Fast-SeqSLAM: Place Recognition and Multi-robot Map Merging

by

Sayem Mohammad Siam

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

© Sayem Mohammad Siam, 2017

Abstract

Loop closure detection or place recognition is a fundamental problem in robot simultaneous localization and mapping (SLAM). SeqSLAM is considered to be one of the most successful algorithms for loop closure detection as it has been demonstrated to be able to handle significant environmental condition changes including those due to illumination, weather, and time of the day. However, SeqSLAM relies heavily on exhaustive sequence matching, a computationally expensive process that prevents the algorithm from being used in dealing with large maps. In this thesis, we propose Fast-SeqSLAM, an efficient version of SeqSLAM. Fast-SeqSLAM has a much reduced time complexity without degrading the accuracy, and this is achieved by (a) using an approximate nearest neighbor (ANN) algorithm to match the current image with those in the robot map and (b) extending the idea of SeqSLAM to greedily search a sequence of images that best match with the current sequence. We demonstrate the effectiveness of our Fast-SeqSLAM algorithm in two related applications: loop closure detection and integration of topological maps independently built by multiple robots operating in the same environment.

Acknowledgements

I would like to express my deepest gratitude to my supervisor and mentor, Prof. Hong Zhang for his support and guidance throughout my journey. It has really been a nice journey for me. I have learned so many exciting things. I have learned how to look problems from the research perspective and also learned how to break down the problems for structured and clean thinking which Hong always emphasized on. I can remember writing things on his whiteboard which always helped me a lot to simplify and understand the problems. He has not only taught me the different things in robotics but also advised me how to be a researcher. He taught me how to write academic papers. I ran into his office whenever I faced any problem. I am extremely fortunate to have such a supervisor who genuinely cared about my learning and career. Thank you, Hong, for all of your invaluable knowledge, feedback, motivation and support that you provided in the last two and a half years.

I am also indebted to my committee members, Nilanjan Ray and Martin Barczyk, for reading my thesis and giving their valuable comments and suggestions.

Last but not least, I would like to thank my family for their unconditional love and mental support. I am here today because of my father and mother who never hesitated to sacrifice anything for me.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

INTRODUCTION

In general, place recognition is finding the most matching place for a given visual data from the prior map of the environment. To compare the current visual data with the prior data we must have an internal representation of our environment which we refer as a map. Most commonly, for place recognition we use a topological map, where nodes represent possible places and edges represent the possible paths between these nodes.

In a navigation task, robots need to build and update an internal map and simultaneously localize the robot in the map. Place recognition allows a robot to localize itself in a prior map. Thus, place recognition becomes fundamental to an autonomous system and vehicles such as self-driving cars, service robots, and unmanned aerial systems. The task of building or updating a map and simultaneously localizing the robot in the map while a robot is exploring an unknown environment is commonly referred to as simultaneous localization and mapping (SLAM). Correctly recognizing previously visited locations, a SLAM algorithm is able to optimize the pose constraints between the nodes, overcome incremental pose drift [34, 71] and build a correct map.

Fig. 1.1 shows places with different level of illumination changes. Images in each row are captured at the same place but with different lighting conditions. In the last row, there is a severe illumination change and two images look completely different from each other although both represent the same place.

For long term and large scale SLAM operation, it is important to recognize locations in changing environments over the course of days, nights and seasonal

Figure 1.1: Places with different illumination. Images in each row are captured at the same place but with different illumination. In the last row, there is a severe illumination change and two images look completely different from each other although both represent the same place.

changes. FAB-MAP 2.0 [14], a successful place recognition algorithm based on single image matching and Bayes filtering, demonstrated its operation over a route of 1000 km. However, a single location may look completely different at different times due to illumination and weather changes. As a result, it becomes difficult to match two images of the same location, and algorithms like FAB-MAP which are based on single image matching, have to resort to filtering techniques or will fail quite easily [45].

Milford and Wyeth [45] proposed a sequence based place recognition algorithm, known as SeqSLAM, which finds a sequence of images in the robot map, rather than a single image, that is the best match for the current image sequence being captured by a moving robot. SeqSLAM has demonstrated better performance in recognizing places that underwent severe appearance changes than the other successful SLAM algorithms like FAB-MAP 2.0 [43]. Most recently, Milford and Shen [43] have demonstrated significant improvement of the viewpoint invariance of SeqSLAM by generating synthetic viewpoints using the state-of-the-art deep learning techniques [35]. However, SeqSLAM, which is based on exhaustive sequence search, is computationally costly when the number of images or nodes in the map is large. This is unfortunate as a robot needs to perform all the computation for localization including loop closure detection online in a SLAM system.

The key motivation for our work is to develop a place recognition algorithm for a large scale and long term operation, based on sequence matching like SeqSLAM but computationally efficient. To achieve a high computational efficiency, we first store the image descriptors of a map in a tree structure. Then we use an approximate nearest neighbor (ANN) algorithm [47] to search for N best nearest neighbor images for a current view and store the distances in a sparse difference matrix where N is constant and much smaller than the number of images in the map, n. In addition, we modify the sequence matching algorithm in SeqSLAM so that we can still find the best match for a current sequence in a sparse difference matrix. Specifically, in our modified sequence matching algorithm, we do not search exhaustively as in the original SeqSLAM algorithm; rather we greedily search only from those locations in the map where the best K matches for the current view of

the robot are found, among the N total matches, where K is smaller than N. Velocity information of the robot is then used to determine where to search next in the map. Due to the efficiency in the construction of the difference matrix and the greedy sequence search algorithm, the computational complexity of our algorithm is $O(n \log n)$ for finding all loop closure nodes between any two traversals of an environment while SeqSLAM has a computational complexity of $O(n^2)$ and where n is the largest number of nodes between the two traversals. Most importantly, when the accuracy of our algorithm is compared with that of SeqSLAM experimentally, the computational efficiency of our proposed algorithm does not come at the expense of accuracy.

The rest of the thesis is organized as follows. In Chapter 2, we summarize relevant background and prior research in place recognition. Chapter 3 describes our algorithm, Fast-SeqSLAM for loop-closure detection. Our experimental design and results are presented in Chapter 4 focusing on computational complexity and the accuracy of our Fast-SeqSLAM algorithm. In Chapter 5, we propose a technique that uses Fast-SeqSLAM to merge multiple maps. Finally, we conclude the work and discuss limitations and future research in Chapter 6.

Chapter 2

Background

2.1 What is a place?

To localize a robot in an environment, we need to represent the environment using some kind of a map. It is challenging to build a metrically accurate map of the environment and doing localization within such a representation. Representing a highly accurate map requires a high amount of memory and localization in such a map will need more computation. We maintain a relational map which is rubbery to represent an environment, instead of maintaining a metrically accurate map [8]. This kind of map is called topological map. Nodes represent the places in the environment and edges represent paths between these places. Places are unique discrete locations in the continuous environment. A place recognition system needs to segment an environment into distinct places. There are different techniques to find the discrete places. One simple approach is to divide the world into discrete places by representing locations at a fixed time or distance interval. We call this a density-based place selection method. Density-based place selection methods are common and practical to implement as they depend on measurable quantities such as distance, time, or sensor values [23].

Another approach is that a new place is detected when the appearance of the place is sufficiently different from the current model of the environment. Change point detection algorithms in video segmentation [73, 29], such as Bayesian surprise [29] and segmented regression [20] have been proposed to define places for the topological map [61, 65]. Image sequencing partitioning technique has been

proposed in the paper [32] to group visually similar images together as places in the topological map. Dynamic vocabulary building [55] and incremental topic modeling [10] together have been proposed by Murphy and Sibley [51] to continuously learn new places in an environment.

2.2 Different representations of a map

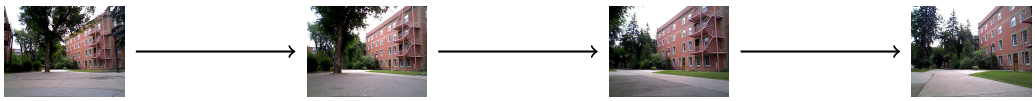


Figure 2.1: Pure topological map. Each image represents a unique place and we only have the sequence information of the places.

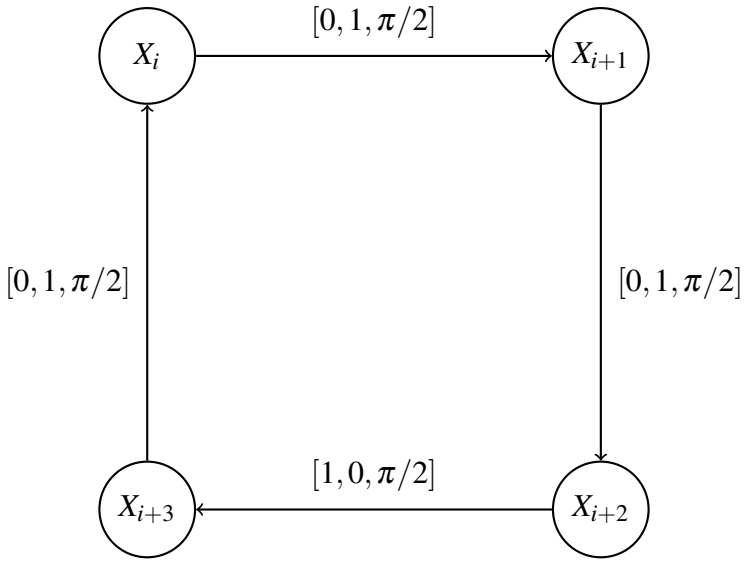


Figure 2.2: Topological-metric map. Relative metric information between the places are also known. $[dx, dy, d\theta]$ represent the relative position changes between any two nodes. dx and dy represent the changes in x and y direction, and $d\theta$ represents the angular change.

A topological map is conceptually similar to the biological notion of a cognitive map [38]. A graph represents an environment where each node represents a place and each edge represents a path between two places. Robot navigation can be defined as going from one node to another node following edges of the map. Places are decision points in navigation and also represents the destination point [65, 17].

The most abstract form of map is just storing the places in a database without their relative positions. So, this form of a map does not have any position information. Place recognition is done by using pure image retrieval techniques.

A pure topological map contains the relative position information of the places. For example, a sequence of images with a time interval between the places fall into this category. Place recognition is done incorporating the spatial or temporal constraints with the place matching score [13]. Fig. 2.1 shows a pure topological map where only the sequence information of the places is known.

Pure topological maps only have the relative position information of the places while topological-metric maps have relative metric information between the places, such as distance, direction or both. Appearance-based topological-metric maps include the relative metric information between two nodes [42]. For example, FAB-MAP [13] and SeqSLAM [45] use a pure topological map to represent the environment; however performance has been improved with the addition of odometry information, and this has been demonstrated by CAT-SLAM [39] and SMART [59] respectively. Fig. 2.2 shows a topological metric map where relative distance and direction between the places are known.

Topological-metric maps also may have metric information about the position of landmarks or objects in addition to the relative metric positions of the places. The metric information about the position of landmarks or objects in a place can be stored within each node [67]. If depth information is extracted from the image data then the metric information within the topological place node can be stored as a sparse landmark map [31] or as a dense occupancy grid map [53]. Methods include MonoSLAM [16], PTAM [30], DTAM [53], LSD-SLAM [18], and ORB-SLAM [48]. They either use stereo camera to get the depth information or structure-from-Motion algorithms [72] to extract depth from the monocular camera images.

2.3 Visual Place Recognition

In the first two sections of this chapter, we described how to represent an environment using different kinds of maps and places. In the appearance-based map, we

represent a place using only visual data. Robots may use their motion sensors or visual odometry [67, 31] to get the metric information between the places for the topological metric map. Metric relationships between places and visual data are used to determine the most likely place, similar to the neuronal firing of the place cells for the cognitive mapping [33].

A visual place recognition system processes all the incoming data from the environment to make a decision about whether the current place is already in the map or a novel one [38]. Visual place recognition or loop-closure detection is independent of the other components of a SLAM system. Fig. 2.3 presents a schematic of a visual place recognition system. A visual place recognition system contains three key components [38]: an image processing module for visual data, a map of the environment and a belief generation module. Image processing module takes the vision data and extracts the features. Belief generation module combines all the sensor data from the environment, uses the map and the location prior to decide whether the current place is already in the map or a new one. A place recognition system can only use the location prior if the system uses a topological map.

In Subsection 2.3.1, we briefly discuss different techniques to describe a place in a map then we discuss how to recognize places by using pure image retrieval techniques in Subsection 2.3.2, and incorporating the motion information for using location prior (see 2.3.3). Finally, we discuss state-of-the-arts visual place recognition algorithms (see 2.3.4) which are related to our proposed algorithm in Chapter 3.

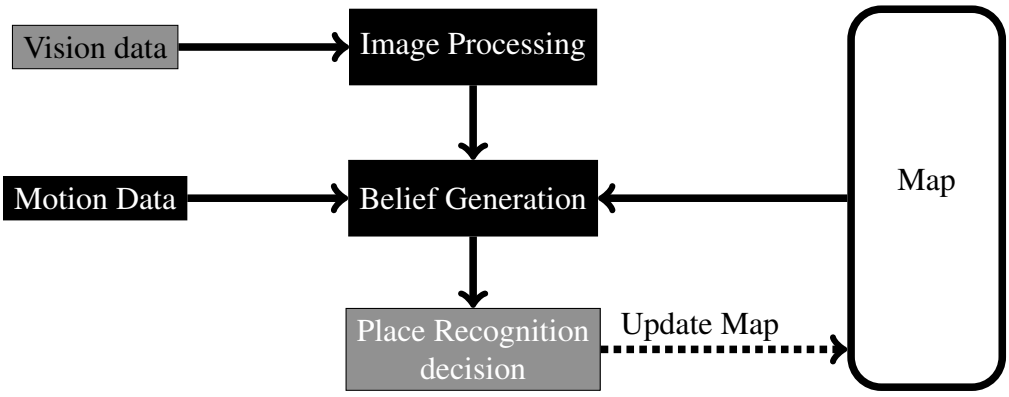


Figure 2.3: Place recognition system

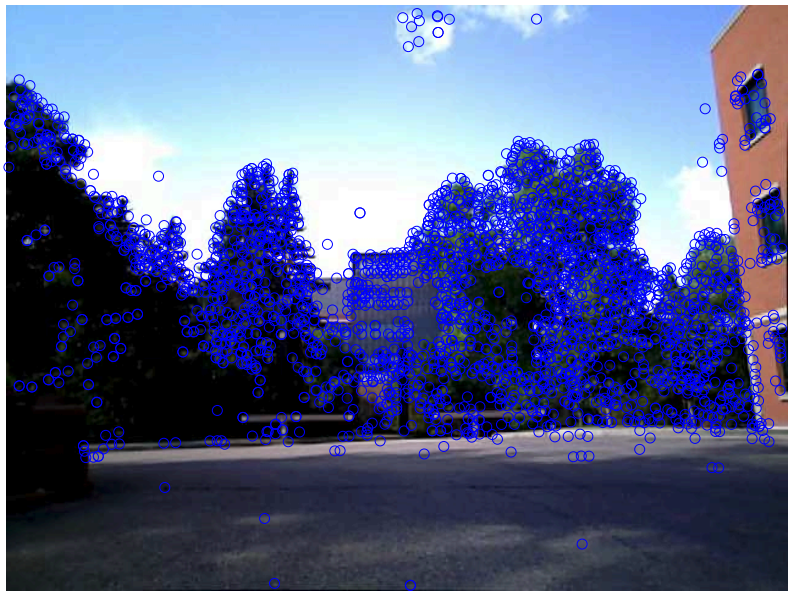


Figure 2.4: SIFT keypoints

2.3.1 Visual Description of a Place

To describe a place in a map we use different visual place description techniques. It falls into two main categories: those that extract parts of the place or image which are important and those that describe the whole scene. For example, scale-invariant feature transforms (SIFT) [37] and speeded-up robust features (SURF) [5] fall into the first category. Fig. 2.4 shows the keypoints extracted by the SIFT descriptor. Local feature descriptors first require a detection phase which determines the interesting parts of the image to retain as local features or key points then in the description phase, we describe the interesting part of the image. In contrast, global or whole-image descriptors such as Gist [57, 58] and HOG [15] do not have a detection phase but describe an image with a single feature vector.

Local Descriptors

Local feature descriptors first detect the interesting keypoints and then describe these keypoints. Local feature descriptors like SIFT [37] and SURF [5] are rotation and scale invariant. However, they suffer from the illumination changes. The use

Figure 2.5: HOG features: each block in the image has a histogram (white shape). Each bin in the histogram is a gradient orientation (9 bins for 0-180 degree) and vote is the gradient magnitude. Finally, each histogram is packed into a single vector as a descriptor of the image.

of local feature descriptors in place recognition became popular with the development of SIFT. There are many local feature descriptors which are also applied in the visual localization and place recognition problem. Ho and Newman [28] used Harris affine regions [41], Murillo et al. [49] and Cummins and Newman [14] used SURF [5], while FrameSLAM [31] used CenSurE [1].

Since the detection and description steps of a local descriptor are independent of each other. So it is not uncommon to combine a detection technique of one descriptor with a description technique of another. For example, Mei et al. [40] combined the detection technique FAST [63] with the description technique of SIFT descriptors.

Global Descriptors

A global descriptor describes an image as a whole to describe the entire scene. Global descriptor finds features including contour representations, shape descrip-

tors, and texture features. Global descriptors can be generated from local feature descriptors, for example, using a grid-based pattern and then finding keypoints in each grid. One big advantage is if we can represent a place using a global descriptor, a place can be described by a single vector. Then we can simply use Euclidean or Manhattan distance to find the similarity scores among these vectors.

HOG, a very popular global descriptor, counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms and SIFT descriptors. In contrast, it is computed on a grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy [15]. Another whole image descriptor is Gist [58], which uses Gabor filters at different orientations and different frequencies to extract information from the image. Gist has been used for place recognition in different occasions. Whole image descriptors based on SURF features known as WI-SURF proposed by Badino et al. [4] to perform localization. Similarly, BRIEF-Gist [70] used BRIEF features [9] to represent an image as a whole.

Global descriptors comprehensively outperform local descriptors at performing place recognition in changing conditions [45, 42, 52]. However, global descriptors are more dependent on image viewpoint than local feature descriptors. As a reason, common global descriptors such as HOG, Gist, and others perform badly in viewpoint changes. Most of the algorithms which use whole-image descriptor to represent images assume that the camera viewpoint remains similar. However, the pose (viewpoint) dependent problem can be reduced by using circular shifts as in [44] or by combining a bag-of-words approach with a Gist descriptor on segments of the image [50].

Condition Invariant Descriptor

With a severe illumination change, the same place may look completely different (shown in Fig. 1.1). For a robust SLAM application, it is very important to reliably match places in changing environments. Most of the conventional local feature descriptors struggle for matching places in changing environments. Furgale and Barfoot [25] observed that the non-repeatability of SURF features due to chang-

ing appearance, particularly lighting change, was a major cause of failure during visual-teach-and-repeat experiments. In [74], Valgren and Lilienthal tested SIFT features and a number of SURF variants across a change in lighting, cloud cover, and seasonal conditions. The SURF variants all outperformed SIFT, but none of the tested features were found to be robust across all conditions. Some options are instead of using a common descriptor, we can use condition-invariant descriptors based on shadow removal [12], learning on different illumination condition [62] or representing places using semantic labels [76].

Instead of using point features such as SIFT or SURF, whole-image descriptors can be chosen. However, as for the whole-image description methods, too drastic a change in appearance will cause system failure [66] and whole image descriptors also suffer from the additional problem of sensitivity to viewpoint change [69]. Whole image descriptors have been used in systems such as SeqSLAM and others [45, 52] that demonstrate robustness against environmental change. In SeqSLAM, they compare a sequence of images instead of a single image and they use downsampled patch normalized raw image pixels as a global descriptor of an image. SeqSLAM has been demonstrated robust against severe illumination changes however, SeqSLAM suffers from the viewpoint changes. Most recently, Milford and Shen [43] have demonstrated significant improvement of the viewpoint invariance of SeqSLAM by generating synthetic viewpoints [35] using the state-of-the-art deep learning techniques.

2.3.2 Image Retrieval

The most abstract form of map is just storing the places in a database without their relative positions. So, this form of a map does not have any position information. Place recognition is done by using pure image retrieval techniques by assuming a uniform location prior for all the nodes. Uniform location prior means that we actually do not have any prior location belief.

Whenever a robot gets a new image, we call it the robot's current view and we need to find the possible matching images in the map with their corresponding matching scores. For example, we can use the Euclidean or Hamming distance to

find the matching scores when each image is represented by a vector using a global descriptor. However, if we represent an image using a local descriptor then we need to search for the matching keypoints for one image in the others and based on the number of matching keypoints we can give a matching score for a pair of images.

Image Retrieval Using Local Image Descriptors

For finding the common keypoints in two images, Lowe used a modification of the k-d tree algorithm called the best-bin-first search method [37] that can identify the nearest neighbors with high probability using only a limited amount of computation. However, this will be inefficient in place recognition because there will be hundreds of keypoints in each image and we need to find the matches of these keypoints in all images in the map.

Scalability is a key concern in place recognition because as a robot visits more places, the number of images in the map increases and search efficiency decreases. Therefore maps need to be designed to ensure large-scale efficiency. A bag-of-words model accelerates the image retrieval process by quantizing the keypoints and using inverted indices; the image ID numbers are stored against the visual words. That is, for each visual word in the vocabulary, a list of the images in which that visual word appears is maintained. Thus, finding all images that contain a visual word or set of words becomes a cheap operation. This reduces the search space rather than requiring a linear search of all images in the map. Hierarchical vocabulary trees [56] have been proposed by Schindler et al. [64] for efficiently quantizing the keypoints into visual words. They only use the most informative features by measuring conditional entropy.

The bag-of-words model creates a feature space with the finite number of visual words in the dictionary. A typical vocabulary contains 5000–10000 words, but a vocabulary of 100000 words has been used for place recognition by FAB-MAP 2.0 [14]. For each image, every keypoint is assigned to a particular word, ignoring any geometric or spatial structure.

Bag-of-words model can build the vocabulary or dictionary from the training

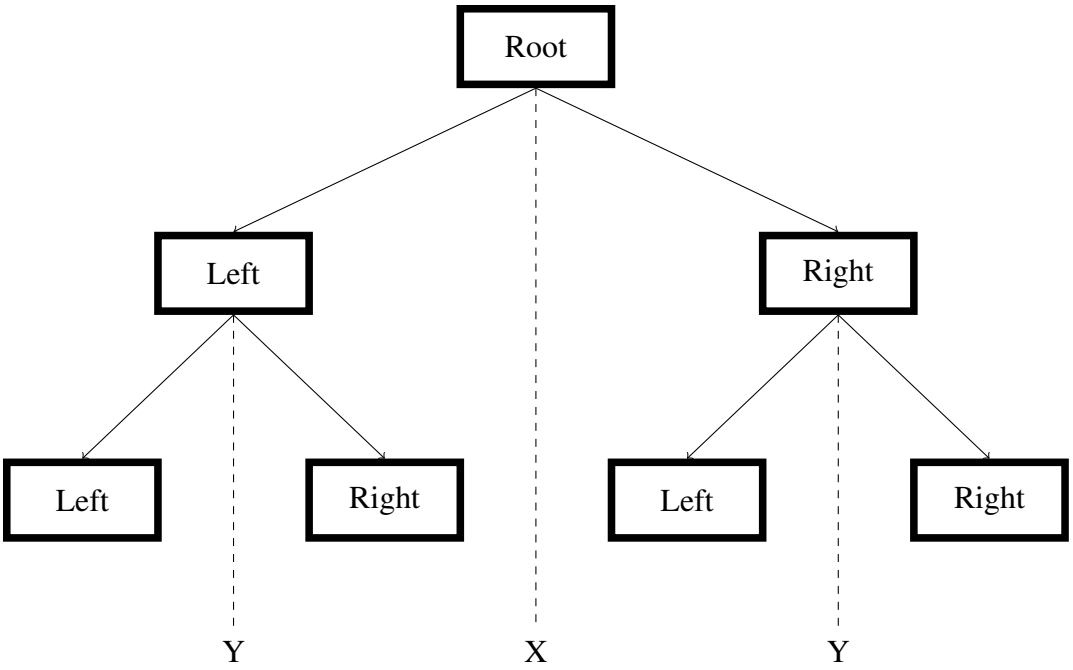


Figure 2.6: Classic k-d tree: it splits the data into two halves alternatively with respect to X and Y at each node.

images or it can use a pre-trained vocabulary. An online method to update the vocabulary based on observations has also been proposed by Nicosevici and Garcia [55].

Image Retrieval Using Global Image Descriptors

Global descriptor describes an image with a single vector. We can linearly match the current view of the robot with all the images in the map and find the top matches. However, the search space is linearly proportional to the number of images in the map. For an efficient image search, we can use a tree-like data structure to find the matching neighbors approximately. In the following, we will discuss k-d trees and hierarchical k-means clustering algorithms for approximate nearest neighbor search.

A popular algorithm for nearest neighbor search is the k-d tree algorithm [24]. Fig. 2.6 shows a classic k-d tree which has only two dimensions, X and Y. The original k-d tree algorithm splits the data into halves at each level of the tree on the dimension for which the data exhibits the greatest variance.

Classical k-d tree performs efficiently in low dimensions. However, the search performance rapidly degrades with an increase in the number of dimensions of the data. Arya et al. [3] modified the original k-d tree algorithm to make the search faster than a linear search by returning an approximate nearest neighbor instead of the exact one. To speed up the search, it maintains a priority queue and visits the tree nodes in order of their distance from the query point. It only checks few nodes from the priority queue which improves the search speed. As a result, it is not always able to return the exact nearest neighbors.

An improved version of the k-d tree algorithm is proposed by Silpa-Anan and Hartley [68] by maintaining multiple randomized k-d trees. Randomized k-d trees maintain a fixed T number of trees. These trees are built by choosing the split-dimension randomly from the first D dimensions on which data has the greatest variance. Muja and Lowe [47] demonstrated that using a fixed value of $D = 5$ performs well; increasing the value of D does not improve the performance. In their implementation, they maintain a single priority queue across all the randomized trees so that it can visit nodes across multiple trees in order of their distance with the query node. The degree of approximation depends on the maximum number of leaf nodes it can visit and the number of randomized k-d tree it maintains. It stops visiting when it has reached the predetermined number of leaf nodes and returns the best candidate from the all visited leaf nodes.

The classic k-means algorithm is used to cluster the data into k subsets. However, in hierarchical k-means clustering, we do not predefine the number of clusters. We decide on a value of k to be the branching factor based on the dimension of data. This restricts the number of clusters at each level of the clustering hierarchy. Finally, we recursively cluster each sub-cluster until we hit some small fixed number of points or until a stop criterion is reached. Fig. 2.7 shows a simple hierarchical k-means tree structure where $k = 2$.

Muja and Lowe [47] developed an algorithm that explores the hierarchical k-means tree in a best-bin-first manner, similar to the exploration technique of the k-d tree. First, the algorithm performs a single traversal through the tree and keeps the unexplored branches in each node it visits in a priority queue. Nodes are extracted

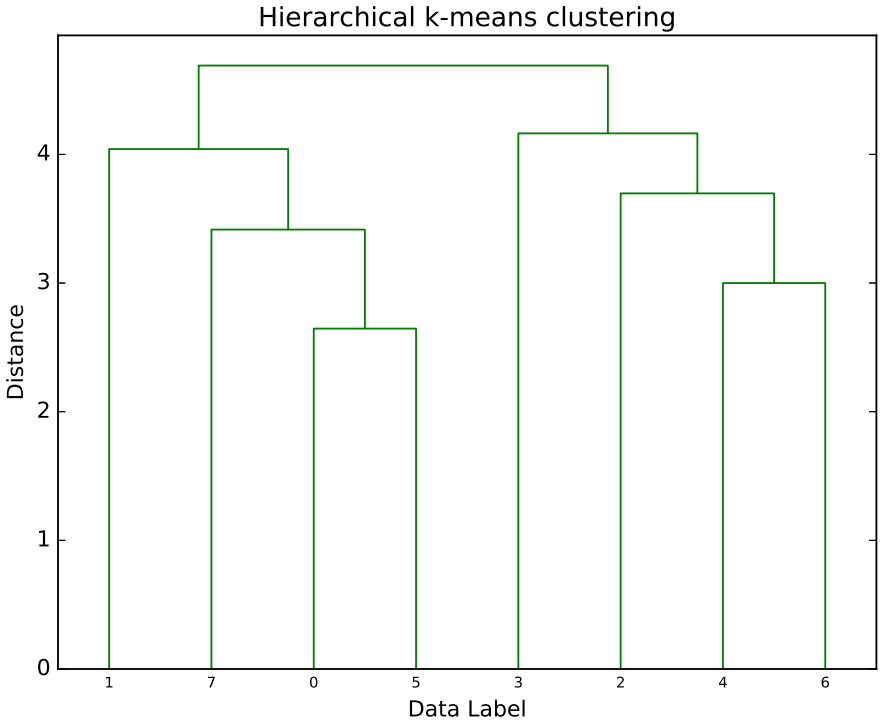


Figure 2.7: Hierarchical k-means with $k = 2$. X axis represents different data. Y axis represents distance for each cluster.

from the priority queue in order of their distances from the branch center. In each traversal, the algorithm adds the unexplored branches along the path to the priority queue. The degree of approximation is specified in the same way as for the randomized k-d trees, by stopping the search after a predetermined number of leaf nodes (dataset points) have been visited.

2.3.3 Incorporating Temporal or Spatial Constraints

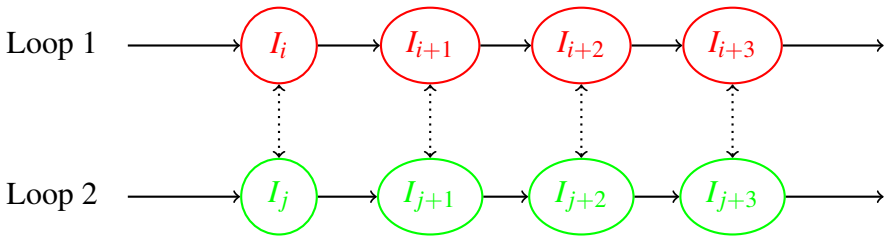


Figure 2.8: Red nodes are from loop 1 and green nodes are from loop 2. In both loops, robot visits each place with the same velocity.

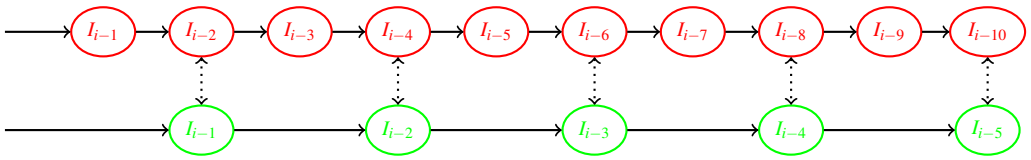


Figure 2.9: Robot has a two time faster velocity while it visits places in the second loop (green) than in the first loop (red).

In the previous section, we discussed how to recognize places if we do not have any relative position information between the places which we call pure image retrieval. However, in reality, most of the robots have relative position information. Maps may also contain the metric information between the places. Knowledge about the relative positions or the metric information of the places can be incorporated with the observation likelihood in finding the correct loop closure nodes.

There are few techniques which try to incorporate the temporal or spatial information of the map with the global best matches. Recursive Bayes filter is a very popular technique to estimate a robot's position incorporating the temporal information of the robot with the image matching score. In Bayes filter, a place

recognition system uses a fixed state transition matrix for the robot to update the location prior for all the places in the map. A probabilistic system like FAB-MAP can be run as a pure image retrieval process based on only observation likelihood or image matching score, but performance improves when transition information is included through Bayesian filtering or similar techniques. Topological maps can also use a location prior to speed up the search. A sampling-based method such as a particle filter can be used to sample possible places [39] which reduces the search space to the number of particles in the environment. Thus, the computation complexity of particle filter is proportional to the number of particles, not the size of the environment.

In contrast to above techniques, there are few algorithms which incorporate temporal information by matching a sequence of images [54, 46]. SeqSLAM, which is inspired by ratSLAM [46], uses an image sequence undergoing different velocities to find a corresponding image sequence that matches the current sequence. As a result of explicitly using an image sequence to enforce spatial consistency, rather than a Bayes filter, it becomes robust with respect to condition changes such as weather and time of the day.

In Fig. 2.8, the robot has the same velocity while traversing in both loops. So, it is more likely that the loop closure nodes will appear continuously. In Fig. 2.9, the robot is moving twice faster in the second loop (green nodes) than the first loop. So, it is more likely that the nodes from the second loop match with the nodes in the first loop sequentially but skipping one node at each time.

2.3.4 Visual Place Recognition Algorithms

In following, we will explore state-of-the-art visual place recognition algorithms which are related to our work.

Bag-of-Words and Bayesian Filter:

Bag-of-words with local feature descriptors have been extensively used for place recognition [22, 2, 13]. FAB-MAP 2.0 [14] is a very successful algorithm which is

an efficient version of FAB-MAP 1.0. It has demonstrated its success in place recognition operation along a route of more than 1000 km using local image descriptors with the Bag-of-Words technique. In the following, we will give an overview of FAB-MAP algorithm.

FAB-MAP 1.0 [13] is a probabilistic appearance-based localization system that uses Bayesian filter to calculate an observational likelihood. FAB-MAP uses a bag-of-words model with SIFT or SURF keypoint features for image descriptions and these keypoints are vector-quantized to serve as words in the vocabulary of the BoW technique. FAB-MAP 1.0 calculates the dependencies between the visual words in the observation likelihood. For example, it is more likely that words associated with car wheels and car doors will appear together. FAB-MAP 1.0 calculates a full joint probability distribution of the observed words using Chow-Liu tree [11]. They capture the dependencies by learning a tree-structured Bayesian network using the Chow-Liu algorithm [11]. Also, Chow-Liu tree provides efficient learning and inference for very large visual vocabulary sizes. The graphical model of the system is shown in Fig. 2.10.

FAB-MAP 1.0 also considers whether the words in common appear frequently. If two locations look similar but the words that appear were frequently observed, FAB-MAP generates a low matching probability by using the denominator as a normalizing constant.

An inverted index provides an efficient way to search for a document in a very large collection [7]. However, FAB-MAP 1.0 model is not directly implementable using an inverted index data structure because every observation component contributes to the appearance likelihood, including negative observations. That is, they need to calculate the observation likelihood for all the previous locations for any new observation. However, FAB-MAP 2.0 solves this problem by calculating the observation likelihood approximately. FAB-MAP 2.0 calculate the appearance likelihood approximately by efficiently handling the negative observation of the words in the probabilistic model, i.e. they do not need to calculate the observation likelihood for all the location for each new observation. Thus, FAB-MAP 2.0 uses the inverted index technique to calculate only the observation likelihood for the loca-

tions which contain the currently observed words or children of these words in the Chow-Liu tree. The computation complexity for calculating observation likelihood for each new observation is $O(W)$, W is the number of words in the dictionary. FAB-MAP 2.0 is a highly scalable algorithm for place recognition. It has been demonstrated with a 100 000 words vocabulary [14] over a route of 1000 km.

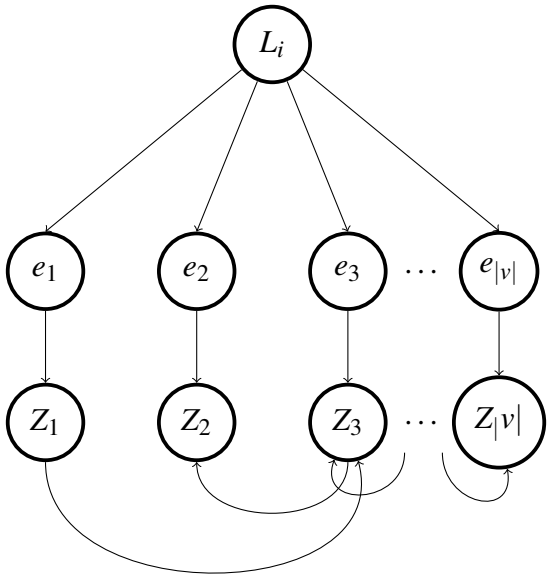


Figure 2.10: A probabilistic graphical model of FAB-MAP system.
Locations L_i independently generate scene elements. $Z_1, Z_2, Z_3 \ldots Z_{|v|}$ are visual words in the vocabulary. Each visual words depend on other visual words. The dependencies between the visual words are modeled via Chow-Liu tree.

Sequence-Based

Milford and Wyeth [45] proposed a sequence based place recognition algorithm, known as SeqSLAM. Instead of a single image, it finds a sequence of images in the robot map that are the best matches for the current image sequence being captured by a moving robot. SeqSLAM has demonstrated a better performance than the other successful SLAM algorithms like FAB-MAP 2.0 [43] in recognizing places that underwent severe appearance changes.

1. Input images are downsampled to (64×32 pixels) in preprocessing step. Then images are converted into grayscale.

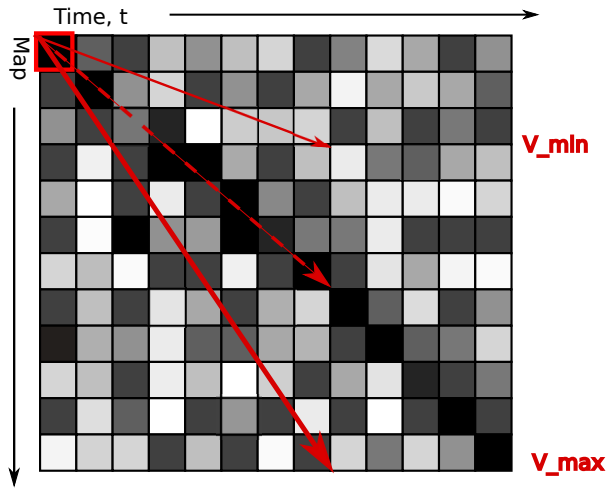


Figure 2.11: Fig. shows a complete difference matrix. Different trajectory lines which correspond to different velocities are originated from the red bordered cell. The dotted trajectory has the minimum difference score.

2. Images are divided into patches of 8×8 pixels, which are then normalized. Each image is converted into a vector and these pixel values are used as feature points.

3. Difference matrix is calculated between each pair of images using the sum of absolute differences.

4. Local contrast enhancement is applied on the difference matrix.

5. SeqSLAM performs an exhaustive search to find the best matching sequence of images.

SeqSLAM uses patch normalized image pixel values as feature points. It creates a difference matrix for all possible pairs of images between the two traversals of the robot. Then it performs contrast enhancement on the difference matrix which forces the matcher to find the best matches in every local neighborhood of the trajectory instead of only one global best match. Finally, for finding a match for the current view of the robot, SeqSLAM performs an exhaustive search.

Each value in the difference matrix is called as a *difference value*. SeqSLAM sums up all the values that each trajectory line passes through and the sum is called *sequence score*. For sequence searching, SeqSLAM checks all the possible se-

quences originating from each location and takes the one which has the minimum sequence score. Fig. 2.11 shows the full difference matrix where all the cells have difference scores. Red lines are trajectory for different velocities. The dotted line has the minimum difference score.

A significant disadvantage of SeqSLAM is computational complexity. For matching, SeqSLAM matches a sequence of images. Since a sequence of images is used, the cost of matching a sequence with all possible sequences in a map is computationally high for a reasonably large map. For finding all loop closure nodes between any two traversals of an environment, SeqSLAM has a computational complexity of $O(n^2)$ where n is the largest number of nodes within the two traversals.

Network flows [52] has been used as an alternative to SeqSLAM's sequence searching. A directed acyclic graph (DAG) is created to store the similarity between pairs of images and formulate image matching is solved as a minimum cost flow problem while incorporating sequence information. Place recognition is equivalent to finding a sequence which has the minimum flow cost. However, the time complexity for this technique remains $O(n^2)$.

Hidden Markov Model (HMM) with Viterbi algorithm [75] has been used for finding the best matching sequence for a given sequence of images [27], inspired by Dynamic Time Warping (DTW) from speech recognition [60]. The state transition matrix is built using local velocity constraints of a robot. However, they need to calculate a complete similarity matrix for all the possible image pairs and hence the time complexity of this algorithm is $O(n^2)$.

To reduce the time complexity of SeqSLAM, a particle filter technique has been proposed by Liu and Zhang [36]. Rather than computing the matching scores for all candidate sequences in the map, a subset of these sequences are sampled and evaluated. The promising ones with high probabilities are kept for further evaluation, the unlikely ones are dropped, and new candidates are introduced in the next round of evaluation. Although a particle filter can alleviate to some extent the computational cost, convergence to the correct solution is probabilistic and the overall complexity of the algorithm is still $O(n^2)$.

2.4 Visual Place Recognition in SLAM

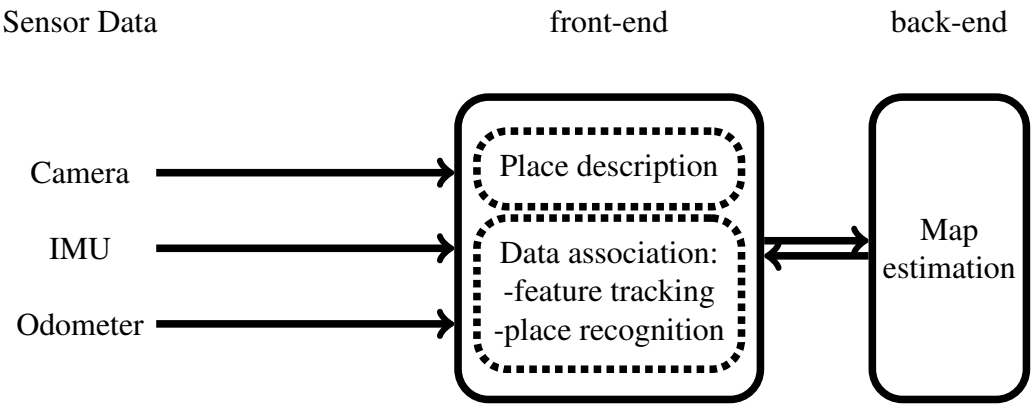


Figure 2.12: SLAM system: It has two main component, front-end, and back-end.

In the previous section, we have described the different modules of a visual place recognition system in detail. Fig. 2.12 shows how the visual place recognition fits into a SLAM system. A common SLAM system has two main components: front-end and back-end. The front-end of SLAM has access to all the sensors and processes the sensor data. Using the information from the front-end, SLAM optimizes the map in the back-end.

Visual place recognition is done in the front-end. Visual place recognition gives the loop-closure information to the back-end and back-end uses that information to rectify the map. The optimizer (back-end) relies on the topological information received from the front-end. SLAM algorithms build a map incrementally. Loop closure detection or place recognition, which is done in the front-end of the SLAM system, can be separated from the online local map update step [67, 26].

When a SLAM system uses a pure topological map, localization is done at the topological level. The loop-closure information only helps the system to identify the most likely location in a pure topological map. However, in a topological metric map, the relative metric poses between the places are known and loop closures help to perform metric correction using pose constraint optimization. If a topological metric map has the metric information both within and between the place descriptors (as in FrameSLAM [31]), the metric information can be used to perform a full metric SLAM solution. Finally, all metric corrections are done at the back-end.

Chapter 3

Fast-SeqSLAM

Our work is motivated by a desire to build a place recognition algorithm which is suitable for a long-term and large-scale SLAM system. Condition invariance and computational efficiency are two key prerequisites for large scale and long term SLAM operations. So, we have to build a computationally fast algorithm which can handle illumination change.

The general setup of SeqSLAM, which we adopt, assumes two traversals of the same environment or route. We assume that the map is built in the first traversal from n images and the second traversal contains m images among which loop closures exist. The goal is to determine if any sequences in the second traversal overlap with any in the first. Both the length and the initial location of a possible matching sequence are unknown.

The key idea in our Fast-SeqSLAM algorithm is that we should avoid searching all the sequences in the map exhaustively for the best matching sequence to the current sequence; instead, we can search greedily among sequences defined by the most likely initial matching images and use motion continuity to extend and continue the search. This initial image set is small, and its size is constant (N) and does not grow with the size of the map.

Further, to identify promising initial locations of matching sequences, rather than linearly comparing with all map images, we use an approximate nearest neighbor algorithm to find the N best nearest neighbor images in the map with respect to the current robot view, and we create a sparse difference matrix with similarity scores only for these nearest neighbors. Then, we greedily search for the best

matching sequence only from the K best matching locations among the N (nearest neighbor nodes) locations. We call each value in the difference matrix as a *difference value*, and the summation of all the difference values of a trajectory sequence as a *difference score*, using the notations in SeqSLAM [45]. Fig. 3.1 shows an example of a difference matrix where $n = m = 13$, and $N = 2$. The brighter a cell is, the larger the difference value. Sequence matching uses this sparse difference matrix, and proceeds as described below where the explanations refer to lines in Algorithm 1, the pseudocode of our sequence matching algorithm.

1. Initialize sparse difference matrix and score matrix with a high value (Lines 2-3).

2. Build a tree using the images of our map (Line 5).

3. Find N best nearest neighbor locations for the current view of the robot (Line 8).

4. Update the sparse difference matrix with difference values returned by ANN function for each of the N locations (Line 9).

5. For each probable loop closure locations, calculate the minimum difference score and update the robot's current location (Lines 14-22).

6. Finally, find the best matching location for the current robot view, and normalize the matching value (Lines 23-26).

In the following subsections, we describe three main components our algorithm in detail which are finding approximate nearest neighbors (Line 8), building a sparse difference matrix (Line 9) and sequence matching (Lines 13-21).

3.1 Approximate Nearest Neighbor

For finding the approximate nearest neighbor (ANN) for an image, we use FLANN by Muja and Lowe [47]. We control the accuracy of the result by setting its precision parameter. FLANN selects the ANN parameters automatically for a given

Algorithm 1: Algorithm for Finding Matches

Input:
Q = Images in the robot views
M = Images in the map
Output:
matches = A vector containing matching indices and matching values

Procedure `findMatches(`*Q, M*`)`
  global D = initialize(len(M), len(Q))
  global S = infinity(size(D)) ▷ matrix for score values
  matches = []
  T = Tree(M)
  foreach *Image* I_i *in* Q **do**
    g_i = descriptor(I_i)
    N-matches, distances = ANN(T, g_i, precision)
    D = updateDifferenceMatrix(D, i, N-matches, distances)
    K-locations = N-matches(1:K)
    probable-loop-closures = K-locations $\cup$ locations
    locations = []
    foreach *location j in probable_loop_closures* **do**
      min_score = $\min_{v_{min},\ldots v_{max}} score(j,T,V)$
      $S_{j,T}$ = min_score
      estimated_velocity = $\arg\min_{v_{min},\ldots v_{max}} score(j,T,V)$
      updated-location = j+estimated_velocity
      if *min_score* $<$ *max_score* **then**
        locations.add(updated-location)
      end
    end
    value = $\min_{j=1,2\cdots m}(\hat{S}_j^T)$
    value = normalize(value)
    index = $\arg\min_{j=1,2\cdots m}(\hat{S}_j^T)$
    matches(T) = (index,value)
  end
  return matches
Procedure `score` (*j, T, V*)
  $S = \sum_{t=T-d_s}^{T} D_k^t$ where, $k = j + V(t-T)$
  return S

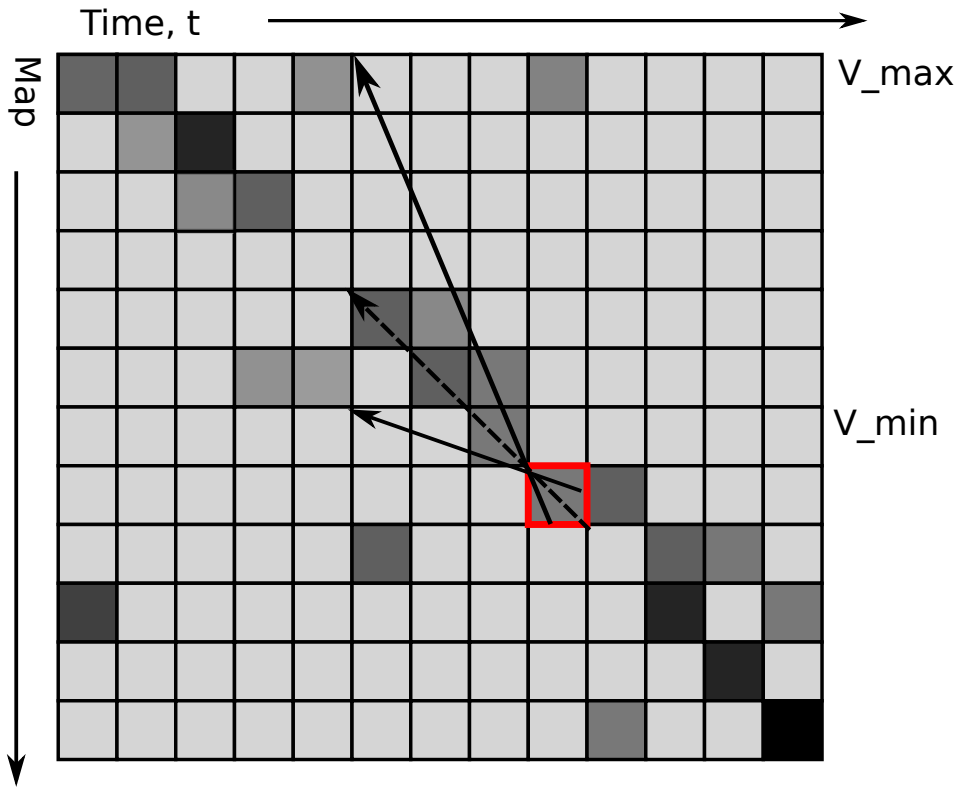


Figure 3.1: The figure shows a sparse difference matrix where $N = 2$. The brighter a cell is, the larger the difference value. Trajectory line with maximum velocity and trajectory line with minimum velocity are shown. We calculate 2 nearest neighbors for each node (only two cells in each column are darker than others). The dotted line is the trajectory which has minimum difference score for the red bordered cell.

dataset and a target accuracy. It uses either randomized k-d trees [68] or hierarchical k-means tree [56] algorithm. One of the factors that has a great impact on the nearest neighbor matching is data dimensionality. Muja and Lowe demonstrated speedup by a factor of 10^3 for data with dimensions from 10^1 to 10^3 and a target accuracy of 80%. In our study, the dimensions of the image descriptors are under 1024.

In one implementation of our algorithm, we first down-sample each image to 32×32 and then extract features using the HOG [15] descriptor, which has 334 dimensions. Alternatively, we can also use raw images as their descriptors as in SeqSLAM so that a descriptor has 1024 dimensions. HOG has a smaller dimension size and therefore less ANN search time, than the raw image descriptor.

3.2 Sparse Difference Matrix

We initialize the difference matrix to large values (bright cells in Fig. 3.1. We use FLANN to create a tree structure to store the map image descriptors to facilitate efficient search. For a map or traversal of n images, each image can find its approximate nearest neighbors in the map in $O(\log(n))$ time, vs. $O(n)$ in the case of linear search. The distance values returned by the ANN algorithm are used to replace the initial values whenever they are computed (where N=2 for the example in Fig. 3.1). Consequently, our difference matrix is sparse containing distances to (approximate) N nearest neighbors along each column (row).

3.3 Sequence Matching

We calculate minimum difference score only from the locations in the map where the best K matches for the current view of the robot are found while we have a total N matching locations (Algorithm 1 Line 10). Then we update the robot's current location using robot's current velocity. We update robot's location so that the loop-closure node between two sequences are temporally continuous. In following, we describe how we calculate difference score and greedily update robot's location.

3.3.1 Difference Score

Each column of our difference matrix $\mathbf{D}$ is a difference vector. Difference vector $\hat{\mathbf{D}}^{(T)}$, where T is the current time, contains N valid image difference values for an image I_{2j} with I_{1T} where $j = 1,2\ldots m$. I_{2j} is an image in the map and I_{1T} is the current robot view. We search for a sequence of images in the map along the rows of $\mathbf{D}$ which best match with the sequence, $I_{1,T-ds}, I_{1,T-1}\cdots I_{1,T}$, across the columns of the robot views in $\mathbf{D}$ the robot views where d_s is the length of sequence. To recognize the current sequence of robot views in the map, a search is performed through the space $\mathbf{M}$ of image difference vectors.

$$\mathbf{M} = \left[\hat{\mathbf{D}}^{\mathbf{T-d_s}}, \hat{\mathbf{D}}^{\mathbf{T-d_s+1}} \ldots \hat{\mathbf{D}}^{\mathbf{T}}\right] \tag{3.1}$$

In the example in Fig. 3.1, different trajectory lines that originate from the red

bordered cell are drawn for different velocities. We calculate difference score S of a trajectory line by summing up the difference values the line passes through. Using Equations (3.2) and (3.3), we calculate difference score for a particular velocity V. In Equations (3.2) and (3.3), D^t_j is the difference value between the robot view t and map image j, and V is the trajectory velocity.

$$S = \sum_{t=T-d_s}^{T} D^t_j \tag{3.2}$$

$$j = s + V(t - T) \tag{3.3}$$

In Algorithm 1, we use the procedure $score(j, T, V)$ in Lines 32-34 to encapsulate the calculation by the above equations.

3.3.2 Outlier Removal

Each time we get the nearest neighbor image from the map, we also check how many recent images of the robot view matched with the nearest neighbor image from the map. If this image from the map previously matched with the other images of the robot view, it is not the true match. So, we penalize the current matching score if some image from the map appears very frequently. We keep a count of how many images in the robot views matched with the same image from the map over a period of time and if the count exceeds a certain threshold then we restore the matching score to its initial value. Fig. 3.2 shows a difference matrix with a red border row. It appears that a single image from the map has matched frequently with the images in the robot view. So, we ignore this matching score and restore it to its initial one.

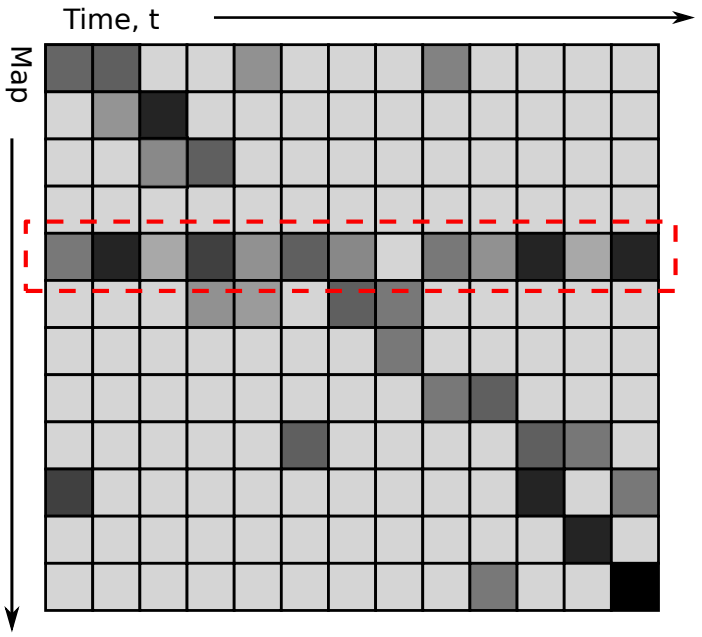
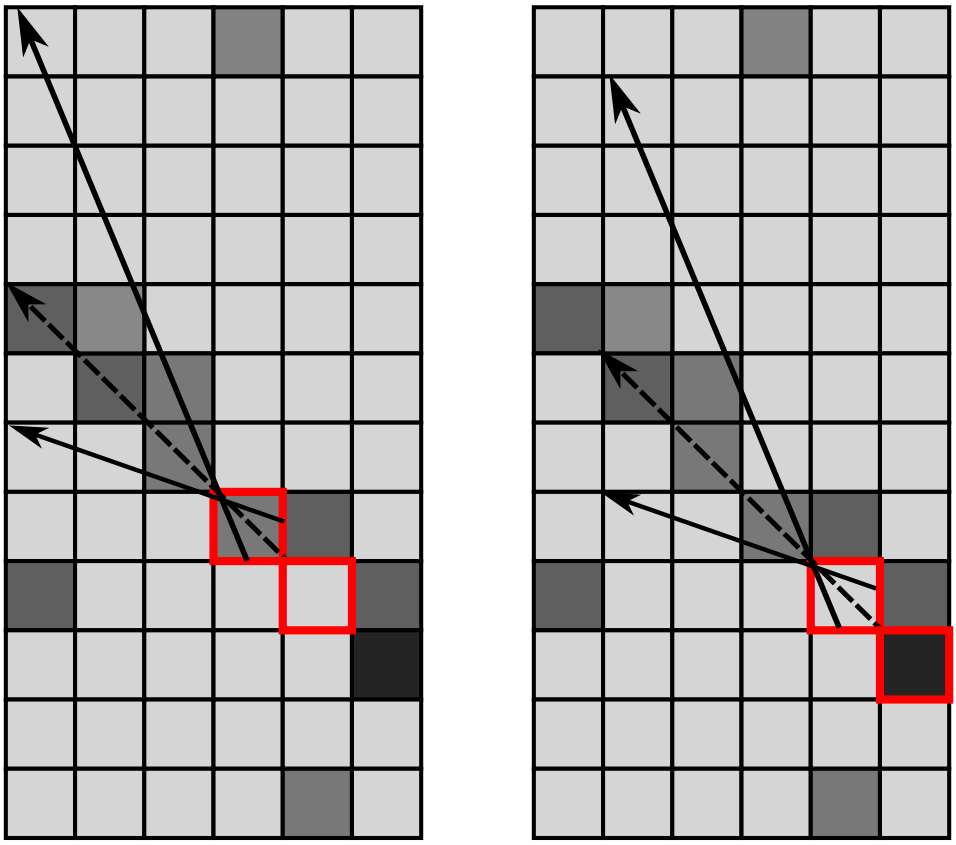


Figure 3.2: In the figure, highlighted row is an outlier. It shows that one image in the map matched with each image in the robot views which should not be the case.

3.3.3 Greedy Motion Model Estimation

To find continuous or temporally consistent loop-closure nodes efficiently, we use a greedy motion model estimation technique. For a current sequence of robot views, we find the best sequence of images in the map. For each probable loop closure node, we calculate a difference score for different trajectories where each trajectory line corresponds to a velocity or motion model. If a node is actually a loop closure node, the robot should move along the trajectory defined by the velocity from previous nodes in the sequence. In other words, the best fit trajectory is the one that has the minimum difference score. We can update the robot's current location using the motion model or velocity that corresponds to the best-fit trajectory.

We only update the robot's current location when the best-fit trajectory line has at least one nearest neighbor; the trajectory is otherwise abandoned. In Fig. 3.3a, the dotted trajectory line has the min-difference-score, and we use that velocity to update robot's current location. Fig. 3.3b shows the current updated location at the red bordered cell.



(a) (b)

Figure 3.3: In Fig. 3.3a, the dotted black trajectory has the minimum difference-score for the red bordered cell. Red bordered cells are current and next locations of the robot. We follow the min-difference score trajectory line to update robot's current location (next red bordered cell). In Fig. 3.3b, we show the updated location.

Chapter 4

Experimental Results of Proposed Place Recognition Algorithm

In this chapter, we demonstrate the effectiveness of our algorithm in place recognition application.

4.1 Datasets

We run our algorithm for loop-closure detection on three different datasets used commonly in previous studies of visual loop closure detection in SLAM, each of which consisted of at least two traversals of the same route but at different times and, as a result, all datasets involve severe illumination changes. The datasets are as follows:

- Nordland Dataset, train ride in northern Norway,
- UA dataset, from the University of Alberta, Edmonton, Canada, and
- Garden Points Walking, from the Queensland University of Technology, Brisbane, Australia.

The Nordland dataset [69] contains four traversals of the same route in winter, spring, summer and fall. It is a 728 km long train ride in northern Norway and has a total 37500 images. The UA dataset was collected by a Husky robot with a normal RGB camera and the Garden Points Walking dataset is collected by Arren Glover

and has two traversals, one during day and the other at night. All these datasets have ground truth information to allow easy performance evaluation. We compared our algorithm with OpenSeqSLAM [69], an implementation of SeqSLAM, in terms of metrics for accuracy and efficiency as described in detail below.

Table 4.1: Parameters

Parameter	Description and Values
R_x, R_y	Reduced image size for all the datasets, we used $[R_x, R_y]$ = [32,32] in all the datasets
Cell size	For HOG, we used Cell size = [8,8]
d_s	Trajectory length in numbers, we used d_s = 20 in Nordland dataset, d_s = 30 in UA and d_s = 20 in Garden points
N	Number of nearest neighbor matches we calculate for current view, we used N = 100 in Nordland dataset, N = 10 in UA and N = 10 in Garden walking dataset
K	Number of best matching locations among N locations where we search for the best matching sequence, K = 5 in Nordland dataset, K = 2 in UA and K = 2 in Garden points
V_{min}	Minimum trajectory speed, $V_{min} = .4V_{ave}$
V_{max}	Maximum trajectory speed $V_{max} = 1.5V_{ave}$

4.2 Evaluation Criteria for Place Recognition

Place recognition systems are typically evaluated using precision and recall metrics. The relation between precision and recall is expressed by a precision-recall curve. For each loop closure node, we get a confidence value. We can select matches based on a particular confidence measure. The correct matches are known as true positives, the incorrect matches are false positives, and matches that the system cannot detect are false negative matches. Precision is defined as the proportion of selected matches that are true positive matches, and recall is the proportion of true positives to the total number of actual matches.

$$Precision = \frac{TP}{TP+FP}$$
$$Recall = \frac{TP}{TP+FN}$$

On precision-recall measurement, a perfect system will have the precision of 100% and recall of 100%. Precision and recall are related to each other via a precision-recall curve, which plots recall against precision for a range of confidence values. Precision-recall curves help to tune the system as we need. For example, some systems (e.g. topological map post-processing step) are very sensitive to the false positive then we have to use a threshold so that we get 100%. As a result, recall at 100% precision was the key metric for place recognition success. However, if we have a system where we rely on multiple sensors for loop closure decision then we can choose a different threshold to find more loop closure nodes.

4.3 Precision-Recall

In order to measure the accuracy of the different algorithms for loop closure detection, we generate precision and recall values while varying the threshold on the similarity between sequences to determine if a loop closure has occurred. We consider a match as positive if it is within a certain offset distance from the ground truth location.

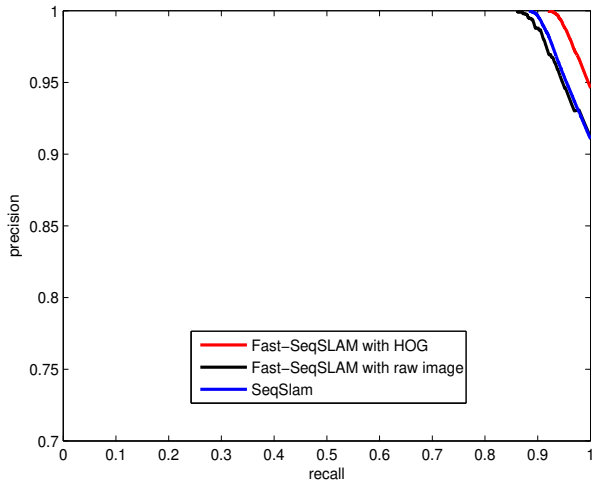


Figure 4.1: Nordland dataset.

We compared our algorithm with SeqSLAM and we plotted the precision-recall curve. As mentioned, we used HOG in our algorithm in order to reduce the di-

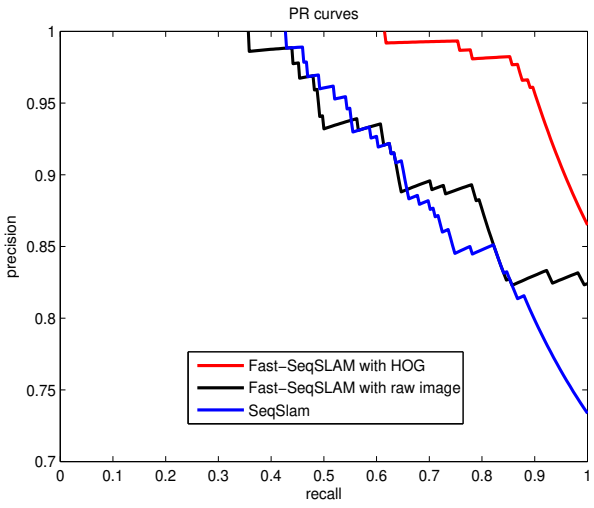


Figure 4.2: Garden Points dataset.

mensionality of the image descriptor and allow efficient and accurate ANN search. However, even if we use the raw image as a descriptor as in SeqSLAM, our Fast SeqSLAM is still able to achieve good performance. Therefore, in our experiments, the following algorithms were included in the comparison, and the results are summarized in Figs. 4.1-4.3.

- Fast-SeqSLAM with HOG (red).
- SeqSLAM (blue).
- Fast-SeqSLAM with raw the image as a descriptor (green).

Fig. 4.1 shows the comparative results for Nordland dataset. All algorithms have almost the same performance. For the Garden Points dataset, in Fig. 4.2, our algorithm with HOG descriptor performs better than SeqSLAM. This result is somewhat surprising as our algorithm is a greedy version of SeqSLAM. We observed that this result is due to the fact that the images in the dataset are captured closely with each other and, as a result, the contrast enhancement step on the difference matrix in SeqSLAM would degrade its performance, as we verified experimentally. Contrast enhancement on the difference matrix did not help SeqSLAM in this case since each image matches with nearby images because of their high similarity. Using the

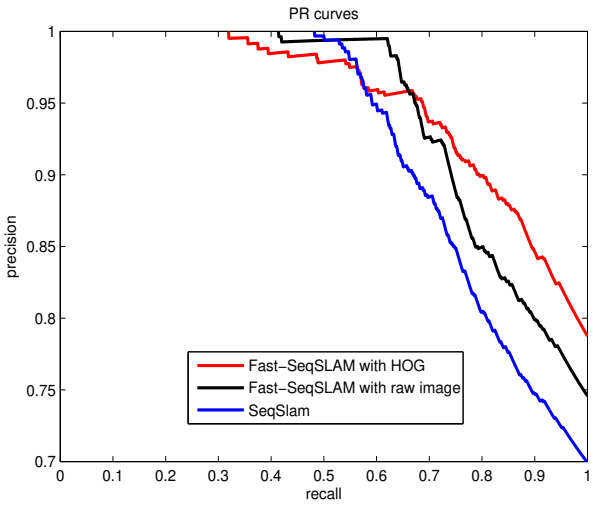


Figure 4.3: UA dataset.

dataset collected on the University of Alberta Campus at two different times of a day with a Husky robot, all three algorithms have a similar performance as shown in Fig. 4.3, with SeqSLAM performing better for low recall values and Fast SeqSLAM better for high recall values. We have given the parameter values we used in our experiments in Table 4.1.

4.4 Computational Complexity and Execution Time

In our algorithm, we use a tree structure to store the image descriptors of a map, and tree construction takes $O(n \log n)$ time. Calculation of the nearest neighbors for the robot's current view has a time complexity of $O(\log n)$. If we use a single nearest neighbor for each image, we will have m total values in the matrix where m is the number of columns in the difference matrix. Since we search sequences greedily and we continue searching until the matching sequence has at least one nearest neighbor. In the worst case, we may calculate a difference score for d_s extra cells in the difference matrix. So we have a time complexity of $O(md_s)$ for finding all the loop closure nodes between the two traversals. Since d_s, the trajectory length, is a constant, we can write $O(m)$ instead of $O(md_s)$. The overall time complexity becomes $O(n \log n + m)$ for finding all loop closure nodes, for a map with n images

and a traversal of m images. If we assume $n \approx m$ in general, then the overall complexity of Fast SeqSLAM is $O(n \log n)$ while SeqSLAM has a time complexity of $O(n^2)$.

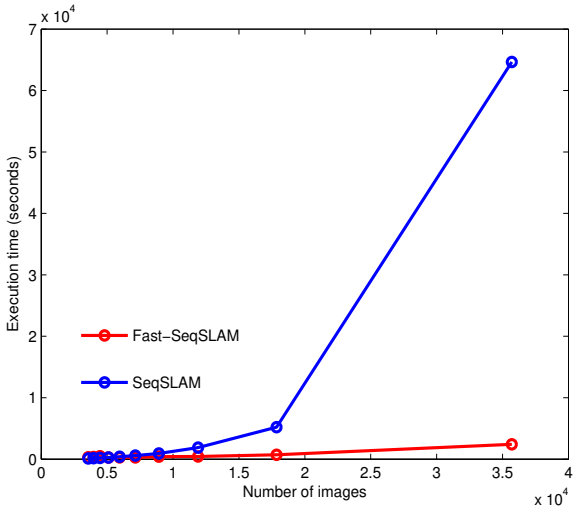


Figure 4.4: Nordland dataset, execution time vs number of images.

Fig. 4.4 demonstrates the comparison of the execution time of our algorithm with SeqSLAM on one dataset using a different number of images of the dataset. The red curve is for our Fast-SeqSLAM algorithm and the blue curve is for SeqSLAM. The y-axis shows the execution time in seconds. For a small map, for example, 5000 images, execution time does not differ much. However, for a large map, for example, 37000 images, our algorithm runs in 40 minutes whereas SeqSLAM takes 18 hours, i.e., Fast-SeqSLAM improves SeqSLAM by a factor of 27 in execution time in this case. Recall that each dataset contains two traversals of the same route, and the above times are what is required to find all corresponding sequences or loop closures between the two traversals.

Chapter 5

Map Merging

In this chapter, we propose a new technique to merge two topological metric maps which are built independently by two robots. We use our Fast-SeqSLAM algorithm proposed in Chapter 3 with switchable pose constraints optimization technique [71].

To work efficiently in the context of multiple-robot systems, robots need to produce a global map of an environment they explore collaboratively. To build a global map accurately and efficiently, we need to merge all the local maps that individual robots build. We will show in this chapter how Fast-SeqSLAM can be used in solving the problem of map merging in the case of two robots, although it can be used also for multiple robots in general.

Two robots must have some overlapping parts in their individual maps to fuse them into a single common map. To fuse two independent maps into a single global map, the robots need to recognize the common locations of their individual maps. In single robot SLAM system, robots need to recognize the previously visited locations. Similarly, in our map merging problem, robots need to recognize the common locations among the individual maps.

5.1 Related Work

Map merging is different from cooperative mapping, where multiple robots concurrently and continuously contribute their data to a single map [77]. While map

building has attracted plenty of research, there are only a few works in map merging that are available for appearance-based map [19]. In our work, we combine multiple maps that have been independently built by different robots.

Erinc and Carpin proposed an anytime algorithm [19] for combining multiple appearance-based maps which are built by multiple robots independently. They also proposed a new metric to evaluate the quality of merged maps, based on algebraic connectivity [21]. They used bag-of-words technique to find the similarity between the images by counting the number of common words in these images. They only established edges between the common nodes in the two maps based on visual similarity and maximizing the algebraic connectivity. However, they do not propose how to convert two maps into a single one.

Ho and Newman proposed an algorithm to find the similar images between the two maps [28]. A visual similarity matrix is constructed for each pair of robots. Then, simply, a 2D transformation is estimated between the maps (using principal moment alignment) and convert them into a single one. However, the solution has a time complexity of $O(n^3)$, where n is the number of observations [28].

5.2 Proposed Map Merging Technique

We extend our Fast-SeqSLAM algorithm to solve the multi-robot map merging problem. The main step in multi-robot map merging is finding the common places in multiple maps or place recognition across multiple maps. We can find the common places across multiple maps applying our Fast-SeqSLAM algorithm and the time complexity will be $O(n \log n)$, where n is the largest number of images among the multiple maps.

Fig. 5.1 shows the major steps of using our map merging solution. It takes two topological metric maps as the input. Each map contains multiple locations or nodes in a graph, and each node is characterized by an image captured at the node. Assuming that the two maps share common locations, then a key step in map

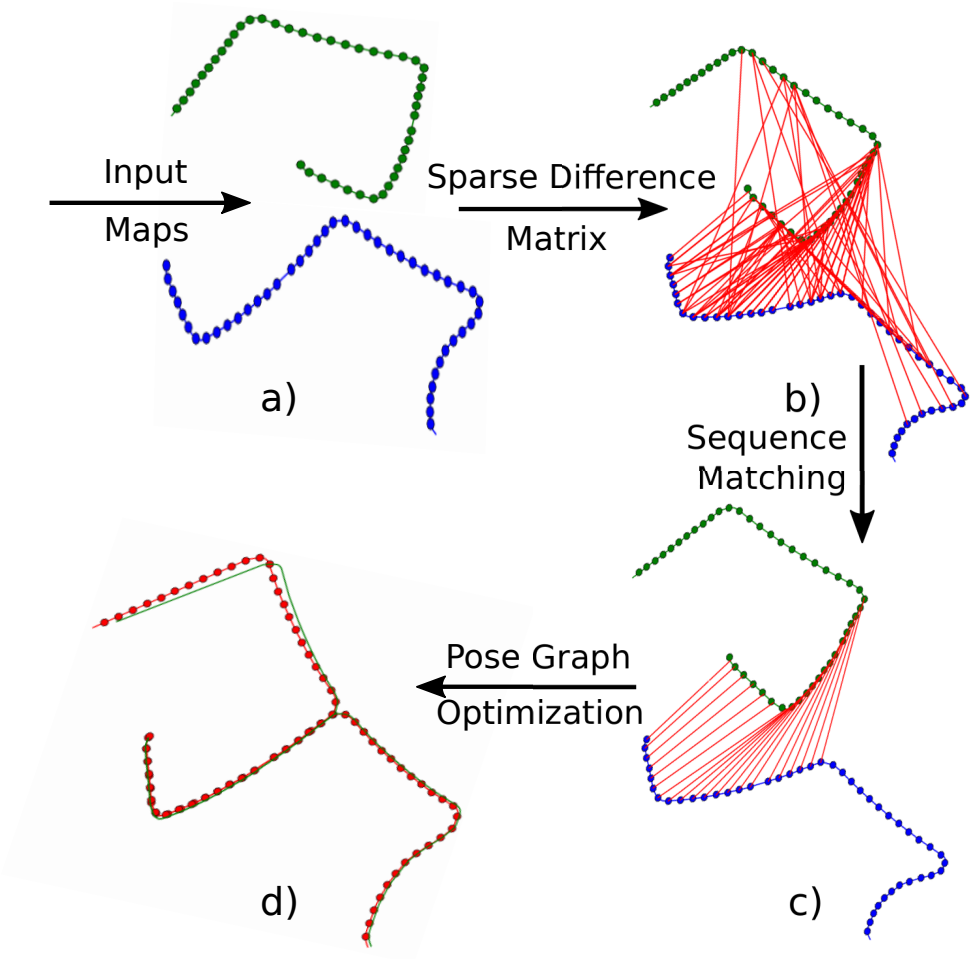


Figure 5.1: Overview of our map merging algorithm.

merging is to find all the pairs of nodes that correspond to the same locations. With the correspondence nodes, we can use a pose graph SLAM algorithm such as g2o to merge the two maps (see Section 5.2.2).

5.2.1 Sparse Difference Matrix

In our map merging setup, we assume that two robots will share their individual map and the system will merge the two maps offline. So, we calculate difference matrix offline. We use Algorithm 2 to create a sparse difference matrix. We first extract hog features for all the images in map 1 and map 2 and pass these as the input to our algorithm. We get a sparse difference matrix D as an output from the algorithm. Tree function in line 2 in Algorithm 2 will return a tree for k-d tree

Algorithm 2: Algorithm for Difference Matrix

Input:
$G1 = \{g_{11}, g_{12} \cdots g_{1n}\}$
$G2 = \{g_{21}, g_{22} \cdots g_{2m}\}$ ▷ $g_{i,j} = Feature(I_{i,j})$
Output:
$D = \{d_{i,j}\}$ ▷ $d_{i,j}$ = distance between image $I_{1,i}$ and $I_{2,j}$
▷ $i = 1, 2 \ldots n$ and $j = 1, 2 \ldots m$

Procedure `getDifferenceMatrix(`$G1, G2$`)`
  $T1 = Tree(G_1)$
  $T2 = Tree(G_2)$
  for $i \leftarrow 1$ ***to*** n **do**
    $neighbour, distance = ANN(g_{2i}, T_1, precision)$
    $S_1 = S_1 \cup (i, neighbour)$
    $d1[i] = distance$
    $neighbour, distance = ANN(g_{1,i}, T_2, precision)$
    $S_2 = S_2 \cup (neighbour, i)$
    $d2(i) = distance$
  end
  $D = initialize(m, n)$
  $S = S_1 \cap S_2$
  foreach (i, j) *in the set* S **do**
    $D(i, j) = (d_1(i) + d_2(j))/2$
  end
  return D

algorithm [6]. The ANN function in line 5 and 8 returns the nearest neighbor for each image with the distance. S_m is a set which contains all the pairs (i, j), where image j is the nearest neighbor of image i. d_m is a vector which contains distances for the nearest neighbors for each image in map m, where $m = \{1, 2\}$. After finding the nearest neighbors for each node, we find the mutually consistent nodes (line 15). Then we initialize the difference matrix with a high difference value in line 12 and use the distance returned by the ANN function to build the sparse difference matrix in line 14-16. After creating the difference matrix, we search for the best matching sequences.

To find the corresponding nodes of the two maps, we use the same concept of our Fast-SeqSLAM algorithm. In Fig. 5.1b, the sparse difference matrix is computed from one map to the other where red lines connect the nearest neighbors. In Step c, we perform sequence matching and use the result to identify corresponding nodes between the two maps, as represented by the surviving red lines in Fig. 5.1c. Finally, we apply pose graph SLAM (Vertigo, a robust version of g2o in this case) to merge maps, considering the relative transformation between the two maps as free variables. Fig. 5.1d demonstrates the result of merging two maps which are built independently by the two robots in an environment. The green line in Fig. 5.1d shows the ground truth map.

5.2.2 Pose Optimization

We find loop-closures using our Fast-SeqSLAM algorithm. Our loop-closure detection algorithm also tells us how confident these loop-closure nodes are. We use these confidence values as initial values for the switch variables in a switchable pose constraint optimization [71]. These switch variables help the loop closure factors to be switched on or off.

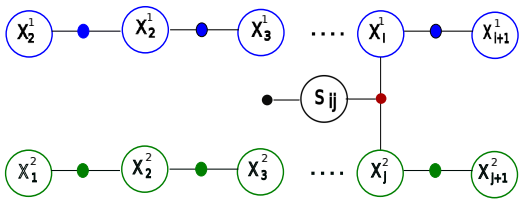


Figure 5.2: In the topological map, the blue nodes are from map 1 and green nodes are from map 2. We use switchable constraint only for the loop closure nodes. The loop closure constraint (red) is controlled by switch variable (black) $s_{2,i}$. The loop closure constraint is switched on or off depending on the value of the switch variable $s_{i,j}$ i.e. it is activated or deactivated as part of the optimization process. Prior factor (black) that controls the switch variable by penalizing deactivation of loop closures [71].

$$
\begin{aligned}
X^*, S^* = \arg\max_{X,S} & \underbrace{\sum_i \| f(\mathrm{x}_i, \mathrm{u}_i) - \mathrm{x}_{i+1} \|^2_{\Sigma_i}}_{\text{Odometry Constraints}} \\
& + \underbrace{\sum_{ij} \| \Psi(s_{ij}) \cdot (f(\mathrm{x}_i, \mathrm{u}_{ij}) - \mathrm{x}_j \|^2_{\Lambda_{ij}}}_{\text{Switchable Loop Closure Constraints}} \\
& + \underbrace{\sum_{ij} \| \gamma_{ij} - s_{ij} \|^2_{\Xi_{ij}}}_{\text{Switch Prior Constraints}}
\end{aligned}
\tag{5.1}
$$

In Equation 5.1, x_i is the node and u_i is the odometry pose constraint from nodes x_i to x_{i+1}. The difference, $\| f(\mathrm{x}_i, \mathrm{u}_i) - \mathrm{x}_{i+1} \|$, measures how well the parameters x_i and x_{i+1} satisfy the constraint u_i. Similarly, this difference $\| (f(\mathrm{x}_i, \mathrm{u}_{ij}) - \mathrm{x}_j \|$ measures how well the parameters x_i and x_j satisfy the loop closure constraints. Additionally, the loop closure constraints have been augmented by a multiplication with the switch function, $\Psi(s_{ij})$. This switch function is defined as $\Psi : R \rightarrow [0,1]$, mapping from the continuous real numbers to the interval $[0,1]$. In the topological map, the switch function can be interpreted as it can enable or disable the constraint edge that it is associated with, i.e. it can completely remove the constraint in the minimization function.

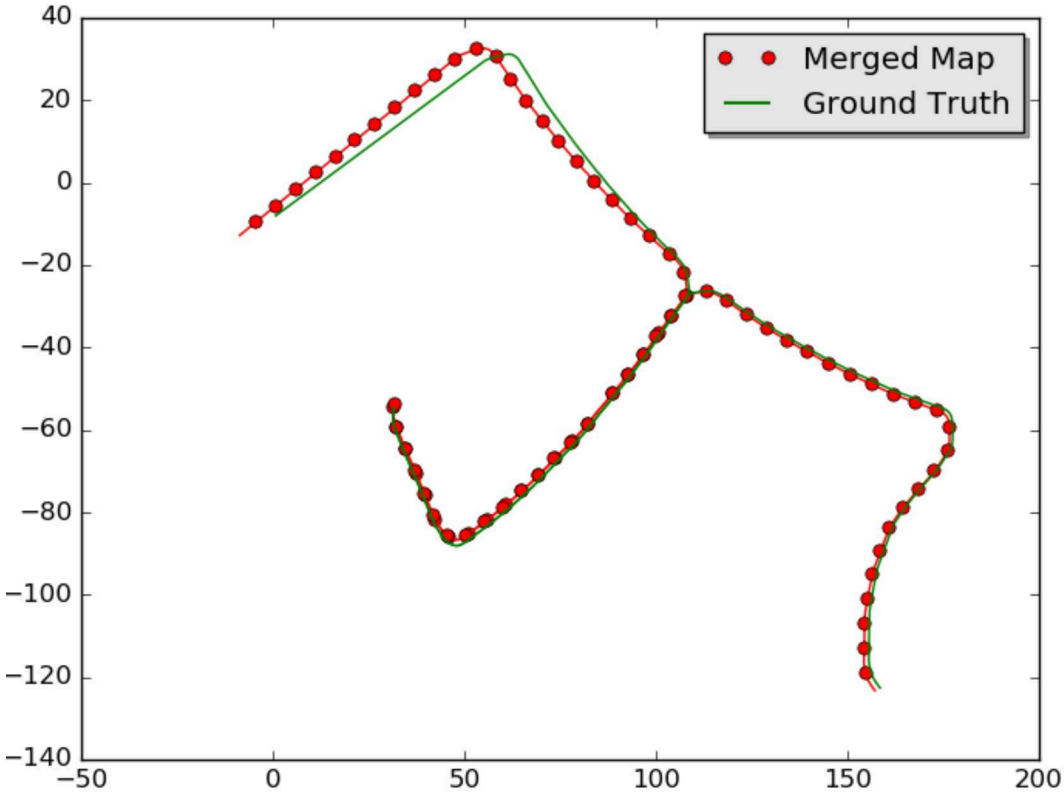


Figure 5.3: Fig. shows the result of our map merging technique on the UA2 dataset. The green line is the ground truth and the red line is the merged map.

5.3 Map Merging Results

We run our map merging technique on the UA2 dataset. We collected images, odometry and GPS data using a Husky robot. Both loops have a common path in their traversals. We found the loop closure nodes by applying our Fast-SeqSLAM algorithm. Then we applied switchable pose constraint optimization to convert two maps into one single map. The result of our map merging technique is shown in Fig. 5.3 on the UA2 dataset. The green line is the ground truth (using GPS data) and the red line represents the merged map.

Map merging result is directly dependent on the place recognition result. We have demonstrated that Fast-SeqSLAM is robust in case of severe illumination change like the SeqSLAM and runs faster than SeqSLAM. However, in this thesis, we did not compare our map merging technique experimentally with the existing map merging algorithms. We posed the multi-robot map merging problem as a special case of a place recognition problem.

Chapter 6

CONCLUSIONS

In this thesis, we have proposed an efficient version of SeqSLAM, that we refer to as Fast SeqSLAM. Our proposed algorithm is able to retain the key advantages of SeqSLAM in terms of place recognition accuracy. At the same time, it is much more efficient than SeqSLAM, speeding it up as much as by a factor of 27, on a map with 37K keyframes.

Our algorithm is greedy, and its key ideas that allow us to achieve this level of efficiency are (a) we avoid computing a complete difference matrix between the map images and those observed by the robot, and (b) possible matching sequences in the map are searched strategically and selectively rather than exhaustively. So, it is possible to use any descriptor with our system. In fact, it is possible to develop an online version of SeqSLAM based on these two key ideas. We are leaving this as our future work. The implementation of Fast-SeqSLAM is available at: https://github.com/siam1251/Fast-SeqSLAM. It is written in Matlab.

Bibliography

[1] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer, 2008.

[2] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037, 2008.

[3] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[4] Hernán Badino, Daniel Huber, and Takeo Kanade. Real-time topometric localization. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1635–1642. IEEE, 2012.

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[6] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[7] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[8] Rodney Brooks. Visual map making for a mobile robot. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 824–829. IEEE, 1985.

[9] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2012.

[10] Tzu-Chuan Chou and Meng Chang Chen. Using incremental plsi for threshold-resilient online event analysis. *IEEE transactions on Knowledge and Data Engineering*, 20(3):289–299, 2008.

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

[12] Peter Corke, Rohan Paul, Winston Churchill, and Paul Newman. Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2085–2092. IEEE, 2013.

[13] 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.

[14] Mark Cummins and Paul Newman. Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011.

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[16] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[17] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. Robotic exploration as graph construction. *IEEE transactions on robotics and automation*, 7(6):859–865, 1991.

[18] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[19] Gorkem Erinc and Stefano Carpin. Anytime merging of appearance-based maps. *Autonomous Robots*, 36(3):241–256, 2014.

[20] SR Esterby, , and AH El-Shaarawi. Inference about the point of change in a regression model. *Applied Statistics*, pages 277–285, 1981.

[21] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[22] David Filliat. A visual bag of words method for interactive qualitative localization and mapping. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3921–3926. IEEE, 2007.

[23] David Filliat and Jean-Arcady Meyer. Map-based navigation in mobile robots:: I. a review of localization strategies. *Cognitive Systems Research*, 4(4):243–282, 2003.

[24] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[25] Paul Furgale and Timothy D Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010.

[26] J-S Gutmann and Kurt Konolige. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*, pages 318–325. IEEE, 1999.

[27] Peter Hansen and Brett Browning. Visual place recognition using hmm sequence matching. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4549–4555. IEEE, 2014.

[28] Kin Leong Ho and Paul Newman. Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74(3):261–286, 2007.

[29] Laurent Itti and Pierre Baldi. A principled approach to detecting surprising events in video. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 631–637. IEEE, 2005.

[30] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[31] Kurt Konolige and Motilal Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008.

[32] Hemanth Korrapati, Jonathan Courbon, Youcef Mezouar, and Philippe Martinet. Image sequence partitioning for outdoor mapping. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1650–1655. IEEE, 2012.

[33] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems*, 8(1):47–63, 1991.

[34] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[35] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170, 2015.

[36] Yang Liu and Hong Zhang. Towards improving the efficiency of sequence-based slam. In *2013 IEEE International Conference on Mechatronics and Automation*, pages 1261–1266. IEEE, 2013.

[37] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[38] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016.

[39] Will Maddern, Michael Milford, and Gordon Wyeth. Cat-slam: probabilistic localisation and mapping using a continuous appearance-based trajectory. *The International Journal of Robotics Research*, 31(4):429–451, 2012.

[40] Christopher Mei, Gabe Sibley, Mark Cummins, Paul M Newman, and Ian D Reid. A constant-time efficient stereo slam system. In *BMVC*, pages 1–11, 2009.

[41] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE, 2001.

[42] Michael Milford. Vision-based place recognition: how low can you go? *The International Journal of Robotics Research*, 32(7):766–789, 2013.

[43] Michael Milford, Chunhua Shen, Stephanie Lowry, Niko Suenderhauf, Sareh Shirazi, Guosheng Lin, Fayao Liu, Edward Pepperell, Cesar Lerma, Ben Upcroft, et al. Sequence searching with deep-learnt depth for condition-and viewpoint-invariant route-based place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 18–25, 2015.

[44] Michael J Milford and Gordon F Wyeth. Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Transactions on Robotics*, 24(5):1038–1053, 2008.

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

[46] Michael J Milford, Gordon F Wyeth, and David Prasser. Ratslam: a hippocampal model for simultaneous localization and mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 403–408. IEEE, 2004.

[47] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[48] Raul Mur-Artal, JMM Montiel, and Juan D Tardós. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[49] Ana Cris Murillo, José Jesús Guerrero, and C Sagues. Surf features for efficient robot localization with omnidirectional images. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3901–3907. IEEE, 2007.

[50] Ana Cris Murillo and J Kosecka. Experiments in place recognition using gist panoramas. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 2196–2203. IEEE, 2009.

[51] Liz Murphy and Gabe Sibley. Incremental unsupervised topological place discovery. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1312–1318. IEEE, 2014.

[52] Tayyab Naseer, Luciano Spinello, Wolfram Burgard, and Cyrill Stachniss. Robust visual robot localization across seasons using network flows. In *AAAI*, pages 2564–2570, 2014.

[53] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[54] Paul Newman, David Cole, and Kin Ho. Outdoor slam using visual appearance and laser ranging. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1180–1187. IEEE, 2006.

[55] Tudor Nicosevici and Rafael Garcia. Automatic visual bag-of-words for online robot navigation and mapping. *IEEE Transactions on Robotics*, 28(4):886–898, 2012.

[56] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. IEEE, 2006.

[57] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[58] 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.

[59] Edward Pepperell, Peter I Corke, and Michael J Milford. All-environment visual place recognition with smart. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1612–1618. IEEE, 2014.

[60] Lawrence Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition. 1993.

[61] Ananth Ranganathan. Pliss: Detecting and labeling places using online change-point detection. *Robotics: Science and Systems VI*, 2010.

[62] Ananth Ranganathan, Shohei Matsumoto, and David Ilstrup. Towards illumination invariance for visual localization. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3791–3798. IEEE, 2013.

[63] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[64] Grant Schindler, Matthew Brown, and Richard Szeliski. City-scale location recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007.

[65] Hagit Shatkay and Leslie Pack Kaelbling. Learning geometrically-constrained hidden markov models for robot navigation: Bridging the topological-geometrical gap. *Journal of Artificial Intelligence Research*, 16(1):167–207, 2002.

[66] Christian Siagian and Laurent Itti. Biologically inspired mobile robot vision localization. *IEEE Transactions on Robotics*, 25(4):861–873, 2009.

[67] Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research*, 2010.

[68] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

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

[70] Niko Sünderhauf and Peter Protzel. Brief-gist-closing the loop by simple means. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1234–1241. IEEE, 2011.

[71] Niko Sünderhauf and Peter Protzel. Switchable constraints for robust pose graph slam. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884. IEEE, 2012.

[72] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[73] Gabriel Tsechpenakis, Dimitris N Metaxas, Carol Neidle, and Olympia Hadjiliadis. Robust online change-point detection in video sequences. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, 2006.

[74] Christoffer Valgren and Achim J Lilienthal. Sift, surf and seasons: Long-term outdoor localization using local features. In *EMCR*, 2007.

[75] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[76] Matthew R Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. Learning semantic maps from natural language descriptions. Robotics: Science and Systems, 2013.

[77] Xun S Zhou and Stergios I Roumeliotis. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1785–1792. IEEE, 2006.