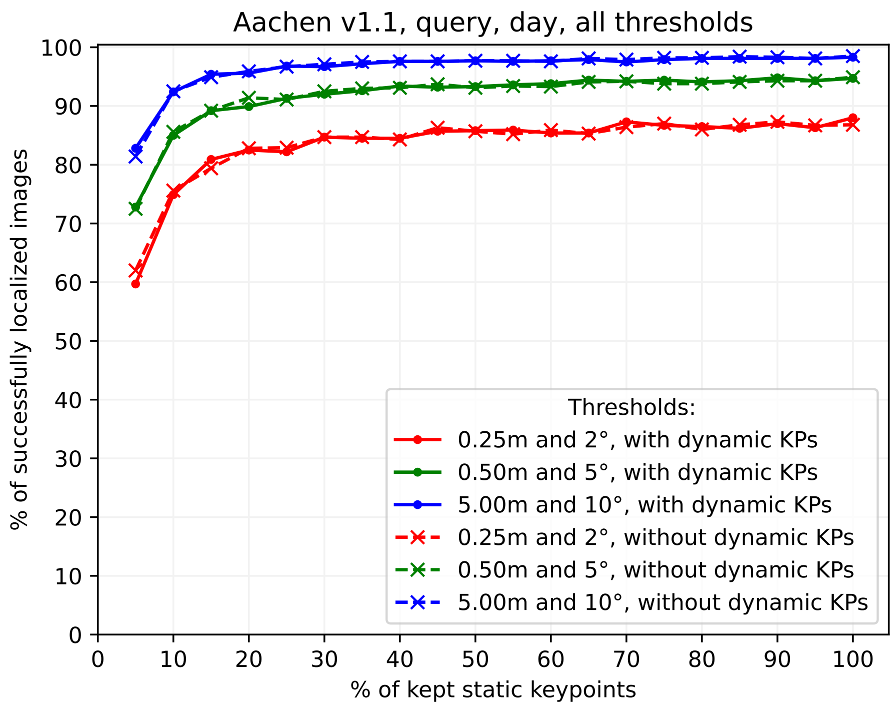

Figure C.5: Aachen v1.1, day query sequence, run 3: the results for all localization thresholds for experiments with and without dynamic keypoints.

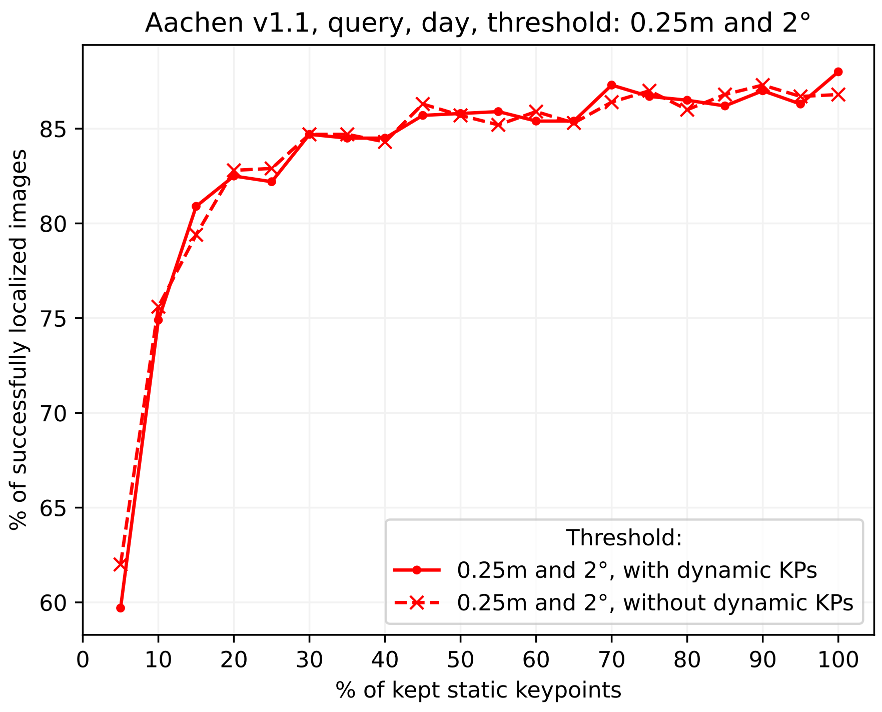

Figure C.6: Aachen v1.1, day query sequence, run 3: the results for 0.25m and 2° localization thresholds for experiments with and without dynamic keypoints.

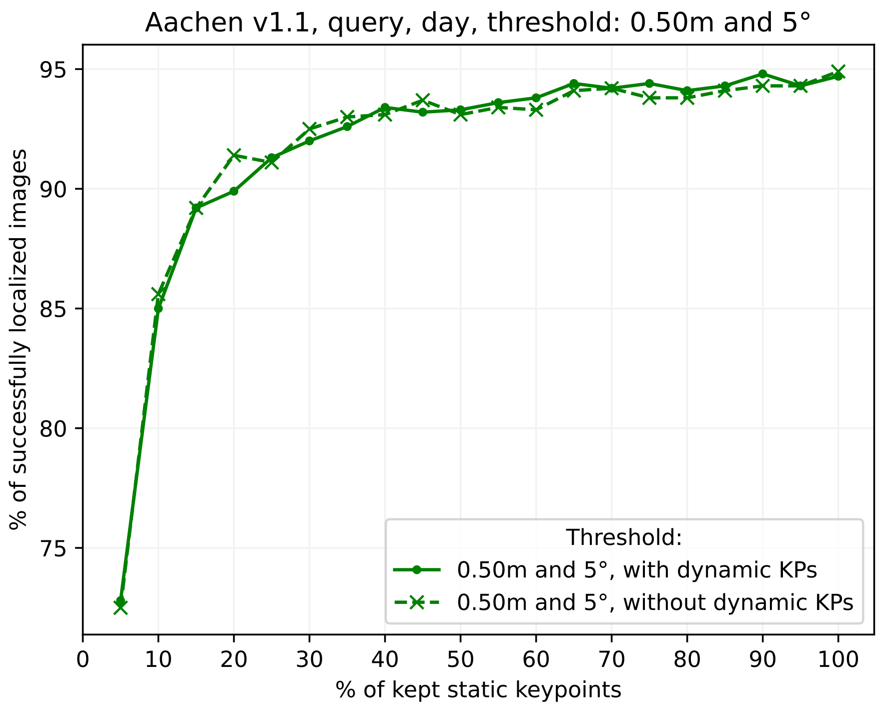

Figure C.7: Aachen v1.1, day query sequence, run 3: the results for 0.50m and 5° localization thresholds for experiments with and without dynamic keypoints.

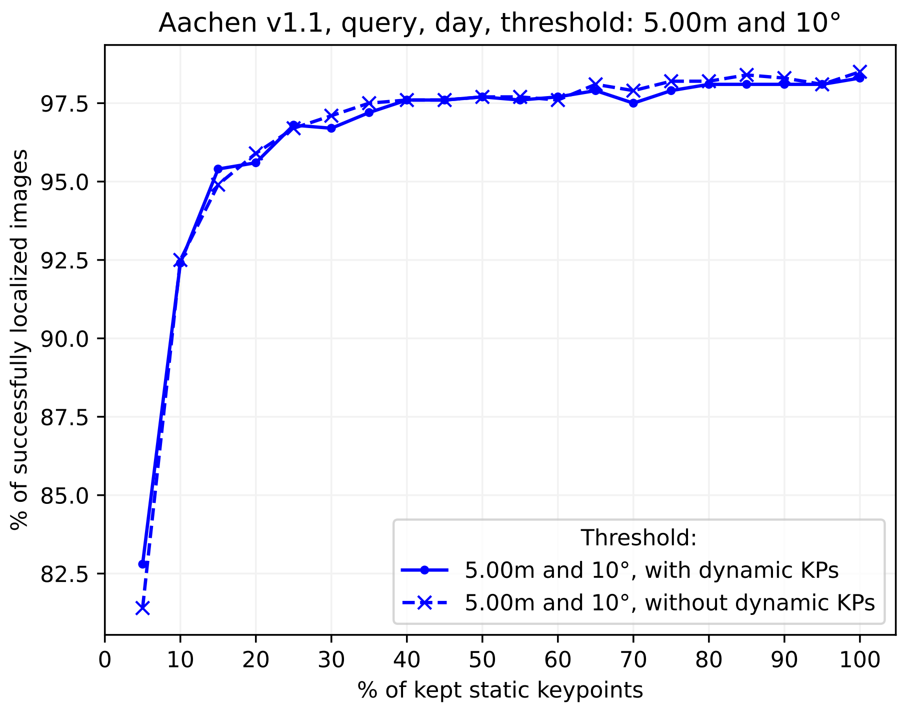

Figure C.8: Aachen v1.1, day query sequence, run 3: the results for 5.00m and 10° localization threshold for experiments with and without dynamic keypoints.

Bibliography

[1] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features," in *Conference on Computer Vision and Pattern Recognition*, 2019.

[2] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," in *The IEEE International Conference on Computer Vision (ICCV)*, pp. 3476–3485, 10 2017.

[3] B. Cao, A. Araujo, and J. Sim, "Unifying deep local and global features for efficient image search," *CoRR*, vol. abs/2001.05027, 2020.

[4] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3640–3649, 2016.

[6] A. Tao, K. Sapra, and B. Catanzaro, "Hierarchical multi-scale attention for semantic segmentation," *CoRR*, vol. abs/2005.10821, 2020.

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

[8] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "Kaze features," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV'12, (Berlin, Heidelberg), pp. 214–227, Springer-Verlag, 2012.

[9] H. P. Moravec, "Towards automatic visual obstacle avoidance," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, p. 584, 1977.

[10] H. P. Moravec, “Visual mapping by a robot rover,” in *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1*, pp. 598–600, 1979.

[11] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, 1988.

[12] J. Shi and C. Tomasi, “Good features to track,” in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR’94)*, pp. 593 – 600, 1994.

[13] S. M. Smith and J. M. Brady, “Susan&mdash;a new approach to low level image processing,” *Int. J. Comput. Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[14] M. Trajkovic and M. Hedley, “Fast corner detection,” *Image and Vision Computing*, vol. 16, pp. 75–87, Feb. 1998.

[15] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004.

[16] D. Lowe, “Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image,” Mar. 23 2004. US Patent 6,711,293.

[17] H. Bay, T. Tuytelaars, and L. Van Gool, *SURF: Speeded Up Robust Features*, pp. 404–417. Springer Berlin Heidelberg, 2006.

[18] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking,” in *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV ’05, (Washington, DC, USA), pp. 1508–1515, IEEE Computer Society, 2005.

[19] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, 2006.

[20] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, 2010.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *Proceedings of the 2011 International Conference on Computer Vision*, 2011.

[22] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Proceedings of the 2011 International Conference on Computer Vision*, 2011.

[23] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint.," in *CVPR*, pp. 510–517, IEEE Computer Society, 2012.

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

[25] G. Levi and T. Hassner, "Latch: Learned arrangements of three patch codes," *CoRR*, vol. abs/1501.03719, 2015.

[26] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 971–987, July 2002.

[27] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761 – 767, 2004. British Machine Vision Computing 2002.

[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, (Washington, DC, USA), pp. 886–893, IEEE Computer Society, 2005.

[29] F. Tombari and L. di Stefano, "Interest points via maximal self-dissimilarities," in *Computer Vision - ACCV 2014 - 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part II*, pp. 586–600, 2014.

[30] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition.," in *CVPR*, 2016.

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

[32] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos.," in *ICCV*, pp. 1470–1477, IEEE Computer Society, 2003.

[33] A. Benbihi, S. Arravechia, M. Geist, and C. Pradalier, “Image-based place recognition on bucolic environment across seasons from semantic edge description,” in *International Conference on Robotics and Automation (ICRA)*, 2020.

[34] J. Revaud, P. Weinzaepfel, C. R. de Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger, “R2D2: repeatable and reliable detector and descriptor,” *CoRR*, vol. abs/1906.06195, 2019.

[35] M. Hassaballah, H. Alshazly, and A. A. Ali, “Analysis and Evaluation of Keypoint Descriptors for Image Matching,” in *Recent Advances in Computer Vision*, 2019.

[36] H. Fan, Y. Zhou, A. Li, S. Gao, J. Li, and Y. Guo, “Visual localization using semantic segmentation and depth prediction,” *CoRR*, vol. abs/2005.11922, 2020.

[37] S. Peng, Z. He, H. Zhang, R. Yan, C. Wang, Q. Zhu, and X. Liu, “Megloc: A robust and accurate visual localization pipeline,” *CoRR*, vol. abs/2111.13063, 2021.

[38] R. Arandjelovic and A. Zisserman, “All about vlad,” in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[39] K. Mikolajczyk and C. Schmid, “Scale &amp; affine invariant interest point detectors,” *Int. J. Comput. Vision*, vol. 60, pp. 63–86, Oct. 2004.

[40] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, “A comparison of affine region detectors,” *International Journal of Computer Vision*, vol. 65, pp. 43–72, Nov. 2005.

[41] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, “Learning local feature descriptors with triplets and shallow convolutional neural networks,” in *Proceedings of the British Machine Vision Conference (BMVC)* (E. R. H. Richard C. Wilson and W. A. P. Smith, eds.), pp. 119.1–119.11, BMVA Press, September 2016.

[42] 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 International Conference on Computer Vision (ICCV)*, 2015.

[43] K. Simonyan, A. Vedaldi, and A. Zisserman, “Learning local feature descriptors using convex optimisation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[44] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys, “Quad-networks: Unsupervised learning to rank for interest point detection,” in *Conference on Computer Vision and Pattern Recognition*, pp. 3929–3937, 2017.

[45] L. Zhang and S. Rusinkiewicz, “Learning to detect features in texture images,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[46] K. M. Yi, E. Trulls Fortuny, V. Lepetit, and P. Fua, “Lift: Learned invariant feature transform,” *Computer Vision - Eccv 2016, Pt Vi*, vol. 9910, pp. 17. 467–483, 2016.

[47] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, “Lf-net: Learning local features from images,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS’18, (Red Hook, NY, USA), p. 6237–6247, Curran Associates Inc., 2018.

[48] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, “Inloc: Indoor visual localization with dense matching and view synthesis,” in *Conference on Computer Vision and Pattern Recognition*, (Salt Lake City, United States), June 2018.

[49] I. Rocco, R. Arandjelović, and J. Sivic, “Convolutional neural network architecture for geometric matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2553–2567, 2019.

[50] G. Tolias, R. Sicre, and H. Jégou, “Particular Object Retrieval With Integral Max-Pooling of CNN Activations,” in *International Conference on Learning Representations*, International Conference on Learning Representations, (San Juan, Puerto Rico), pp. 1–12, May 2016.

[51] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, “Pyramid methods in image processing,” *RCA Engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[52] 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 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 4826–4837, Curran Associates, Inc., 2017.

[53] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, “Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors,” *CVPR*, 2017.

[54] H. Lim, S. Sinha, M. Cohen, and M. Uyttendaele, “Real-time image-based 6-dof localization in large-scale environments,” *Conference on Computer Vision and Pattern Recognition*, 03 2012.

[55] R. Castle, G. Klein, and D. Murray, “Video-rate localization in multiple maps for wearable augmented reality,” *2008 12th IEEE International Symposium on Wearable Computers*, pp. 15–22, 2008.

[56] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[57] W. Wei-Tung, W. Yi-Leh, T. Cheng-Yuan, and H. Maw-Kae, “Adaptive density-based spatial clustering of applications with noise (dbscan) according to data,” *International Conference on Machine Learning and Cybernetics*, vol. 1, p. 445–451, 2015.

[58] S. Ullman, “The interpretation of structure from motion,” *Proceedings of the Royal Society of London. Series B. Biological Sciences*, p. 405– 426, 1979.

[59] J. Fuentes-Pacheco, J. Ascencio, and J. Rendon-Mancha, “Visual simultaneous localization and mapping: A survey,” *Artificial Intelligence Review*, vol. 43, 11 2015.

[60] C. Valgren and A. Lilienthal, “Sift, surf and seasons: Appearance-based long-term localization in outdoor environments,” *Robotics and Autonomous Systems*, vol. 58, pp. 149–156, 02 2010.

[61] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, “Complete solution classification for the perspective-three-point problem,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, pp. 930– 943, 09 2003.

[62] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof, “From structure-from-motion point clouds to fast location recognition,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2599–2606, 06 2009.

[63] M. Teichmann, A. Araujo, M. Zhu, and J. Sim, “Detect-to-retrieve: Efficient regional aggregation for image search,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5104–5113, 06 2019.

[64] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, “Deep image retrieval: Learning global representations for image search,” in *Proceedings of the European Conference on Computer Vision*, vol. 9910, pp. 241–257, 10 2016.

[65] A. Yandex and V. Lempitsky, “Aggregating local deep features for image retrieval,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1269–1277, 12 2015.

[66] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6230–6239, 07 2017.

[67] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, “Bisenet: Bilateral segmentation network for real-time semantic segmentation,” in *Proceedings of the European Conference on Computer Vision*, p. 325–341, 08 2018.

[68] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” *Proceedings of the European Con- ference on Computer Vision*, 02 2018.

[69] Z. Li and N. Snavely, “Megadepth: Learning single-view depth prediction from internet photos,” in *Proceedings of the IEEE Conference on Com- puter Vision and Pattern Recognition*, pp. 2041–2050, 06 2018.

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

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

[72] M. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision, ECCV 2014 - 13th European Conference, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp. 818–833, Springer Verlag, 2014.

[73] F. Radenović, G. Tolias, and O. Chum, “Fine-tuning cnn image retrieval with no human annotation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 1655–1668, 2019.

[74] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *International Journal of Computer Vision*, vol. 124, pp. 237–254, Sept. 2017.

[75] A. Araujo, W. Norris, and J. Sim, "Computing receptive fields of convolutional neural networks," *Distill*, 2019.

[76] F. Wang, X. Xiang, J. Cheng, and A. Yuille, "Normface: L2 hypersphere embedding for face verification," *Proceedings of the 25th ACM international conference on Multimedia*, 2017.

[77] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[78] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation.," *PAMI*, 2017.

[79] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.

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

[81] M. M. Chun, "Contextual cueing of visual attention," *Trends in Cognitive Sciences*, vol. 4, no. 5, pp. 170 – 178, 2000.

[82] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *Science*, vol. 328, no. 5980, pp. 876–878, 2010.

[83] V. Nicosia, G. Bianconi, V. Latora, and M. Barthelemy, "Growing multiplex networks," *Physical review letters*, vol. 111, p. 058701, 08 2013.

[84] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *CoRR* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 1263–1272, PMLR, 06–11 Aug 2017.

[85] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner,

Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *CoRR*, vol. abs/1806.01261, 2018.

[86] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, Curran Associates, Inc., 2017.

[87] G. Peyré and M. Cuturi, "Computational optimal transport: With applications to data science," *Foundations and Trends in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[88] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 2292–2300, Curran Associates, Inc., 2013.

[89] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices.," *Pacific J. Math.*, vol. 21, no. 2, pp. 343–348, 1967.

[90] Y. Yuan, X. Chen, and J. Wang, "Object-contextual representations for semantic segmentation," in *Computer Vision – ECCV 2020*, (Cham), pp. 173–190, Springer International Publishing, 2020.

[91] D. Lin, D. Shen, S. Shen, Y. Ji, D. Lischinski, D. Cohen-Or, and H. Huang, "Zigzagnet: Fusing top-down and bottom-up context for object segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7482–7491, 2019.

[92] J. Fu, J. Liu, Y. Wang, Y. Li, Y. Bao, J. Tang, and H. Lu, "Adaptive context network for scene parsing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[93] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3146–3154, 2019.

[94] H. Zhang, H. Zhang, C. Wang, and J. Xie, "Co-occurrent features in semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[95] Y. Yuan and J. Wang, "Ocnet: Object context network for scene parsing," *CoRR*, vol. abs/1809.00916, 2018.

[96] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017.

[97] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L. Chen, "Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 12472–12482, IEEE, 2020.

[98] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. D. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 8856–8865, Computer Vision Foundation / IEEE, 2019.

[99] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[100] D. Lee, "Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks," 2013.

[101] Y. Li, L. Liu, and R. T. Tan, "Certainty-driven consistency loss for semi-supervised learning," *CoRR*, vol. abs/1901.05657, 2019.

[102] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 1195–1204, 2017.

[103] H. Badino, D. Huber, and T. Kanade, "Visual topometric localization," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 794 – 799, 07 2011.

[104] T. Sattler, W. Maddern, A. Torii, J. Sivic, T. Pajdla, M. Pollefeys, and M. Okutomi, "Benchmarking 6dof urban visual localization in changing conditions," *CoRR*, vol. abs/1707.09092, 2017.

[105] J. Dong, J. G. Burnham, B. Boots, G. C. Rains, and F. Dellaert, "4d crop monitoring: Spatio-temporal reconstruction for agriculture," *CoRR*, vol. abs/1610.02482, 2016.

[106] H. Germain, G. Bourmaud, and V. Lepetit, “Sparse-to-dense hypercolumn matching for long-term visual localization,” in *2019 International Conference on 3D Vision (3DV)*, pp. 513–523, 2019.

[107] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 Year, 1000km: The Oxford RobotCar Dataset,” *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[108] C. Linegar, W. Churchill, and P. Newman, “Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 787–794, IEEE, 2016.

[109] J. Spencer, R. Bowden, and S. Hadfield, “Same features, different day: Weakly supervised feature learning for seasonal invariance,” in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[110] M. Tanner, P. Piniés, L. M. Paz, and P. Newman, “What lies behind: Recovering hidden shape in dense mapping,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 979–986, 2016.

[111] W. Maddern, G. Pascoe, and P. Newman, “Leveraging experience for large-scale lidar localisation in changing cities,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, 05 2015.

[112] M. Larsson, E. Stenborg, L. Hammarstrand, M. Pollefeys, T. Sattler, and F. Kahl, “A cross-season correspondence dataset for robust semantic segmentation.,” in *Conference on Computer Vision and Pattern Recognition*, pp. 9532–9542, Computer Vision Foundation / IEEE, 2019.

[113] S. Griffith, G. Chahine, and C. Pradalier, “Symphony lake dataset,” *International Journal of Robotic Research*, vol. 36, 08 2017.

[114] N. Sünderhauf, P. Neubert, and P. Protzel, “Are we there yet? challenging seqslam on a 3000 km journey across all four seasons,” in *ICRA 2013*, 2013.

[115] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla, “Benchmarking 6dof outdoor visual localization in changing conditions,” in *Conference on Computer Vision and Pattern Recognition*, pp. 8601–8610, 2018.

[116] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt, “Image retrieval for image-based localization revisited,” in *British Machine Vision Conference*, 09 2012.

[117] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[118] J. L. Schönberger, T. Price, T. Sattler, J.-M. Frahm, and M. Pollefeys, "A vote-and-verify strategy for fast spatial verification in image retrieval," in *Asian Conference on Computer Vision (ACCV)*, 2016.

[119] A. Resindra, A. Torii, and M. Okutomi, "Structure from motion using dense cnn features with keypoint relocalization," *IPSJ Transactions on Computer Vision and Applications*, vol. 10, 12 2018.

[120] Z. Zhang, T. Sattler, and D. Scaramuzza, "Reference pose generation for long-term visual localization via learned features and view synthesis," *International Journal of Computer Vision*, 2020.

[121] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2 - a large-scale benchmark for instance-level recognition and retrieval," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[122] P. Wenzel, R. Wang, N. Yang, Q. Cheng, Q. Khan, L. von Stumberg, N. Zeller, and D. Cremers, "4Seasons: A cross-season dataset for multi-weather SLAM in autonomous driving," in *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2020.

[123] E. Wijmans and Y. Furukawa, "Exploiting 2d floorplan for building-scale panorama rgbd alignment," in *Computer Vision and Pattern Recognition, CVPR*, 2017.

[124] B. Vasileios, R. K. Duncan, Frost, B.-L. Axel, T. Arjang, H. Huub, and M. Krystian, "Silda: Scape imperial localisation dataset," 2019.

[125] M. Schulze, "A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method," *Social Choice and Welfare*, vol. 36, pp. 267–303, 02 2011.

[126] P. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12708–12717, 2019.

[127] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual slam," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2466–2473, 2018.

[128] M. Humenberger, Y. Cabon, N. Guérin, J. Morat, J. Revaud, P. Rerole, N. Pion, C. R. de Souza, V. Leroy, and G. Csurka, “Robust image retrieval-based visual localization using kapture,” *CoRR*, vol. abs/2007.13867, 2020.

[129] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, 2014.

[130] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[131] N. Capece, U. Erra, and R. Scolamiero, “Converting night-time images to day-time images through a deep learning approach,” in *21st International Conference Information Visualisation*, pp. 324–331, 07 2017.

[132] A. Anoosheh *et al.*, “Night-to-day image translation for retrieval-based localization,” *CoRR*, vol. abs/1809.09767, 2018.

[133] L. Sun, K. Wang, K. Yang, and K. Xiang, “See clearer at night: towards robust nighttime semantic segmentation through day-night image conversion,” in *Artificial Intelligence and Machine Learning in Defense Applications*, p. 8, 09 2019.