MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Mentzl Leonard** Personal ID number: **474391**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Computer Vision and Image Processing**

II. Master’s thesis details

Master’s thesis title in English:

Visual Localisation and Navigation in Changing Environments

Master’s thesis title in Czech:

Vizuální lokalizace a navigace v proměnném prostředí

Guidelines:

1. Familiarise yourself with the problems related to visual navigation of mobile robots in changing environments.
2. Perform review of methods that can use a series of images captured at the same location to construct environment models aimed to achieve long-term vision-based navigation or localisation.
3. Select datasets useful for perspective visual navigation methods.
4. Propose improvements of these datasets, methodology of their collection and eventually gather the relevant data yourself.
5. Select a set of methods suitable for long-term localisation and navigation in changing environments.
6. Select a set of key performance criteria to evaluate the aforementioned methods.
7. Perform the evaluation and discuss the results.

Bibliography / sources:

[1] Krajník, T., Fentanes, J.P., Santos, J.M. and Duckett, T., 2017. Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments. IEEE Transactions on Robotics, 33(4), pp.964-977.
[2] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. and Leonard, J.J., 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on robotics, 32(6), pp.1309-1332.
[3] Lowry, S., Sünderhauf, N., Newman, P., Leonard, J.J., Cox, D., Corke, P. and Milford, M.J., 2015. Visual place recognition: A survey. IEEE Transactions on Robotics, 32(1), pp.1-19.
[4] Rosen, D.M., Mason, J. and Leonard, J.J., 2016, May. Towards lifelong feature-based mapping in semi-static environments. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1063-1070). IEEE.
[5] Krajník, T., Vintr, T., Molina, S., Fentanes, J.P., Cielniak, G., Mozos, O.M., Broughton, G. and Duckett, T., 2019. Warped hypertime representations for long-term autonomy of mobile robots. IEEE Robotics and Automation Letters, 4(4), pp.3310-3317.
[6] Sünderhauf, N., Shirazi, S., Jacobson, A., Dayoub, F., Pepperell, E., Upcroft, B. and Milford, M., 2015. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. Robotics: Science and Systems XI, pp.1-10.
[7] Zdeněk Rozsypálek et al.: Semi-Supervised Learning for Image Alignment in Teach and Repeat navigation. In SAC 2022 (to appear)

Name and workplace of master's thesis supervisor:

Ing. Zdeněk Rozsypálek Department of Computer Science FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **17.01.2022** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

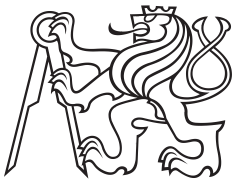
Ing. Zdeněk Rozsypálek	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Visual Localisation and Navigation in Changing Environments

Leonard Mentzl

Supervisor: Ing. Zdeněk Rozsypálek
Field of study: Open Informatics
Subfield: Computer Vision and Image Processing
May 2022

Acknowledgements

Thanks to CTU for its friendly study environment.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information within it in accordance with the methodological instructions for observing the ethical principles in the preparation of university theses.

Mirošovice, date 7th May 2022

Abstract

The goal of this thesis is comparison of different approaches for visual localisation and navigation in changing environments. The tests are performed on a custom dataset collected on a car park. We compare the methods of image retrieval and image alignment separately. We test feature-based methods, namely Bag-of-Words and VLAD, with features SIFT, SURF, BRIEF and their variants. We test feature-less methods GIST and neural-network-based methods, namely ConvNet Landmarks, NetVLAD and image alignment using fully-convolutional neural network. We test the influence of some modelling methods, namely FAB-MAP and FreMEn. It turns out that NetVLAD is the best method for image retrieval and fully-convolutional siamese neural network is the best method for image alignment.

Keywords: computer vision, visual localisation, visual navigation, changing environments, autonomous vehicles

Supervisor: Ing. Zdeněk Rozsypálek

Abstrakt

Tato práce se zabývá srovnáním různých metod pro vizuální lokalizaci a navigaci v proměnném prostředí. Metody testujeme na vlastním datasetu nasbíraném na parkovišti. Srovnáváme metody pro vyhledávání a zarovnávání obrázků. Testujeme metody využívající bodové vzorky: Bag-of-Words a VLAD; s vzorky: SIFT, SURF, BRIEF a jejími variantami. Testujeme metody bez bodových vzorků: GIST a metody využívající neuronové sítě: ConvNet Landmarks, NetVLAD a zarovnání pomocí plně konvoluční siamské neuronové sítě. Testujeme vliv různých metod pro modelování: FAB-MAP a FreMEn. Z výsledků plyne, že pro vyhledávání obrázků je nejelepší NetVLAD a pro zarovnání je nejlepší plně konvoluční siamská neuronová síť.

Klíčová slova: počítačové vidění, vizuální lokalizace, vizuální navigace, proměnné prostředí, autonomní vozidla

Překlad názvu: Vizuální lokalizace a navigace v proměnném prostředí

Contents

Figures

Tables

ctuthesis t1606152353

Chapter 1

Introduction

Visual localisation is a field in robotics which is being researched actively. There are many difficult challenges in visual localisation. The real-world environment changes over time and therefore robots operating for longer time periods have to deal with the varying environment. The scene is illuminated differently in different phases of day, surfaces have different optical features based on weather, foliage colour differs based on seasons etc. [1]. Other than that, there are also way more complicated changes, such as vehicles being parked differently and different items being stocked in a storage house. If the robot navigates based on parked cars or stocked items, the navigation may yield completely irrelevant output on the next day [2]. Since it is possible to operate robots for longer time periods, the methods used for navigation have to be adjusted in a way so they give meaningful results throughout longer periods of time. There are many approaches to this problem, for example [3, 4, 5, 6]. In this thesis, we compare the quality of several such approaches on a custom dataset.

The basic task of image retrieval is that the robot is given a dataset which consists of images from different places and then it is given a query image and it has to find the corresponding image from the dataset. The corresponding image is supposed to come from the same place and orientation as the query image. There are several methods to tackle this problem, such as Bag-of-Words [7] or VLAD [8]. However, these methods assume that the dataset was captured in the same conditions as the query images. It can also happen that the dataset was captured in day and the query was captured in night. Or also some mobile objects such as cars could have moved in the meantime. If we talk about long-term visual localisation, we no longer assume that the dataset images and the query images were taken under the same conditions.

Usually, the methods are tested on a specific change in the environment, such as daytime changes or seasonal changes. In this thesis, we will test the methods on more changes in the environment, including daytime changes, seasonal changes and changes of mobile objects.

There are also methods which utilise more information gathered from the world. There are structure-based approaches such as Active Search[9], which build a 3D representation of the world from the obtained images. The task then changes to matching the query image with the obtained 3D representation. There are some hierarchical approaches such as Hierarchichal Localisation [10] and Visual Localization Using a Sparse Semantic 3D Map [11], which build a hierarchy out of the places in the dataset. The hierarchical structure then serves to reduce the search space. When the robot already knows that it is in a certain building, it does not have to look for correspondences in a different city. There are also sequential methods which are based on the thought that the robot does not occur at a place out of sudden but it got to the place somehow capturing other places. The methods such as PFSL [12] and SeqSLAM [13] do not use only the last photo for localisation but also a short sequence of preceding photos.

In this thesis, the methods localise based on a single photo but the dataset may contain more than one photo for each place, captured in different times and conditions. Note that this problem is different from the sequential approach. In our scenario, we have a sequence of images for each place in the dataset and only one query image to estimate the location. The sequential approach is the inverse problem, where there is only one image for each place in the dataset and the location is estimated from a sequence of query images.

The methods for long-term visual localisation can be compared based on various criteria. The methods can be compared by how often they find the correct matching image in the database. When the world coordinates are provided for each image in the dataset, the method may output the coordinates in the world, not just the index of the matching image. Therefore, the quality of the method may be compared based on the error in the position estimation. And when the image orientations are also provided in the dataset, the tested methods may estimate the 6-DOF pose of the robot. And then another criterion can be based on how well the orientation of the query image is estimated. However, none of these criteria involve the practical impact on navigation. If the robot needs to navigate to a given target, the exact location might not be required for this task. The robot may think that it is somewhere else but if it decides for the correct direction anyway, it does not mind if the robot then localises again and fixes the error. After all, the objective of navigation is to navigate the robot towards the given target and not reporting its exact location.

In this thesis, we focus mainly on criteria which are directly related to long-term navigation. Different errors in visual localisation may cause different problems in navigation, so we do not compare only how often the tested methods fail but also how severe these failures are. The error rate may also change with the amount of captured photos at given places. If the robot has seen the place twice before, it may navigate more effectively than when it has seen the place only once before. However, the second measurement may also confuse the robot. Therefore, we also focus on how the quality of localisation methods changes with the amount of captured data. For the teach-and-repeat approach for navigation, we will also compare the methods according to how well they are able to registrate one photo over another given that they are from the same place.

To compare those methods, a custom dataset was collected. The dataset consists of 8389 photographs taken from selected points in a car park in Čestlice, Czech Republic. These photographs were taken in different day times, different week days and different months. As stated above, long-term visual navigation on car park is more challenging because the vehicles are varying dramatically and therefore some methods may turn to be completely unusable on this dataset. Some photographs capture a significant amount of foliage, so the seasonal changes are also present in this dataset.

Chapter 2

Related Work

Many methods have been proposed to tackle the long-term visual localisation. They were tested on different datasets. Usually, each presented method comes with its own dataset. There are also several benchmarks which test how these methods perform on different datasets. Let us discuss these datasets, methods and benchmarks.

2.1 Related datasets

There are several datasets suitable for long-term visual localisation and place recognition. There are several datasets which were created by driving a car on the same route in different conditions, such as the **Nurburgring** dataset [13], **St. Lucia** dataset [14], **Alderley** dataset [13], the **CMU Seasons** dataset [15, 16], the **Oxford RobotCar** dataset [17] and the **EU Car** dataset [18]. The Nurburgring [13] consists of two traversals of the car racing track in Nurburg. The St. Lucia dataset [14] was captured by passing a street network in the suburbs of St. Lucia in various daytimes. In each traversal, the same set of streets were passed. The map is provided in the paper. The Alderley dataset [13] consists of two traversal of a selected loop in the suburbs of Alderley. The CMU Seasons dataset [15, 16] was taken by driving though a set of trajectories through the suburbs near Pittsburgh in different seasons. The trajectories were captured from two different views. However, they were all captured in daytime. The Oxford RobotCar dataset [17] captures a loop in Oxford in various daytime and weather conditions. The EU Car dataset [18] is similar, it differs in that it covers two loops in

France, one is located in a city, the other one is located in suburbs.

In some cases, the datasets were captured from a train, for example the **Nordland** dataset [19], or by a mobile robot, for example the **NCLT** dataset [20], the **Consolidated** dataset [21] and **GRIEF** datasets [22], or by a pedestrian, for example the **Mall** dataset [23]. The Nordland dataset [19] was captured on a long railway track in Norway, it was traversed four times in different seasons. The NCLT dataset [20] consists of traversals of a street grid in the University of Michigan's North Campus. The traversals are different but they usually cover a huge part of the street netwrok. A map of the street network is provided in the paper. The consolidated dataset [21] was captured by a robot in a relatively small area in Kralupy nad Vltavou. All traversals cover the same path, both day and night expeditions were performed.

GRIEF datasets [22] consist of four independent datasets focused on seasonal changes. The **Carlevaris** dataset consists of two traversals in an urban area, one in summer and one in winter. The **Stromovka** dataset is similar, it is located in the park Stromovka in Prague. The **Michigan** dataset consists of images of 5 places, each of them was captured 12 times, each month once. The **Planetarium** dataset is similar but it captures one place from 5 different angles which is located in front of the planetarium in Prague.

Some datasets were created from photos which are publicly available on the Internet, such as the **Pittsburgh** dataset [24], the **San Francisco** dataset [24], the **SPED** dataset [25], **Dubrovnik** dataset [26], the **Rome** dataset [26] and the **Landmarks** dataset [27]. The Pittsburgh and San Francisco datasets [24] were created from images available on Google Street View. At each place, a set of 24 images were created. After that, a subset of those places were visited and photographed. The SPED dataset [25] was created by downloading data from selected public webcams available on the Internet. The Dubrovnik and Rome datasets [26] were created from publicly available geo-tagged photos in the Internet. The Dubrovnik dataset consists of photos located in the historical district in Dubrovnik and the Rome dataset consists of photos of various landmarks in Rome. The Dubrovnik dataset consists of one huge component while the Rome dataset consists of 69 insulated components. The Landmarks dataset [27] consists of 6 selected landmarks across the world.

Some datasets were created by volunteers visiting the location on foot, such as the **Tokyo 24/7** dataset [28], **7 Scenes** dataset [29], the **Achen** dataset [30], the **Cambridge Landmark** dataset [31], the **Vienna** dataset [32], or by an autonomous robot in the **Witham** dataset [4]. The Tokyo 24/7 dataset [28] consist of 125 selected places in Tokyo, each place was captured 3 times in 3 different angles. The 7 scenes dataset [29] consists of 7

insulated indoor scenes. The Aachen dataset [30] covers the historical district of Aachen and includes both day and night photos. The Cambridge Landmark dataset [31] captures different walks in areas close to 5 selected landmarks in Cambridge. The walks around given landmark differ across expeditions. The Viena dataset [32] consists of a set of photos of the landmarks of Vienna. The paper tells only a little information about the methodology of how the photos were captured.

Other than that, the **Greg's Office** dataset [33] was collected by a static camera in an office and the dataset contains the estimated depth map.

Table 2.1 contains the summary of all these datasets. The photos in the datasets form various topologies. In some cases, they are only insulated points. In some cases, they form long lines or loops. When multiple connected streets are captured, they form a network. And in some datasets, the photos are taken arbitrarily within an area or a small set of areas. In most of the datasets, only one photo is taken at a place but in some datasets, more photos with different orientations were being taken simultaneously at given places. Every time the location was visited in order to take photos for the dataset, we call it one expedition. Another criterion to catalogise the dataset it cross-coverage, which says how often the places were visited. In some datasets, it was intended to visit each place in each expedition, we call it complete cross-coverage. Sometimes, only some of the places were intended to be visited more times, we call it partial cross-coverage. And in the rest of datasets, the authors were capturing the photos in the same areas throughout the expeditions but it was not required to capture any concrete place more than once. We call it sparse cross-coverage. The column Poses says whether or not ground-truth orientations are provided in the dataset. We focus on three real-world periodicities, the day period, the week period and the year period. The day period affects the overall illumination and traffic density. The week period affects the density of traffic and the year period affects the foliage colour and visibility. There is also month periodicity but we do not focus on it because it has minimal impact in our scenario. We say that the dataset covers a periodicity when it is observable from the data. Only a few of these datasets cover all three periodicities. Question marks in the table mean that the term either does not make sense for the given setting or the paper does not offer such information.

In this thesis, we represent each place by a sequence of images and the methods are offered another image and then they have to guess to which sequence the image belongs. That implies that each place must be covered by at least three expeditions. Therefore, the datasets Aachen, Alderley, Cambridge Landmark, Dubrovnik, GRIEF–Carlevaris, GRIEF–Stromovka, Landmarks, Mall, Nurburgring, Pittsburgh, Rome, San Francisco and Vienna

do not deserve any additional comments.

We do not compare the methods only based on how well they localise but also on the actual impact in navigation. Therefore, it should be possible to perform at least simple navigation tasks in the topology of the dataset. If the topology is only a single line, a loop or a small area, like in the datasets Consolidated, EU Car, Greg's Office, GRIEF–Michigan, GRIEF–Planetarium, Nordland and Oxford RobotCar, it is insufficient for even a simplified navigation task.

Datasets such as 7 Scenes, SPED, Tokyo 24/7, and Witham contain several different places so it may make sense to navigate between them but there is no topological information provided in these datasets. The 7 Scenes dataset contain 7 insulated indoor scenes so there is no navigation task possible. In the Witham, some places are seen from other places, so this information could be added manually. However, it contains only 8 places and the connections between them are quite sparse. In the Tokyo 24/7 dataset, there is a map which shows a street network with some of the places. 125 places were observed and the map displays only 35 of them. The rest is possibly very far from these 35 points. The photos are geo-tagged, so it would be possible to place these locations onto a street map and deduce the topology from there. Other than that, this dataset has only 3 expeditions, which is sufficient for our task but it is still very few. The SPED dataset observes the places continuously, so there is no problem with lack of expeditions. The topology could be obtained in a similar way as in the Tokyo 24/7 dataset but it would most likely turn out that these places are scattered over many different towns and are too far apart. And navigation in such sparse grid does not make sense.

Out of these 27 datasets, only the NCLT, CMU Seasons and St. Lucia datasets could be potentially used for this task but these datasets have other weaknesses. The datasets should cover more real-world periodicities, ideally all of them. The NCLT and the CMU Seasons dataset do not cover the day/night changes and the St. Lucia dataset does not cover any seasonal changes.

2.2 Related Methods

There are two tasks which use similar methods: image retrieval and image alignment. These tasks share some of their methods. The framework for image

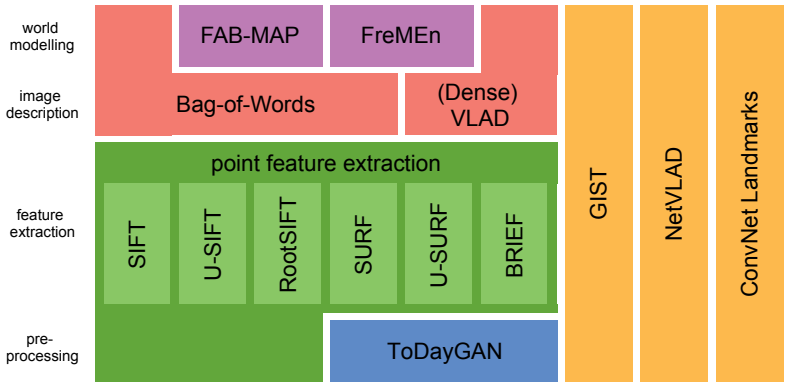


Figure 2.1: The architecture of our image retrieval framework

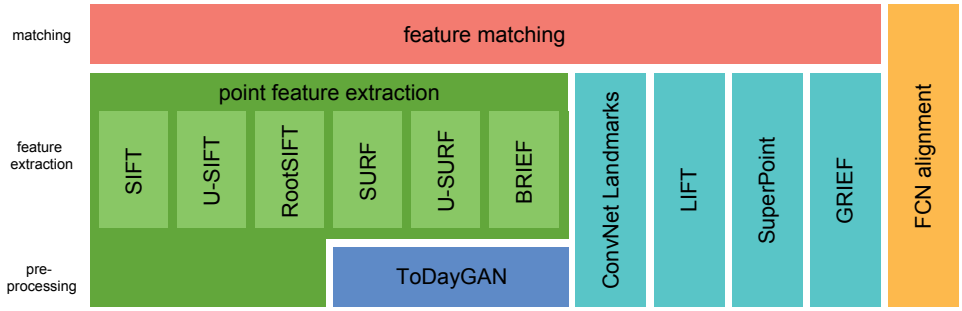


Figure 2.2: The architecture of our image alignment framework

retrieval has four layers: optional pre-processing layer, feature extraction layer, image description layer and optional world modelling layer. In the pre-processing layer, the images are pre-processed to eliminate some of the disruptive features, for example night images might be transferred to day images. In the feature extraction layer, some kind of features are extracted from the image. They can be either point features or more complex features such as area features. Each feature is assigned a vector of fixed size. In the image description layer, each image is assigned a global descriptor, which is one vector of fixed size. In the world modelling layer, a more complex model of the world might be created in order to find the relevant data effectively. The figure 2.1 depicts an overview of these layers together with the methods discussed later. Image alignment has a similar framework, it has three layers: optional pre-processing layer, feature extraction layer and matching layer. The first two layers are the same as in the image retrieval framework. In the matching layer, the extracted features are matched to obtain the mutual displacement. The overview of the image alignment framework is depicted in the figure 2.2.

2.2.1 Pre-processing Layer

ToDayGAN [5] is an approach which transforms the images using a generative adversial neural network first to make them look like they were captured in day. After that, a standard method for image retrieval is used, the paper [5] uses the DenseVLAD [28] algorithm.

2.2.2 Feature Extraction Layer

BRIEF [34] is a method for describing image features based on binary comparisons between pixels. BRIEF features are invariant with respect to rotation, scale and intensity change.

GRIEF [22] is a variant of BRIEF, which is less variant to outdoor seasonal changes. It does not involve any neural networks, it is trained using evolutionary algorithm. It is not as robust as deep learning approaches but it is fast to compute.

LIFT [35] is a feature detection and description method which uses a deep neural network. One neural network performs keypoint detection, rotation estimation and feature description sequentially. Experiments show that LIFT performs significantly better than standard approaches such as SIFT or SURF. Other experiments [36] show it fails on extremely rotated pairs of images, although it has its own learnt rotation estimation.

RootSift [37] is a description method based on SIFT. Unlike SIFT, the histograms [38] in the descriptor are being compared statistically correctly. This is attained by performing the element-wise square root on the SIFT descriptor and measuring the distance of two descriptors by dot product.

SuperPoint [36] is a deep neural network which performs feature detection and description jointly. It is trained on synthetic images first and then on real images. Experiments show that SuperPoint performs significantly better than LIFT and standard approaches such as SIFT or SURF but it fails to match extremely rotated scenes.

SIFT [39] is a description method for image features which extracts Harris corners [40] from the image and describes them based on gradients. SIFT

features are invariant with respect to rotation, scale and intensity change.

SURF [41] is a description method similar to SIFT but it uses different operations to make the algorithm way faster. Like SIFT, SURF features are also invariant with respect to rotation, scale and intensity change.

U-SIFT [42] and **U-SURF** [41], also called upright-SIFT and upright-SURF are variants of the SIFT and SURF descriptors respectively which are not invariant with respect to rotation unlike standard SIFT and SURF descriptors.

ConvNet Landmarks [43] is an algorithm for image description which uses neural networks to extract and describe landmarks. At first, a set of landmark proposals are selected using the Edge Boxes algorithm [44]. Then each landmark is described by AlexNet [45], which is a neural network pre-trained on the ImageNet [46] dataset. The dimension of these descriptors is then reduced using Gaussian Random Projection [47]. The similarity between two images is then calculated by counting the mutual best matches of individual landmarks.

2.2.3 Image Description Layer

Bag-of-Words [7] is a representation of images which was previously used for text documents and is based on occurence of words. At first, a database of image is taken and a huge set of features is extracted from them. These features are clustered using the k-means algorithm [48] into k clusters, which are then reffered as visual words. Each image is assigned a vector based on occurence of those visual words. These vectors can be then used for image retrieval. Any kind of feature extraction method can be used for this purpose.

DenseVLAD [28] is an algorithm which uses the VLAD image descripror [8] on densely sampled featured described by RootSift [37], which is derived from SIFT [39]. VLAD is an approach similar to bag-of-words [7] but it does not accumulate the count of words but differences from the centroids of words. The image descriptors can be then used for image retrieval.

Gist [49] is a method for creating a global image descriptor. Unlike other approaches, it does not extract any local features from the images in advance. This global descriptor can be used for image retrieval.

NetVLAD [50] is an approach which involves a neural network with a custom VLAD layer. This VLAD layer does something similar to the VLAD descriptor [8] and is fully differentiable. Such network can be then trained to an arbitrary task, such as visual localisation.

2.2.4 World Modelling Layer

FAB-MAP [3] is a localisation method which models the probabilities of feature visibility using Chow-Liu trees [51]. The paper uses the SURF [41] features but it may work with other features as well. It is not only capable of recognising from which place the image is, it can also decide whether or not a new place was observed.

FreMEn [4] creates a temporal model for feature appearance. The paper [4] uses the BRIEF features [34]. At first, the appearance of features is predicted using discrete Fourier Transform [52] and then an image retrieval method based on occurence of features is used.

2.2.5 Matching Layer

Image alignment using fully-convolutional siamese neural networks (FCN) [6] is a method to obtain the horizontal displacement of two images, which relies on a fully-convolutional neural network [53]. The neural network outputs a 1-dimensional signal for both images. The displacement of both images is then given by the maximum of cross-correlation [54] of both signals. The maximal value of cross-correlation can be also used as a score of how much the images match each other, so this method can be also used for image retrieval but it is too much time-demanding.

Feature matching is defined for each kind of point features separately. The definition of each feature also specifies how the matching score shall be calculated. To select the matching pairs, there are different strategies. We will use the mutual best match. After that, the image displacement is calculated using RANSAC [55]. In this case, only one match is needed to calculate the displacement, so RANSAC degenerates to a sequential search.

2.3 Related Benchmarks

The papers presenting a method for visual localisation usually contain a comparison with a different method proving that it is better in the given setting. Recently, a comparison of many long-term visual localisation methods was done [56]. Unlike this thesis, it provides also a comparison of localisation methods which rely on more aspects than a single photo, for example an already-built point cloud or a sequence of preceding photos. However, it does not compare any methods which can utilise more images per place and the selected criteria do not involve the practical impact on navigation.

There is the Image Matching Benchmark [57] for the Image Matching Challenge 2021 which aims on matching pairs of images from different viewpoints. The conditions vary across the images but it is not the main objective of the challenge. Unlike this thesis, the objective of the challenge is 6-DOF pose estimation. In this thesis, we aim only on one degree of freedom, the horizontal displacement.

There is a long-term visual localisation benchmark [58], where the objective is to correctly estimate the translation and rotation of query images. This benchmark does not aim on the practical impact on navigation. In this benchmark, the rotation is computed without the prior knowledge of what image the rotation shall be estimated against. However, in our benchmark, the methods for rotation estimation are tested on ground-truth matching pairs.

Dataset	Topology	Ors.	Exps.	CC.	Poses	Setting	Periodicities		
							Day	Week	Year
7 Scenes [29]	7 Points	1	?	C	✓	Indoor	✗	✗	✗
Aachen [30]	Area	1	?	S	✓	Historic city	✓	✗	✗
Alderley [13]	Loop	1	2	C	✗	Suburban	✓	✗	✗
Cambridge Landmark [31]	5 Areas	1	2	S	✓	Outdoor	✗	✗	✗
CMU Seasons [15, 16]	17 Lines	2	12	C	✓	Suburban	✗	✗	✓
Consolidated [21]	Area	1	179	C	✓	Outdoor	✓	✗	✗
Dubrovnik [26]	Area	1	2	S	✓	Historic City	✗	✗	✗
EU Car [18]	2 Loops	1	20	C	✓	City, Suburban	✓	✓	✓
Greg's Office [33]	1 Point	1	?	C	✓	Indoor	✓	✓	✓
GRIEF–Carlevaris [22]	Line	1	2	C	✓	City	✗	✗	✓
GRIEF–Michigan [22]	5 Points	1	12	C	✓	City	✗	✗	✓
GRIEF–Planetarium [22]	1 Point	5	12	C	✓	City	✗	✗	✓
GRIEF–Stromovka [22]	Line	1	2	C	✓	City	✗	✗	✓
Landmarks [27]	6 Areas	1	?	?	✓	Landmarks	✗	✗	✗
Mall [23]	Line	4	2	S	✓	Indoor	✗	✗	✗
NCLT [20]	Network	1	27	P	✓	Indoor, Outdoor	✗	✓	✓
Nordland [19]	Line	1	4	C	✓	Railway	✓	✗	✓
Nurburgring [13]	Loop	1	2	C	✗	Road	✓	✗	✗
Oxford RobotCar [17]	Loop	1	72	C	✓	City	✓	✓	✓
Pittsburgh [24]	Network	24	2	P	✗	City	✗	✗	✗
Rome [26]	69 Areas	1	2	S	✗	Landmarks	✗	✗	✗
San Francisco [24]	Network	24	2	S	✓	City	✗	✗	✗
SPED [25]	2543 Points	1	?	C	✓	Outdoor	✓	✓	✓
St. Lucia [14]	Network	1	10	C	✗	Suburban	✓	✓	✗
Tokyo 24/7 [28]	125 Points	3	3	C	✗	City	✓	✗	✗
Vienna [32]	Set of Areas	1	?	?	✗	Historic City	✗	✗	✗
Witham [4]	8 Points	1	10080	C	✓	Indoor	✓	✓	✗
Čestlice	34 Poins	8	31	C	✓	Car Park	✓	✓	✓

Table 2.1: Table summarising the existing datasets, our dataset is on the bottom of the table. Ors. = Orientations, Exps. = Expeditions, CC. = Cross-Coverage, S = sparse, P = partial, C = complete

Chapter 3

Čestlice Dataset

To train and test our methods, we need a dataset which consists of photos of real-world places. The photos should be captured on a few fixed places in the world over a long time period. This thesis focuses on localisation from a sequence of images (i.e. there is a sequence of images for each place), so there must be at least two training measurements and one testing measurement. So we can use only datasets which have at least 3 measurements from different times for each place. There are three mandatory periods in the real world: day period, week period and year period. Ideally, more time periods should be covered by the dataset.

For our purpose, we need a dataset with many expeditions to make a good temporal model. The count of places may improve the robustness of trained methods but it is not as important. To improve the robustness more, the captured places should be farther apart. However, this distance must be considered logically by the similarity of the views, which does not always correspond to the same metric distance. When the scene is densely populated by smaller objects, then smaller changes of camera position are sufficient to reach a different view, unlike the situation when there is a few bigger objects.

To test the impact of errors in visual localisation to navigation, we should be able to perform some non-trivial navigation tasks in the dataset, at least in a simplified form. Therefore, the places should be organised in a topology which is more complicated than a long path or long circle. Unfortunately, the places in most of the datasets are organised this way.

I decided to collect my own dataset, I call it the Čestlice dataset. The

No.	date			time		wx.	notes
	wd.	D	M	from	to		
0	Fri	9	Jul	13:17:54	14:04:28		1 photo missing at place 0
1	Sat	10	Jul	14:14:24	15:04:34		
2	Wed	14	Jul	16:06:32	17:03:36		1 extra photo at place 32
3	Thu	15	Jul	05:19:00	05:47:30		1 extra photo at place 33
4	Sun	18	Jul	14:10:30	14:53:20		Times in EXIF are 12 hours behind.
5	Mon	19	Jul	08:06:56	08:45:16		
6	Tue	20	Jul	18:07:40	18:55:08		
7	Wed	11	Aug	19:04:40	19:44:42		
8	Thu	12	Aug	07:06:54	07:55:40		
9	Sun	15	Aug	06:08:34	06:39:24		
10	Mon	23	Aug	14:08:24	14:59:08		Places 17, 19 and 21 were skipped.
11	Tue	24	Aug	05:22:16	05:56:06		
12	Fri	27	Aug	14:56:44	15:37:24		
13	Sat	28	Aug	20:14:54	20:52:02		
14	Wed	8	Sep	20:04:58	20:42:10		
15	Thu	9	Sep	14:08:06	14:41:34		
16	Sat	11	Sep	04:55:04	05:31:04		
17	Tue	14	Sep	09:35:14	10:03:36		
18a	Fri	17	Sep	05:18:54	05:34:06		only places 22–29, 30 (5 photos), 31–33
18b	Fri	24	Sep	05:20:32	05:45:44		only places 0–21, 30
19	Sun	19	Sep	04:55:34	05:28:38		
20	Mon	20	Sep	18:01:24	18:36:18		
21	Tue	5	Oct	10:03:58	10:36:58		
22	Fri	8	Oct	07:01:26	07:30:02		
23	Sun	10	Oct	06:09:12	06:45:06		
24	Mon	11	Oct	04:56:20	05:33:52		
25	Wed	13	Oct	11:07:28	11:56:50		
26	Sat	16	Oct	14:14:52	14:59:50		
27	Thu	21	Oct	19:09:18	20:01:42		1 photo missing at place 25
28	Thu	11	Nov	12:07:22	13:07:04		1 extra photo at place 27
29	Mon	5	Nov	07:13:12	07:49:10		Place 31 was skipped.
30	Fri	18	Nov	20:04:08	20:51:04		Places 1 and 12 were skipped.

Table 3.1: Conditions under which the photos were taken. No. = expedition number, wd. = week day, D = day, M = month, wx. = weather

Figure 3.1: Showcase of the Čestlice Dataset. Vehicle registration plates were blurred additionally in this showcase.

photos were taken in the year 2021 in the commercial district of Čestlice, Czech Republic. I picked 34 places and I took 8 photos at each of them in each expedition. The aim of this dataset is to cover three real-world periods: day period, week period and month period.

I segmented the day to four intervals: 0:00 - 6:00, 6:00 - 12:00, 12:00 - 18:00, 18:00 - 24:00, all with respect to the local time zone. Each week has 7 days. Due to time constraints, I was not able to capture the dataset for more than half a year, so I divided this period into months from July to December. That makes 168 combinations in total. That would require too much time, so I decided to cover only each combination of each couple of periods, like in pairwise testing [59]. That can be attained by only 42 expeditions. An exhaustive list of the conditions under which the photos were taken is given in the table 3.1.

The photos were captured by the camera Fujifilm Finepix C10, using the broadest view and resolution 1600×1200 pixels. The calibration matrix is:

$$K = \begin{bmatrix} 1778 & 0 & 800 \\ 0 & 1778 & 600 \\ 0 & 0 & 1 \end{bmatrix}$$

The collection of the dataset had to be interrupted after 31 st expedition due to unpleasing social conditions.

I annotated each photo by a set of two vanishing points. These vanishing points are matched to the photographical map in Mapy.cz. Each vanishing point has direction either +1 (the matched direction is in front of the camera), or -1 (the matched direction is behind the camera), or 0 (the matched direction lies in the focal plane). Using these annotations, it is possible to reconstruct the orientation of the camera.

The annotation is semi-automatic. I wrote a script which searches for vanishing points in images and I used this script for preprocessing. After that, I selected some good vanishing points from the detected vanishing points manually and assigned them a direction in the real world. If there were not enough detected good vanishing points, I selected them from the image manually.

Line segments are detected using probabilistic Hough transform and vanishing points are detected using MLESAC [60] with least-square optimisation. In this process, all points and lines are in homogeneous coordinates in the basis γ. They are also normalised to the unit ball, so the error is not distorted for points which are far from the principal point. As the error function, I take the absolute value of the dot product of line and detected vanishing point.

3.1 Estimation of Rotation in Čestlice Dataset

This step should be done manually but it is intractable for such huge dataset. Therefore I decided for a semi-automatic annotation. Note that any standard method for pose estimation cannot be used if it is supposed to serve as ground truth for the tested methods.

Each photo in the dataset has two vanishing points ${u'}_\beta$ and ${v'}_\beta$ which are mapped to directions u and v in the photographical map. These two vanishing points are normalised to the unit sphere and have sign according to their direction. Our task is to find the rotation of the camera. At first, we undo the calibration by multiplying the vanishing points ${u'}_\beta$ and ${v'}_\beta$ by the inverse of the calibration matrix K^{-1} from left and normalising the vectors again. We get ${u'}_\gamma = \frac{K^{-1}{u'}_\beta}{||K^{-1}{u'}_\beta||}$ and ${v'}_\gamma = \frac{K^{-1}{v'}_\beta}{||K^{-1}{v'}_\beta||}$. Now we need to find a rotation matrix R which maps u and v to ${u'}_\gamma$ and ${v'}_\gamma$. The angle between u

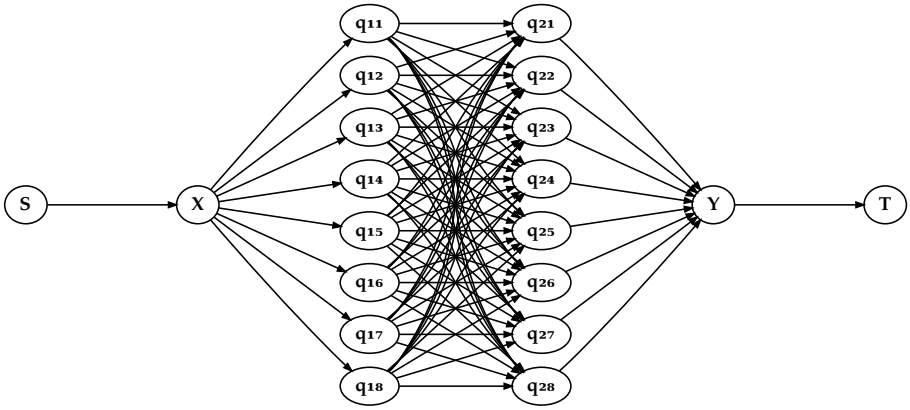


Figure 3.2: Graph of the flow network to solve. S is the source node, T is the target.

and v may not match the angle between ${u'}_\gamma$ and ${v'}_\gamma$ because of the noise in the data, so we minimise the mean square error. That can be done by setting:

$$x = \frac{u+v}{||u+v||} \quad y = \frac{u-v}{||u-v||} \quad z = x \times y$$

$$x' = \frac{{u'}_\gamma + {v'}_\gamma}{||{u'}_\gamma + {v'}_\gamma||} \quad y' = \frac{{u'}_\gamma - {v'}_\gamma}{||{u'}_\gamma - {v'}_\gamma||} \quad z' = x' \times y'$$

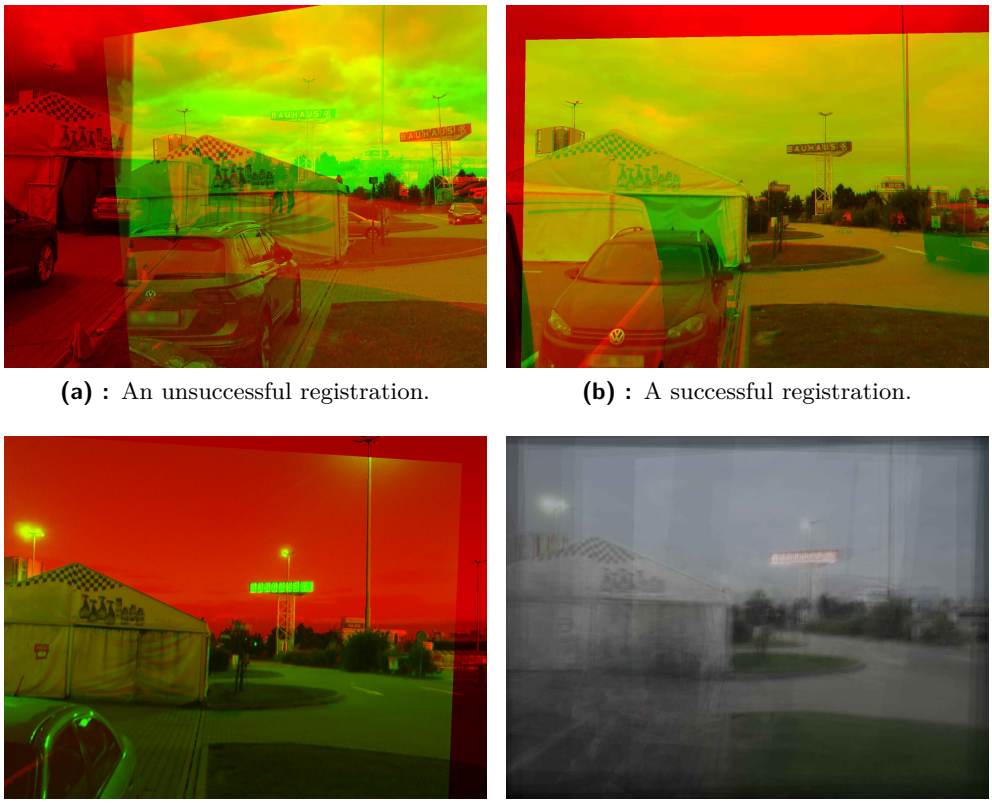
Note that the vectors x, y and x', y' are orthogonal. Therefore x, y, z and x', y', z' form two orthonormal bases. We can then express the rotation as $R = B'B^T$ where:

$$B = \begin{bmatrix} x & y & z \end{bmatrix}$$

$$B' = \begin{bmatrix} x' & y' & z' \end{bmatrix}$$

This method is not accurate enough but we can use it to classify the photos into 8 groups per place, one group for each orientation. We pick 8 model orientations and then we pair them with photos for given expedition and place. There are typically 8 such photos. One of the typical orientation points roughly northwards so we set the model orientations to point to 8 world directions exactly. We set the cost of pairing rotation q_1 with q_2 as $-\left(\Re(q_1\bar{q}_2)\right)^2$ where q_1 and q_2 are expressed as quaternions. We are looking for an optimal pairing which minimises this cost. That can be done by solving the following min-cost flow problem [61].

The figure 3.2 shows the network of the min-cost problem to be solved. The node S is the source node, the node T is the target node. The nodes



(a) : An unsuccessful registration.

(b) : A successful registration.

(c) : A successful day vs. night registration.

(d) : Registration of all 31 images.

Figure 3.3: Illustration images for registration. Faces and car registrations were blurred additionally. Red channel is the target image and green channel is the source image.

$q_{11}, \ldots, q_{18}$ correspond with the model rotations and $q_{21}, \ldots, q_{28}$ correspond with the actual rotations. The cost of edges between the nodes $q_{11}, \ldots, q_{18}$ and $q_{21}, \ldots, q_{28}$ are set to the cost of pairing the given rotations, the cost of all other edges are zero. For all edges except the edges from S to X and from Y to T holds that the minimum flow is 0 and the maximum flow is 1. The minimum and maximum flow for the edges from S to X and from Y to T are set to the number of pairs to match, which is typically 8. It can be less if some of the photos are missing. This is to enforce that given number of pairs must be matched. We use the cycle cancelling algorithm [62] to solve this problem. When the problem is solved, we select the pairs which were assigned a nonzero flow.

After we classify the angles for whole the dataset, we have a set of photos pointing nearly the same direction for each place and angle. We can estimate the rotation accurately using registration. At first, I will explain how to register one pair of photos. We have two grayscale photos I_1 and I_2 related by homography KR^TK^{-1}, where K is the calibration matrix and R is a rotation

matrix. The photos in the dataset are colourful, so we need to convert them to grayscale first. We maximise the mutual information which is equal to:

$$MI(I_1, I_2) = H(I_1) + H(I_2) - H(I_1, I_2)$$

where H is the entropy, which is calculated as:

$$H(X) = -\sum_{s \in S} P(X = s) \log P(X = s)$$

where S is the set of symbols in the signal and $P(X = s)$ is the probability of receiving symbol s. These probabilities are estimated using histograms. For calculating the entropy of one image, we have $S = \{0, \ldots, 255\}$, which refer to the intensity of pixels and for calculating the entropy of two images, we have $S = \{0, \ldots, 255\}^2$, which refer to the intensity of pixels of the first image and the intensity of the corresponding pixels in the other image. If $P(X = s) = 0$, the term $P(X = s) \log P(X = s)$ does not make sense. In that case, we use the limit $\lim_{P(X=s)\to 0^+} P(X = s) \log P(X = s)$, which equals zero. We calculate the mutual information only on the intersection of both images.

We maximise the mutual information by a probabilistic method similar to simulated annealing. For this optimisation, we represent the rotation as Euler angles and start by rotation r which has all angles set to zero. Then we optimise the mutual information iteratively for different scales from s_{min} to s_{max}. For each scale s, we scale the images by factor 2^s and then blur them by Gaussian filter with constant σ. We set the step to $t = 2^{-s} t_{base}$ We also have to recalculate the calibration matrix.

$$K' = \begin{bmatrix} 2^s & 0 & 0 \\ 0 & 2^s & 0 \\ 0 & 0 & 1 \end{bmatrix} K$$

For each scale, we iterate the following steps for i_{max} iterations. We create a new rotation r' which is rotation r shifted by a random vector whose elements are uniformly chosen from the range $[-t, t]$. Then we create a matrix R' which represents the rotation r' and transform the image I_2 by homography $K'R'K'^{-1}$ and then we evaluate the mutual information. If the mutual information is higher, we set r to r'.

For this dataset, it turns out that the optimisation yields satisfying results for $s_{min} = -4$, $s_{max} = -3$, $\sigma = 4$, $t_{base} = 0.001$ and $i_{max} = 100$.

It turns out that this algorithm often yields results which are way off (fig. 3.3a). However, many of them are pretty accurate (fig. 3.3b), and sometimes it manages to register two photos where one was captured in day and the

other was captured in night (fig. 3.3c). If we register all pairs of photos from the same place pointing roughly in the same direction and evaluate the quality of registration, we can choose the maximum spanning tree. Then we can calculate the rotation for each photo. As a visual check, we can transform each photo by corresponding homography and lay them all over each other. Then we get an image similar to figure 3.3d.

It turns out that the mutual information does not give us a good measure of the quality of registration. We calculate the Canny edges [63] for both images, then we blur them by Gaussian filter and then we calculate the correlation of both results. Higher correlation means better quality. For Canny edge detection, we choose parameters $\sigma = 4$, lower threshold $T_l = 2$ and higher threshold $T_h = 50$. And for Gaussian blur, we choose $\sigma = 10$.

For image alignment, the roll angle has to be zero, so we rotate each image to make the roll zero. Then we calculate the displacement as the difference of the principal points. However, it turns out that this method does not give a good enough estimate to be used as ground-truth for benchmarking alignment methods. There are many outliers and some places were registrated poorly. Therefore, a subset of 10 places and 10 expeditions were chosen and the outliers were fixed there manually. Namely, expeditions 0, 3, 10, 11, 13, 16, 23, 25, 27, 29 and places 0 (rotation 0), 3 (rotation 3), 5 (rotation 6), 7 (rotation 0), 10 (rotation 3), 11 (rotation 0), 11 (rotation 7), 20 (rotation 1), 28 (rotation 3), 29 (rotation 0) were chosen. A diagram of all the places is depicted in the figure 4.1

Chapter 4

Criteria for evaluation

The methods regarding image retrieval are compared based on zero-one loss [64] and first-step loss. The methods regarding image alignment are compared based on how many image pairs are aligned correctly within a given threshold. Now let us discuss the zero-one loss and first-step loss.

The selected methods are given the photos from all preceding expeditions. There are 34 places captured in 8 different rotations, which make 272 combinations in total. We will call them oriented places. The index of an oriented place is the index of place multiplied by 8 plus the index of rotation. The indices of rotations are depicted in the small star diagram in the figure 4.1. The method is given a photo from the dataset and it is supposed to tell what oriented place it corresponds to.

In the zero-one loss function, the method is penalised by 1 point for a wrong estimation and 0 points for a correct estimation. This error function reflects only the quality of the localisation and does not involve the practical impact on navigation. In the first-step loss, we measure the practical impact in navigation in a simplified world graph in figure 4.1. The places X and Y are not captured in the dataset but they may serve as good navigation points. All edges are assigned the same length. We call it first-step loss because it cares only about the first step in the navigation towards a given target.

Let us have a ground-truth oriented place and estimated oriented place. For each destination point, we calculate the error separately and calculate an arithmetic average of them, which corresponds to choosing the destination at random. We calculate the trajectory from the ground-truth oriented place

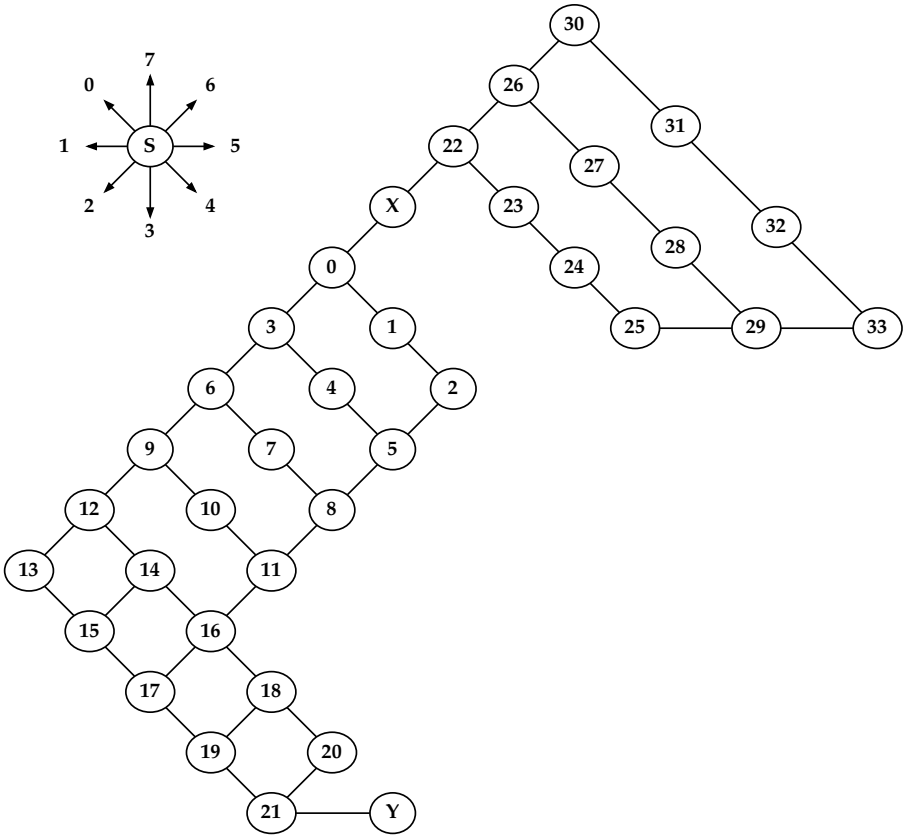


Figure 4.1: Simplified graph of world of the Čestlice dataset. The star figure in the top-left corner depicts the indices of rotations.

and from the estimated oriented place. We take only the first step in the trajectory, which is either *go forward, go forward-left, go left, go back-left, go back, go back-right, go right, go forward-right* or *stop*. The last option is chosen only when the starting and destination places are equal. The directions are relative to the orientation of the starting oriented place. We can assume that the robot navigation system will not be responsible for keeping the robot on the road, so we do not need exact orientation for that.

There can be more shortest paths leading to the given destination which differ in the first step. So we choose the path where the first step is closest to the forward motion. If there are still more options, we treat them all equally. If there are more options for the first step from the ground-truth oriented place, we say that they are both correct. If there are more options for the first step from the estimated oriented place, we calculate an arithmetic average of errors for both of them, which corresponds to choosing the path at random. If the first step from the estimated oriented place is equal to at least one correct step, the error is 0, otherwise, the error is 1.

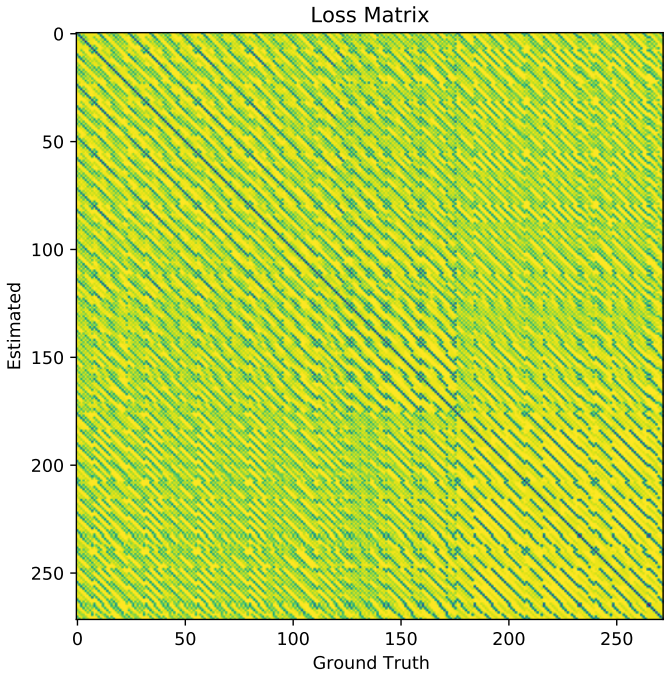


Figure 4.2: Visualisation of the loss matrix for the first-step loss criterion. Brighter colours refer to larger error. Some kind of mistakes are penalised differently from other errors. The value at row y and column x says how the method is penalised if it estimates y while the correct result is x.

To accelerate the evaluation, we can pre-calculate the loss matrix for all combinations of two oriented places. The figure 4.2 is a visualisation of the loss matrix. The main diagonal is zero because the planner always makes good decisions when it has the correct starting position. Other than that, there are pairs for which the errorneous estimation of place does not matter as much as other pairs. For example, when the method correctly estimates the orientation but the estimated place is one aisle off from the ground-truth place, most of the navigation steps are similar.

The losses are then averaged over each evaluated expedition. We plot a graph of the loss with respect to the evaluated expedition.

Chapter 5

Implementation

I decided to write the benchmark in Python 3 [65], using the libraries NumPy [66] and SciPy [67] because they are easy to work with and therefore they help in eliminating many implementation errors.

For detection and description of SIFT, RootSIFT, U-SIFT, SURF, U-SURF and BRIEF features, I used the existing implementation in the OpenCV library [68]. There is no direct support for the U-SIFT features in the OpenCV library [68], so I used a workaround copied from the Image Matching Benchmark [57]

I implemented FreMEn on my own. I tested the results against the implementation [69] in Haskell, which was earlier tested against implementation [70] in C++. All these implementations seem to be equivalent. To test the performance of GIST, I used an existing Python 3 wrapper [71] for an implementation [72] written in C. I used the library imageio [73] to load the images.

The paper presenting ToDayGAN also contains a link to publicly available code together with the neural network pre-trained on the Oxford RobotCar dataset. It is implemented in Python using the PyToch library [74]. I used this implementation to transform photos from the night expeditions of Čestlice dataset (namely expeditions 11, 13, 14, 16, 18, 19, 23, 24, 27 and 30). However, the output size is smaller than the original size. Therefore, I rescaled the transformed photos to their original size using bilinear interpolation [75] by Imagemagick [76]. I used these transformed photos instead of the original photos where applicable.

The paper presenting NetVlad also contains a link to a publicly available implementation in Matlab using the MatConvNet library [77] together with a set of pre-trained neural networks which calculate image descriptors. I downloaded the model which they call the best model (VGG-16 [78] + NetVLAD + whitening, trained on Pittsburgh). However, I did not have enough memory to run this model, so I downloaded the model AlexNet [45] + NetVLAD + whitening (250 MB) trained on Pitts30k. I pre-calculated all descriptors using the Matlab code and used them in my Python framework.

I did not find any implementation of the ConvNet Landmarks algorithm but I found implementations of all its non-trivial building blocks in Python. The Edge Box [44] algorithm is present in the OpenCV library [68], pre-trained AlexNet neural network [45] is available for the PyTorch library [74] and Gausssian Random Projection [47] is available in the Scikit-Learn library [79]. I set the number of extracted landmarks to 50 and landmark descriptor dimension to 512. The extracted landmarks have to be resized to the size 231×231 before they are put into the neural network. I did that using the bilinear interpolation [75] from the library Scikit-Image [80].

The paper presenting the image alignment using fully convolutional siamese neural network [6] also offer a working implementation for the PyTorch library [74] together with a pre-trained neural network. The neural network was trained on the Nordland and EU Car datasets.

The authors of GRIEF also offer an implementation in C++. However, I did not manage to make it work. At first, I had to make several changes in the code to make it compile at least. When I launched the program as it was described in the documentation, it crashed with a segmentation fault. I could implement it on my own in theory but the limited time budget for this thesis does not allow me to do so. Therefore, I decided to skip this method.

I found an implementation of LIFT [81] for the TensorFlow library [82]. There is also a pre-trained model of the neural network. However, it was written for a version of TensorFlow which does not seem to be compatible with the current one and the code is no longer maintained. I tried downgrading the TensorFlow library to a version which could work but the code was throwing different errors. Therefore, I declared the implementation as unusable. I did no find any other implementation of LIFT. It could be possible to implement and train the network on my own but training neural networks is beyond the scope of this thesis.

The authors of SuperPoint also offer an implementation in python using the PyTorch library. Unlike the implementation of LIFT, I managed to run

this implementation without problems. It also offers a pre-trained model trained on the MS COCO dataset [83].

All calculated descriptors are being cached for later use. It accelerates the computation because the descriptors calculated for evaluating one expedition can be later used for composition of the database for another expedition. I used the library Matplotlib [84] to plot the graphs.

5.1 Implementation of Bag-of-Words and VLAD

The methods such as Bag-of-Words or VLAD require a composition of a vocabulary of visual words [85]. This step is too time-demanding, so I decided to create a vocabulary in advance. That must be done on a different dataset, so I composed a set of photos from other datasets for visual localisation. I prioritised the datasets which are free and easy to download. Datasets such as NCLT, Oxford RobotCar and EU Car are free to download but they can be downloaded only per chunks of size between tens and hundreds of gigabytes. For this purpose, I need to download a small sample across all these chunks. Though, I managed to find a meaningful sample of the Oxford RobotCar dataset suitable for this task. I composed a dataset from 8139 randomly chosen photos from the CMU Seasons dataset, 2634 night photos chosen from the Oxford RobotCar dataset and 2464 photos from the Aachen dataset, containing both photos captured in day and night.

Preliminary experiments show that assembling a vocabulary from another bigger database yields better result than assembling the vocabulary from a small database. It is also a common practice in the FAB-MAP method, which cannot work without any already-built vocabulary.

I decided to cluster the extracted features into 1000 visual words [85] for the Bag of Words and 64 visual words for VLAD. The dimension of the extracted features is between 32 (e.g. BRIEF) and 128 (e.g. SIFT). It is clear then that it is not needed to employ all the dimensions to cluster the features and a dimension around 10 should be sufficient because $\sqrt[10]{1000} \approx 1.995262315$. Therefore, I decided to reduce the dimension to 10 using principal component analysis [86]. However, the set of vectors is too huge to fit them into a matrix and then calculating an SVD decomposition [87] of the matrix. Namely, we need to compute the matrix V in SVD decomposition of the matrix $M \ominus t = USV^T$ where M is a matrix of rows where each row corresponds to one feature extracted from the images and $\ominus t$ is row-wise subtraction of row

vector t, which is the mean of all rows in M. We can obtain the matrix V from the eigenvector decomposition [88] of the matrix $(M \ominus t)^T(M \ominus t) = V\Lambda V^T$. The matrix $(M \ominus t)^T(M \ominus t) = C$ is relatively small and can be calculated online by a single pass of all the images using these formulae:

$$t = \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{F_i}(M_i)_j$$

$$B = \frac{1}{N}\sum_{i=1}^{n}M_i^T M_i$$

$$C = N(B - t^T t) \quad N = \sum_{i=1}^{n} F_i$$

where M_i is the part of matrix M which corresponds to the i-th image, F_i is the count of features extracted from the i-th image, $(M_i)_j$ is the j-th row in the matrix M_i, n is the count of all images and N is the total number of features extracted from all images. The scale is not important for the principal component analysis [86], so we can drop the multiplication by N in the final step.

Clustering that many points would be still too much time and resourse demanding, so I used the algorithm BICO [89] to reduce the amount of points. It works online, so it does not need too much memory. For the algorithm BICO, I set the number of projections to 15 and number of clusters to 10^4 for Bag-of-Words and 640 for VLAD (i.e. 10 times more than the number of visual words we need). The clustering works in two passes. In the first pass, the matrix for the dimension reduction is calculated. And in the second pass, the dimension of the features are reduced and passed into the BICO algorithm. Although there exists an implementation in Python [90], it turns to be very slow, so I decided to do this step in C++. I used an existing implementation of BICO for C++ [91] which also contains the implementation of the k-means algorithm [48] together with the k-means++ initialisation [92]. The library OpenCV [68] is available for C++ also and the workarounds used to attain the variants of the used descriptors can be easily ported to C++. For operation with matrices, I used the library Eigen [93].

The implementation of DenseVLAD is problematic because the paper offers contradictive information on what the dense sampling means. It says that the key points are sampled in a regular grid but the key points in the illustration certainly do not come from a regular grid. The paper does not even offer any working implementation which would clarify what the authors meant. The benchmark [58] contains an implementation of DenseVLAD but the link pointing there is broken. After all, it turns out that the method DenseVLAD is just VLAD with specific settings. Therefore, we will test only the method VLAD with different kinds of descriptors, including RootSIFT, which is used

in DenseVLAD. The readers of this thesis may decide on their own whether this sampling is dense enough to call it DenseVLAD.

5.2 Implementation of FAB-MAP

The implementation of FAB-MAP used in the paper [3] was published. There is also an open-source implementation called openFABMAP [94]. However, they do not fit our scenario. So I decided to implement FAB-MAP on my own.

To describe our implementation of FAB-MAP, let us explain its basic principes first. There is a set of observations $\mathcal{Z}^k = \{Z_1, Z_2, Z_3, \ldots, Z_k\}$, where the current obsservation Z_k consists of observations of different words $Z_k = \{z_1, z_2, z_3, \ldots, z_n\}$ in the visual vocabulary. In FAB-MAP, we say that we either observe the word at least once or we do not observe it in the photo. If we observe multiple instances of the word in the photo, we do not utilise the information of how many instances we observe. There is a set of places $\mathcal{L}^k = \{L_1, L_2, L_3, \ldots, L_{n_k}\}$ known after k observations. There is a set of word occurences $e_1, e_2, e_3, \ldots, e_n$ which say that a given visual word occurs at a place. The word maay occur at a place but we may not observe it. And vice-versa, the word may not occur at a place and we may obsserve it anyway.

Now let us describe the probabilistic model used in FAB-MAP. The words randomly occur at the places. $P(e_i|L_j)$ denotes the probability that word i occurs at place j. We assume that the random variables $e_1, e_2, e_3, \ldots, e_n$ are conditionally independent given L_j. We may or may not observe a visual word. $P(z_i|e_i, L_j)$ denotes the probability that we observe word i given that word i occurs there and we are at place j. We assume that z_i is conditionally independent on the place given e_i, so we can write $P(z_i|e_i, L_j) = P(z_i|e_i)$. We also assume that z_i is independent on e_j if $i \neq j$, meaning that the occurence of one word can not influence the observation of another word. And we assume that $P(z_i|e_i)$ is equal for each $i \in \{1, 2, 3, \ldots, n\}$, so the probability distribution is given by two values: false positive probability $P(z_i = 1|e_i = 0)$ and false negative probability $P(z_i = 0|e_i = 1)$. The paper [3] sets these parameters to $P(z_i = 1|e_i = 0) := 0$ and $P(z_i = 0|e_i = 1) := 0.39$. The probability distribution of Z_k is modelled by a Chow-Liu tree [51]. It is a Bayesian network with a root node z_1 and each other node z_q has its parent node z_{p_q}.

The probability that we observe place i is given by:

$$P(L_i|\mathcal{Z}^k) = \frac{P(Z_k|L_i, \mathcal{Z}^{k-1})P(L_i|\mathcal{Z}^{k-1})}{P(Z_k|\mathcal{Z}^{k-1})}$$

where $P(Z_k|L_i, \mathcal{Z}^{k-1})$ is the observation likelihood, $P(L_i|\mathcal{Z}^{k-1})$ is our prior belief and $P(Z_k|\mathcal{Z}^{k-1})$ is the normalisation factor. We further assume that Z_k is independent on $\mathcal{Z}^{k-1}$ given L_i. The paper further specifies how $P(Z_k|L_i)$ shall be calculated. It is approximated by the Chow-Liu tree:

$$P(Z_k|L_i) \approx P(z_1|L_i) \prod_{q=2}^{n} P(z_q|z_{p_q}, L_i)$$

The term $P(z_q|z_{p_q}, L_i)$ can be further decomposed as:

$$P(z_q|z_{p_q}, L_i) = \sum_{s\in\{0,1\}} P(z_q|e_q = s, z_{p_q})P(e_q = s|L_i)$$

The terms $P(z_1|L_i)$, $P(z_q|e_q = s, z_{p_q})$ and $P(e_q = s, L_i)$ can be further expressed using $P(z_i)$, $P(z_i|e_i)$, $P(z_q|z_{p_q})$ and $P(e_i|L_j)$, where $P(z_i)$ and $P(z_q|z_{p_q})$ are taken from the Chow-Liu tree, which is learned in advance, $P(z_i|e_i)$ is a parameter of the method and $P(e_i|L_j)$ are learnt place models.

When a place is initialised, the probabilities $P(e_i|L_j)$ are set to $P(z_i)$. If we occur at place L_i, the probabilities are updated to:

$$P(e_i = 1|L_j, \mathcal{Z}^k) = \frac{P(Z_k|e_i = 1, L_j)P(e_i = 1|L_j, \mathcal{Z}^{k-1})}{P(Z_k|L_j)}$$

The paper [3] does not specify how this term shall be evaluated. I looked into the implementation openFABMAP but I failed to find which part of code is responsible for evaluating this term. Using the formulae above, we can approximate this term as:

$$P(e_i = 1|L_j, \mathcal{Z}^k) \approx \frac{P(e_i = 1|L_j)P(z_1|e_i = 1, L_i)\prod_{q=2}^{n} P(z_q|e_i = 1, z_{p_q}, L_i)}{P(z_1|L_i)\prod_{q=2}^{n}\sum_{s\in\{0,1\}} P(z_q|e_q = s, z_{p_q})P(e_q = s|L_i)}$$

Using the independence assumptions, it can be simplified to:

$$P(e_1 = 1|L_j, \mathcal{Z}^k) \approx \frac{P(z_1|e_1 = 1)P(e_1 = 1|L_j)}{P(z_1|e_1 = 1)P(e_1 = 1|L_j) + P(z_1|e_1 = 0)P(e_1 = 0|L_j)}$$

for the root element in the Chow-Liu tree. And for the other elements, it is:

$$P(e_q = 1|L_j, \mathcal{Z}^k) \approx \frac{P(z_i|e_i = 1, z_{p_i})P(e_i = 1|L_j)}{P(z_q|e_q = 1, z_{p_q})P(e_q = 1|L_j) + P(z_q|e_q = 0, z_{p_q})P(e_q = 0|L_j)}$$

That is suspicious because if the false positive probability is zero, then once we observe z_i ot place L_j, we set the probability $P(e_i|L_j)$ to 1.

The paper [3] uses a non-trivial prior belief for testing but we do not have any prior belief in our scenario. So we set it to the uniform distribution over all places. Therefore, the estimated place is given by:

$$\operatorname*{argmax}_{L_i} P(L_i|\mathcal{Z}^k) = \operatorname*{argmax}_{L_i} \frac{P(Z_k|L_i, \mathcal{Z}^{k-1})P(L_i|\mathcal{Z}^{k-1})}{P(Z_k|\mathcal{Z}^{k-1})}$$

Both $P(L_i|\mathcal{Z}^{k-1})$ and $P(Z_k|\mathcal{Z}^{k-1})$ are constant with respect to place, so it holds that:

$$\operatorname*{argmax}_{L_i} P(L_i|\mathcal{Z}^k) = \operatorname*{argmax}_{L_i} P(Z_k|L_i) = \operatorname*{argmax}_{L_i} \ln P(Z_k|L_i)$$

The term $P(Z_k|L_i, \mathcal{Z}^{k-1})$ is calculated as a long product, so it is better to calculate its logarithm instead to attain better numerical stability.

FAB-MAP is supposed to localise on-line, meaning that it starts with an initial model, which is improved gradually. In our scenario, the localisation method is given a set of data in advance. We combine those approaches so that our implementation of FAB-MAP starts with the initial model and then it is updated by all images from the dataset. In this step, the ground-truth locations are given, so FAB-MAP does not need to estimate the location of the given dataset images. After that, FAB-MAP is given test images one by one and the locations are estimated from the obtained model. The model is not updated in this step, otherwise the order of tested images would matter. FAB-MAP may also calculate the probability that a new place was visited. We assume that we know all places in advance, so we do not calculate this term in this thesis.

We use the same visual word database as in the implementation of Bag-of-Words and VLAD. FAB-MAP was tested on a visual word database containing 11000 visual words, so most of the visual words were contained at most once in the images. However, we use a smaller visual word database, containing only 1000 words, and it turns out that most of the words occur at least once in the images. So we limited feature extraction in a way so at most 100 features are extracted from each image. If there are more features in the image, we choose the most significant ones. We train the Chow-Liu trees on the same dataset as the visual word database.

Chapter 6

Experiments

We run the implemented methods on the Čestlice dataset and calculate the average loss for each of the method. We calculate both zero-one loss and first-step loss. For each expedition, we feed the methods by all the data from the preceding expeditions and use the current expedition as query. We observe how the losses change with respect to the changes in the environment. The background colour in the graphs correspond to the mean colour of whole the expedition, so columns with light colours correspond with day expedition and columns with dark colours correspond to night expeditions.

6.1 Image Retrieval Results

The feature-based approaches can be applied to any kind of features. Plotting all the combinations into one plot would result in an unreadable plot. Therefore, we offer multiple views to the obtained results. In this chapter, there is only an overview in figure 6.1, where we selected one of the best features for each approach and all approaches which do not use any features. The other plots are presented in appendix B.

Both Bag-of-Words and VLAD estimate the location with reasonable accuracy. The accuracy is different for each feature extraction algorithm. For Bag-of-Words, the SURF and U-SIFT features seem to be the best. For VLAD, the U-SIFT features seem to be the best. Both Bag-of-Words and VLAD approaches perform similarily in our setting.

Surprisingly, the methods developed on top of Bag-of-Words and VLAD in order to deal with changing environments have significantly worse performance than the methods Bag-of-Words and VLAD on their own. In FAB-MAP, there are more potential causes. It can be because the default settings cannot be used in our scenario or because we wrongly update the models, as discussed in the section 5.2. FreMEn creates a constant model after the first expedition, so the performance is the same as without FreMEn. However, after more expeditions, FreMEn creates a more complicated model. It may seem that it could help increasing the performance in the way that it filters out the irrelevant expeditions. For examplle, the localiser would not localise based on data obtained in day when it is night at that moment. However, it turns out that simplifying the data this way actually harms. Surprisingly, not even ToDayGAN works. It does transform the night photos to day photos but it introduces so many additional artifacts so it is aactually harder to localise using such photos. An example output of ToDayGAN is shown in the figure 6.2. The possible cause is that Oxford Robotcar dataset and Čestlice dataset are too different so it is not possible to transfer the network from one dataset to another without re-training it. Therefore, it turns out that the best temporal model is keeping a sequence of descriptors for each place and then localising using whole the sequence.

GIST seems to fail completely in Čestlice dataset, its accuracy is about the same as random guessing. It seems that GIST always chooses a small subset of places and then most of the guesses are in this small subset. The possible cause is that GIST describes how the photo looks overall. And indeed, all photos look similarily overall, they are all photos of a car park.

The methods NetVLAD and ConvNet Landmarks perform significantly better than Bag-of-Words and VLAD. It can be seen that almost all methods have a significant rise of error at expedition 11, which is the first night expedition. The only exceptions are the methods which perform badly overall. NetVLAD and ConvNet Landdmarks have also a rise of error here but not as significant as in the other methods. NetVLAD seem to perform better than ConvNet Landmarks overall but when an surprising event comes, they perform similarly.

NetVLAD turns out to be the best method for localisation in our benchmark, GIST turns out to be the worst. From the methods which do not involve any neural networks, VLAD with U-SIFT features and Bag-of-Words with SURF features turn out to be the best methods in our benchmark.

It seems that the errors for both zero-one loss and first-step loss highly correlate and the comparison of different methods differ only very little on which loss we choose. The first-step loss is smaller but that is most likely

caused by that the decision space is also smaller, so a random guess is more likely to hit the correct solution. From this observation, it seems that none of the methods exploit the fact that the exact position is not always needed to make the correct decision.

6.2 Image Alignment Results

The results of the image alignment benchmark are shown in the figure 6.3. All tested approaches led to an alignment which is significantly better than estimating constant zero displacement.

ToDayGAN failed completely in this task also. Similarly as in the previous task, ToDayGAN transformed the photos to day photos but it introduced so many additional artefacts which confused the alignment methods. The result with ToDayGAN was always significantly worse than without ToDayGAN. Therefore, ToDayGAN does not deserve any additional comments in this section.

The performance of tested methods can be sorted linearly in this order:

1. Fully-convolutional Siamese Neural Network (FCN)
2. SuperPoint
3. U-SURF
4. U-SIFT
5. SURF
6. RootSIFT
7. SIFT
8. BRIEF
9. ConvNet Landmarks

ConvNet Landmarks turns to be good for image retrieval but the area features are too inaccurate for image alignment. The upright variants of SIFT and SURF features seem to perform better than standard SIFT and SURF

because the roll of the camera changes only slightly and therefore the rotation of features is a valuable information in feature matching. Standard SIFT and SURF discard the information about feature rotation and therefore it cannot be used for matching. The approaches using neural networks turn out to be the best in this scenario. The fully-convolutional siamese neural network is better than SuperPoint possibly because it was trained solely for estimating the horizontal displacement, while SuperPoint was trained generally for image matching.

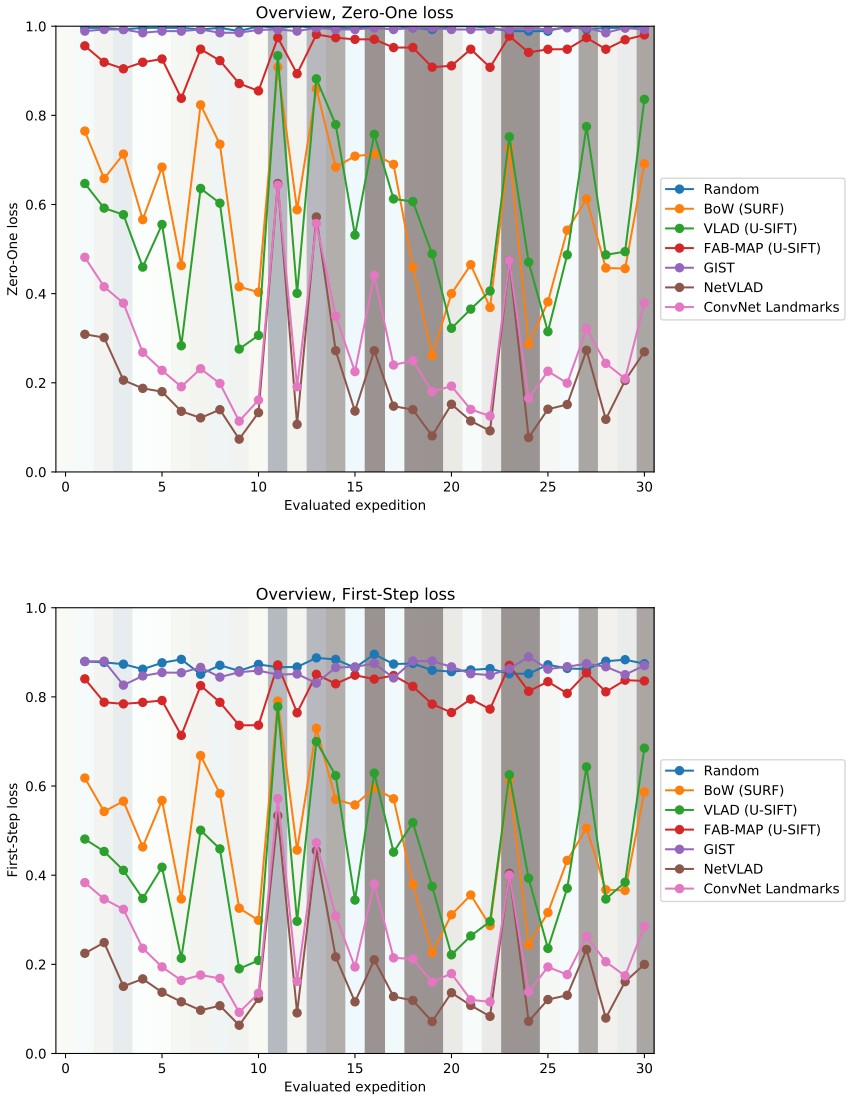


Figure 6.1: Zero-one and First-step loss of Bag-of-Words with SURF features, VLAD with U-SIFT features, Gist, NetVLAD and ConvNet Landmarks.

Figure 6.2: Performance of ToDayGAN. Original photo is on the left, photo transformed to day is on the right.

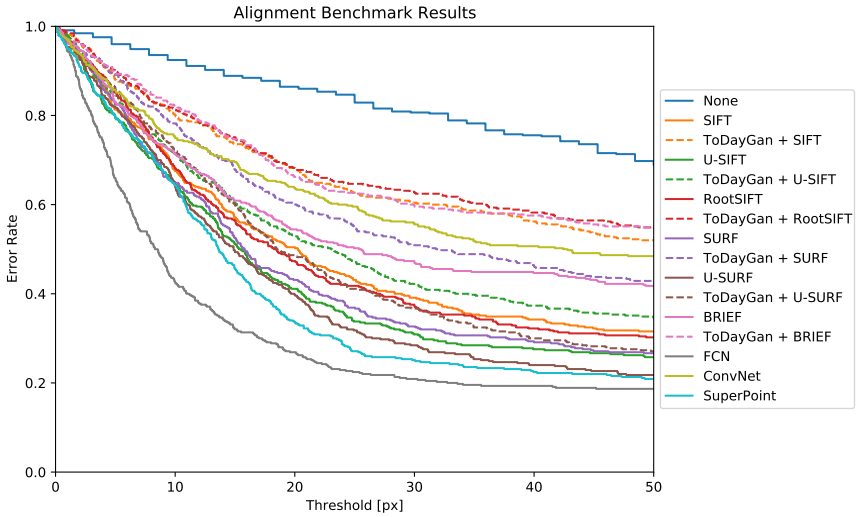


Figure 6.3: Result of the image alignment methods. The plot shows how many pairs of photos are aligned wrongly with respect to chosen threshold.

Chapter 7

Conclusion

We tested various methods for image localisation and navigation in changing environments. For visual localisation, the methods NetVLAD and ConvNet Landmarks turn out to be the best, feature-based approaches such as Bag-of-Words and VLAD are worse and GIST is unusable in our scenario.

What is surprising, the improvements to the feature-based approaches which are supposed to deal with changing environments, such as ToDayGAN, FreMEn and FAB-MAP, do not work at all. All of them actually seem to decrease the performance. The best model to tackle the changes in the environment seems to be keeping a sequence of photos for each place and then finding matches across all the photos.

For FreMEn, it is not that surprising that the performance is worse because FreMEn is a method for compressing the time sequences and therefore it is expected that using the original sequence yields better results. However, the compression ratio is very small in this scenario, so it is not worth it. The result seems to be slightly better than FAB-MAP, which denotes that the temporal modelling in FreMEn improves the performance.

ToDayGAN seems to be sensible on switching scenarios. It was trained on the Oxford RobotCar dataset and it performed poorly on the Čestlice dataset. The night photos were transformed to day photos but it introduced so many additional artefacts which confused the methods used on top of ToDayGAN. The poor performance of ToDayGAN on the Čestlice dataset caused that the methods using photos pre-processed by ToDayGAN performed significantly worse in both image retrieval and image alignment than the same methods

without ToDayGAN pre-processing.

There are several possible causes why FAB-MAP failed. One possibility is simply that FAB-MAP is not designed for long-term visual localisation and it expects that the scene did not vary that much since it was visited last time. It is also possible that the settings provided in [3] cannot be used for our scenario for a reason. However, the paper [3] does not specify how the settings shall be chosen, so we assume that the provided settings can be used in any scenario to get a meaningful result. Another possibility is that we wrongly update the probability model because the paper [3] does not specify how one particular term shall be evaluated, as it was discussed in section 5.2.

The failure of GIST is possibly caused by that GIST describes the image by its overall look and all the places look similarly overall. They all show a car park. The streets in the car park look very similar, they vary only in details. In such scenario, GIST seems to choose a few distinguishable representative photos from the dataset and then most of the predictions are one of these photos. It is usually wrong, so GIST performs about the same as random guessing.

The results for both zero-one loss and first-step loss were similar. The order of different methods varies only slightly in some expeditions. That does not mean that it would be pointless to try both of these losses, it means that none of the tested methods exploit the fact that the exact position is not always needed to take the correct decision. Or at least, they exploit it nearly equally. Therefore, it opens the question whether there could be a method which actually exploits this fact. This could be possibly attained by providing some kind of information about the world structure or assigning weights to features based on how much they are important for navigation.

The surprise for all the methods was the expedition 11, which was the first night expedition in the dataset. In this expedition, the error of all methods skyrocketed except the methods which performed badly overall. The other nights were not as surprising because the methods already had at least one night photo in the database already. There are visible peaks at the night expeditions though. The expedition 11 was a surprise for the deep-learning approaches, namely NetVLAD and ConvNet Landmarks, also but they still performed significantly better than other approaches.

For image alignment, the best method seems to be image alignment using fully-convolutional siamese neural network, then SuperPoint, then upright variants of SURF and SIFT, then standard features in this order: SURF, RootSift, SIFT, BRIEF and ConvNet Landmarks was the worst. All these

methods turned out to be significantly better than predicting constant zero displacement. ToDayGAN worsened the quality of all methods in which it was used.

ConvNet Landmarks is a good method for image retrieval because it seems to be capable of distinguishing various objects in different conditions. The position of the object is not required for image retrieval, so it does not matter that it is inaccurate. However, this inaccuracy matters a lot in image alignment, which is the possible cause of that it performed badly there.

The upright variants of SIFT and SURF features preformed better than the standard SIFT and SURF features. This is caused most likely by that the camera roll angle changes only slightly and therefore the feature rotation can be used to distinguish the key points. This is possibly the case of all road vehicles because the roll angle of camera is given by the inclination of the road, which usually does not change.

The fully-convolutional siamese neural network turned out to be better than SuperPoint. The possible cause it that the fully-convolutional siamese neural network was trained solely for the purpose of estimating the horizontal displacement of two image pairs while the SuperPoint neural network was trained generally for matching images of the scene from any two viewpoints.

Bibliography

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, Cambridge, Mass., 2005.

[2] David M Rosen, Julian Mason, and John J Leonard. Towards lifelong feature-based mapping in semi-static environments. In *RSS 2015 Workshop on the problem of mobile sensors*, 2015.

[3] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[4] Tomáš Krajník, Jaime Pulido Fentanes, Jo ao Santos, Keerthy Kusumam, and Tom Duckett. FreMEn: Frequency Map Enhancement for Long-Term. *Mobile Robot Autonomy in Changing Environments*, 2015. In review.

[5] Asha Anoosheh, Torsten Sattler, Radu Timofte, Marc Pollefeys, and Luc Van Gool. Night-to-day image translation for retrieval-based localization. *CoRR*, abs/1809.09767, 2018.

[6] Zdeněk Rozsypálek, George Broughton, Pavel Linder, Tomáš Rouček, Jan Blaha, Leonard Mentzl, Keerthy Kusumam, and Tomáš Krajník. Contrastive learning for image registration in visual teach and repeat navigation. *Sensors*, 22(8), 2022.

[7] Josef Sivic and Andrew Zisserman. Video google: a text retrieval approach to object matching in videos. *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2, 2003.

[8] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Perez. Aggregating local descriptors into a compact image representation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304 – 3311, 07 2010.

[9] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(9):1744–1756, sep 2017.

[10] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. *CoRR*, abs/1812.03506, 2018.

[11] Tianxin Shi, Shuhan Shen, Xiang Gao, and Lingjie Zhu. Visual localization using sparse semantic 3d map. *CoRR*, abs/1904.03803, 2019.

[12] Erik Stenborg, Carl Toft, and Lars Hammarstrand. Long-term visual localization using semantically segmented images. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6484–6490, 2018.

[13] Michael J. Milford and Gordon. F. Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE International Conference on Robotics and Automation*, pages 1643–1649, 2012.

[14] Arren Glover, William Maddern, Michael Milford, and Gordon Wyeth. Fab-map + ratslam: Appearance-based slam for multiple times of day. In *2010 IEEE International Conference on Robotics and Automation in Anchorage, Alaska*, pages 3507 – 3512, 06 2010.

[15] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomáš Pajdla. Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] Hernan Badino, Daniel Huber, and Takeo Kanade. The CMU Visual Localization Data Set. `http://3dvis.ri.cmu.edu/data-sets/localization`, 2011.

[17] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[18] Zhi Yan, Li Sun, Tomáš Krajník, and Yassine Ruichek. Eu long-term dataset with multiple sensors for autonomous driving. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[19] Niko Sünderhauf, Peer Neubert, and Peter Protzel. Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. *Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA)*, page 2013, 01 2013.

[20] Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035, 2016.

[21] Filip Majer, Lucie Halodová, Tomáš Vintr, Martin Dlouhý, Lukáš Merenda, Jaime Fentanes, David Portugal, Micael Couceiro, and Tomáš Krajník. *A Versatile Visual Navigation System for Autonomous Vehicles*, pages 90–110. ResearchGate, 01 2019.

[22] Tomáš Krajník, Pablo Cristóforis, Keerthy Kusumam, Peer Neubert, and Tom Duckett. Image features for visual teach-and-repeat navigation in changing environments. *Robotics and Autonomous Systems*, 2016.

[23] Xun Sun, Yuanfan Xie, Pei Luo, and Liang Wang. A dataset for benchmarking image-based localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[24] Akihiko Torii, Josef Sivic, Masatoshi Okutomi, and Tomáš Pajdla. Visual place recognition with repetitive structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11):2346–2359, 2015.

[25] Zetao Chen, Adam Jacobson, Niko Sünderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian D. Reid, and Michael Milford. Deep learning features at scale for visual place recognition. *CoRR*, abs/1701.05105, 2017.

[26] Yunpeng Li, Noah Snavely, and Daniel P. Huttenlocher. Location recognition using prioritized feature matching. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 791–804, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[27] Yunpeng Li, Noah Snavely, Daniel Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *European Conf. on Computer Vision*, 2012.

[28] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomáš Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[29] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2013.

[30] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *Proceedings of the British Machine Vision Conference*, pages 76.1–76.12. BMVA Press, 2012.

[31] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Convolutional networks for real-time 6-dof camera relocalization. *CoRR*, abs/1505.07427, 2015.

[32] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pages 2599–2606, 06 2009.

[33] Tomáš Krajník, João Santos, Bianca Seemann, and Tom Duckett. Froctomap: An efficient spatio-temporal environment representation. In Michael Mistry, Aleš Leonardis, Mark Witkowski, and Chris Melhuish, editors, *Advances in Autonomous Robotics Systems*, volume 8717 of *Lecture Notes in Computer Science*, pages 281–282. Springer International Publishing, 2014.

[34] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[35] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 467–483, Cham, 2016. Springer International Publishing.

[36] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Deep Learning for Visual SLAM Workshop*, 2018.

[37] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918, 2012.

[38] Karl Pearson and Olaus Magnus Friedrich Erdmann Henrici. X. contributions to the mathematical theory of evolution.—ii. skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London. (A.)*, 186:343–414, 1895.

[39] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 11 2004.

[40] Christopher G. Harris and M. J. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[41] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[42] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[43] Niko Sünderhauf, Sareh Shirazi, Adam Jacobson, Edward Pepperell, Feras Dayoub, Ben Upcroft, and Michael Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. *Proceedings of Robotics: Science and Systems XII*, 07 2015.

[44] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.

[45] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014.

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[47] Sanjoy Dasgupta. Experiments with random projection. *CoRR*, abs/1301.3849, 2013.

[48] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[49] Aude Oliva and Antonio Torralba. Building the gist of a scene: the role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006.

[50] Relja Arandjelovic, Petr Gronát, Akihiko Torii, Tomás Pajdla, and Josef Sivic. Netvlad: CNN architecture for weakly supervised place recognition. *CoRR*, abs/1511.07247, 2015.

[51] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[52] GEORGE ARFKEN. 14 - fourier series. In GEORGE ARFKEN, editor, *Mathematical Methods for Physicists (Third Edition)*, pages 760–793. Academic Press, third edition edition, 1985.

[53] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. Fully-convolutional siamese networks for object tracking. *CoRR*, abs/1606.09549, 2016.

[54] Eric W. Weisstein. Cross-correlation. From MathWorld—A Wolfram Web Resource.

[55] H. Cantzler. Random sample consensus (ransac), 1981.

[56] Carl Toft, Will Maddern, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Tomáš Pajdla, Fredrik Kahl, and Torsten Sattler. Long-term visual localization revisited. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):2074–2088, 2022.

[57] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image Matching across Wide Baselines: From Paper to Practice. *International Journal of Computer Vision*, 2020.

[58] Lars Hammarstrand, Fredrik Kahl, Will Maddern, Tomáš Pajdla Marc Pollefeys, Torsten Sattler, Josef Šivic, Erik Stenborg, Carl Toft, and Akihiko Torii. Long-term visual localisation. `https://www.visuallocalization.net/`, 2020.

[59] Richard Kuhn, Raghu Kacker, and Yu Lei. Practical combinatorial testing. *National Institute of Standards and Technology*, 2010.

[60] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:2000, 2000.

[61] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1 edition, February 1993.

[62] Morton Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220, 1967.

[63] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[64] Claude Sammut and Geoffrey I. Webb, editors. *Zero-One Loss*, pages 1031–1031. Springer US, Boston, MA, 2010.

[65] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[66] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane,

Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[67] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[68] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[69] Leonard Mentzl. Time sequence prediction framework. `https://gitlab.fel.cvut.cz/mentzleo/framework2`, 2020.

[70] Tomáš Krajník, Tomáš Vintr, Sergi Molina Mellado, Jaime Pulido Fentanes, Grzegorz Cielniak, and Tom Duckett. Warped hypertime representations for long-term autonomy of mobile robots. *CoRR*, abs/1810.04285, 2018.

[71] Yuichiro Tachibana. lear-gist-python. `https://github.com/whitphx/lear-gist-python`, 2021.

[72] Matthijs Douze, Hervé Jégou, Sandhawalia Harsimrat, Laurent Amsaleg, and Cordelia Schmid. Evaluation of gist descriptors for web-scale image search. In *CIVR 2009 - International Conference on Image and Video Retrieval*, pages 19:1–8, Santorini, Greece, July 2009. ACM.

[73] Almar Klein, Sebastian Wallkötter, Steven Silvester, Anthony Tanbakuchi, Paul Müller, actions user, Juan Nunez-Iglesias, Mark Harfouche, Antony Lee, Matt McCormick, OrganicIrradiation, Arash Rai, Ariel Ladegaard, Tim D. Smith, Ghislain Vaillant, jackwalker64, Joel Nises, Miloš Komarčević, rreilink, lschr, Dennis, Hugo van Kemenade, Maximilian Schambach, Chris Dusold, DavidKorczynski, Felix Kohlgrüber, Ge Yang, Graham Inggs, Joe Singleton, and Michael. *imageio/imageio: v2.16.1*. Zenodo, February 2022.

[74] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit

Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[75] Alan C. Bovik. Chapter 3 - basic gray level image processing. In Al Bovik, editor, *The Essential Guide to Image Processing*, pages 43–68. Academic Press, Boston, 2009.

[76] The ImageMagick Development Team. Imagemagick. `https://imagemagick.org`, 2021.

[77] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[78] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[79] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[80] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[81] Kwang Moo Yi and Eduard Trulls. Tf-lift: Tensorflow implmentation for learned invariant feature transform. `https://github.com/cvlab-epfl/tf-lift`, 2017.

[82] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[83] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[84] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[85] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 604–610 Vol. 1, 2005.

[86] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[87] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[88] Eric W. Weisstein. Eigen decomposition. From MathWorld—A Wolfram Web Resource.

[89] Hendrik Fichtenberger, Marc Gillé, Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Bico: Birch meets coresets for k-means clustering. In *ESA*, 2013.

[90] gallmerci at GitHub. bico. `https://github.com/gallmerci/bico`, 01 2019.

[91] Department of Computer Science Technical University Dortmund. Bico. `https://ls2-www.cs.tu-dortmund.de/grav/en/bico`, 10 2014.

[92] David Arthur and Sergei Vassilvitskii. K-means++: the advantages of careful seeding. In *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[93] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010.

[94] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth. Openfabmap: An open source toolbox for appearance-based loop closure detection. In *The International Conference on Robotics and Automation*, St Paul, Minnesota, 2011. IEEE.

Appendix A

DVD Content

Folder `benchmark` contains the benchmark written in Python which we use. The script `Benchmark.py` runs the benchmark for image retrieval and the script `AlignmentBenchmark.py` runs the benchmark for image alignment. The root directory of the dataset is passed as argument. The folder `benchmark/vocabularies` contains the vocabularies and Chow-Liu trees for the Bag-of-Words and VLAD approaches. The Chow-Liu trees can be generated using the script `chow_liu.py`. If NetVLAD is tested, the dataset has to contain a folder `netvlad` which contains files with the same names as the names of the photos with the `.mat` extension (The file names therefore end on `.JPG.mat`). If ToDayGAN is tested, the results of ToDayGAN is put into folders `todaygan##`, where `##` stands for the expedition number.

Folder `bow` contains a script in C++ which creates bags of visual words. It is based on the code [91].

Folder `cestlice` contains the complete Čestlice dataset. The folder `cestlice/places` contains files specifying which photos belong to which places. Each line corresponds to one photo. The first column is the file name, the second column is the time stamp and the other columns are the estimated rotation in quaternions. The coordinates of the quaternions are in the order i, j, k, r. The subfolder `registraion` contains the results of registration. For each place, there is a photo created by overlying all photos from given place, the calculated rotations of all photos and the files `angles` contains the results of pairwise registrations. The files `angles` have 6 columns. The first two are the indices of photos, the next four columns are the mutual rotation in quaternions and the last column is the matching score.

Folder `cestlice_alignment` contains a subset of the Čestlice dataset used for benchmarking image alignment. The photos are rotated so they have all the same roll angle. The ground-truth displacements are provided in the files `displacement.txt` in each subfolder.

Folder `netvlad` contains a short script used to generate NetVLAD descriptors of the dataset. The location of the dataset is hardcoded.

Folder `semiauto_alignment` contains a toolkit for semi-automatic annotation of the dataset. The annotation is provided together with the dataset so doing it again is not required for running the benchmark. The python scripts must be ran from the directory in which the dataset is located. The C++ tools can be ran from any place but the location of the dataset is hardcoded in `mainwindow.cpp`. The tools are used in this order: `extract_lines.py` (extract line segments and vanishing points), `vp_selector` (assign vanishing points to directions (manual)), `opravy.py` (list vanishing points in focal plane), `vp_selector_fix` (fix vanishing points in focal plane (manual)), `match_angles_mcf.py` (match photos with world angles), `rotation_test_1.py` (register photos (time demanding)), `align.py` (calculate displacements and set roll to zero).

Folder `thesis` contains the thesis itself in LaTex format together with all figures.

Appendix B

More Plots

The figures B.1, B.2, B.3, B.4, B.5, B.6 and B.7 contain plots for each feature-based approach, where the same approaches with different features are compared. The figures B.8, B.9, B.10, B.11, B.12 and B.13 contain plots for each kind of feature, where different approaches with the same features are compared. There are plots for both zero-one loss and first-step loss.

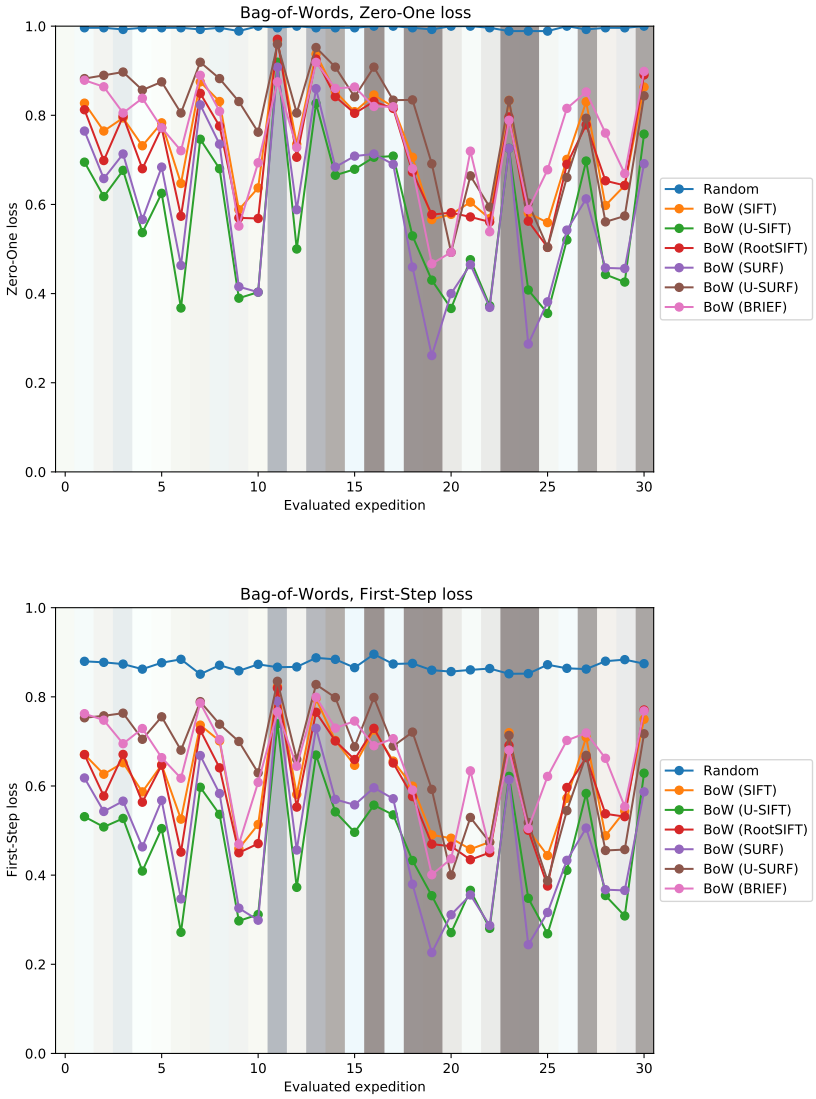


Figure B.1: Zero-one and First-step loss of Bag-of-Words with different features.

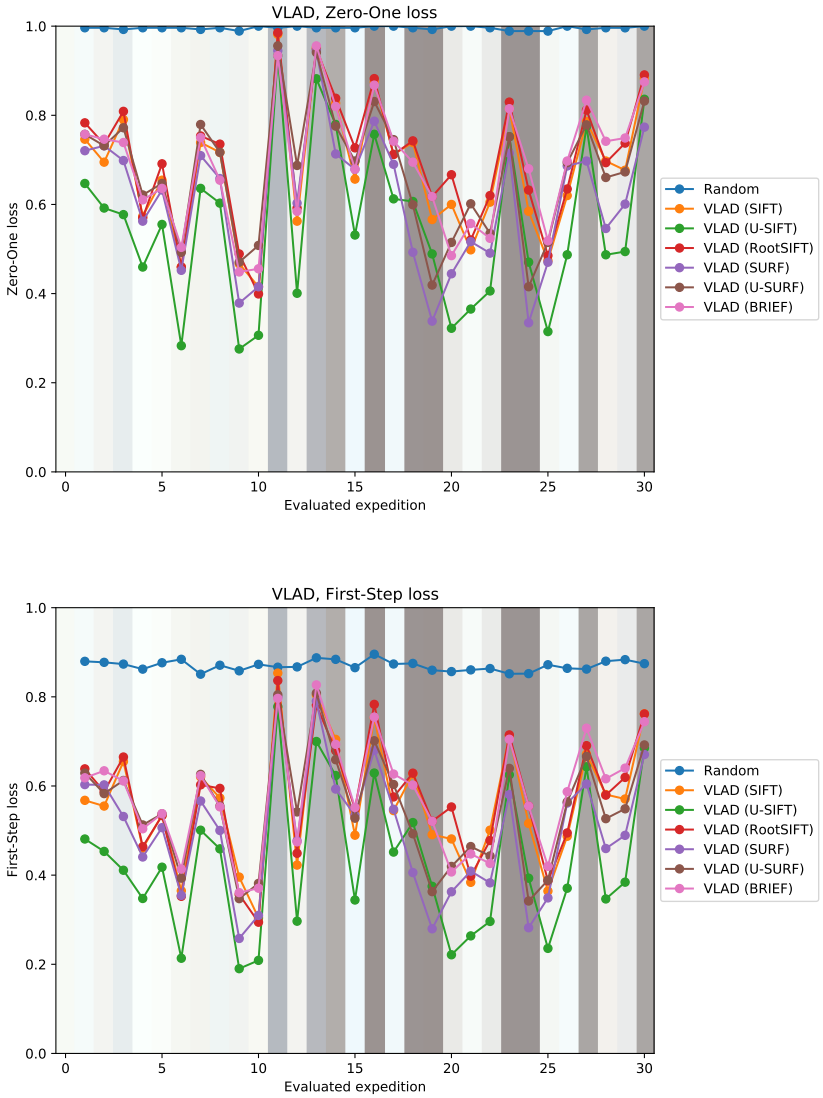


Figure B.2: Zero-one and First-step loss of VLAD with different features.

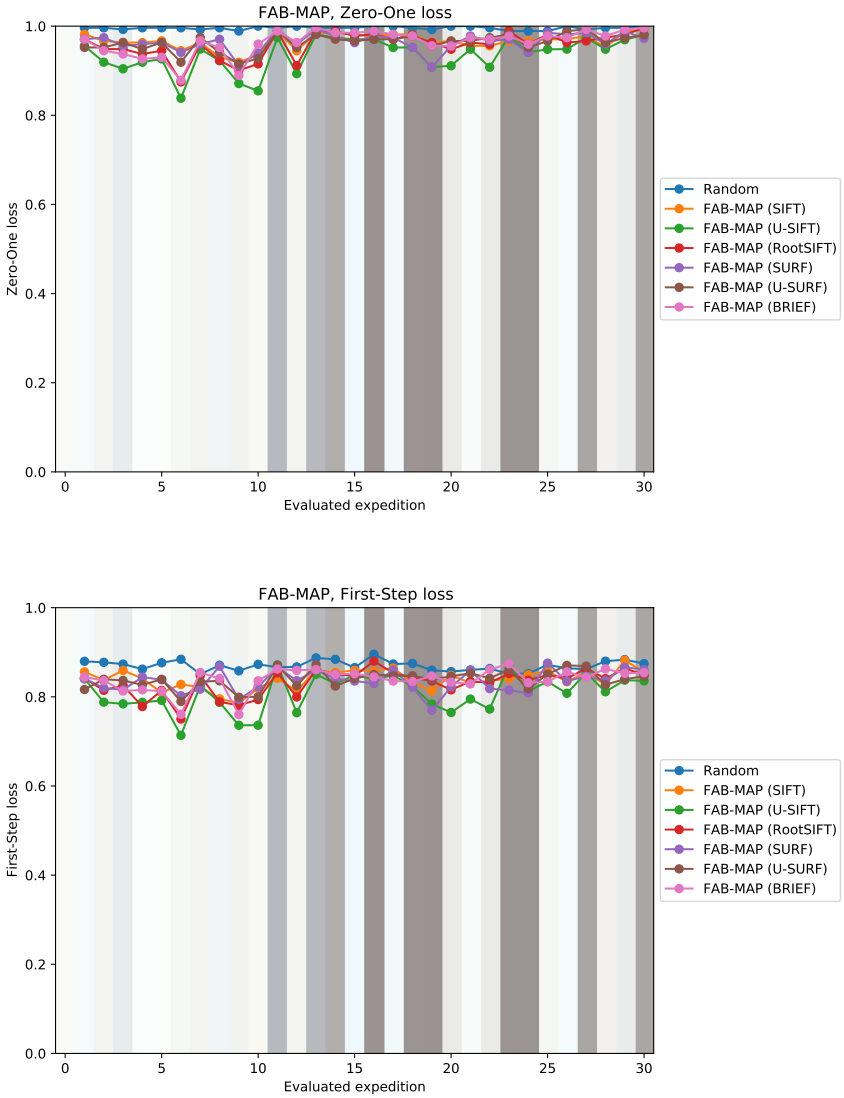


Figure B.3: Zero-one and First-step loss of FAB-MAP with different features.

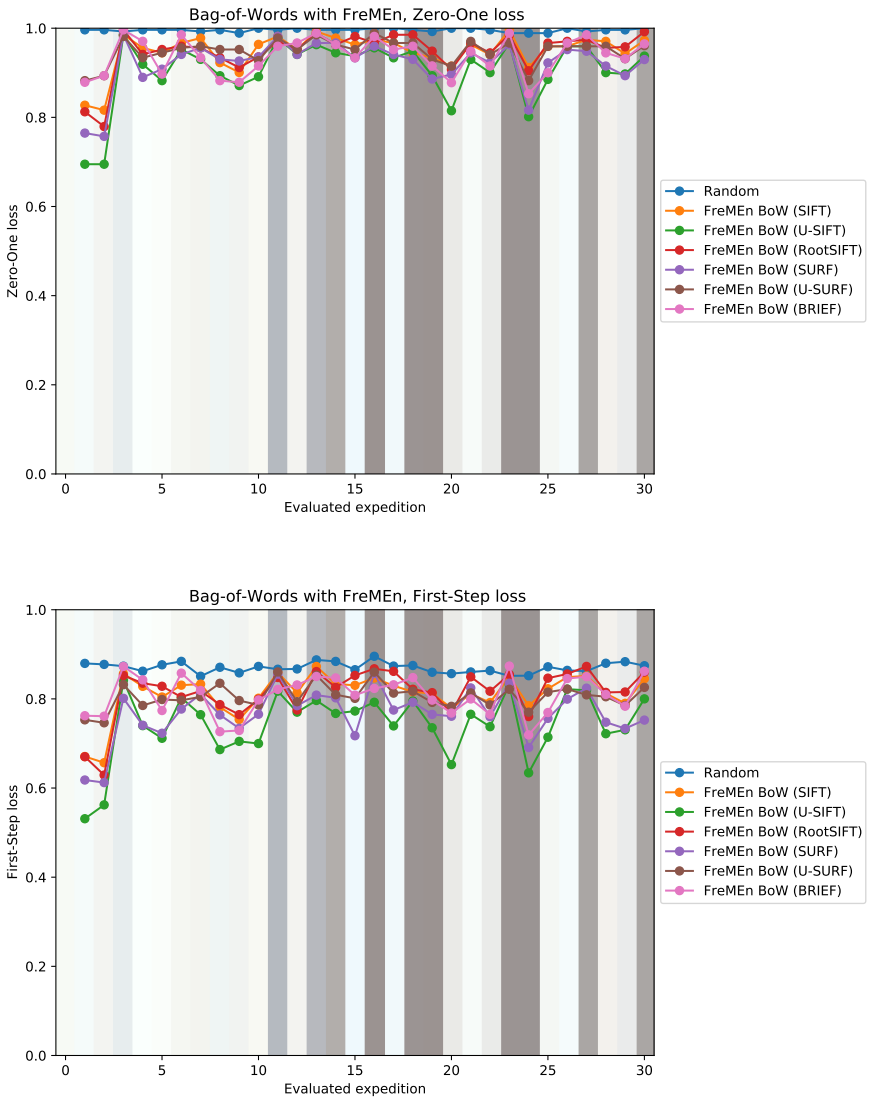


Figure B.4: Zero-one and First-step loss of FreMEn applied to Bag-of-Words with different features.

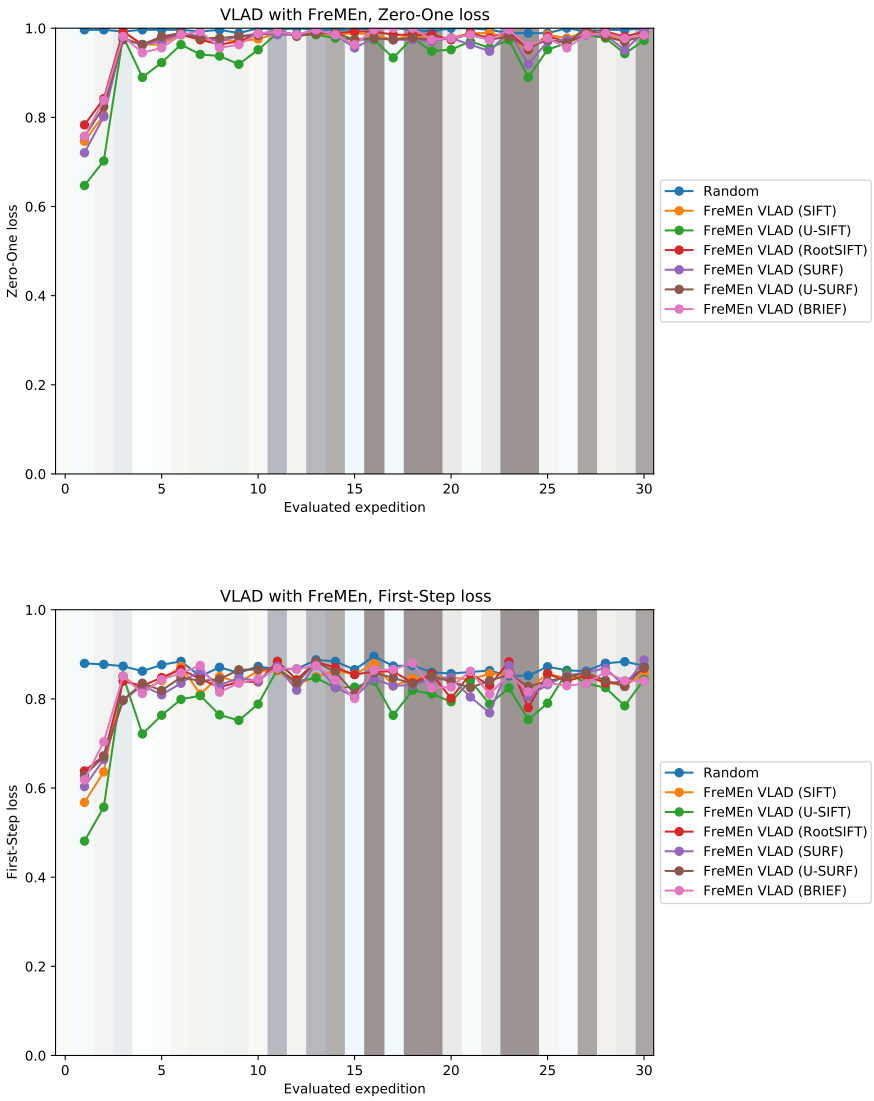


Figure B.5: Zero-one and First-step loss of FreMEn applied to VLAD with different features.

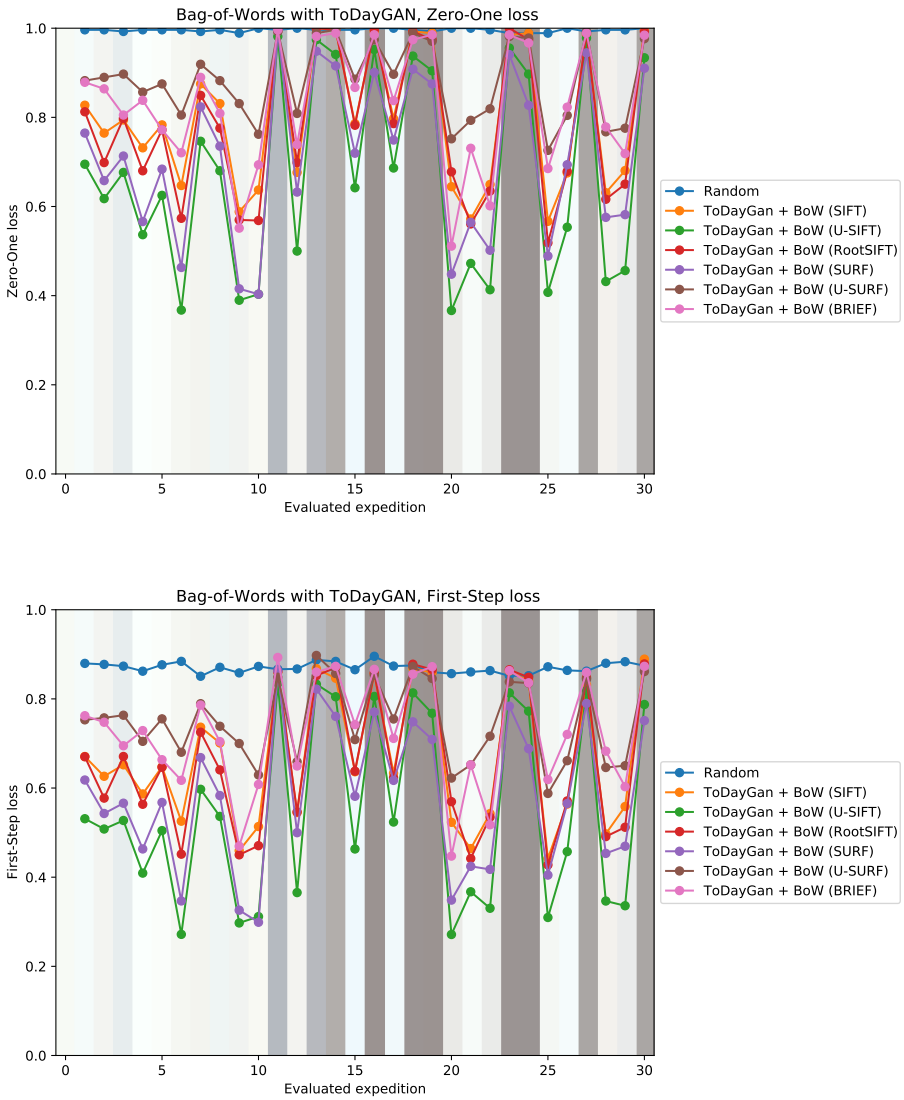


Figure B.6: Zero-one and First-step loss of Bag-of-Words with different features using ToDayGAN.

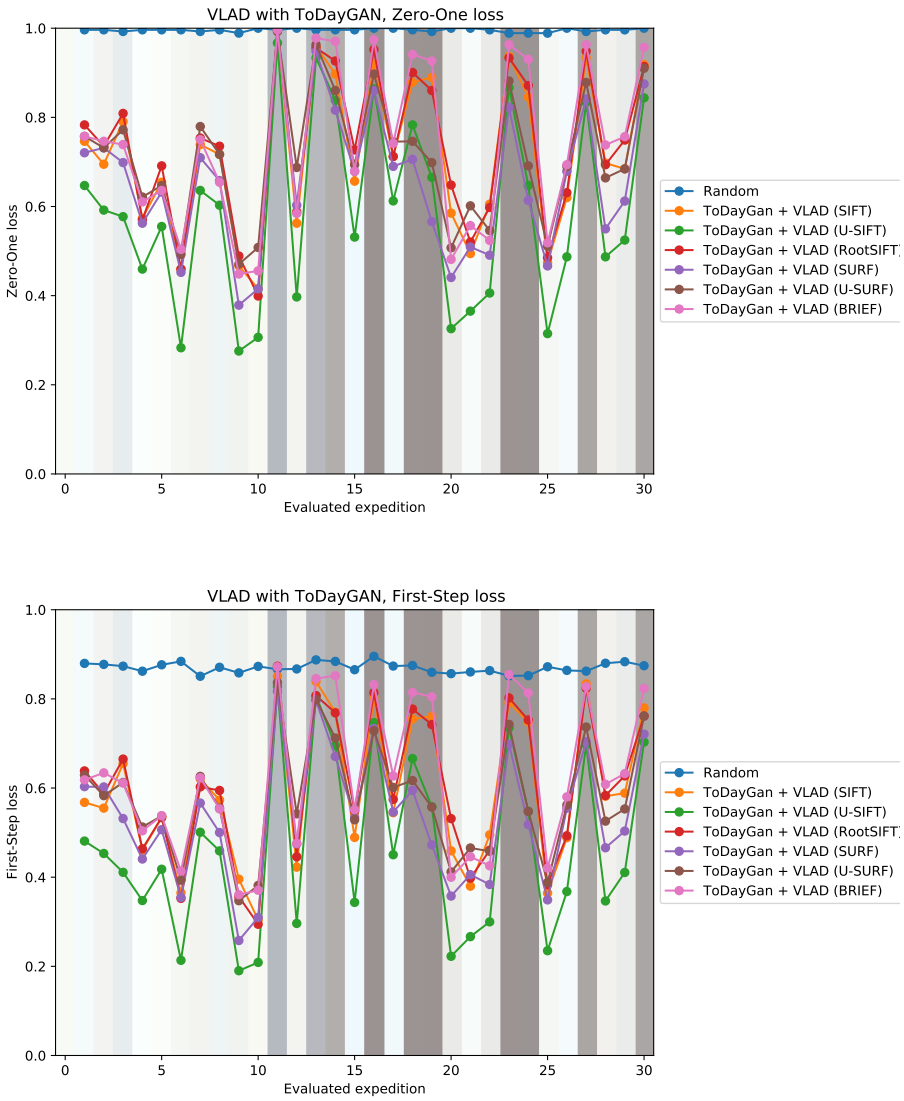


Figure B.7: Zero-one and First-step loss of VLAD with different features using ToDayGAN.

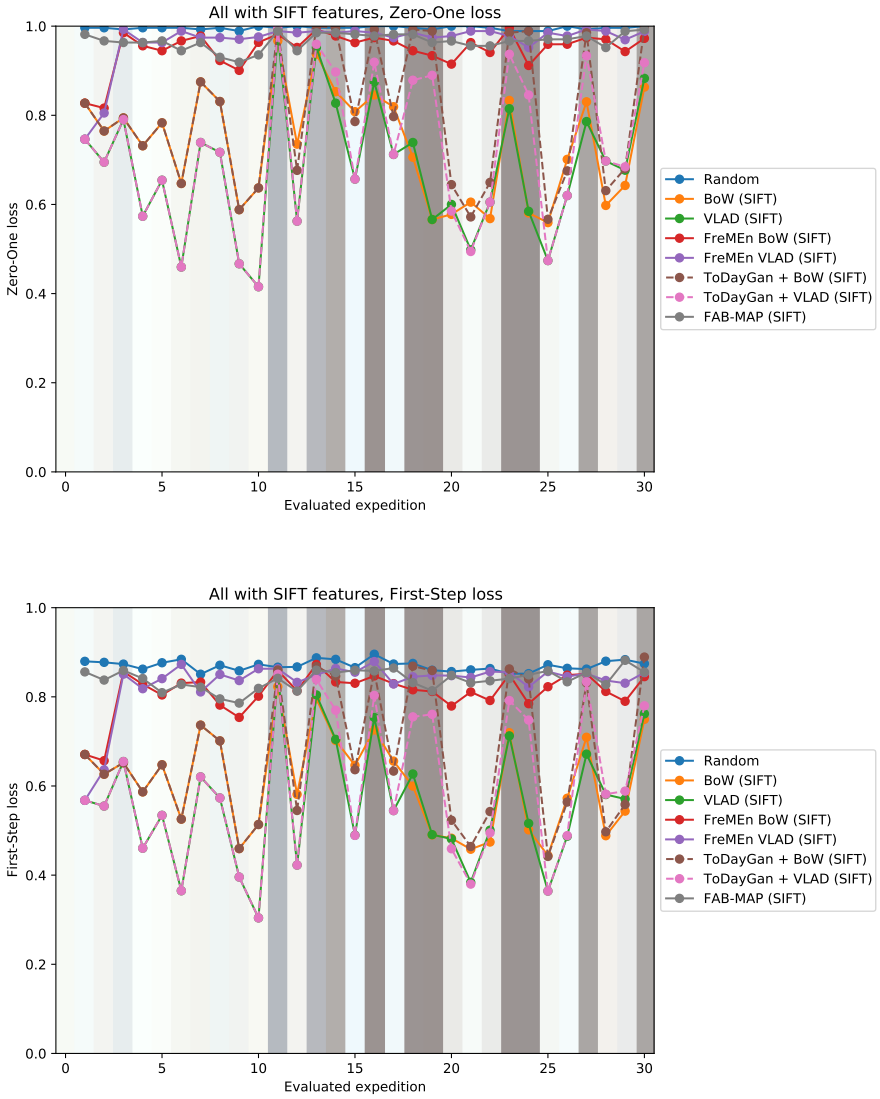


Figure B.8: Zero-one and First-step loss of different methods using SIFT features.

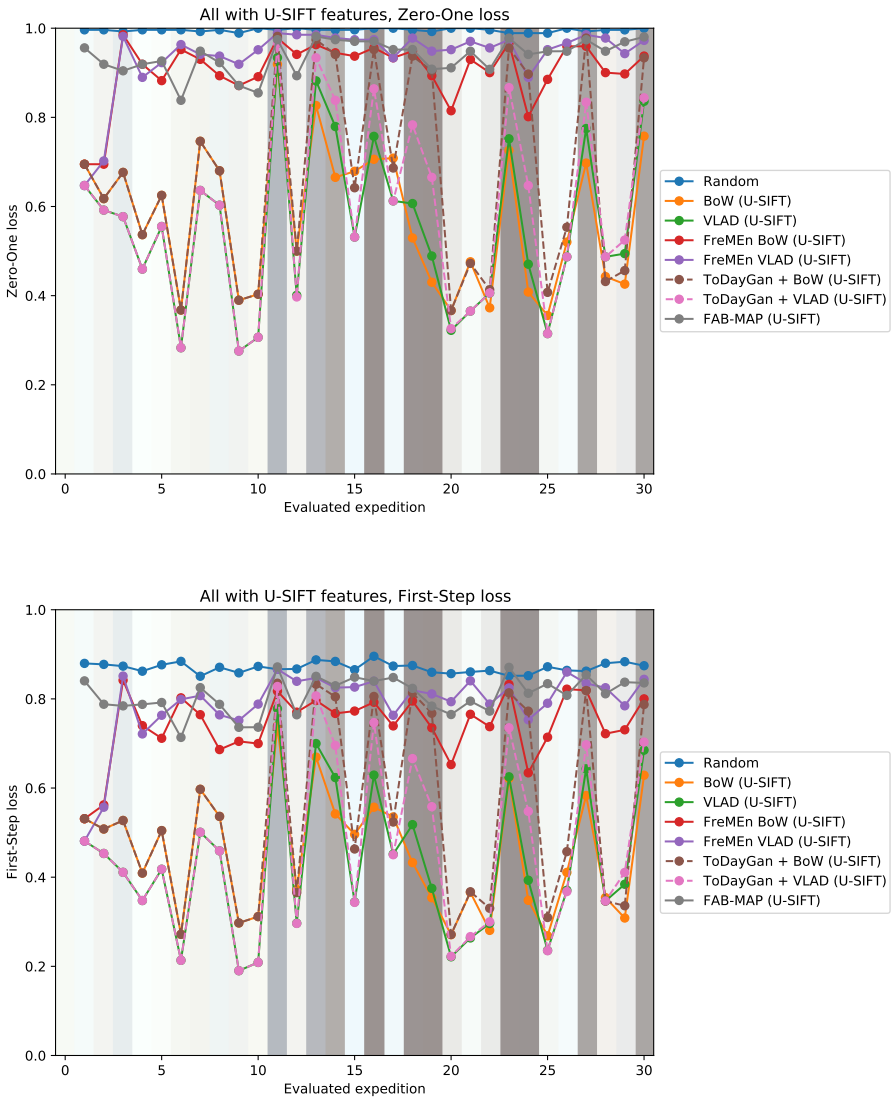


Figure B.9: Zero-one and First-step loss of different methods using U-SIFT features.

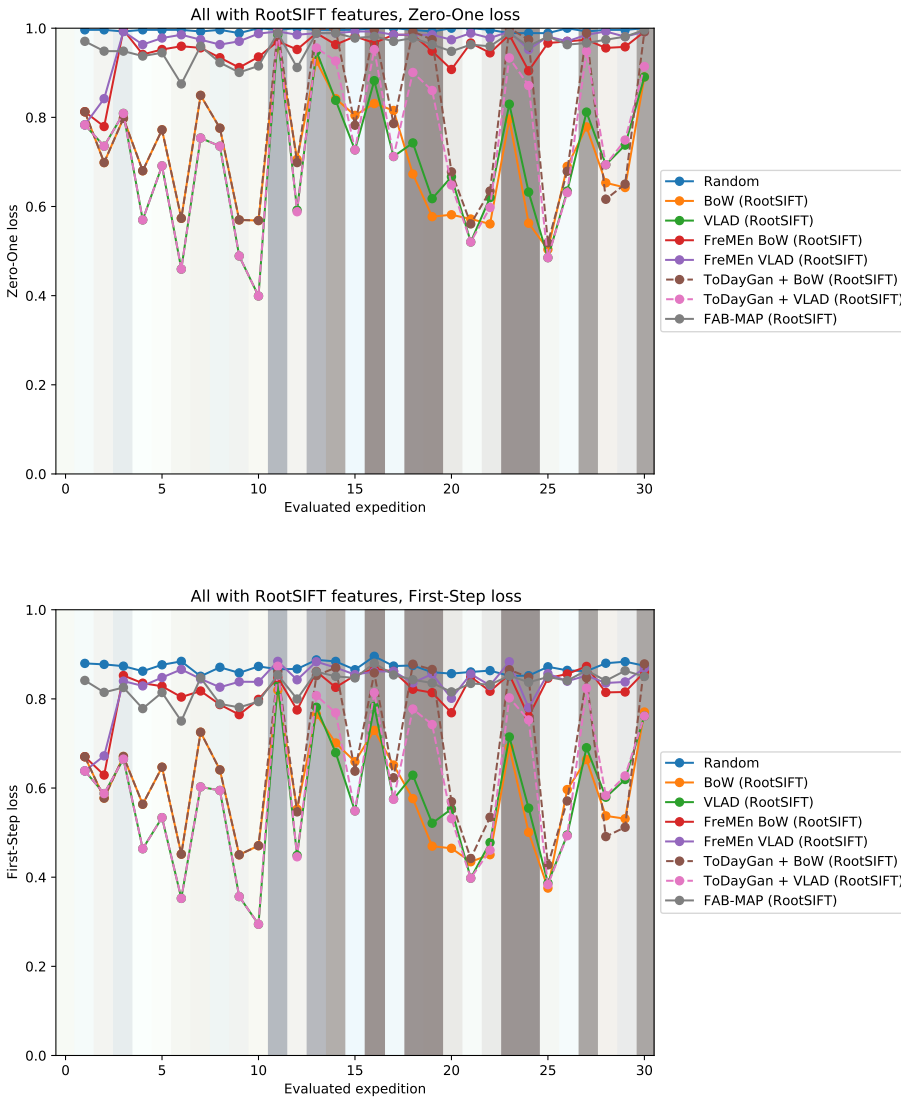


Figure B.10: Zero-one and First-step loss of different methods using RootSIFT features.

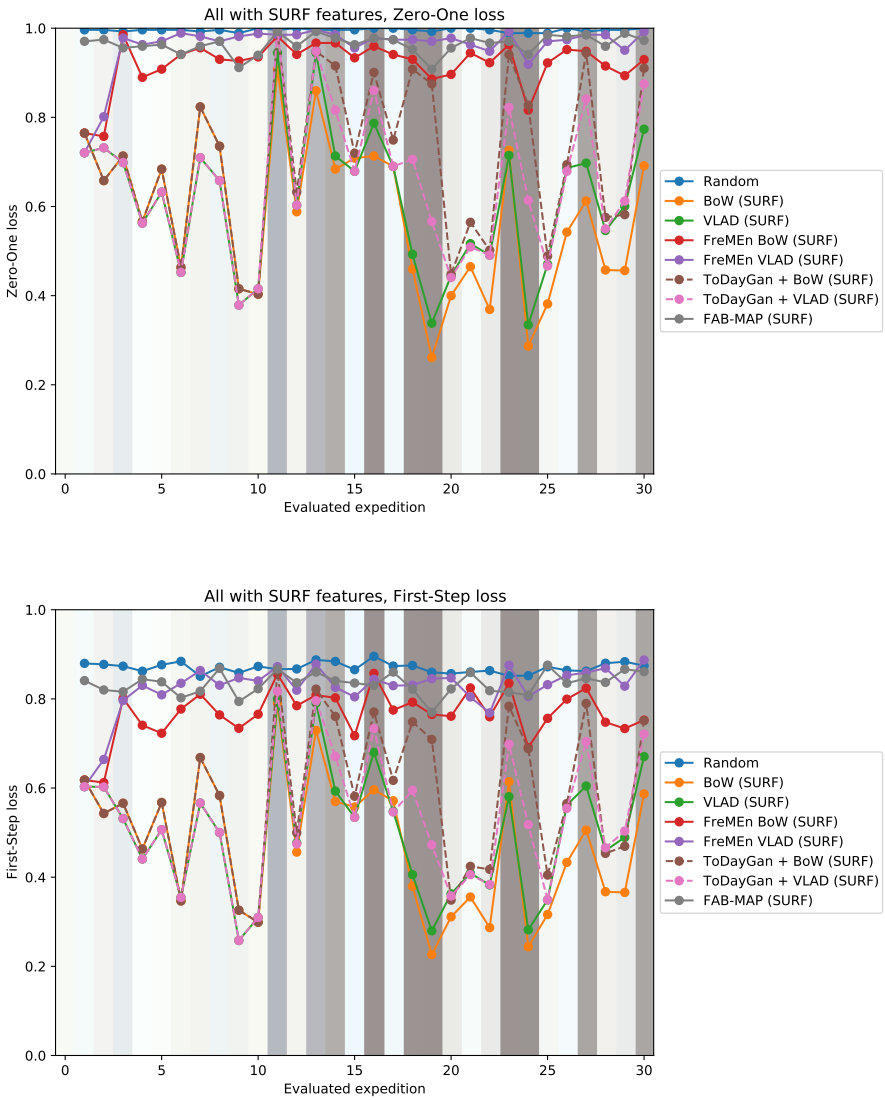


Figure B.11: Zero-one and First-step loss of different methods using SURF features.

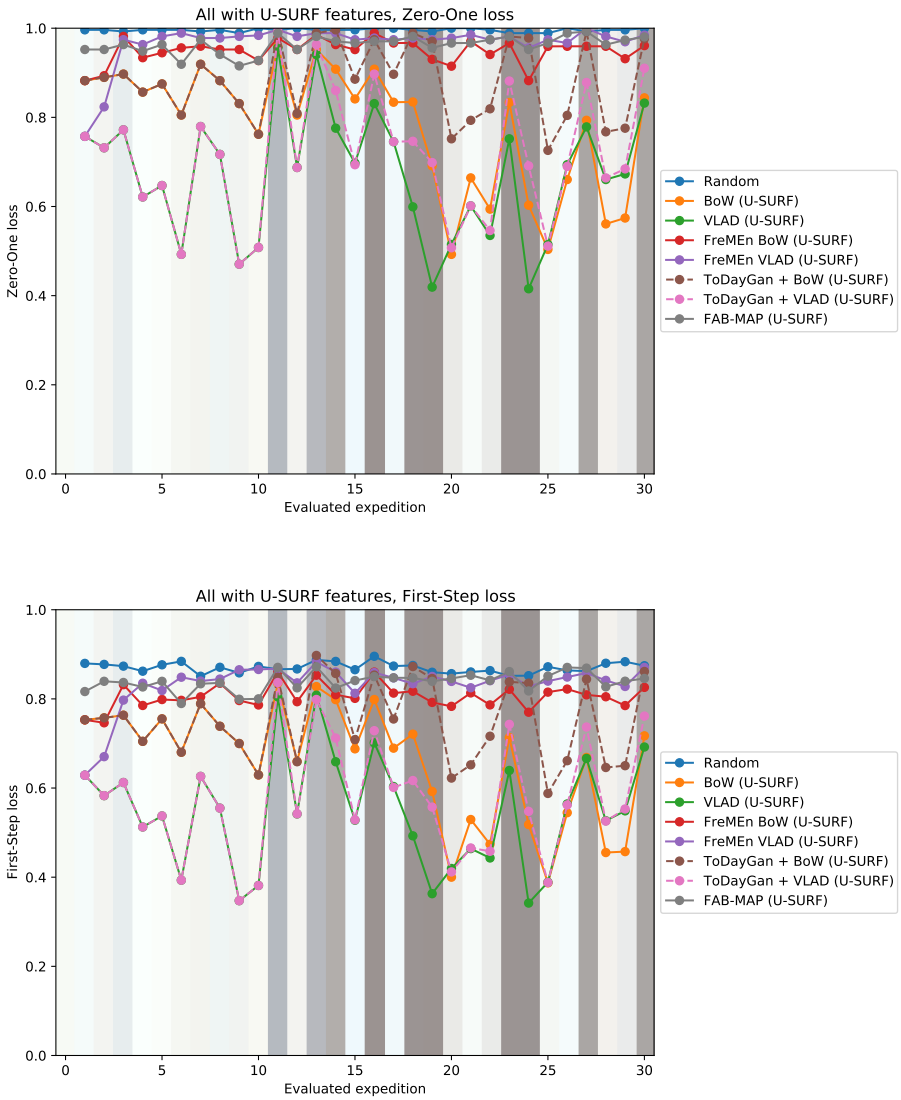


Figure B.12: Zero-one and First-step loss of different methods using U-SURF features.

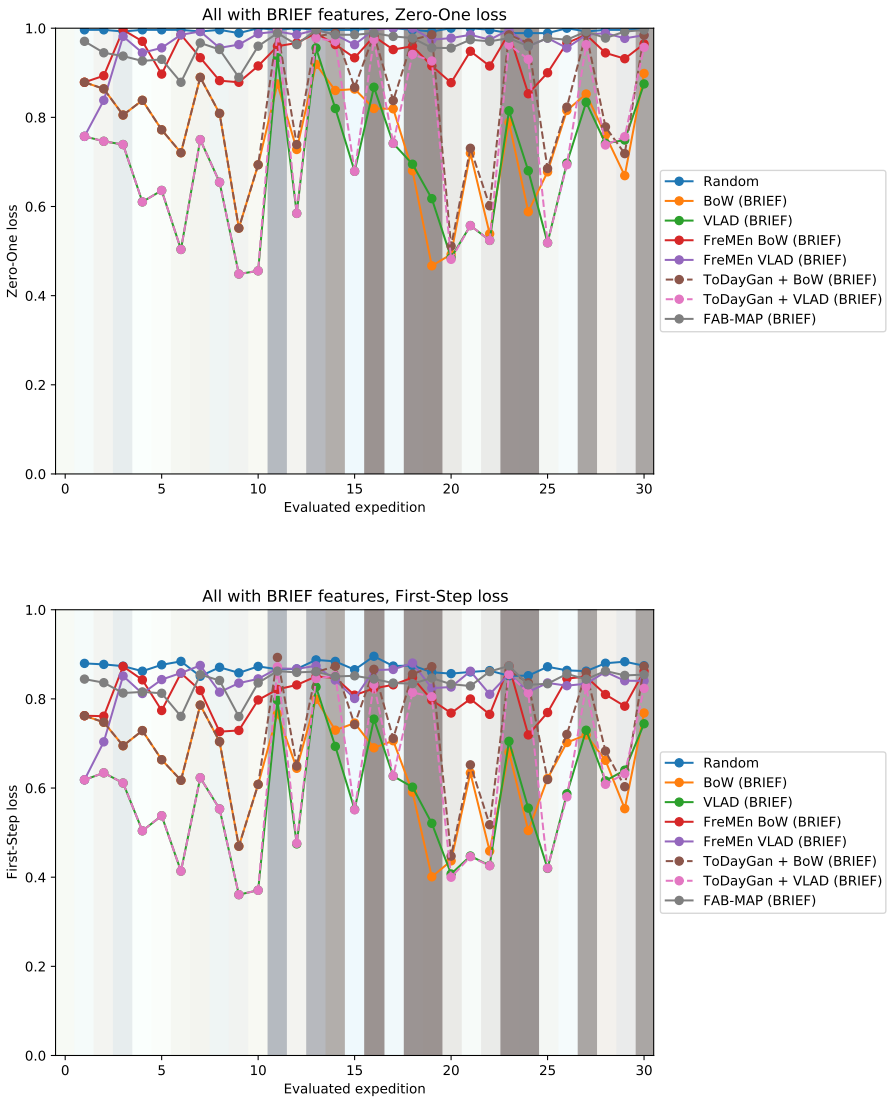


Figure B.13: Zero-one and First-step loss of different methods using BRIEF features.