Efficient Visual Search in Appearance-based SLAM

by

Kiana Hajebi

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

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

Simultaneous localization and mapping (SLAM) in an unknown environment is a prerequisite to have a truly autonomous mobile robot. In this thesis, we focus on appearance-based visual SLAM, for which we develop a graph-based nearest-neighbor search algorithm to speed up bag-of-words (BoW) image retrieval. In appearance-based SLAM, the robot uses the visual appearance of the locations taken along its route to build the map of the environment and localizes itself within this map by recognizing places it has visited before. To solve the challenging problem of appearance-based place recognition (*a.k.a.* loop closure detection in the context of SLAM) in large-scale environments, we employ the bag-of-words model, because of its efficiency in image representation and retrieval. Moreover, because the complexity of BoW does not grow with the size of the dataset, as much as that of other search techniques (*e.g.* direct feature matching) do, it can be employed for large-scale image search applications. Although BoW provides an efficient search technique, its vector-quantization (VQ) step can be computationally expensive in order to provide sufficient discriminating power for image matching.

In order to speed up the VQ process, we propose a graph-based nearest neighbor search method (GNNS) that builds a k-NN graph index, and is efficiently integrated into the SLAM system. The k-NN graph is constructed over the visual words of the vocabulary. At search time, the standard GNNS algorithm starts from a randomly sampled node in the graph and performs hill-climbing until reaching a

local minimum which is then an approximate nearest neighbor of the queried feature. We modify the GNNS algorithm to be initialized judiciously by exploiting the following properties of SLAM and BoW model: (1) the sequential property of images acquired in appearance-based SLAM, (2) the perceptual aliasing problem inherent to BoW. This smart initialization of GNNS improves the speedup of the vector-quantization considerably. We experimentally show that our search method outperforms the popular KD-trees and hierarchical k-means algorithms.

Furthermore, we develop a SLAM system by integrating the BoW's loop closure detection in a Bayesian filtering framework. This probabilistic framework then discards false loop closures by enforcing the temporal coherency of estimation. We run our SLAM system on different datasets to test our loop closure detection in challenging environments. The system successfully detects loop closures with 100% precision and high recall rates in real-time.

Acknowledgements

I would like to sincerely acknowledge my supervisor Prof. Hong Zhang for all the guidance, kindness and support he has provided me throughout this study. I cannot imagine completing my thesis without his patience and continuous support and the freedom he gave me to explore my ideas. His insight and feedback over the course of this research work was invaluable.

I would also like to express my appreciation to my supervisory committee members Prof. Martin Jägersand and Prof. Nilanjan Ray, and other thesis examiners Prof. Martin Müller, Prof. Huosheng Hu and Prof. Biao Huang for reading my thesis and providing me with valuable comments.

In addition, I would like to thank all the staff at the Department of Computing Science for their assistance and support. I would like to thank my colleagues in the Robotics lab, CIMS lab and RLAI lab for their help, discussions and the great time we had over my PhD years. I am also grateful to all my friends who have been there for me whenever I needed advice, support, encouragement and motivation.

And last, but not least, I would like to thank my family for their never-ending love, support and encouragement along the way.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

SLAM:	Simultaneous Localization and Mapping
VQ:	Vector-Quantization
BoW:	Bag-of-Words
SIFT:	Scale Invariant Feature Transform
LSH:	Locality Sensitive Hashing
HKM:	Hierarchical K-means
LCD:	Loop Closure Detection
LC:	Loop Closure
MVG:	Multiple View Geometry
PDF:	Probability Density Function
PA:	Perceptual Aliasing
NN:	Nearest Neighbor
KNN:	K-Nearest Neighbor
ANN:	Approximate Nearest Neighbor
GNNS:	Graph Nearest Neighbor Search
SGNNS:	Sequential Graph Nearest Neighbor Search
RANSAC:	Random Sample Consensus

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Autonomous mobile robots that can navigate over long periods of time will help in the automation of tasks operated by humans. Working in hazardous environments or places out of range of human operators, such as underground and on other planets, can benefit from autonomous mobile robotics.

In many environments, robots have no prior knowledge about their surroundings or any external infrastructure like Global Positioning System (GPS) to assist them localize themselves reliably. In underwater, underground or even indoor environments, GPS is not available or reliable. The navigation problem in unknown environments without any external assistance is known as Simultaneous Localization and Mapping (SLAM). The SLAM problem has been investigated for several decades in mobile robot research field and many developments have been made. Several solutions have been proposed for long-term SLAMs in small-scale environments. Working in a large-scale environment is challenging and has remained unsolved. Solving the SLAM problem for large-scale environments with real-time performance is the purpose of our research.

Autonomous navigation can benefit better from the schemes that work based on the visual appearance of locations rather than geometric information about them. When the robot gets lost or the metric information about its position is erroneous, the robot can still recognize the places through their appearances, and therefore localize itself. This can be achieved by mapping the area based on the appearance of distinct locations. In this research, we employ appearance-based schemes to design our SLAM algorithm.

Place recognition in the context of SLAM is known as *loop closure detection*, which is a fundamental problem in SLAM. All SLAM algorithms, including ours, aim to solve the loop closure detection problem. The problem that is addressed in this thesis is therefore summarized by the following: in large-scale environments, where the map is large, how can we efficiently determine whether the new observation of the robot matches one of the previously visited locations?

The problems that loop closure detection faces are typically due to the dynamic changes in environment which are caused by changes in illumination, viewpoint or moving objects, or due to the repetitive structures in the environment that make different locations look similar (perceptual aliasing). Many SLAM solutions have been proposed to tackle these problems. However, they are mostly successful in small-scale environments but not efficient for long-term large-scale operations. The largest appearance-based navigation system that we are aware of is the FAB-MAP by Cummins and Newman [2011]. Although they have tried to make different parts of their system scalable, they achieved low recall rates at high precision. We suspect that the degradation in their performance on large datasets might be due to inefficiency in the search index they use for the vector-quantization process and also their handling of perceptual aliasing using Chow Liu trees. A Chow Liu tree is a data structure used to improve the observation model by capturing the correlations among visual words. However, Cummins and Newman [2011] demonstrate that in

large-scale datasets, when the size of the vocabulary is larger, using a Chow Liu tree worsens the likelihood estimates in some fraction of datasets and it is no longer more effective than naïve Bayes at high precision. To address these issues, we propose to use an efficient search structure that can be utilized by a bag-of-words (BoW) model to benefit SLAM. Moreover, we improve BoW's search performance by integrating it with a probabilistic particle filtering scheme. A more detailed overview of our solution is discussed in the next section.

1.2 Solution Overview

The bag-of-words (BoW) method was originally proposed for document retrieval [Jones 1972, Salton and Buckley 1988]. In recent years, the method has been successfully applied to image retrieval tasks in computer vision community [Sivic and Zisserman 2003] and has been extensively used in appearance-based SLAM as a standard method for loop closure detection due to its computational efficiency. BoW represents an image as a sparse vector of visual words, and thus images can be searched efficiently using an inverted index file system [Witten et al. 1999]. Moreover, because the complexity of BoW does not grow with the size of the dataset as much as that of other search techniques (*e.g.* direct feature matching) do, it can be employed for large-scale image search applications.

In large-scale environments, SLAM maps contain a large number of images to match in order to solve the loop closure detection problem. The image search in such large maps is challenging and still an open problem. One problem with BoW-based SLAM in large-scale environments is the size of the vocabulary. Larger vocabularies improve the performance of BoW matching and are essential for large-scale environments. However, the larger the vocabulary, the more expensive the vector-quantization process of the BoW retrieval. A vector-quantization process

finds the nearest neighbor visual word to each image feature. Typically hundreds to thousands of features are extracted from an image and need to be matched against tens or hundreds of thousands of visual words.

Approximate nearest neighbor (NN) search methods such as KD-trees [Bentley 1980], hierarchical k-means (HKM) [Fukunaga and Narendra 1975] and hashing techniques [Indyk and Motwani 1998] have been used to speed up the search, but at the expense of search accuracy. In spite of extensive and continuing research on the NN-search problem, finding a practical solution that works well in high dimensional spaces is still an open problem. In this thesis, we propose a graph-based k-nearest neighbor search algorithm, called GNNS, that outperforms the popular NN-search methods: randomized KD-trees, Locality Sensitive Hashing (LSH) and HKM that are widely used in SLAM systems.

We show how nicely our graph search structure can be integrated into our SLAM system, as it allows us to easily implement methods such as soft quantization [Philbin et al. 2008] or take advantage of the sequential property of SLAM data to speed up the vector-quantization process.

We also develop a SLAM framework which integrates the BoW model with a particle filtering scheme to estimate the likelihood of loop closures by exploiting the temporal coherence between sequential images. We experimentally show that we can successfully detect loop closures with high precision and recall values in real-time and in different challenging environments.

1.3 Thesis Contributions

This section presents the main contributions of this thesis:

– We propose to use a graph-based nearest neighbor search algorithm (GNNS) for visual search in BoW-based appearance SLAM. We empirically show

that GNNS can be used as an effective approximate NN search method and outperforms popular NN search algorithms on real-world and synthetic datasets [Hajebi et al. 2011].

- We provide theoretical guarantees for the accuracy and computational complexity of GNNS algorithm [Hajebi et al. 2011].
- We propose Sequential GNNS (SGNNS) to improve the performance of GNNS for BoW's vector-quantization in SLAM [Hajebi and Zhang 2014]. SGNNS exploits the sequential property of SLAM data and the perceptual aliasing problem in BoW. We show SGNNS outperforms GNNS and the state-of-the-art NN search algorithms. SGNNS can also be used in similar applications that run feature tracking or registration-based tracking [Roy 2015].
- We develop a probabilistic SLAM framework to work with our GNNS-based visual search for efficient loop closure detection. We validate our system by testing on challenging datasets.

1.4 Thesis Outline

The remainder of the thesis is organized as follows. Chapter 2 provides background and literature review on SLAM, BoW image retrieval and nearest neighbor search algorithms. Chapter 3 describes our graph-based search algorithm (GNNS) and evaluates its performance as a stand-alone method. In Chapter 4, we utilize GNNS presented in Chapter 3 in BoW's vector-quantization and provide experimental evaluation. Chapter 5 integrates all components of our new SLAM system and presents the SLAM framework. In Chapter 6, we evaluate our SLAM system on different datasets and finally Chapter 7 concludes the thesis and discusses future directions.

Chapter 2

Background

This chapter provides the background and literature review relevant to our research work. We will review three strands of literature in the following sections: the work on SLAM and specifically appearance-based SLAM, bag-of-words image retrieval and nearest neighbor search.

2.1 Visual SLAM

Simultaneous localization and mapping (SLAM) is a fundamental problem in mobile robotics in which a robot navigates in the environment, builds the map of the environment and at the same time uses the map to localize itself [Smith and Cheeseman 1986]. It is a challenging problem as the robot does not have any prior knowledge about the environment or its pose. It is only given the sensor data, a motion model, and an observation model.

Vision sensors for the SLAM algorithm have been a strong focus of SLAM research. Vision sensors such as cameras have advantages over laser, ultrasonic and sonar range-finders, as they are relatively cheap and easily available and moreover provide a good source of visual information which allows for more meaningful representations of the environment.

Localization and mapping are challenging problems. There is always uncertainty about the robot's actual pose. Sensor readings are noisy, motion models are approximations, and robot actuation is subject to the noise of controllers. A Bayesian inference framework has been proposed to to deal with the uncertainty in SLAM. Various Bayes filters have been proposed, depending on how the environment is represented. The Extended Kalman filter (EKF) and Rao-Blackwellized particle filter are the ones that have extensively been used in SLAM algorithms.

Visual SLAM methods can be categorised into metric/geometric landmarks based and topological appearance-based, which will be described in the next section.

2.1.1 Metric SLAM

Metric or landmark-based SLAM methods represent the map of the environment by 3D-point landmarks which are obtained from the interesting features extracted from sensor readings. The pose of the robot (2D position + heading) is computed with respect to a global reference frame, and is tracked along with the landmarks' positions.

EKF-SLAM

EKF-SLAM, the earliest SLAM algorithm that was introduced by Smith and Cheeseman [1986], applies an extended Kalman filter to the SLAM problem. In EKF-SLAM, a map is composed of point landmarks. The observed landmarks are used to update the mean and covariance of the state vector (*i.e.* the robot pose and the map), which is approximated by Gaussian distributions.

EKF-SLAM has been widely used in SLAM systems. However, there are limitations to EKF-SLAM. It cannot tolerate a high amount of uncertainty.

Because of the linearization technique used in EKF, the filter might diverge if the posterior is highly uncertain. EKF-SLAM needs sufficiently distinct landmarks to approximate the posterior well. If the landmarks are ambiguous or sparse, EKF-SLAM performs poorly in data association. The other drawback is the update time of the state vector, which is quadratic in the number of map landmarks.

FastSLAM

FastSLAM is another SLAM algorithm that is based on Rao-Blackwellized particle filters and was introduced by Montemerlo et al. [2002]. It represents the robot's state-space using a set of particles. Weighted particles then provide a sample-based representation of the PDF (probability density function) over the state of the robot. Each particle contains an estimated robot pose and set of EKFs, one for each landmark in the map, to track the landmarks independently. This is more efficient than EKF-SLAM which maintains only one Gaussian to estimate the location of all landmarks jointly. Using low-dimensional separate EKFs on individual landmarks allows the update time of the filter to be logarithmic in the number of landmarks, unlike EKF-SLAM complexity which was quadratic.

Instead of tracking the most likely data association, FastSLAM maintains a posterior over multiple data associations. This makes FastSLAM more robust to data association problems than other algorithms. FastSLAM does not depend on the assumption of Gaussianness, which is another advantage over EKF-SLAM, as FastSLAM can cope with non-linear robot kinematics as well as highly uncertain robot poses.

Problems of Metric SLAM Approaches

Metric visual SLAM methods suffer from the complexity bottleneck when handling a large number of state variables in large-scale environments. In EKF,

this complexity is due to the increase in the number of landmarks that have to maintain their correlations; while in FastSLAM, it is caused by updating multiple maps. Another limitation observed in metric SLAM is the data loss due to image projections, when the 3D environment is mapped to 2D images. 2D landmarks are then processed to construct a 3D map, and the 3D map is again re-projected into 2D image frames for the purpose of data association.

2.1.2 Appearance-Based SLAM

Although research in metric SLAM has been active and has made improvements, appearance-based SLAM (aSLAM) showed better performance in a variety of indoor and outdoor environments, and has been of more interest during the past decade. In appearance-based SLAM, the environment is modeled topologically using a graph whose nodes are associated with the visual appearance of the locations that have been visited by the robot. The links between nodes represent whether two nodes are accessible from one another. Mapping is done by adding new nodes to the graph. The localization is carried out by matching the current view of the robot to the previously visited map locations. If a match is found, a connection will be established between the new node and its matching node.

The 3D information of locations is not accounted for in aSLAM, as it only works in appearance space and it avoids the hassle of 3D to 2D and 2D to 3D projections of metric SLAMs[1]. Another advantage over metric SLAM is that in aSLAM only the robot poses are estimated along the trajectory, not the landmarks!

[1] However, there are *hybrid* SLAM algorithms which incorporate metric data into the appearance-based SLAM.

2.1.3 Loop Closure Detection

Loop closure detection (LCD) is a classic problem and the core of a SLAM system, which decides whether the current view of the robot matches one of the previously visited locations (closes a loop) or is a new location. In this thesis, we focus on loop closure detection in appearance SLAM. LCD in aSLAM is addressed as an image retrieval task which is performed by matching the new observation to the existing map that contains the images of the previously visited locations. False positive loop closures might happen when the robot associates the current view with a previous location in the map incorrectly, and thereby makes one node in the graph map represent two distinct locations. Falsely connecting unrelated areas is catastrophic and results in inconsistent maps. Perceptual aliasing also generates false positives. When environments are highly self-similar, as with floor tiles or corridors in indoor environments, distinct places look similar. If the SLAM algorithm does not handle this problem properly, false positive loop closures occur.

If robot does not detect a true loop closure (false negative), then there will be two nodes in the map that represent the same location. This is not as catastrophic as wrong loop closures and can be recovered by their neighboring loop closures.

False positive loop closures are usually eliminated in a post-processing step using multi-view geometric verification.

2.1.4 Multi-View Geometric (MVG) Verification

In order to handle wrong loop closures, a post verification step is applied to the result of image matching. After the image search process, a list of matching candidates that have high similarity to the query image is retrieved. The MVG verification step re-ranks the candidates based on their geometric consistency to the query and hence reduces the number of false positive loop closures.

MVG is performed by first identifying the corresponding features between two images (using direct feature matching), and then filtering out the putative matches that do not satisfy the geometric constraints (called *epipolar* constraints). 2D feature points x in the first and x' in the second image, that are the projections of the same 3D point X, should satisfy the relation $xFx' = 0$. F is the Fundamental matrix that describes the epipolar geometry between two views. RANSAC (random sample consensus) [Fischler and Bolles 1981], as a fast and robust estimation algorithm, is used to fit a Fundamental matrix to the corresponding points and identify the inliers/outliers. Images that contain a higher percentage of inliers are geometrically more consistent with the query image.

Although in a large-scale navigation system, the MVG step is inevitable as it improves the performance noticeably [Cummins and Newman 2009], it can be a costly process when the number of matching candidates is high.

2.1.5 Optimization-Based SLAM Back-ends

Filter-based approaches to SLAM (such as EKF-SLAM and FastSLAM) achieve fast estimation results as they only estimate the current/recent state of the robot and marginalize out the previous ones. However, these approaches introduce linearization errors that accumulate over time and might result in SLAM drifts. In contrast, graph-based SLAM algorithms (called graphSLAM in [Thrun and Montemerlo 2006]) have been introduced to solve the *full* SLAM problem, which is an offline problem defined over all robot poses and map features. GraphSLAMs find the optimal solution over the entire past trajectory of the robot and thereby are not suffering from linearization errors. The posterior of the *full* graph SLAM is formulated as a sparse graph. The graph nodes are robot's poses (and/or map

landmarks[2]) and the edges are the spatial constraints between nodes that can be the odometric data between poses or loop closure constraints. These constraints are generally non-linear (due to the effect of the orientation of the robot) and are resolved using non-linear least squares optimization techniques into a globally consistent estimate of the robot trajectory. This category of SLAM algorithms that are solved using graph optimization techniques and are applied to the offline SLAM problem (where the robot states at all time-steps and all measurements are available), is referred to as SLAM back-end. SLAM front-end, in contrast, is involved with online map building and loop closure detection and generating the graph structure of poses.

Pose graph SLAM, formulated as a non-linear least square problem, can be solved efficiently because of the sparsity inherent in the graph structure. However, the optimization can be difficult because of its dependence on the initial estimate of the map. Noisy odometry data can accumulate the error of the global structure of the map over time and make the odometry-based constraints erroneous. Loop closure constraints can be affected by wrong data associations as well, leading to poor initial estimates.

Lu and Milios [1997] first proposed the graph-based formulation for offline SLAM algorithms and performed global relaxation on the map from laser range scan and odometry based constraints. Their method is computationally expensive, as it requires the inversion of a large matrix for a large environment. Duckett et al. [2000] formulate the relaxation problem as energy minimization of a spring-mass system. They simplify and linearize the problem by assuming absolute knowledge of the robot's orientation. Frese et al. [2005] improve the previous work by introducing a multi-level relaxation (MLR) method, that applies Gauss-Seidel relaxation at different levels of map resolution and offers less computational time.

[2]When the nodes represent only poses not landmarks, the graph representation is referred to as *pose graph*

Thrun and Montemerlo [2006] and Folkesson and Christensen [2004] speed up the convergence by graph reduction and perform the linear approximation only on some sections of the graph. Olson et al. [2006] apply a variant of Stochastic Gradient Descent that recovers the robot trajectory even with poor initial estimates. Kaess et al. [2007] propose *incremental smoothing and mapping* (iSAM) that, unlike *full* graph SLAM problem that needs to be solved in batch mode, performs incremental smoothing in which a new observation or pose variable is added to the system without the need to re-build or re-calculate everything from scratch.

The work by [Kümmerle et al. 2011] presents a general framework, called g^2o ("general graph optimization"), to perform optimization for non-linear least square problems like SLAM and BA (bundle adjustment [Triggs et al. 2000]). Similar to [Kaess et al. 2007], g^2o supports incremental optimization. It also exploits the sparsity of the information matrix and therefore can be used to solve large-scale SLAM problems containing around 10K variables (poses + map landmarks).

Although online solutions for pose graph SLAM have been presented, they are not reliable for long-term mapping as the graph grows unbounded in time [Biber and Duckett 2009]. Johannsson et al. [2013] propose a reduced pose graph approach that does not grow with the duration of travel. It only scales with the size of the environment. Although it is efficient for long-term operation in an indoor environment of limited size, it is not suitable for large-scale environments.

In this research, we mainly focus on the SLAM front-end. However, the integration of a back-end would be possible to run in parallel to the front-end, with the addition of inter-frame motion information to our appearance-only based system.

2.2 BoW Image Retrieval

Bag-of-words is a popular model that has been used in image classification, objection recognition, and appearance-based navigation. Because of its simplicity and search efficiency it has been used as a successful method in Web search engines for large-scale image and document retrieval [Sivic and Zisserman 2003, Nister and Stewenius 2006, Philbin et al. 2007].

The bag-of-words model represents an image by a sparse vector of visual words. Image features, such as SIFTs [Lowe 2004], are sampled and clustered (*e.g.* using *k-means*) in order to quantize the space into a discrete set of visual words. The centroid of clusters are then considered as visual words which form the visual vocabulary. Once a new image arrives, its local features are extracted and vector-quantized into the visual words. Each word might be weighed by some score which is either the word frequency in the image (*i.e.* "term frequency" or *tf*) or the "term frequency-inverse document frequency" or *tf-idf* [Sivic and Zisserman 2003]. A histogram of weighted visual words, which is typically sparse, is then built and used to represent the image.

An inverted index file, used in the BoW framework, is an efficient image search tool in which the visual words are mapped to the database images. Each visual word serves as a table index and points to the indices of the database images in which the word occurs. Since not every image contains every word and also each word does not occur in every image, the retrieval through inverted-index file is fast.

2.2.1 Visual Vocabulary Construction and Vector-Quantization

A visual vocabulary is required to vector-quantize image descriptors into sparse representations. Vocabulary is typically built in an offline processing from a training dataset. The sampled feature descriptors from training data are clustered

and the cluster centroids are considered as visual words. The set of visual words form the visual vocabulary. The clustering is usually done through k-means [Sivic and Zisserman 2003] or variations [Philbin et al. 2007, Nister and Stewenius 2006]. K-means has the time complexity $O(kN)$ per iteration, where N and k are the number of data points and clusters, respectively, and thus it is not feasible for large vocabularies ($k >$10K). Nister and Stewenius [2006] have reduced this time complexity to $O(N \log k)$ by using hierarchic k-means (HKM) approach [Gersho and Gray 1991] to create vocabulary trees. Other approaches replace the nearest neighbor search of k-means by approximate search algorithms, such as KD-trees, to reduce the time complexity. Philbin et al. [2007; 2008] and Cummins and Newman [2009] demonstrate that this approach has similar complexity to vocabulary trees but has far better performance.

Vector-quantization as a nearest-neighbor classification maps the high-dimensional feature descriptors into visual words. When the vocabulary is large, the vector-quantization process can be computationally expensive for real-time search. An efficient approximate nearest-neighbor search method is therefore required to speed up the search while maintaining the accuracy.

2.2.2 Perceptual Aliasing and BoW

Perceptual aliasing occurs when different places in the environment are perceived as identical. In this thesis, two types of perceptual aliasing (PA) are referred to. The first type is the natural perceptual aliasing that is caused by the environment, *e.g.* due to repetitive structure that make different locations look similar and thus the place recognition difficult. Figure 2.1 illustrates an example of this type of PA. Down-weighing the words that repeat often in the dataset, by using an inverse-document-frequency (idf) weighting scheme [Sivic and Zisserman 2003], can help to reduce this perceptual aliasing.

Figure 2.1: An example of perceptual aliasing

The other type is the "artificial" perceptual aliasing inherent in the BoW model —the error caused by the vector-quantization process in which feature descriptors are mapped to visual words of the vocabulary. This PA happens when two descriptors from different objects in the scene fall into the same cluster (*i.e.* map to the same visual word), and therefore are forced to match. This problem can be overcome by increasing the size of the vocabulary or by a soft quantization scheme (see Section 2.2.3).

2.2.3 BoW for Large-Scale Image Retrieval

Most state-of-the-art large-scale image retrieval systems rely on the BoW model [Sivic and Zisserman 2003, Nister and Stewenius 2006, Philbin et al. 2007, Zhang et al. 2010b, Philbin et al. 2008, Wu et al. 2009, Jegou et al. 2008; 2007, Wang et al. 2011, Zhou et al. 2013, Zhang et al. 2011, Zhou et al. 2011, Wang et al. 2013a, Zhou et al. 2010, Zheng et al. 2013]. However, the traditional BoW generates many inaccurate matches because of the error in the vector-quantization process where visual features are mapped to visual words, which diminishes the discriminative power of the local descriptors. Two types of solutions have been proposed to tackle this problem. One type tries to improve the discriminative ability of local features. Soft quantization [Philbin et al. 2008, Jegou et al. 2007] is an example in which each local descriptor is mapped to a weighted set of visual

words, *i.e.* its k nearest words, to improve the matching between images. This is in contrast to "hard-assignment" that maps the local descriptor to only one visual word. Hamming embedding by Jegou et al. [2008] is another approach, that attaches to each quantized descriptor a binary signature that encodes the localization of that descriptor within its Voronoi cell. Building high-order features has also been proposed to boost the discriminative power of local features. [Wu et al. 2009] bundle SIFT image features within local MSER regions, while [Zhang et al. 2010b] group the co-occurring features within a certain spatial distance threshold so that the local feature group is scale invariant and repeatable.

The other type of approaches utilize spatial information among features/words in the BoW model. Some methods work as a post-processing step in which the geometric consistency among matched features are verified and the matches that do not satisfy the geometric constraints are discarded. One popular approach is RANSAC [Fischler and Bolles 1981] which is usually utilized to improve the result of BoW [Philbin et al. 2007].

Since the global geometric verification is computationally expensive, it is usually applied only to the top-ranked images obtained after the searching step. Sivic and Zisserman [2003] propose geometric verification by checking the neighboring feature consistency. However, this approach can be sensitive to the background noise. Jegou et al. [2008] integrate weak geometric consistency (WGC) into the inverted index, enforcing descriptors to agree on the scale and orientation. They exploit partial geometrical information without explicitly estimating the full transformation mapping the features from one image to another.

There are other approaches that encode the spatial relationships among features into image representation or an inverted index. Zhang et al. [2011] introduce the *geometry-preserving visual phrase (GVP)*, a set of words in a particular spatial layout, to encode the spatial information among words in an

image into the BoW representation and inverted index. Using GVPs they capture only translation invariance. While it is possible to add more invariance such as scale and rotation, but that increases the dimension of offset space and incurs more computational cost and memory usage. Wang et al. [2011] try to enhance the discriminative power of visual words using the spatial context that is obtained from the statistics in a local neighborhood of invariant features. Cao et al. [2010] propose *ordered bag-of-features* that are obtained by projecting local image features into certain lines and circles to capture some basic geometric information in images. However, when there are too many image features, encoding such geometric relationships can be expensive.

In summary, utilizing geometric information among visual features to improve the performance of BoW's search has been studied in many ways. The methods that perform geometric verification (using RANSAC) in a post-processing step can be very expensive in large-scale applications, as retrieved images are too many to process and RANSAC estimation can be expensive. Other methods that encode the spatial relationships among visual words into the image representation, make the image representation very complex, which increases the image matching time.

2.3 Related Work on Appearance-based SLAM

Appearance-based SLAM (aSLAM) has been gaining more popularity in the past decade. The method makes use of rich appearance information from vision sensors to handle loop closure detections. In this section, we will describe the major research that has been done in this field.

Kuipers [1978] pioneered the concept of topological maps for mobile robotics. Much research has been done in the context of appearance-based localization and SLAM using topological maps. Among the works that have studied the issues

relevant to appearance-based loop closure detection, there has been extensive research towards making image representations and image search more efficient.

Many methods employ global features to describe the entire image. Ulrich and Nourbakhsh [2000] propose to use color histograms as compact image representations that exhibit some invariance properties to rotation and translation changes. Sünderhauf and Protzel [2011] propose an appearance-based place recognition system that can simply implement the image descriptors. They use BRIEF-Gist descriptors, inspired by the work in [Oliva and Torralba 2006; 2001] that examines holistic descriptors and introduces Gist.

Compared to local descriptors, global ones are not very robust to illumination and perspective changes. Wang et al. [2005] propose a global localization strategy that uses SIFT descriptors to represent local features. They build a visual vocabulary upon the features, and images are represented by visual words that are the indices of an inverted index system. They employ RANSAC for geometric verification to confirm putative matches. Many other works have proposed extensions to this basic approach. Newman et al. [2006] use BoW representation for images, where the vocabulary is built using agglomerative clustering. They combine a 3D laser ranging sensor with vision, for loop closure detection. They use vision for loop closure detection and laser readings for geometric map building. They construct a similarity matrix that shows the similarity between all images and then detect loop closures as off-diagonal stripes in the similarity matrix. They try to remove the effects of repetitive features (perceptual aliasing) in the similarity matrix by a rank reduction technique. Filliat [2007] proposes a new approach that builds the visual vocabulary online and incrementally.

Other works have also tried to remove the effect of perceptual aliasing, by representing images using discriminant local features instead of visual words. Li and Kosecka [2006] and Zhang [2011] propose reducing the number of features in

each image. They keep track of the local features that are repeatable over time. However, these approaches are more specific to the navigation problems where the data is sequential and there exists considerable overlap between consecutive images. Kawewong et al. [2011] use position-invariant robust features (PIRFs) to preserve the distinctive power of local features. PIRFs are detected incrementally and despite their simplicity they are robust against dynamic changes. These approaches use direct feature matching between locations of the map and the query image. Although such matching methods are reliable, they are not efficient for large-scale environments.

Probabilistic frameworks have shown the most suitable solutions to deal with uncertainties in mapping and localization. Kröse and Bunschoten [1999] were the first to propose a probabilistic framework for appearance-based localization. They use Principal Component Analysis (PCA) to reduce the dimensionality of images. Linear PCA is invariant to rotation, scale and translation but not illumination.

The work by Angeli et al. [2008] and Cummins and Newman [2008; 2009] addresses SLAM issues more properly. They both use BoW representations, however their systems differ in several aspects. Angeli et al. [2008] use a Bayesian filtering framework to compute the likelihoods of loop closure candidates. They use SIFT features and local color histograms to provide more distinct image representations. They then build two visual vocabularies incrementally, without any prior information on the environment and use two BoW representations as an input of a Bayesian filtering framework to estimate the likelihood of loop closures.

FAB-MAP [Cummins and Newman 2008; 2011] is another major work in appearance-based SLAM that proposes a probabilistic framework over the bag-of-words representation of locations for estimating loop closure likelihoods. Along with the visual vocabulary they learn a Chow Liu tree as an offline process to capture the co-occurrences of visual words. Therefore they can handle the

perceptual aliasing problem more properly.

Maddern et al. [2011; 2012] use particle filters in combination with FAB-MAP. They incorporate odometric information into the place recognition process to improve the robustness of FAB-MAP.

Milford and Wyeth [2012] proposed a SeqSLAM approach, to tackle the problem of perceptual aliasing on long-term datasets. Their datasets comprise two runs along the same route, taken at different times of the day and under severe weather conditions with a changing environment. Instead of feature-based image matching methods, they use whole image template matching for the scene recognition. They divide each image into normalized patches and calculate the distance between two images by the Sum of Absolute Differences. A matrix of differences between all images is stored to find the best matching sequence of images to the query at search time. Experiments on two datasets show that their method outperforms FAB-MAP when tackling extreme perceptual aliasing. However, their image representation and matching approach is not suitable for navigation in large-scale environments. Sünderhauf et al. [2013] run their own implementation of SeqSLAM on a dataset taken four times, once in every season on the same route of 728km. They show that the good matching performance of SeqSLAM highly depends on the stability and repeatability of viewpoints in the different traversals of the environment. With mild shifts in viewpoints the performance drops dramatically.

2.3.1 Large-Scale SLAM

In spite of the considerable amount of work on SLAM, research in large-scale environments is not extensive and not yet sufficiently well-developed. When the size of the map increases, handling the map and localization become major problems. Many appearance-based navigation methods for large-scale

environments have been proposed, however most of them are only demonstrated to operate on image datasets of up to 30,000 images [Milford and Wyeth 2008; 2012, Schindler et al. 2007]. The largest experiment that we are aware of is the work by Sünderhauf and Protzel [2011] and FAB-MAP by Cummins and Newman [2009].

Sünderhauf and Protzel [2011] propose a pose graph SLAM system that is evaluated with a dataset containing 58,758 images collected from a 66km trajectory. Although the dataset has several loops, its traversal direction does not change, which is suitable to evaluate their place recognition system. The appearance-based place recognition systems that use the appearance of whole scenes (*a.k.a.* holistic descriptors) for place recognition might fail to recognize the places revisited from different directions.

Cummins and Newman [2009] with FAB-MAP2 have conducted the largest experiment, on a dataset of 103,000 images captured from a 1,000km trajectory. They built FAB-MAP2 upon the basic FAB-MAP system proposed in [Cummins and Newman 2008]. Their basic FAB-MAP model had two drawbacks: first, it was not scalable to large datasets and second, it was not suitable for real-time SLAM. Because of the high computational cost of their SLAM algorithm, it was only applicable to maps up to around 1,000 locations. They tried to improve their basic algorithm in a number of ways to deal with large-scale environments. In [Cummins and Newman 2009], they tried to make their system scalable by introducing sparse approximations to their FAB-MAP model and hence accommodating inverted index approaches into their probabilistic framework to speed up the search. Moreover, they added a geometric post-processing verification stage to discard the false loop closure candidates. They also reduced the construction time of the visual vocabulary by using approximate nearest neighbor search algorithms for k-means, and learned an approximate version of a Chow Liu tree to reduce its memory usage and computational complexity

Chow-Liu tree. They demonstrate that in large-scale datasets, when the size of the vocabulary is larger, using a Chow Liu tree worsens the likelihood estimates in some fraction of the dataset and it is no longer more effective than naïve Bayes at high precision. FAB-MAP2 achieved a recall rate of 3.1% at 100% precision and 14.3% at 90% precision. It can be observed from their low recall rate that the large-scale problem is very challenging and is not considered solved.

2.4 Nearest-Neighbor Search

Bag-of-words image search plays a key role in large-scale appearance-based navigation and is one focus of this research. For search in a very large collection of images, a vocabulary of hundreds of thousands or even millions of visual words is required. Vector-quantization with such large vocabularies is an expensive nearest-neighbor (NN) search process that affects the computational time of BoW search dramatically. In this section, we briefly review the background of nearest-neighbor search and the relevant related work.

Nearest neighbor search, also known as similarity search or proximity search, can be defined as one of the following problems:

- k-Nearest Neighbor Search (kNN): Given a set $\mathcal{D} \subset \mathcal{U}$ of n objects (points in vector space $\mathbb{R}^d$), and a query $Q \subset \mathcal{U}$, construct a data structure that efficiently returns the closest k objects in $\mathcal{D}$ to the query Q. 1NN search is the special case of kNN that returns the single nearest neighbor to the query.

- Range Search: Given a set $\mathcal{D} \subset \mathcal{U}$ of n objects (points in vector space $\mathbb{R}^d$), and a query $Q \subset \mathcal{U}$, construct a data structure that efficiently returns the objects X that are within a given distance r from the query: $\{X \subset \mathcal{D} \mid \rho(X, Q) \leq r\}$, where ρ is a distance measure.

The naïve solution to find the nearest neighbor is to compare the query to every point in the dataset (linear search). This approach is very inefficient, as the search time increases linearly with the size of the dataset. Other search algorithms pre-process the data into a structure or *index* so as to reduce the number of distance computations at search time [Bentley 1980]. However, these methods are only efficient for low dimensions ($d < 10$). Many of the current algorithms still suffer from a search time/space complexity that is exponential in dimensionality d; as the dimensionality grows, the search time complexity gets close to the complexity of the linear search. This is called *the curse of dimensionality*. In order to improve the time complexity in high-dimensional spaces, *approximate* algorithms can be used. In many applications, a fast approximate nearest neighbor solution is more desirable than an exact but slow solution. Approximate nearest neighbor (ANN) search algorithms are sometimes formulated as c-approximate NN search methods that return an approximate nearest neighbor X^c, such that $\rho(X^c, Q) \leq c\rho(X^*, Q)$, where X^* is the closest point to the query Q.

In the following, we present an overview of the literature on nearest neighbor search algorithms.

2.4.1 Space partitioning algorithms

Space partitioning algorithms have been utilized for NN search since the 1970s. These methods mostly use tree-based index structures and apply branch and bound techniques to prune the tree at search time. Search structures that have been developed for vector spaces are also called *spatial access* or *spatial index* methods. KD-trees, quad-trees, R-trees and X-trees are popular choices.

The classical KD-tree algorithm [Bentley 1975, Friedman et al. 1977, Bentley 1980] partitions the space by hyper-planes that are perpendicular to coordinate axes. At the root of the tree a hyperplane orthogonal to one of the dimensions

splits the data into two halves according to some splitting value, which is usually the median or mean of the data to be partitioned. Each half is recursively partitioned into two halves with a hyperplane through a different dimension. Partitioning stops after $\log n$ levels, where n is the total number of data points, so that at the bottom of the tree each leaf node corresponds to one data point. The splitting values at each level are stored in the nodes. The query point is then compared to the splitting value at each node while traversing the tree from root to leaf to find the nearest neighbor. Since the leaf point is not necessarily the nearest neighbor, to find approximate nearest neighbors, a backtrack step from the leaf node is performed and the points that are closer to the query point in the tree are examined. Instead of simple backtracking, Arya and Mount [1993] and Beis and Lowe [1997] propose to use a Best-Bin-First (BBF) heuristic to perform the search faster. BBF maintains a sorted queue of nodes that have been visited and expands the bins in the order of their distance to the query point (*priority search*).

Silpa-Anan and Hartley [2008] proposed randomized KD-trees as an improvement over the original KD-trees. A set of KD-trees is created and queried instead of a single tree. In each random KD-tree, the data points are rotated randomly, so that the initial choice of axes does not affect the points that are retrieved. At query time, the same rotation is applied to the query point before searching each tree. The union of the points returned by all KD-trees is the candidate list. The best nearest neighbors are selected using linear search over the candidate list.

Quad-trees [Samet 1984] recursively partition the space into quadrants. In R-trees [Guttman 1984] data points are grouped into hyper-rectangles. The input to the search is also a hyper-rectangle. These methods are inspired by the fact that spatial data cannot be well-represented by point locations and rather needs to be presented by multi-dimensional areas. A dynamic index structure is also used to

efficiently update the tree. X-trees [Berchtold et al. 1996] has been proposed to tackle a problem of R-tree-based structures: the overlap of bounding rectangles that grows with increasing dimension. In X-trees, this overlap is minimized with a new splitting strategy.

Hierarchical k-means trees (HKM), proposed by Fukunaga and Narendra [1975], are another type of partitioning trees based on k-means clustering. The tree is built by running k-means clustering recursively. k is called the branching factor. Data points are first partitioned into k distinct clusters to form the nodes of the first layer of the tree. Each cluster is recursively partitioned into k clusters and this process continues until there are no more than k data points in each node. A depth-first search is a common tree traversal approach for searching the tree. Muja and Lowe [2009; 2014] propose to use priority queues to search the tree more efficiently. Similar to the Best-Bin-First approach [Beis and Lowe 1997], when traversing the tree, the unvisited branches of the nodes along the path are added to the priority queue. When backtracking, the branches are extracted and expanded in the order of the distance of their cluster centroid to the query.

Brin [1995] generalized Fukunaga and Narendra's method to generic metric space, by removing the cluster mean computation step and using some of the data points as cluster centers.

2.4.2 Mapping-based techniques

Random projections [Kleinberg 1997, Kushilevitz et al. 2000] and locality-sensitive hashing [Indyk and Motwani 1998, Indyk 2000] are popular mapping-based techniques for c-approximate NN search. They have been developed for high-dimensional approximate nearest neighbor search. These techniques use random projections for dimensionality reduction. Kleinberg [1997] first introduced the idea of random projections for NN search. His idea was later

improved by Indyk and Motwani [1998] and Indyk [2000] who used non-linear mapping based on Locality-Sensitive Hashing (LSH). LSH uses a family of hash functions of the same type to create a hash value for each point of the dataset. Each function reduces the dimensionality of the data by projection onto random vectors. The data is then partitioned into bins by a uniform grid. Since the number of bins can be still too large, a second hashing step is performed to obtain a smaller hash value. At query time, the query point is mapped using the hash functions and all the data points that are in the same bin as the query point are returned as candidates. The final nearest neighbors are selected by a linear search through candidate data points.

2.4.3 Graph-based search algorithms

Papadias [2000] constructs a graph over data points. He assumes that each data point (*e.g.* an image configuration) is specified as a collection of components (*e.g.* objects). Each point has the form of $X_i = (V_1, \ldots, V_m)$, where each V_j is an object and can take values from a finite set (*e.g.* a set of squares of different sizes). The objective is to find the point in the dataset that has the closest configuration to the query Q. Papadias [2000] defines X_i and X_j as neighbors if one can be converted to the other by changing the value of one of its variables. Several heuristics to perform hill-climbing on such a graph are proposed [Papadias 2000].

Paredes and Chávez [2005] aim at minimizing the number of distance computations during the nearest neighbor search. A k-NN graph is built from dataset points and when queried with a new point, the graph is used to estimate the distance of all points to the query, using the fact that the shortest path between two nodes is an upper bound on the distance between them. Using the upper and lower bound estimates, Paredes and Chávez [2005] eliminate points that are far away from the query point and exhaustively search in the remaining dataset.

Lifshits and Zhang [2009] define a *visibility graph* and then perform nearest neighbor search by a greedy routing over the graph. They make strong assumption about the dataset.

Chapter 3

The Graph Nearest Neighbor Search (GNNS)

In this chapter, we present a graph-based approximate nearest neighbor search algorithm (GNNS) and study the complexity and efficiency of the algorithm when compared to widely-used NN search methods. GNNS builds a k-nearest neighbor graph in an offline phase and when queried with a new point, it performs a greedy search on the graph structure.

A background review on widely-used nearest neighbor search algorithms and search indices was provided in the previous chapter. In the first section of this chapter, we extend the review to previous research on k-NN graph construction and graph-based search algorithms. In the second section, we first describe our GNNS search algorithm and then provide theoretical guarantees for the accuracy and computational complexity of the algorithm. Finally, we show the effectiveness of this algorithm empirically.

3.1 Related Work

3.1.1 k-Nearest-Neighbor Graph Construction

k-NN graphs are widely used in many applications of machine learning, data mining, computer graphics, bio-informatics and others. A k-NN graph is a directed graph $\mathcal{G} = (\mathcal{D}, \mathcal{E})$, where $\mathcal{D}$ is the set of nodes (*i.e.* data points) and $\mathcal{E}$ is the set of links. Node X_i is connected to node X_j if X_j is one of the k-NNs of X_i.

The naïve construction of k-NN graph takes $\theta(dn^2)$ time, where n is the number of datapoints and d is the dimensionality. However, this solution is not practical for large-scale applications. Earlier research on the exact construction of k-NN graphs [Bentley 1980, Clarkson 1983, Vaidya 1989, Dickerson and Eppstein 1996, Paredes et al. 2006] has tried to make the graph construction more efficient computationally. Bentley [1980] proposed a multi-dimensional divide-and-conquer method for the nearest-neighbor problem which takes $O(n\log^{d-1} n)$ time in the worst case. Clarkson [1983] introduced a NN algorithm with expected running time of $O(c^d n\log n)$ for some constant c. Vaidya [1989] also proposed an alternate approach using a modified form of quad-trees with a worst case $O((cd)^d n\log n)$ time for some constant c.

Alternative methods to solve the k-NN graph problem take advantage of the proximity properties of the Voronoi diagram or its dual, Delaunay triangulation (DT). Using an order (k+1) Voronoi diagram, for each point p in the dataset the other k points lying in the same Voronoi region are chosen as the nearest neighbors of p. Lee [1982] and Aggarwal et al. [1987] showed how to efficiently construct an order k Voronoi diagram in $O(k^2 n\log n)$ and $O(k^2 n + n\log n)$ time, respectively. However, these methods are only efficient for small values of k in the plane. Dickerson et al. [1992] and Eppstein and Erickson [1993] presented faster algorithms that search a Delaunay triangulation. The former requires

$O(kn\log k + n\log n)$ time for the planar case and the latter $O(kn + n\log n)$ time for the simpler L_∞ metric. However, these approaches are not still efficient in high dimensions. For higher dimensions, Dickerson and Eppstein [1996] make use of the linear-sized Delaunay triangulation of [Bern et al. 1994] and propose a more efficient search method using the higher dimensional Delaunay triangulation, which is also called Steiner Delaunay triangulation. In Steiner DT extra points are added at construction time to keep the triangles well-shaped and maximum vertex degree of nodes bounded by a constant. For each point p in the dataset, a breath-first search (by distance) on the Stiener DT finds the k nearest neighbors. With an amortized analysis, Dickerson and Eppstein showed that the running time of enumerating k nearest neighbors to each point of the dataset, in order, is $O(kn\log n)$ and $O(n\log n + kn)$ when order is not required. The preprocessing time of Steiner DT construction is $O(n\log n)$. Although the method is theoretically optimal, it has not been used in practice so far, perhaps because of the large hidden constants involved in theoretical algorithms. Also since the Delaunay triangulation is not unique, the result of construction is sensitive to the triangulation.

Paredes et al. [2006] developed a k-NN graph construction algorithm for a general metric space where coordinate information is not necessarily available. Their algorithm works well for low dimensions but is not efficient on high dimensions. The authors empirically estimate the computational cost of $O(c_1 n^{1.27})$ (distance evaluations) for low and medium dimensions ($d \leq 16$) and $O(c_2 n^{1.90})$ for higher dimensional space ($d \simeq 24$). They removed k from complexity computations as based on their experiments k has only a mild impact on the computational cost.

Since there is still no exact k-NN graph construction method that works well in high dimensions, some effort has gone into building approximate k-NN graphs [Chen et al. 2009, Dong et al. 2011, Wang et al. 2012]. A straightforward solution is

employing an approximate nearest neighbor search algorithm, such as KD-trees and locality sensitive hashing, to build a search index over data points and then using the index structure to find the k-nearest neighbors of each point. One drawback of this approach is that these nearest neighbor search methods have been developed for the general NN-search problem in which the query points are not necessarily inside the dataset. For the problem of k-NN graph construction the query points are in the dataset and this approach might not be efficient because of the extra effort that is made to produce better results for general queries.

Chen et al. [2009] proposed an approximate method which employs a divide-and-conquer method and has been developed specifically for the l_2 metric. Datapoints are divided into possibly overlapping subsets, and the approximate k-NN graph of subsets is recursively computed. The division part uses a spectral bisection technique: a hyperplane bisects the datapoints according to the direction of largest singular value. The paper proposes two ways to perform the divide step; overlap and glue. In overlap, two overlapping subsets are generated and a parameter α is tuned to change the size of the overlap area. In glue, another separate area is used to merge the two disjoint subsets. In the conquer phase, if a datapoint belongs to more than one subset (the points in the overlap or merging zone), its k-nearest neighbors are selected based on its neighbors in each subset. The time complexity of the algorithm is $\theta(dn^t)$ for high dimensional data, where $t \in (1,2)$ is a function of the internal parameter α which is used to control the size of the overlap area in the divide phase. k is removed from the complexity computation assuming d, the dimensionality, dominates k.

Another recent work on approximate k-NN graph construction is an algorithm by Dong et al. [2011], called NN-DESCENT. This simple method is based on the principle that a neighbor of a neighbor is also likely to be a neighbor: If we start with an approximation of a k-NN graph (can be a random graph) we can improve the

approximation of the k-nearest neighbors of each datapoint by searching through the neighbors of neighbors of each point. The authors quantify this observation with a heuristic argument that suggests the algorithm works under some assumptions. The computational complexity of the algorithm that has been computed empirically is $O(n^{1.14})$. One advantage of NN-DESCENT over the algorithm of [Chen et al. 2009] is that it can work with arbitrary similarity measures.

Wang et al. [2012] and Wang et al. [2013b] proposed a divide-and-conquer method to build an approximate graph for large-scale data. The method divides the data-points into subsets, hierarchically and randomly. It builds an exact k-NN graph over each subset and repeats the process until the multiple graphs can be combined into a relatively accurate approximate k-NN graph. The authors experimentally show considerable speedup over large datasets, compared to some other similar methods.

3.1.2 k-NN Graph-based Search Algorithms

In this section, we review the related work on nearest neighbor search algorithms that employ k-NN or a *neighborhood* graph as a search index.

Arya and Mount [1993] build a neighborhood graph as follows: for each point p in the dataset D, they maintain the list of other points sorted by distance to p. The closest point r in the list has a direct link to p. The other points whose distance to r is less than the distance from p are removed from the list and a directed link between p and any remaining point in the list is created. The graph search is performed using a *best-first* strategy. The starting point is chosen by constructing a KD-tree on the dataset (in a pre-processing phase) and searching the leaf node that contains the query. That node will then be considered as the starting point for the graph search. A heap is used to maintain the nearest neighbor candidates which are the visited nodes (initially the starting point) and their neighbors, sorted based on their

distance to the query. The closest candidate to the query that is retrieved from the heap is the next node to expand. The method reduced the degree of the graph but at the expense of increasing the number of search steps.

Sebastian and Kimia [2002] build a neighborhood graph for shape-based indexing and search in a metric space. They use k-NN graph as a local approximation to Delaunay graph that is used for Euclidean space. The search strategy is *best-first* and starts from a few well-separated seeds. At each step the search moves towards the *extended*-neighbors of the current node p and add them to a priority queue. The *extended*-neighbors are the immediate neighbors of p as well as the neighbors of neighbors of p that are τ-closer to the query with respect to p, where τ is a fixed threshold (≥ 1.0). A neighbor of p, p_{nn}, is called τ-closer to query q w.r.t. p if $d(p_{nn},q) \leq \tau d(p,q)$. The front element in the queue, that has the smallest distance to the query, is then chosen to proceed the search from. If the front element is not τ-closer to the query than the current node, the search will stop and return the current node as the nearest neighbor.

Similarly and independently in [Hajebi et al. 2011], we proposed a graph-based nearest neighbor search algorithm. We construct a k-NN graph and when queried with a new point, perform hill climbing starting from a randomly sampled node of the graph. The distance of the neighbors of the chosen seed from the query is calculated and the closest neighbor to the query is chosen to proceed the search from. If no neighbor is closer than the parent node, the search stops and the parent node is returned as the approximate nearest neighbor. The paper compares the method with randomized KD-trees and LSH and empirically shows the effectiveness of our algorithm. We also provide theoretical guarantees for the accuracy and computational complexity of the algorithm.

More recently, Aoyama et al. [2011], Wang and Li [2012] and Wang et al. [2013b] have used similar greedy search strategies but with differently constructed

neighborhood graphs. Aoyama et al. [2011] propose an undirected degree-reduced k-nearest neighbor graph. The graph is constructed with an incremental procedure: it starts with $k' = 1$ and builds a k'-NN graph, it increments k' and links the k'th neighbor y to node x, only if node x is not reachable through the existing edges when the search starts from y. This procedure continues until $k' = k$. The number of edges of each node is thereby reduced and is less than or equal to k. Wang et al. [2013b] develop an effective data structure by combining a neighborhood graph with a bridge graph to boost the performance of approximate NN search. The authors demonstrate the effectiveness of their approach on a dataset of one billion SIFT descriptors.

3.2 The Graph Nearest Neighbor Search Algorithm (GNNS)

Our graph nearest neighbor search (GNNS) algorithm, described in this section, builds a k-NN graph as the search structure, and when queried with a new point, it performs hill-climbing starting from a randomly sampled node of the graph [Hajebi et al. 2011]. In the following subsections, we explain the graph index construction and the search algorithm in more detail. We also analyze the complexity of GNNS and provide experimental results to demonstrate the performance of GNNS compared to other search algorithms.

3.2.1 k-NN Graph Search Index

k-NN graph for the set of data-points $\mathcal{D} = \{X_1, \ldots, X_n\} \in \mathbb{R}^d$ is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{D}$ is the set of nodes and $\mathcal{E} \subset \{\langle X_i, X_j \rangle : i, j \in \{1, \ldots, n\}\}$ is the set of links. Node X_i is connected to node X_j if X_j is one of the k nearest neighbors of X_i. The k nearest neighbors of node X_i are sorted and ranked based on their

increasing distance to X_i. The computational complexity of the naïve construction of this graph is $O(dn^2)$, where n is the size of the dataset and d is the dimensionality. In our application, the graph is constructed in an offline phase, so the cost of the graph construction is not our concern.

The choice of k is crucial to have a good performance. A small k makes the graph too sparse or disconnected so that the hill-climbing method frequently gets stuck in local minima. Choosing a big k gives more flexibility during the runtime, but consumes more memory and makes the offline graph construction more expensive.

3.2.2 Algorithm Description

The GNNS Algorithm, which is basically a best-first search method to solve the K-nearest neighbor search problem, is shown in Algorithm 1. Throughout this section, we use capital K to indicate the number of queried neighbors, and small k to indicate the number of neighbors to each point in the k-nearest neighbor graph. Starting from a randomly chosen node in the k-NN graph, the algorithm replaces the current node Y_{t-1} by the neighbor that is closest to the query:

$$Y_t = \underset{Y \in N(Y_{t-1},E,\mathcal{G})}{\operatorname{argmin}} \rho(Y,Q),$$

where $N(Y,E,\mathcal{G})$ returns the first $E \leq k$ neighbors of Y in $\mathcal{G}$, and ρ is a distance measure (we use Euclidean distance in our experiments). The algorithm terminates after a fixed number of greedy moves T. If $K = 1$, we can alternatively terminate when the algorithm reaches a node that is closer to the query than its best neighbor. At termination, the closest K visited nodes to the query are returned as the K-nearest neighbors. Figure 3.1 illustrates the algorithm on a simple nearest neighbor graph with query Q, $K = 1$ and $E = 3$.

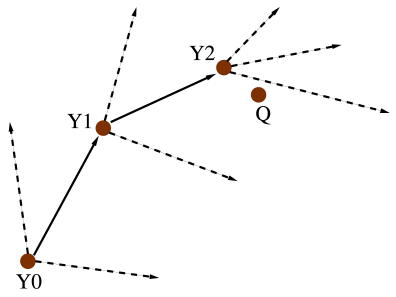


Figure 3.1: The GNNS Algorithm on a simple nearest neighbor graph.

Algorithm 1 The Graph Nearest Neighbor Search (GNNS) algorithm for K-NN Search Problems.

Input: A k-NN graph $\mathcal{G} = (\mathcal{D}, \mathcal{E})$, a query point Q, the number of required nearest neighbors K, the number of random restarts R, the number of greedy steps T, and the number of expansions E.

Output: The K nearest approximate neighbors to Q

ρ is a distance function. $N(Y, E, \mathcal{G})$ returns the first E neighbors of node Y in $\mathcal{G}$.
$\mathcal{W} = \{\}$.
$\mathcal{U} = \{\}$.
for $r = 1, \ldots, R$ **do**
  Y_0: a point drawn randomly from a uniform distribution over $\mathcal{D}$.
  for $t = 1, \ldots, T$ **do**
    $Y_t = \text{argmin}_{Y \in N(Y_{t-1}, E, \mathcal{G})} \rho(Y, Q)$.
    $\mathcal{W} = \mathcal{W} \bigcup N(Y_{t-1}, E, \mathcal{G})$.
    $\mathcal{U} = \mathcal{U} \bigcup \{\rho(Y, Q) : Y \in N(Y_{t-1}, E, \mathcal{G})\}$.
  end for
end for
Sort $\mathcal{U}$, pick the first K elements, and return the corresponding elements in $\mathcal{W}$.

Parameters R, T, and E specify the computational budget of the algorithm. By increasing each of them, the algorithm spends more time in search and returns a more accurate result. The difference between E and k and K should be noted. E and K are two input parameters to the search algorithm (online), while k is a parameter of the k-NN tree construction algorithm (offline). Given a query point Q, the search algorithm has to find the K-nearest neighbors of Q. The algorithm, in each greedy step, examines only E out of k neighbors (of the current node) to choose the next node. Hence, it effectively works on an E-NN graph.

Next, we analyze the performance of the GNNS algorithm for the nearest neighbor search problem ($K = 1$).

3.2.3 Theoretical Analysis

Theorem 1. *Consider the version of the GNNS algorithm that terminates when the greedy procedure terminates in a local minimum. Assume that the n data points are drawn uniformly randomly from a d-dimensional hypercube of volume 1. Let* $0 < \delta < 1$. *Choose M such that*

$$n \geq (M^d - 1)(d \log M + \log 1/\delta). \tag{3.1}$$

Partition the unit hypercube into hypercubical cells of side equal to $1/M$. *Denote the set of cubical cells by* $\{A_1, \ldots, A_{M^d}\}$. *Construct a k-NN graph* $\mathcal{G}$ *by connecting each point* $X \in A_i$ *to* $k \geq 2d$ *arbitrary points (if exist) each in a neighboring cell of* A_i. *The set of neighboring cells of* A_i *is denoted by* $V_i = \{A_j : \left\|A_i - A_j\right\|_1 = 1/M\}$, *where the* $\left\|A_i - A_j\right\|_1$ *is the* L_1 *distance between the center of two cells,* A_i *and* A_j. *For a cell* A_i *there would be at most two immediate neighbors on either side and in each dimension, thus* $|V_i| \leq 2d$.

Then with probability at least $1 - \delta$, *for any query point Q and any starting point* Y_0, *the GNNS algorithm returns an approximate nearest neighbor to Q, whose distance from the exact nearest neighbor is no more than* $\frac{\sqrt{d}}{M}$, *and its computational cost is bounded by*

$$\min\{nd, Md^2\}.$$

Proof. As we discretize each hypercube edge into M equal intervals, the unit hypercube is partitioned into M^d cube cells of volume $(1/M)^d$, $\{A_1, \ldots, A_{M^d}\}$. We

compute the probability that there exists at least one point in each cell.

$$\begin{aligned}
\mathbb{P}\left(\forall j, \exists i, X_i \in A_j\right) &= 1 - \mathbb{P}\left(\exists j, \forall i, X_i \notin A_j\right) \\
&= 1 - \mathbb{P}\left(\bigcup_{j=1}^{M^d} \forall i, X_i \notin A_j\right) \\
&\geq 1 - \sum_{j=1}^{M^d} \mathbb{P}\left(\forall i, X_i \notin A_j\right) \\
&= 1 - M^d \left(1 - \frac{1}{M^d}\right)^n.
\end{aligned}$$

Let $M^d \left(1 - \frac{1}{M^d}\right)^n \leq \delta$. After reordering, we get

$$n \geq \frac{d \log M + \log 1/\delta}{\log\left(1 + \frac{1}{M^d - 1}\right)}.$$

Because $\log(1 + \frac{1}{M^d-1}) \leq \frac{1}{M^d-1}$, we get

$$n \geq (M^d - 1)(d \log M + \log 1/\delta).$$

In summary, we have shown that for any $0 < \delta < 1$, if Inequality (3.1) holds, then $\mathbb{P}\left(\forall j, \exists i, X_i \in A_j\right) \geq 1 - \delta$. Thus, with probability $1 - \delta$ all cells contain at least one data point.

Now let $X \in A_i$ be an arbitrary point in $\mathcal{D}$, and $Q \in A_q$ be a query point that is not necessarily in $\mathcal{D}$. There are at most $2d$ cells in V_i. Under the condition that all cells contain at least one data point, there is at least one cell A_k in V_i such that $\left\|A_k - A_q\right\|_1 < \left\|A_i - A_q\right\|_1$, which is easy to see because we use L_1 norm. Thus, the greedy approach makes progress. Further, recall that each axis is partitioned into M intervals. Hence, the algorithm takes at most Md steps, and in each step, an arbitrary point from the closest cell, A_k, to the query is chosen to proceed the search from. Therefore, at most $Md \times 2d$ distance computations are needed until we get to the query cell, A_q. The number of points in A_q is unknown. An exact solution will require to compute the distance of all points in A_q to Q. An approximate solution

will return an arbitrary point from A_q as an approximate nearest neighbor to Q, whose distance from the exact nearest neighbor is no more than $\frac{\sqrt{d}}{M}$. Thus, the computational complexity for an approximate solution will be $\min\{nd, 2Md^2\}$.

□

Remark. Because $M = O(n^{1/d})$, the computational cost is $\min\{nd, n^{1/d}d^2\}$. As a data point is not visited more than once during the graph search, $n^{1/d}d^2 \leq nd$. The theorem can be proven for other distributions. The crucial assumption is the assumption on the independence of the data points. The uniformity assumption is made to simplify the presentation.

3.3 Experiments

To validate our method GNNS, we compared the performance of our algorithm with state-of-the-art nearest neighbor search techniques: randomized KD-trees with the best-bin-first search heuristic and LSH[1]. The experiments are carried out on a real-world publicly available image dataset [Howard and Roy 2003] as well as our synthetically generated dataset. We compare the methods in terms of both the speedup over the linear search and the number of Euclidean distance computations.

First, we explain the experiments with the real-world datasets. We generated five datasets of 17,000, 50,000, 118,000 and 204,000 (128-dimensional) SIFT descriptors[2] [Lowe 2004]. For each dataset, the query set containing 500 SIFT descriptors is sampled from different images than the ones used to create the dataset. The experiments are performed for $K = 1$ and 30. The accuracy is measured by first computing the percentage of the K nearest neighbors reported correctly, and then averaging over 500 queries.

[1]We used the publicly available implementations of KD-trees [`http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN`] and LSH [`http://www.mit.edu/~andoni/LSH/`]

[2]We used SIFT descriptors due to their popularity in feature matching applications.

For each dataset, instead of building a new graph for each value of E, we constructed a single large graph with $k = 1000$, and re-used it in all experiments. We exhaustively tried LSH and KD-trees with different values of parameters and chose combinations that result in better speedup and precision (parameter sweep).

In Figures 3.2, 3.3 and 3.5, K, the number of NNs returned by algorithms, is set to 1; and in Figure 3.4, K is set to 30. These figures are produced by varying the number of node expansions E; The other parameter R is fixed and set to 1 and T is not used as we alternatively terminated the search when it reached the node which is better than its neighbors.

Figures 3.2 and 3.4 compare different search methods in terms of their speedup over linear search, while Figure 3.3 compares the methods in terms of the number of distance computations that they perform (normalized by dividing over the number of distance computations that the linear search performs). The experiments have been performed on datasets of different sizes. The error bars, shown in all figures, are standard deviations over 500 queries. As can be seen in these figures, the GNNS method outperforms both the KD-trees and LSH algorithms. The figures also show how the performance improves with the size of the dataset.

The second set of experiments were performed on synthetically generated datasets of different dimensions to show how the performances of different methods degrade as dimensionality increases. To construct a dataset of dimension d, we sampled 50,000 vectors from the uniform distribution over $[0,1]^d$. We also sampled 500 query vectors from the same distribution. Figure 3.5 show the results for the randomly generated datasets. Figure 3.5(a) compares GNNS and randomized KD-trees. The GNNS method outperforms the KD-trees. Figure 3.5(b) shows the results for the LSH method, which is much inferior to the two other methods. The figures also show how the speedup of different algorithms with respect to the linear search degrades as we increase the dimensionality and

the precision.

3.4 Summary

In this chapter, we present a graph-based approximate NN-search algorithm, called GNNS. We analyze GNNS and obtain upper bounds on the number of iterations before the algorithm finds an approximate nearest neighbor. We also discuss that the k-NN graph index used in our algorithm has an expensive construction; however there is a large amount of research to improve the computational complexity of graph construction.

We evaluate the performance of GNNS by comparing it with popular NN search methods, randomized KD-trees and LSH, in terms of the search precision and speedup over the baseline (linear search). We also examine and compare the performance of all algorithms when the size of datasets and data dimensionality change. Our experiments show the superior performance and effectiveness of GNNS on high dimensional real-world datasets as well as synthetically generated datasets.

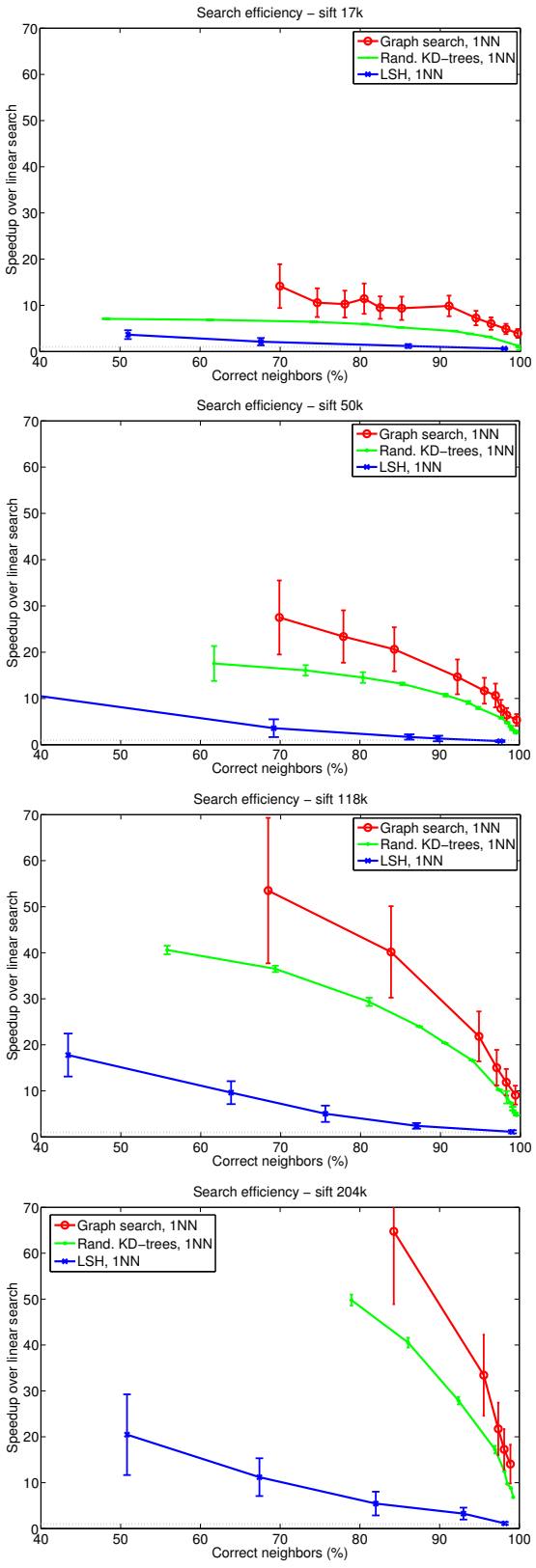


Figure 3.2: Performance comparison of GNNS, KD-trees and LSH on datasets of (a) 17K, (b) 50K, (c) 118K and (d) 204K points for 1-NN search problem ($K = 1$). The gray dashed line indicates the speedup of 1.

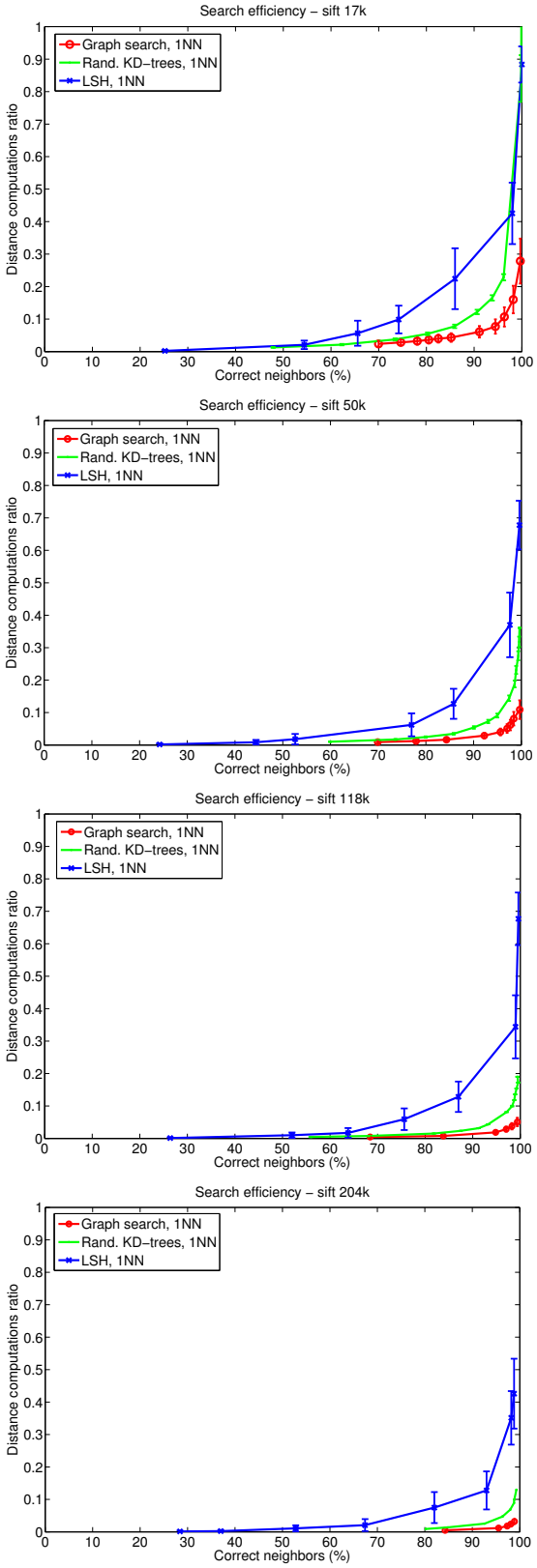


Figure 3.3: Performance comparison of GNNS, KD-trees and LSH on datasets of (a) 17K, (b) 50K, (c) 118K and (d) 204K points for 1-NN search problem ($K = 1$).

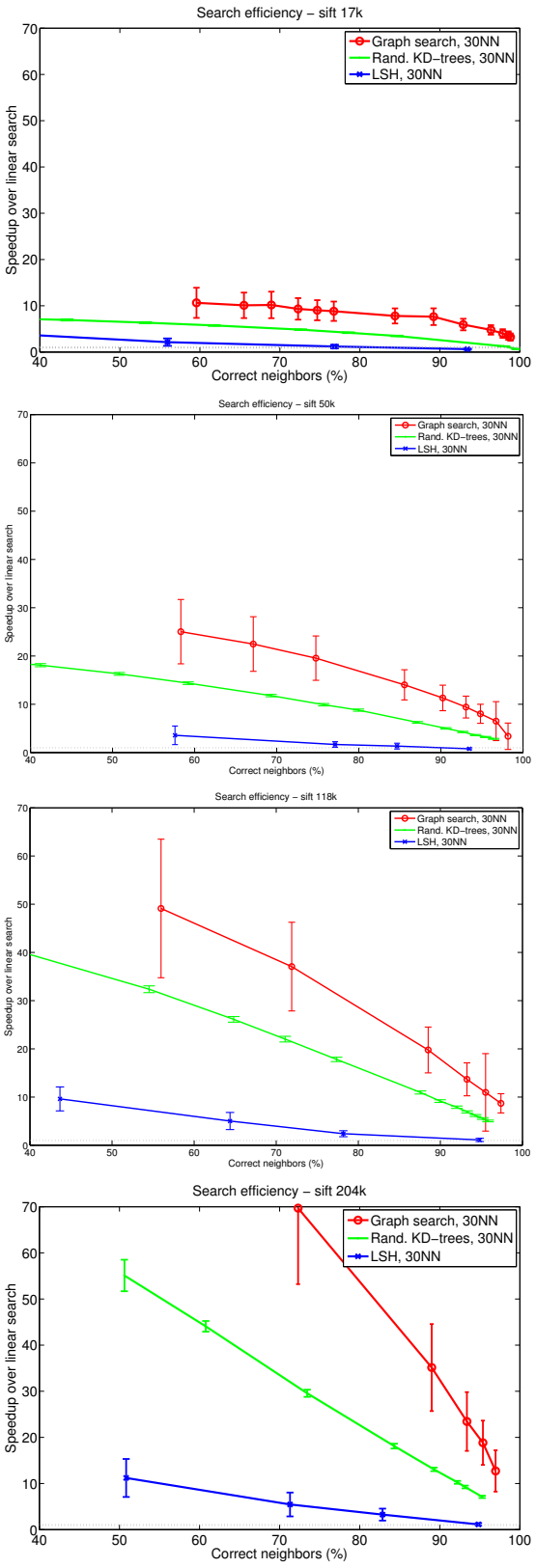
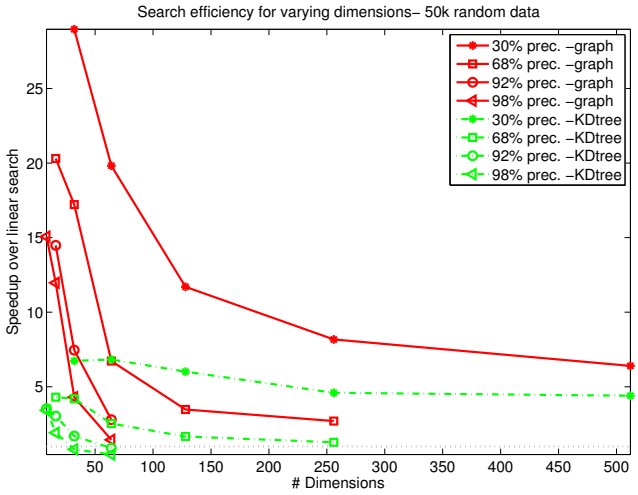
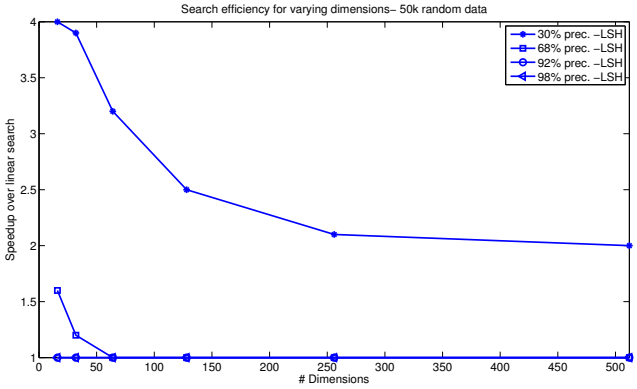


Figure 3.4: Performance comparison of GNNS, KD-trees and LSH on datasets of (a) 17K, (b) 50K, (c) 118K and (d) 204K points for 30-NN search problem ($K = 30$). The gray dashed line indicates the speedup of 1.



(a)



(b)

Figure 3.5: Search speedup and precision for data of varying dimensionality (1-NN search). GNNS is compared with KD-trees (a) and LSH (b). Datasets have 50k points. The gray dashed line indicates the speedup of 1.

Chapter 4

Vector Quantization for SLAM

In large-scale environments, SLAM maps contain a large number of images to match in order to solve the loop closure detection problem. The image search in such large maps is challenging and still an open problem. Although BoW is an efficient search technique, its vector-quantization (VQ) step can be computationally expensive. Vector-quantization maps the image feature descriptors to the words in a visual vocabulary. Typically hundreds to thousands of features are extracted from an image and need to be matched against tens or hundreds of thousands of visual words. Approximate nearest neighbor search algorithms, such as hierarchical k-means tree [Nister and Stewenius 2006] and randomized KD-trees [Silpa-Anan and Hartley 2008], have been used to speed up the quantization process, however at the cost of search accuracy.

In this chapter, we propose an efficient search algorithm to improve the performance of vector-quantization in a BoW-based appearance SLAM, which is another main contribution of this thesis. We employ the approximate nearest neighbor search algorithm (GNNS) presented in the previous chapter to solve the VQ. We show that the graph-based search structure used in GNNS can efficiently be integrated into the BoW model and the SLAM framework and that we can exploit their properties to increase the efficiency of the system. As these properties

are difficult to exploit by other search indices, this motivates the use of GNNS rather than other search algorithms in solving the loop closure detection problem. We experimentally show that GNNS, when applied to solve the vector-quantization problem, outperforms the state-of-the-art search methods in terms of search precision and speedup over the baseline linear search.

This chapter is organized as follows. In Section 4.1, we describe the construction of BoW's vocabulary and the graph structure as part of the offline pre-processing of our system and also discuss their space and time complexity. In Sections 4.2 and 4.3, we show how GNNS can exploit the sequential dependency in SLAM data as well as the perceptual aliasing in BoW to improve the search. Finally, we provide experimental results in Section 4.4 to demonstrate the effectiveness of our vector-quantization method.

4.1 Graph Vocabulary Construction

In the BoW model, the vocabulary is usually constructed using k-means clustering. In an offline phase, the feature descriptors extracted from a training dataset are clustered and the centroids of clusters are used to represent the visual words of the vocabulary. The standard k-means clustering using linear search is expensive and therefore not practical for building large vocabularies. The hierarchical k-means method (HKM) [Nister and Stewenius 2006] has been proposed to reduce the computational cost of k-means. Philbin et al. [2008] also propose a modification of k-means where the exact nearest neighbor search is replaced by an approximate NN-search algorithm, such as KD-trees. They show that this modification achieves the complexity of HKM, and demonstrate that approximate k-means (AKM) outperforms HKM when applied to the vector-quantization problem.

Once the vocabulary is built, we can construct the k-NN graph over the visual

words, such that each node of the graph corresponds to a visual word in the vocabulary. However, the graph construction is also a computationally expensive process especially when the dataset is large. Even though this search index is constructed only once offline, it is still desirable to minimize its computational cost. Similar to AKM [Philbin et al. 2008], an approximate NN-search method can be adopted to speed up the k-NN graph construction. Such other approximation methods have been studied in Section 3.1.1.

The time complexity of graph construction is negligible with respect to the time complexity of the k-means clustering and we can show that it can be considered as a small additional cost to the last iteration of k-means: In every iteration of k-means, the distance of data points to each cluster centroid is computed to update their membership in the new clusters, and this continues until convergence. Given n data points and C clusters' centroids[1], the complexity of each iteration is then $O(nC)$. The k-NN graph construction, can be embedded into k-means processing as follows: in the last iteration, in addition to other data points, the distance of each centroid from the other centroids is computed and the nearest neighbors are used to build the k-NN graph. The additional computational cost is negligible, as $C \ll n$. The complexity of the last iteration will then change to $O((n+C)C)$ which is slightly higher than $O(nC)$. This shows that the complexity of graph construction is comparable to the complexity of one iteration of k-means clustering and in applications where k-means clustering is performed (like in vocabulary construction for BoW), the construction time of the graph-based index is absorbed by the k-means algorithm.

A k-NN graph has space complexity of $O(nk)$. As the memory increases linearly with the size of the vocabulary, the k-NN graph might not be as scalable as

[1]The number of clusters generated by k-means is generally denoted by "k", however to avoid confusion with other k and K notations we use for k-NN graph and K-NN search, we use C to denote the number of cluster centroids.

KD-trees and HKM structures. However, as shown in [Cummins and Newman 2009], for a large-scale SLAM application ($\sim$1000km trajectory) a vocabulary of 100,000 words suffices. The memory consumption of a k-NN graph built over a vocabulary of this size is not major practical problem for our algorithm. For example, a 1000-NN graph built over a 100,000-word vocabulary requires 100,000$\times$1000$\times$4Bytes$\sim$381M of memory. Later in this chapter, we experimentally show that reducing the size of a k-NN graph by half, which can be obtained by changing the branching factor k to $k/2$, decreases the accuracy of VQ by only 2 to 7%. This degradation has a negligible effect on the result of image matching and loop closure detection.

4.2 Exploiting Sequential Dependencies in Data

The sequential property of data can be utilized to the advantage of image retrieval. Unlike many image retrieval and classification applications that search in a pool of unordered images, in appearance-based SLAM we can take advantage of the temporal coherency of images to make the image search for loop closure detection efficient. We propose to use GNNS for vector-quantization. In standard GNNS, search is initiated from a random node. By taking the sequential property of images into account, we can replace the random initiation of GNNS by a smarter method: sequential images usually have some overlap with their neighboring images and hence share a certain amount of features and visual words. This property can reduce the amount of computations required for the vector-quantization step, as we can quantize a feature once in the image where it is first observed, and use its visual word in subsequent images as long as the feature is observed. This requires us to match each new frame to the previous frame(s) to find the repeatable features. This step does not incur a significant cost as feature matching in two images can

be performed efficiently and can in fact incur no additional cost if the matching between consequent frames is already done in the process of key-frame detection: a new frame is matched against the previous key-frame in order to decide whether it is sufficiently different from the previous key-frame, in terms of appearance, to be considered a new location in the map. This process is done through direct feature matching between images [Zhang et al. 2010a].

Our approach works as follows: once a new image is captured, the features are extracted. Each feature is vector-quantized through GNNS. Let f be a feature in the current image that has a match f' in the previous key-frame. Let the visual word assigned to f' be w'. Intuitively, there is a good chance that the visual word w' is also the word or one of the neighbors of the word corresponding to feature f. Therefore we start the GNNS search from w', rather than a random node (see Figure 4.1 and Algorithms 2 and 3). This can significantly reduce the number of iterations and distance computations in the GNNS search. This is an advantage of the graph-based index over other search indices when images to be processed are temporally dependent as in visual SLAM, as it is not trivial to employ such prior knowledge in other search indices. Experimental results are shown in the following section.

4.3 Exploiting the Perceptual Aliasing Problem in BoW

As described in the last section, the sequential dependencies in data can improve the speedup of VQ on the features that have a match in the previous key-frame. For each feature in the current frame, we find the first nearest neighbor (1-NN) feature in the previous frame. The 1-NN is then verified as a true match if it passes the distance-ratio test (where the ratio between the distances of the closest and second

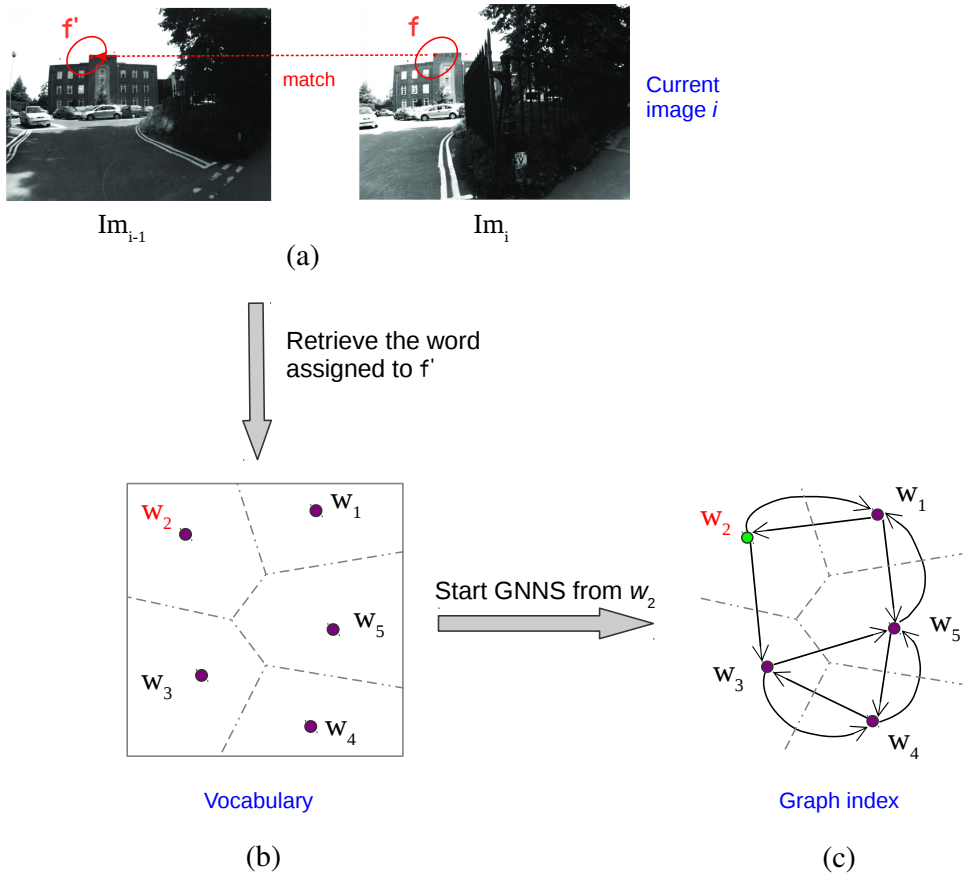


Figure 4.1: Vector-quantization with GNNS. (a) The match to the given feature, f, is found in the previous image; (b) The corresponding word to the matched feature is retrieved (w_2 in this example); (c) The word is used to start the GNNS from. The result of GNNS is the word that will be assigned to f.

closest neighbor from the query feature is below some threshold [Lowe 2004]). However, for features that do not have a true match, GNNS will normally start the search from a random node. In this section, we explain how we can improve the VQ speedup on non-matched features by taking advantage of the perceptual aliasing problem in BoW.

Perceptual aliasing (identifying two different locations as similar) is a well-known problem associated with the bag-of-words model, and is caused by the vector-quantization process in BoW: two features that are not verified as matches by the distance-ratio test might be mapped to the same visual word and hence considered as a match by BoW. Although this VQ error is inevitable, we can exploit it to improve the performance of GNNS on non-matched features as follows.

Let f be a feature in the current frame and f'' be its 1-NN, but not its true match (*i.e.* not verified by distance-ratio test), in the previous frame. Because of the vector-quantization error inherent to BoW, f and f'' might fall into the same cluster and thereby map to the same visual word. Taking this chance into account, choosing the visual word assigned to f'', rather than a randomly sampled node, to initiate GNNS, will be judicious. Our experimental results, in the following section, show the effectiveness of our algorithm.

We call our proposed method sequential GNNS, or SGNNS, in the rest of the thesis.

4.4 Vector-Quantization Performance Evaluation

In this section, we first describe the datasets and evaluation metrics we use in our experiments in this Chapter. In Section 4.4.2, we investigate the performance of sequential GNNS (SGNNS) while varying k in the k-NN graph and compare it to

Algorithm 2 VQ with Sequential GNNS

Input: F_{curr}: features in current image, F_{last}: features in the last key-frame, W_{last}: visual words assigned to F_{last}, M: maps the index of a feature in current image to its 1NN feature in the last key-frame, $\mathcal{G}$: k-NN graph index

Output: W_{curr}: visual words corresponding to the features in F_{curr}

for $i = 1 \ldots |F_{curr}|$ **do**
  $j = \mathrm{M}(i)$.
  $seed = W_{last}(j)$.
  $query = F_{curr}(i)$.
  $W_{curr}(i) = \mathrm{SGNNS}(\mathcal{G}, seed, query)$.
end for

Algorithm 3 SGNNS

Input: $\mathcal{G}$: k-NN graph index, Q: query point, Y_0: the seed to start the search from.

Output: The first nearest approximate neighbor to Q

ρ is a distance function. $N(Y,\mathcal{G})$ returns the neighbors of node Y in $\mathcal{G}$.
$\mathcal{W} = \{\}$.
$\mathcal{U} = \{\}$.
$t = 1$.
$Y_t = \mathrm{argmin}_{Y \in N(Y_0,\mathcal{G})}\, \rho(Y,Q)$.
while $\rho(Y_t,Q) < \rho(Y_{t-1},Q)$ **do**
  $\mathcal{W} = \mathcal{W} \bigcup Y_t$.
  $\mathcal{U} = \mathcal{U} \bigcup \rho(Y_t,Q)$.
  $t = t+1$.
  $Y_t = \mathrm{argmin}_{Y \in N(Y_{t-1},\mathcal{G})}\, \rho(Y,Q)$.
end while
Sort $\mathcal{U}$, pick the first element, and return the corresponding element in $\mathcal{W}$.

GNNS when applied to the problem of vector-quantization in SLAM. In Section 4.4.3, we study how a k-NN graph constructed using approximate methods can affect the performance of SGNNS on vector-quantization. And finally in Section 4.4.4, we evaluate the performance of our method (SGNNS) by comparing to hierarchical k-means (HKM) and randomized KD-trees, when applied to the problem of vector-quantization in the context of visual SLAM.

4.4.1 Datasets and Evaluation Metric

We performed our experiments on two datasets: an outdoor and an indoor dataset. The outdoor dataset is the City Center dataset[2] (right-side sequence) from [Cummins and Newman 2008] that contains 1237 images. The indoor dataset is a *lab* dataset[3] that has been taken inside a research laboratory using an ActivMedia Pioneer P3-AT mobile robot equipped with a Dragonfly IEEE-1394 camera, and contains 384 images. Different vocabularies with different sizes, 5000-word, 100,000-word and 204,000-word, have been used in our study, that have been constructed using k-means clustering. We clustered 128-dimensional SIFT [Lowe 2004] feature descriptors extracted from different datasets than the above-mentioned. The 100K and 204K-word vocabularies are used to evaluate the performance of our method on large-scale data.

Our performance evaluation metric is the speedup over linear search while fixing the search accuracy in terms of the true nearest neighbors found. We select the parameters in KD-trees, HKM and GNNS/SGNNS such that we can obtain a fixed accuracy and then calculate the speedup of the algorithms over the linear search at the same accuracy. The speedup over linear search is computed as the ratio of the query time by the ANN algorithm over the query time by linear search.

[2] `http://www.robots.ox.ac.uk/~mobile/IJRR_2008_Dataset/`

[3] Dataset #8 (lab2) at `http://webdocs.cs.ualberta.ca/~hajebi/datasets/`

4.4.2 Accuracy vs. Speedup

In this section, we investigate how the performance of SGNNS and GNNS change by varying k, the branching factor of the k-NN graph. Each row ((a), (b) and (c)) in Figure 4.2 shows the trade-off between the speedup (top row) and the accuracy (bottom row) of GNNS versus SGNNS for vector-quantization on matched features and all features.

The first two rows use the 5000-word vocabulary. Figure 4.2(a) and 4.2(b) show the results on the Lab and City Center datasets, respectively. Figure 4.2(c) shows the results of the City Center dataset with the 100K-word vocabulary.

Within all rows, increasing the branching factor k increases the accuracy of the search at the expense of speedup. Increasing the branching factor increases memory and construction time for the graph, as shown in Table 4.1.

In the experiments shown in Figure 4.2, GNNS search starts from randomly chosen nodes for vector-quantization on all features. SGNNS1, exploiting the sequential property of data (see Section 4.2) chooses better initiation seeds for VQ on matched features. For non-matched features, the search is similar to GNNS. SGNNS is similar to SGNNS1 for VQ on matched-features; however, it replaces the random initiation of search on non-matched features with a smarter one, by exploiting the perceptual aliasing problem (see Section 4.3). This improvement over SGNNS1, both in terms of VQ accuracy and speedup, can be seen in Figure 4.2.

Since in GNNS, the search starts from a random node for every feature, matched or non-matched, the performance of the GNNS on "all features" is very close to the one on "matched features"; therefore, we represent only the result for "all features". However, in SGNNS the search initiation is smarter, which results in better performance over GNNS.

The better performance of SGNNS on "matched features" against "all

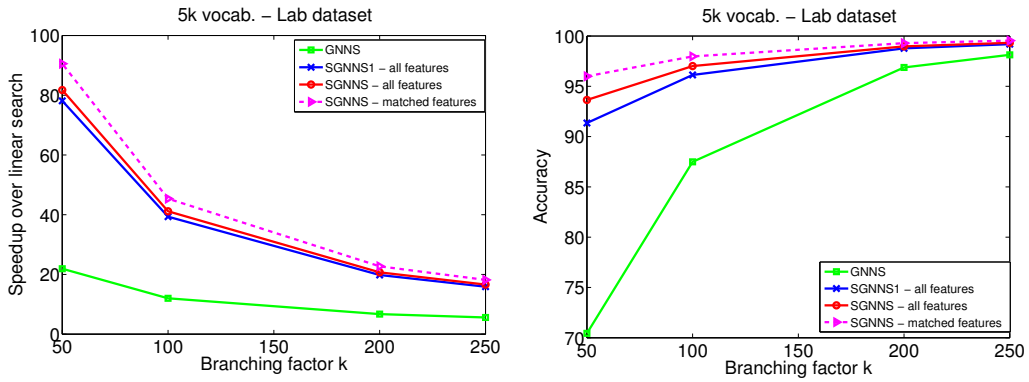
Table 4.1: Construction time and memory usage of k-NN graphs with different branching factors

5K Vocabulary					
Branching factor	50	100	200	250	300
Build time (sec)	5.25	5.40	5.92	6.03	6.04
Memory (MB)	0.95	1.91	3.81	4.77	5.72

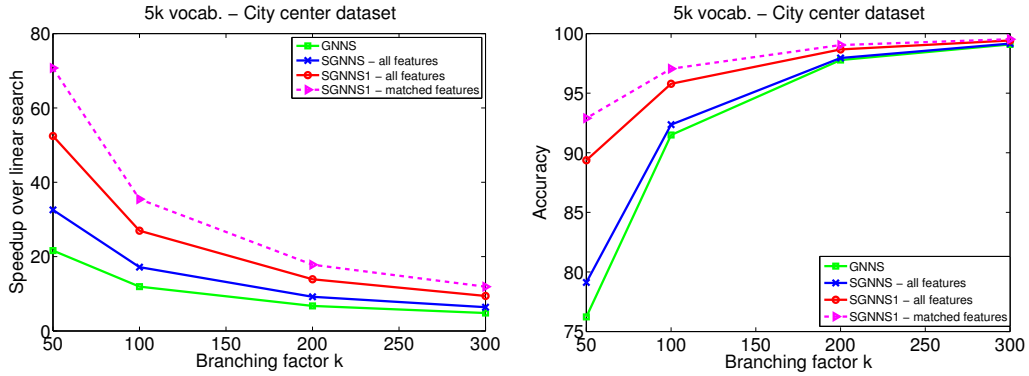
100K Vocabulary							
Branching factor	50	100	200	300	400	500	1000
Build time (min)	25.30	25.88	25.64	24.80	25.96	26.17	27.13
Memory (MB)	19.07	38.15	76.29	114.44	152.58	190.73	381.47

features" curve can be explained as the result of choosing the initiation seeds for the graph search more wisely for matched features than for non-matched features (1-NNs verified by distance-ratio test versus 1-NNs that are not); and because the number of non-matched features are higher than the number of matched features, in our experiments, the performance on "all features" is slightly worse than that on "matched features".

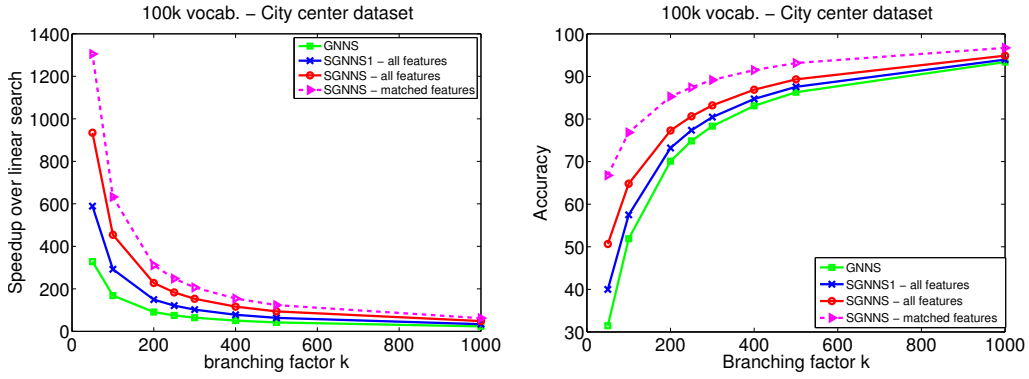
We also observe from our experiments in Figure 4.2(b) and (c), that by increasing the size of the vocabulary from 5K to 100K, the performance on all features (with SGNNS) reduces and the curve gets closer to the GNNS curve. This can be explained by the fact that increasing the size of the vocabulary decreases the vector-quantization error in BoW, which is the property we were exploiting to improve the performance on non-matched features. With a larger vocabulary, the non-matched features and their 1-NNs (in the previous frame) are less likely to share the same or a neighboring visual word.



(a)



(b)



(c)

Figure 4.2: Accuracy and speedup of GNNS and SGNNS for different branching factors

4.4.3 Approximate Graph Construction

In this section, we investigate how the accuracy of vector-quantization with SGNNS degrades by using an approximate graph instead of an exact one. When the graph construction time is a concern, especially when the vocabulary is large, an approximate graph, constructed more quickly, can be a good replacement for an exact graph, at the expense of small reduction in VQ accuracy.

In order to build such an approximate k-NN graph we can use an approximate nearest neighbor search algorithm. In these experiments, we use hierarchical k-means (HKM). We calculate the accuracy of an approximate graph $\mathcal{G}'$, with respect to the exact graph $\mathcal{G}$ that is built using the brute-force method, as:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|N(i,\mathcal{G})\cap N(i,\mathcal{G}')|}{|N(i,\mathcal{G})|}, \tag{4.1}$$

where n is the number of nodes, $N(i,\mathcal{G})$ denotes the neighbors of node i in graph $\mathcal{G}$, and $|\cdot|$ denotes the cardinality of a set. A graph with higher accuracy is closer to the exact graph.

Experimental results are shown in Tables 4.2 and 4.3. In Table 4.2, the k-NN graphs are built over the 5K vocabulary and the queries are from the Lab dataset while in Table 4.3, the vocabulary and query set are the 100K-word vocabulary and the City Center datasets, respectively. Corresponding to each branching factor k, one exact and two approximate graphs are constructed. We adjust the parameters of HKM to obtain accuracies of $\sim 80\%$ and $\sim 52\%$ for each branching factor. The reported accuracies are in the range of [0,1]. The accuracy of the exact graph is 1. The accuracy and speedup of vector-quantization with SGNNS using each graph is shown in the third to sixth columns of each table. The third and forth columns report the results of VQ on all image features and the fifth and sixth columns report the results on only matched features. The last two columns of each table, show the

construction time of the graphs and HKM indices. To construct the exact graph we do not build any search index. All experiments were executed single-threaded with a 2.5GHz Intel Core i5 processor.

By reducing the degree of accuracy of the graph to 80% in Table 4.2, we can observe a reduction in graph construction time by 23 to 29%, while the accuracy of VQ on "all features" is slightly decreased by 1 to 2%. Also, when the graph accuracy is decreased to 52%, the graph construction time is decreased by 29 to 35%, while the VQ accuracy is decreased by 4 to 6%.

However in Table 4.3, we can observe more reduction in the construction time of approximate graphs, as they are built on a larger vocabulary. On approximate graphs of accuracy 80%, we observe a build time reduction by 80 to 86%, over the decreasing branching factor of 1000 to 300. The slight accuracy reduction of VQ on "all features" is by 1 to 2%. When the accuracy of the graph changes to 52%, the construction time is reduced by 88 to 92%, while the VQ accuracy decreases by 5 to 9%.

4.4.4 Comparison with Other Methods

We compare the performance of four methods for vector-quantization: randomized KD-trees, hierarchical k-means tree (HKM), GNNS and our proposed method, Sequential GNNS (SGNNS). For KD-trees and HKM we used the FLANN library[4] implementations.

The only parameter in GNNS and SGNNS is k, the branching factor of the k-NN graph index. For SGNNS we choose the same k as the one we selected for GNNS to obtain the given accuracy. E, the number of expansions in GNNS, is set to k in our experiments.

We choose the version of GNNS in which the search terminates at local minima,

[4] `http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN`

Table 4.2: Performance of SGNNS with approximate k-NN graphs. 5K vocab., Lab queryset

k-nn Graph	Graph build Acc.	VQ on all fts		VQ on matched fts		Graph build time (s)	Index build time (s)
		Acc.	Speedup	Acc.	Speedup		
250 -nn	1	0.9932	16.5914	0.9954	18.1087	6.0371	-
	0.7974	0.9833	16.6233	0.9887	18.1512	4.6241	0.1555
	0.5340	0.9501	16.7422	0.9662	18.3078	4.0165	0.1348
200-nn	1	0.9897	20.6957	0.9929	22.6258	5.9244	-
	0.7929	0.9781	20.7491	0.9853	22.6901	4.4108	0.1550
	0.5120	0.9373	20.9509	0.9581	22.9337	3.8690	0.1367
100-nn	1	0.9701	41.1416	0.9797	45.2212	5.4035	-
	0.8098	0.9566	41.2879	0.9717	45.3728	4.0763	0.1541
	0.5204	0.9153	41.7084	0.9456	45.8251	3.8615	0.1047
50-nn	1	0.9364	81.7385	0.9598	90.2351	5.2541	-
	0.8089	0.9196	82.0912	0.9509	90.5401	3.7507	0.1307
	0.5183	0.8714	83.4794	0.9222	91.7453	3.6036	0.1565

Table 4.3: Performance of SGNNS with approximate k-NN graphs. 100K vocab., City Center queryset

k-nn Graph	Graph build Acc.	VQ on all fts		VQ on matched fts		Graph build time (s)	Index build time (s)
		Acc.	speedup	Acc.	speedup		
1000-nn	1	0.9487	48.4465	0.9671	62.0685	1628.0	-
	0.8078	0.9356	48.4215	0.9559	62.0835	311.1534	3.3799
	0.5146	0.8907	48.3039	0.9185	62.2296	186.4528	3.4921
500-nn	1	0.8931	93.8105	0.9315	123.6187	1570.2	-
	0.8079	0.8769	93.9168	0.9167	123.9136	279.7694	3.8644
	0.5212	0.8195	94.5834	0.8646	125.1055	175.9058	3.7626
400-nn	1	0.8689	116.2389	0.9149	154.6434	1557.8	-
	0.8031	0.8512	116.4652	0.8975	155.1761	219.8821	3.9417
	0.5158	0.7896	117.5713	0.8424	156.8864	137.1464	3.9472
300-nn	1	0.8321	153.4519	0.8919	206.4186	1488.2	-
	0.8001	0.8128	153.9414	0.8718	207.3598	210.7279	4.0707
	0.5120	0.7458	156.1846	0.8108	210.7741	128.4566	4.0361

instead of having T greedy moves. Note that improving the speedup is not possible if we always make a fixed number of greedy moves.

In the case of KD-trees, the FLANN parameters that we set include *trees*, the number of randomized KD-trees, and *checks*, the number of leaf nodes to check in one search. In the case of HKM, the parameter set includes *iterations*, the maximum number of iterations to perform in one k-means clustering, *branching*, which is the branching factor of the tree, and *checks*, the number of leaf nodes to check.

Tables 4.4-4.7 compare the performance of the four search algorithms on vector-quantization. The first two columns show the results when all image features are quantized, and the last two columns show the results when only matched features (*i.e.* the features in current image whose 1-NNs in the previous key-frame have passed the distance-ratio test) are quantized. In SGNNS, for each feature the search starts from the visual word assigned to their 1-NN feature in the previous frame. GNNS starts the search from a random node for each feature.

Tables 4.4-4.6 show the experiments on the City Center dataset. The average number of SIFT features extracted from each image is 300 and the average number of matched features is 63, which is roughly 25% of all features. The speedups for accuracies of $\sim$91% and $\sim$99% are shown in the first two tables. The first two experiments use the 5000-word vocabulary.

In Table 4.4, we used a 100-NN graph for GNNS and SGNNS. For KD-trees, we set *trees* and *checks* to 4 and 140, respectively. For HKM, we set *iterations*, *branching* and *checks* to 4, 8 and 50, respectively.

In Table 4.5, we used a 300-NN graph for GNNS and SGNNS. For KD-trees, we set *trees* and *checks* to 4 and 2200, respectively. For HKM, we set *iterations*, *branching* and *checks* to 7, 8 and 600, respectively.

In Table 4.6, we show the vector-quantization results when a 204,000-word vocabulary is used. We used a 300-NN graph for GNNS and SGNNS. For

KD-trees, we set *trees* and *checks* to 4 and 2200, respectively. For HKM, we set *iterations*, *branching* and *checks* to 7, 8 and 500, respectively.

The last part of experiments has been done on an indoor dataset (Table 4.7) where the overlap between images is larger —around 82% of features are matched. We used the 5000-word vocabulary for this experiment and the average number of SIFT features extracted from each image in the dataset is 98. For GNNS and SGNNS we used a 250-NN graph and for KD-trees, we set *trees* and *checks* to 6 and 400, respectively. For HKM, we set *iterations*, *branching* and *checks* to 3, 8 and 160, respectively.

As can be seen in all experiments, both GNNS and HKM outperform KD-trees. SGNNS outperforms HKM by as much as 250% for the case of vector-quantizing all features, and as much as 400% if only matched features are used in creating the BoW representation of an image. This superior performance is due to two factors: first, the efficiency of graph-based search (GNNS) —as indicated by the third row of each table, over the first two rows of each table —and second, by the exploitation of the perceptual aliasing of BoW along with the sequential property of images whose features are to be vector-quantized, as indicated by the last row (SGNNS) of each table over the third row (GNNS).

Note that in each table, GNNS and SGNNS share the same branching factor. The branching factor of k-NN graph has been chosen such that GNNS, KD-trees and HKM achieve a fixed accuracy. However, SGNNS achieves higher accuracy than GNNS given the same branching factor.

We also observed that the features that are matched between two images do not share common visual words, necessarily. On average, 41% of corresponding features share the same visual words in the experiments presented in Tables 4.4-4.5. This amount was reduced to 24%, with the 204K vocabulary (Table 4.6).

Table 4.4: Comparison of different search algorithms on Vector-Quantization - City Center dataset, Accuracy fixed at ∼91%, 5000-word vocab., 100-NN graph

	VQ of all features		VQ of matched features	
	Accuracy	Speedup	Accuracy	Speedup
KD	0.9164	6.6826	0.9339	6.6866
HKM	0.9078	10.4415	0.9202	10.4477
GNNS	0.9149	10.0432	0.9313	10.0070
SGNNS	0.9579	23.2553	0.9717	29.9503

Table 4.5: Comparison of different search algorithms on Vector-Quantization - City Center dataset, Accuracy fixed at ∼99%, 5000-word vocab., 300-NN graph

	VQ of all features		VQ of matched features	
	Accuracy	Speedup	Accuracy	Speedup
KD	0.9951	1.2449	0.9962	1.2452
HKM	0.9938	3.4861	0.9955	3.4871
GNNS	0.9909	4.0868	0.9924	4.0809
SGNNS	0.9942	8.0012	0.9954	9.9273

Table 4.6: Comparison of different search algorithms on Vector-Quantization - City Center dataset, Accuracy fixed at ∼92%, 204K-word vocab., 300-NN Graph

	VQ of all features		VQ of matched features	
	Accuracy	Speedup	Accuracy	Speedup
KD	0.9364	16.6498	0.9485	16.6489
HKM	0.9185	35.6192	0.9205	35.6177
GNNS	0.9297	59.9617	0.9457	59.0481
SGNNS	0.9343	129.2809	0.9512	172.7866

Table 4.7: Comparison of different search algorithms on Vector-Quantization - Lab dataset, Accuracy fixed at ∼98%, 5000-word vocab., 250-NN graph

	VQ of all features		VQ of matched features	
	Accuracy	Speedup	Accuracy	Speedup
KD	0.9815	2.5448	0.9821	2.5432
HKM	0.9843	3.9291	0.9839	3.9283
GNNS	0.9813	4.6228	0.9825	4.6279
SGNNS	0.9931	13.4739	0.9954	14.6996

4.5 Summary

In this chapter, we propose to use the graph nearest neighbor search (GNNS) algorithm to speed up the vector-quantization task in BoW. We show how GNNS can be integrated into the BoW and SLAM framework and thus improve the performance of the vector-quantization by taking advantage of their properties. The first property is that the images in SLAM are acquired sequentially. This reduces the cost of vector-quantization on the features that are repeated over subsequent frames. The other property that is inherent to the standard BoW model is the perceptual aliasing problem (*i.e.* forcing dissimilar visual features to map to similar visual words). This property allows GNNS to improve the speedup and search precision of VQ on non-matched features.

Our experiments validate the effectiveness of our algorithm and its superior performance over state-of-the-art algorithms.

Chapter 5

Integrated SLAM System

In this chapter, we contribute by developing a SLAM system that integrates the proposed algorithms in the previous chapters. We present a detailed description of different parts of the SLAM system and how they are integrated, in the following subsections. In the next chapter, we will evaluate the performance of our integrated system.

5.1 SLAM framework

In an appearance-based SLAM, the loop closure detection, as the main part, is addressed as an image retrieval task. Because the bag-of-words model has been a successful approach for image search in large datasets in vision and robotics community, we also employ BoW in our SLAM system. However, the BoW model has some deficiencies and needs to be improved based on the application it is utilized for (mentioned in Section 2.2.3). Search in large vocabularies and challenging dynamic environments and perceptual aliasing are the major issues that need to be tackled. We propose an efficient search index to speed-up the vector-quantization process in SLAM and at the same time reduce the error of vector-quantization in BoW that causes perceptual aliasing. Further, we present a

SLAM framework, integrating the improved BoW model with a particle filtering scheme to estimate the likelihood of loop closures exploiting the temporal coherence between sequential images.

An overview of the system is illustrated in the diagram of Figure 5.1. In an offline processing, the visual vocabulary from the training data and the graph search index over the visual words of the vocabulary are constructed. When a robot observes a new image, the interesting features are identified and their descriptors are computed. The image is matched to the previous key-frame (map location), using feature matching, to see if it is sufficiently different to be considered as a new node in the map. Using our graph index and the soft-quantization technique, the features are vector quantized. New word entries are inserted/updated into the inverted index file. In the search step, the BoW's voting system is used to calculate the likelihood of the map locations which are then plugged into a particle filtering scheme to update the loop closure likelihoods. The most likely loop closure locations are then considered for a post verification step. Based on the result we decide whether there is a loop closure.

In the following sections, we describe the components of our SLAM framework in more detail.

5.2 Key-frame Detection

In appearance-based SLAM, the map of the environment is topologically modeled using a graph whose nodes represent the distinct places viewed by the robot. The edges of the graph show the connectivity relationships between physical locations. Key-frame detection, as an important component of SLAM, is required in order to decide when to add a new node to the map. There should be sufficient appearance change between subsequent frames to prevent over-sampling the environment.

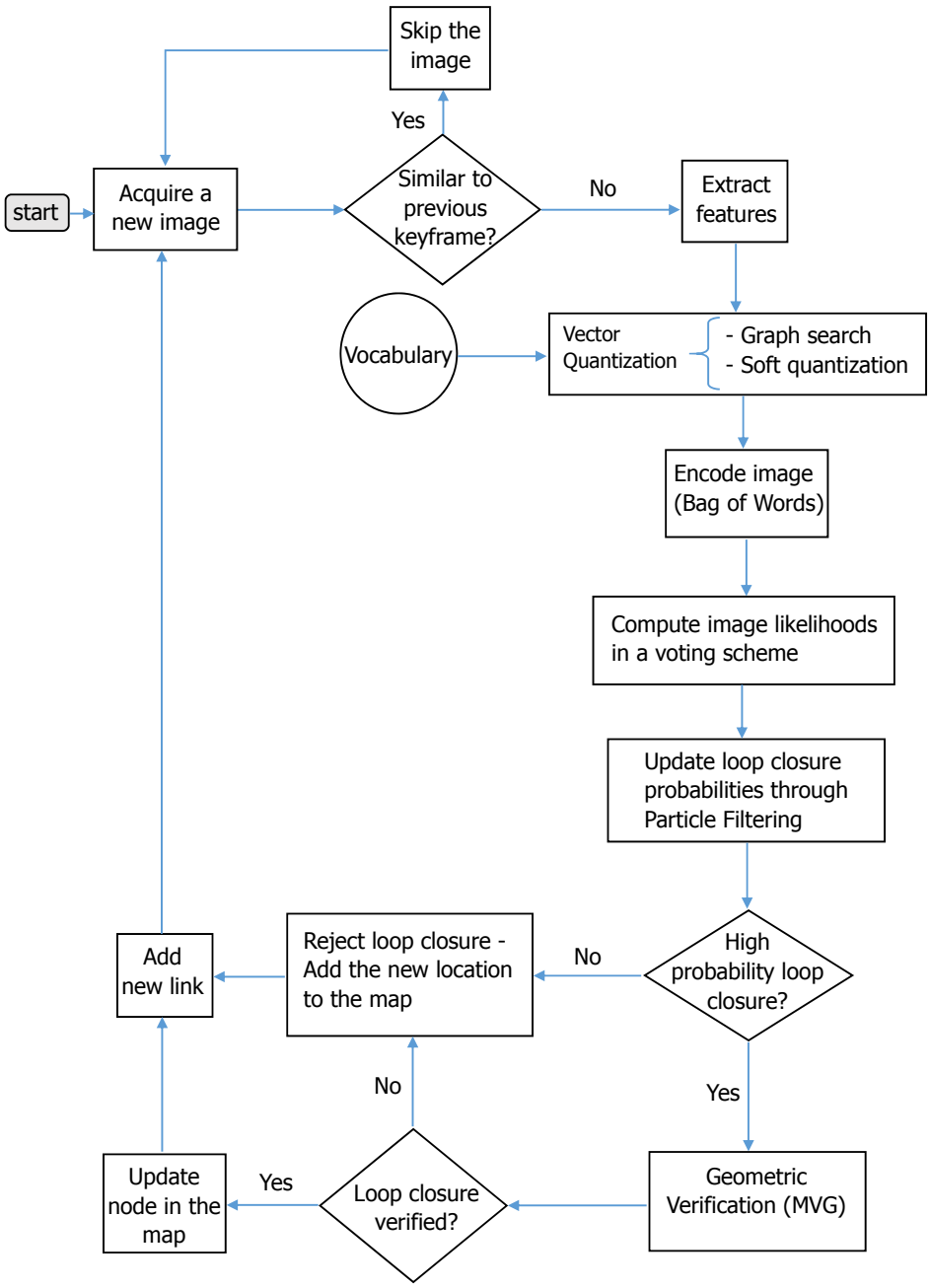


Figure 5.1: An overview of our appearance-only SLAM framework

This allows the map representation to be computationally efficient for loop closure detection and at the same time, provide good visual coverage of the surroundings.

For appearance-based key-frame selection we require an approach to measure the similarity between images. If the similarity is lower than some threshold, we introduce the new observation as a new key-frame to the map. Different image similarity measures have been studied and compared in [Zhang et al. 2010a], among which feature matching performs the best. Visual features of two images are extracted and their descriptors are matched. The similarity score between images can then be computed as:

$$Sim_{fm} = \frac{N_m}{min(N_{im1}, N_{im2})}, \tag{5.1}$$

where, N_{im1} and N_{im2} are the number of features in the first and second image, respectively and N_m is the number of matched features between two images. A feature f_1 and its corresponding descriptor d_1 in the first image will be a match to f_2 and d_2 in second image, if they pass the distance-ratio test: $\text{dist}(d_1, d_2) \leq \text{ratio\_th} * \text{dist}(d_1, d_{22})$; where, d_2 is the first and d_{22} is the second nearest neighbor to d_1. We set ratio_th $= 0.6$ in our experiments. The distance-ratio test is used to reject ambiguous and false matches.

5.3 BoW's Image Representation

We use the BoW model to represent images. Image features are extracted using SIFT [Lowe 1999] or SURF [Bay et al. 2008] detectors/descriptors, as they have proven highly successful in SLAM systems. Features are then mapped to the visual words of the vocabulary. We build the vocabulary using offline processing from a training dataset. An image can then be represented by the set of visual words that occur in that image. As this representation is very sparse, it is stored in an inverted

index data structure to speed-up the image search in large-scale retrieval.

5.3.1 Vector-Quantization

As described above, in the BoW model an image is represented using visual words. This is performed by vector-quantization task. Vector-quantization maps the visual feature descriptors extracted from an image to the visual words of the vocabulary. When the vocabulary is large, vector-quantization process can be computationally expensive. In Chapter 4, we proposed to use a graph-based search algorithm (GNNS) for the search in vocabularies and we showed how efficient our method is against popular search indices. We integrate our method into the SLAM framework to perform vector-quantization. In the next chapter, we will experimentally show the computational cost of loop closure detection can be improved using our search method.

5.3.2 Soft Quantization

Vector-quantization is typically performed as an assignment of one visual word to each visual feature, *i.e.* hard assignment. The discriminative power of feature descriptors is reduced in the quantization process. Two descriptors that match might fall into different clusters and then be considered as different visual words. This is called the effect of fine quantization. Conversely, different descriptors might fall into the same cluster due to coarse quantization.

Philbin et al. [2008] propose a soft assignment approach to assign a feature descriptor to a weighted combination of visual words and hence reduce the error of the quantization process. The method is shown to be more effective than the hard assignment for large scale image datasets ([Philbin et al. 2008]). As we will explain, the soft assignment method can be easily integrated into our graph-based search index. Thus, we aim to employ it to improve the efficiency of the quantization and

retrieval tasks.

In soft assignment, a weight is assigned to each visual word based on the distance of the feature descriptor to the centroid of clusters. This weight is exponential in the distance to the cluster centroid and computed as $\exp(-\frac{d^2}{2\sigma^2})$, where d is the distance from the descriptor to the cluster centroid and σ is a spatial scale. The smaller σ is, the fewer clusters get a substantial weight. σ and r are the main parameters. r is the number of nearest neighbor clusters to which the distance of the feature descriptor is computed. Each feature descriptor is then defined by an r-vector. In experiments in [Philbin et al. 2008], $r = 3$ showed the best results.

Some features are close in descriptor-space, however, the quantization process puts them in different clusters and assigns them different words. Soft assignment gives such close descriptors another chance to match after quantization.

To perform soft assignment, we need to search for the r nearest neighbor clusters to each feature f_i, a process that is computationally costly. However, in the process of vector-quantization(VQ) through our k-NN graph index to find the nearest cluster w_i for feature f_i, we have already computed the distance of w_i to its k nearest neighbors, and the r nearest neighbor clusters to f_i are most likely among the k nearest neighbor clusters to w_i (assuming $k > r$). Thus, performing soft assignment adds no additional cost to our system.

5.4 BoW's Image Matching

Once a new key-frame is acquired by the robot, its visual features are mapped to visual words using SGNNS. The words are then weighed according to the *tf-idf* statistics ([Sivic and Zisserman 2003]) that show how important a visual word is to an image in the dataset. *tf* or term frequency is the number of times a word appears in an image, and is normalized by dividing by the total number of words in

the image. *idf* or inverse-document-frequency measures how common or rare the visual word is across images in the dataset, so that it down-weighs or up-weighs the words, accordingly. The *tf-idf* weight of word i in image t, is therefore computed as follows:

$$tf_{it} idf_{it} = \frac{n_{it}}{n_t} \log \frac{N}{n_i}, \tag{5.2}$$

where, n_{it} is the frequency of word i and n_t represent the total number of words in image t, respectively. N represents the total number of images in the dataset. n_i is the number of images in which word i has occurred.

An inverted index is then constructed/updated to store the mapping from visual words to the images in which the words have appeared. Beside the index of an image, for each word the *tf-idf* weight computed with respect to that image, is also stored and updated in the index. Once a new image is captured, N and n_i change and therefore for each word in the index, the *idf* score should be updated.

At image retrieval stage, each word of the query image is looked up in the inverted index and its the associated image indices and *tf-idf* weights are retrieved. The weights are then used to contribute to the score of retrieved images. A histogram of image scores is formed once all words in query image contributed.

5.5 Loop Closure Detection using Particle Filtering

In the previous sections, we described how images, *i.e.* loop closure candidates, are scored through our BoW voting scheme, once a new frame is captured. Next we explain how loop closures are detected.

Image scores are plugged into a particle filtering framework to estimate the likelihood of loop closure hypotheses. In SLAM systems built on particle filter, the map locations are represented by particles. The set of weighted particles provide a sample-based representation of the PDF distributed over the possible locations of

the robot. Each particle k can then be considered as a loop closure candidate and the loop closure detection problem can be formulated as estimating the full posterior:

$$p(x_t^{(k)}|Z_{0:t}) = w_t^{(k)} \propto w_{t-1}^{(k)}\ p(z_t|x_t^{(k)}, Z_{0:t-1})\ p(x_t^{(k)}|x_{t-1}^{(k)}) \tag{5.3}$$

where $w_t^{(k)}$ is the weight of the particle k at time t. $Z_{0:t}$ denotes the observations up to time t, z_t is the current observation at time t, and x_t is the robot's state in appearance space at time t.

Motion model: The prediction step of a particle filter is performed through a motion model that comprises the state transition probability $p(x_t|x_{t-1})$. The motion model assumes that if the robot is currently at location x_{t-1}, it is likely to move to one of its neighboring locations (nodes) in the topological map at the next time step t. Similar to [Angeli et al. 2008], we choose p to be a Gaussian that has a significant amount of mass on four immediate neighbors of the mean, x_{t-1} (two previous and two next neighbors of x_{t-1}). Motion model enforces the temporal coherence of likelihood updates.

Observation Likelihood: $p(z_t|x_t^{(k)}, Z_{0:t-1})$ denotes the observation likelihoods that are obtained by BoW's voting scheme.

The new weights of particles at time t are then calculated using the motion model and the observations likelihoods and the previous weights. A particle resampling scheme will be adopted to give the particles with higher weights a chance to propagate.

5.6 New Location Detection

Once the full posterior over loop closure hypotheses is updated, we need to decide whether a loop closure has occurred. One approach can be picking the hypothesis that has the highest probability and check if its probability is above some threshold,

then consider it as a loop closure. However the loop closure might not be among the single peaks of the posterior and might rather be diffused over a set of neighboring hypotheses. Similar to [Angeli et al. 2008], we look for the hypotheses for which the sum of the probabilities of their neighboring hypotheses are above some threshold. Once the candidate hypotheses are identified, we verify their geometric consistency with the current observation, as described in the next section.

5.7 Geometric Verification

In order to improve the performance of loop closure detection and maintaining a satisfactory recall rate while improving precision, a post-verification stage is required (especially for large datasets [Cummins and Newman 2008]), to check whether the putative matched locations to the current observation satisfy the geometric constraints. The geometric verification step consists of two parts. First, the N most likely loop closure hypotheses, identified in the previous section, are shortlisted. The similarity of each candidate to the current view is then measured by the ratio of the number of matched features to the minimum number of features between two images. The candidates with similarity above a certain threshold (called `match_th` in our experiments), are passed to the next step of verification which is RANSAC (see Section 2.1.4). RANSAC identifies a set of inliers between each candidate key-frame and the current view. The candidates whose ratio of inliers to matched features is above some threshold (called `inlier_th` in our experiments) are then accepted as correct loop closures.

5.8 Map Optimization with g^{2}o

This thesis is mainly focused on the SLAM front-end in which the loop closure detection is performed using appearance, and a topological map is constructed.

However, a practical SLAM system can considerably benefit from the use of metric data such as odometry data.

Odometry metrical data can be integrated into the topological map, such that the nodes of map graph represent the robot poses, $x_i = (x, y, \theta)^T$, and the edges represent the spatial constraints between poses. Such a graph is referred to as *pose graph* in the literature (see Section 2.1.5). However, the uncertainty in odometry pose estimate accumulates over time, and it can lead to map drift. In order to recover the global structure of the map, a global optimization back-end is suggested for SLAM system. This optimization can be formulated as a nonlinear least squares problem to relax the pose graph. In our experiments, we use the g^{2}o implementation [Kümmerle et al. 2011][1] for SLAM to create a globally consistent trajectory of the robot. Once a new frame is acquired, the odometry constraints and loop closure constraints are fed into the optimizer as its inputs, and the final trajectory estimate is returned. Because of the sparsity of pose graph constraints, g^{2}o can provide real-time performance. Section 6.4 shows our experiments on the integration of loop closure detection, odometry data and the g^{2}o map optimization.

5.9 Summary

In this chapter, we describe the components of our appearance-based SLAM framework. We use BoW image representation and inverted index retrieval along with a particle filtering framework for efficient loop closure detection. The particle filter is used to exploit the temporal coherence of sequential images and BoW model improves the scalability of the system. We also show that the search structure of SGNNS, that is employed for vector-quantization in BoW, allows us to easily implement soft quantization to reduce the vector-quantization error.

[1] `http://openslam.org/g2o.html`

Chapter 6

SLAM Evaluation and Experimental Results

In the previous chapters, we explained our proposed methods for different parts of a large-scale SLAM system, and verified the effectiveness of the proposed sub-components of the system on stand-alone problems. In this chapter, we evaluate the performance of the overall SLAM system with all sub-components integrated.

6.1 Datasets

In order to evaluate the feasibility of our system, we use different types of datasets that have been mostly used as benchmarks in SLAM publications before. We describe each dataset in the following. Some sample images of each dataset are shown in Figure 6.1.

Oxford City Center[1] [Cummins and Newman 2008]: this dataset has been taken along roads near city center, with pedestrians and traffic to test the robustness in dynamic environments and against scene changes. The dataset contains two sets of image sequences. One sequence is taken from the right camera and the other

[1] `http://www.robots.ox.ac.uk/~mobile/IJRR_2008_Dataset/data.html`

from the left camera mounted on the robot. Each sequence of the City Center dataset contains 1237 images and the resolution of images is 640×480. The Oxford City Center dataset has been used as the benchmark for localization and SLAM evaluations. The ground truth data is also publicly available for this dataset.

New College[1] [Cummins and Newman 2008]: this dataset has been taken from large areas of a university campus with repetitive structures to test the robustness against strong perceptual aliasing. The dataset contains two sets of image sequences. One sequence is taken from the right camera and the other from the left camera mounted on the robot. Each sequence of the New College dataset contains 1073 images with a resolution of 640×480. The New College dataset has been used as the benchmark for localization and SLAM evaluations. The ground truth data is also publicly available for this dataset.

Google Street View[2]: this dataset has been collected by a Point Grey Ladybug camera mounted on a moving vehicle, from the streets of Pittsburgh. The dataset contains 12,556 Google street view panoramas of a 13 mile-long run. Each panoramic image is 640×320 pixels, down-sampled from higher resolution images, and is the concatenation of four sub-views taken by front, rear, left and right side cameras. GPS coordinates have also been provided for each location. This dataset has been also used as benchmark in SLAM and image retrieval applications [Singh 2010].

UofA Quad: this dataset is taken from the quad of the University of Alberta campus, using an Xtion Kinect camera mounted on a Clearpath Robotics Husky UGV. The dataset contains 1129 images of resolution 640×480.

Shopping Mall: this dataset is an indoor dataset that has been taken in Alexis Nihon Plaza in Montreal, using a Husky robot, equipped with a Xtion Kinect camera. The dataset contains 484 images of resolution 640×480.

[2]Dataset provided by Google for research purposes.

(a)

(b)

(c)

(d)

(e)

Figure 6.1: Sample images from (a) City Center (right sequence), (b) New College (left sequence), (c) Google Street View, (d) UofA Quad, and (e) Mall datasets.

6.2 Performance Metrics

We quantitatively evaluate the performance of loop closure detection (LCD) as the main part of the SLAM system, using precision-recall curves when the ground truth is available. Precision and recall are defined as:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{6.1}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{6.2}$$

where, *TP* are the true-positive LCDs that have a match in ground truth. *FP* or false-positive is generated when the detected loop closure is not among the true LCs for the current view, or the current view does not have any loop closure at all. Precision is then defined as the ratio of the true-positive detections to the total number of detections. *FN* or false-negative means that the robot does not detect any loop closure and reports the current location as a new location. Recall is defined as the ratio of the true-positive detections to the number of loop closures in ground truth. It is desirable for a SLAM system to detect loop closures without any false-positives as they will lead to filter divergence.

6.3 Loop Closure Detection Performance

In this section, we discuss the performance of loop closure detection on different datasets. We use different types of datasets to demonstrate that our SLAM system can work in real-time with sufficient accuracy, under perceptual aliasing and large changes in appearances, in dynamic environments, with large-scale datasets and can detect different types of loop closures.

We also demonstrate, in sections 6.3.1 and 6.3.2, the significance of our search algorithm SGNNS for the speed improvement of the vector-quantization task, as

it is a computationally expensive task for a real-time system when dealing with large-scale data.

We used 500 particles for the particle filter, in all the following experiments.

6.3.1 City Center Dataset

The City center dataset has been taken along a 2km path as the robot traverses one big loop twice. We used only the right sequence of the dataset in our experiment to evaluate the loop closure detection performance. We used the left sequence of images (which contains different images from the test dataset - right sequence) to build the visual vocabulary for BoW. A 2000-word vocabulary was constructed offline using k-means clustering. We used 128-dimensional SIFT feature descriptors as the input to the clustering.

Precision-recall

We evaluate the performance of our loop closure detection using precision and recall metrics explained in Section 6.2. Figure 6.2 shows the precision-recall curves when a different number of N top loop closure candidates are retrieved, for $N = 1$, 2, 3, 5, and 10. Each curve has been generated by varying the verification threshold, `match_th` (explained in Section 5.7). We used a 300-NN graph, built over the vocabulary, as the VQ search index.

Processing Time

Below, we compare the computational costs of different components of SLAM for processing a new observation. Table 6.1 presents the timing performance of the most time-consuming parts: the SIFT feature extraction, key-frame detection, vector-quantization and geometric verification (MVG) including feature matching, distance-ratio test and RANSAC. We set $N = 10$ for MVG. The times are the

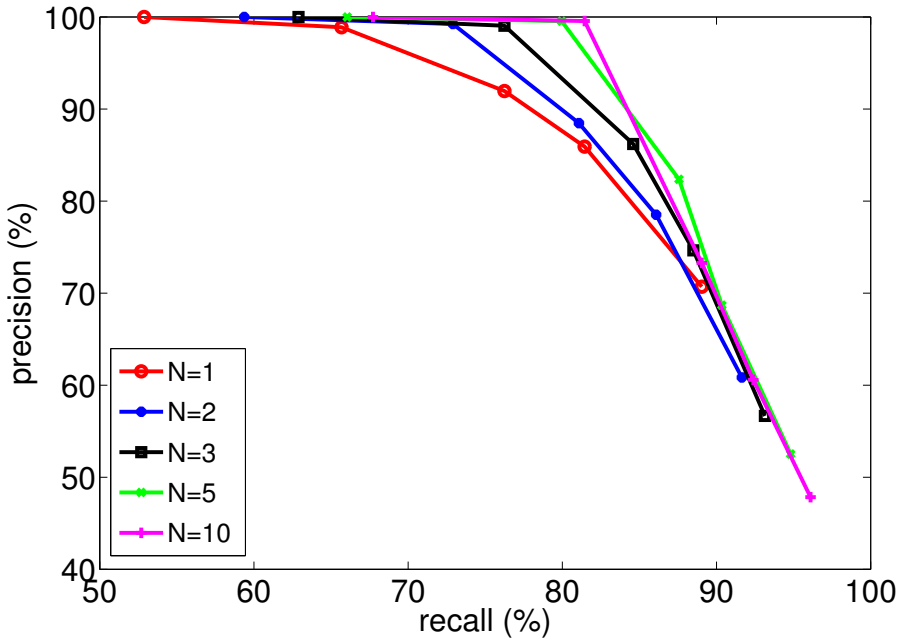


Figure 6.2: Precision-recall curves for the City Center (right sequence) dataset. Each curve shows the precision vs. recall after verification on different number of candidates (N =1, 2, 3, 5 and 10).

average over all 1237 dataset images. Images contain 358 SIFT features on average. The processing times of VQ using linear search and VQ using SGNNS are shown in columns four and five of the table. For SGNNS, we chose the branching factor of k-NN graph so that we can achieve approximately 99% accuracy on vector-quantization with respect to linear search (as the baseline). When the size of vocabulary increases, the vector-quantization using linear search becomes the computationally dominant part in a real-time system. The SGNNS algorithm is very effective in reducing this time.

The times in Table 6.1, have been obtained on a single core of a 2.5GHz Intel Core i5 processor. As our implementation is not optimized, the execution times reported serve mainly to show how the speedup of VQ over linear search changes as vocabulary grows. We believe the times can still be improved by optimizing the implementation.

The total time of other main components, including updating the inverted index,

Table 6.1: Time of different components of SLAM (in sec.) - City Center

Vocabulary size	Feature detection	Key-frame detection	VQ with linear search	VQ with SGNNS	MVG	Others
C = 2K	0.1000	0.0242	0.2098	0.0690	0.2924	0.05
C = 5K			0.5477	0.0948		
C = 10K			1.0947	0.1539		
C = 100K			11.0027	1.4654		

inverted index search (*i.e.* BoW voting) and particle filter update and re-sampling of 500 particles is approximately 0.05 second. The average processing time of each frame on our current system is therefore be less than one second with the 2K-, 5K- and 10K-word vocabularies, and less than two seconds when the 100K-word vocabulary is used. As the robot collected images approximately every two seconds [Cummins and Newman 2008], we achieved the desired real-time performance.

In the following, we demonstrate that we can reduce the processing time of VQ with SGNNS substantially, while the degradation in the performance of BoW-based loop closure detection is insignificant. In Table 6.2 and Figure 6.3, we use k-NN graphs with different branching factors in the VQ process. We can observe, in Table 6.2, that by decreasing the branching factor k from 300 to 30, the processing time of VQ decreases by $\sim 94\%$. Although the accuracy of VQ with respect to the baseline degrades by $\sim 13\%$, this change does not affect the precision and recall values of loop closure detection much. Figure 6.3 also shows that the precision and recall values of LCD, when SGNNS with different graphs is used for vector-quantization, are very close to the result of linear search.

The results shown in Table 6.2 and Figure 6.3 have been produced by a BoW-based LCD and no particle filter is involved. Verification has been performed using distance-ratio based feature matching over the top N=5 loop closure candidates. No RANSAC has been performed.

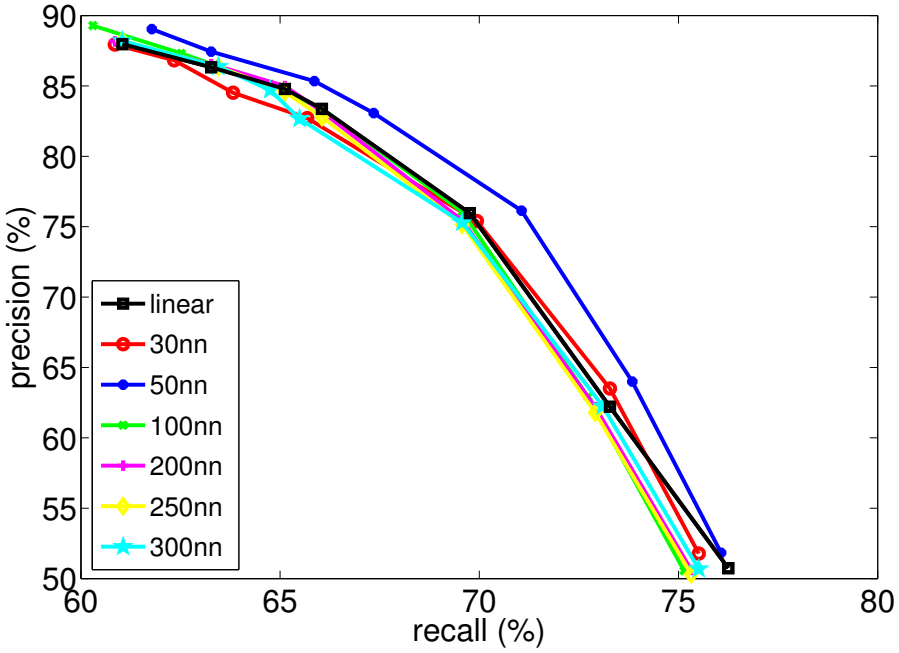


Figure 6.3: Precision-recall curves for the City Center dataset, comparing the performance of BoW-based LCD when linear search is used for vector-quantization vs. when SGNNS is used. Different curves show the results when SGNNS is run on k-NN graphs with different branching factors (k = 30, 50, 100, 200, 250, and 300).

Table 6.2: LCD results on City Center dataset, with VQ using linear search vs SGNNS (2K vocabulary is used)

	Linear search	300nn graph	250nn graph	200nn graph	100nn graph	50nn graph	30nn graph
VQ Time(s)	0.2960	0.0987	0.0706	0.0608	0.0404	0.0224	0.0175
VQ Acc.	1	0.9985	0.9975	0.9956	0.9809	0.9376	0.8679
LCD recall	0.6104	0.6104	0.6104	0.6085	0.6030	0.6178	0.6085
LCD precision	0.8797	0.8820	0.8820	0.8817	0.8929	0.8904	0.8794

6.3.2 New College Dataset

The New College dataset has been taken along a 1.9km path and contains several loops. We used the left sequence of the dataset in our experiment to evaluate the loop closure detection performance. We use a different dataset to build the visual vocabulary for BoW. A 5000-word vocabulary was constructed offline using k-means clustering. We used 128-dimensional SIFT feature descriptors as the input to the clustering.

Precision-recall

Similar to the previous section, we evaluate the performance of our loop closure detection using precision and recall metrics explained in Section 6.2. Figure 6.4 shows the precision-recall curves when different number of top loop closure candidates are retrieved, $N = 1$, 2, 3, 5, and 10. Each curve has been also generated by varying the verification threshold, `match_th` (explained in Section 5.7). We also used a 300-NN graph, built over the vocabulary, as the VQ search index.

Processing Time

Below, we compare the computational costs of different components of SLAM. Similar to Section 6.3.1, we present the run time of the most time-consuming parts in Table 6.3. The times are the average over 1073 images, and each image contains 510 SIFT features on average. With growing size of the vocabulary the vector-quantization becomes expensive and can be speeded-up using SGNNS.

As the number of features per frame has been increased compared to the City Center dataset, we can expect a higher execution time in key-frame detection and MVG (for N=10). However, we can reduce the MVG time by choosing smaller

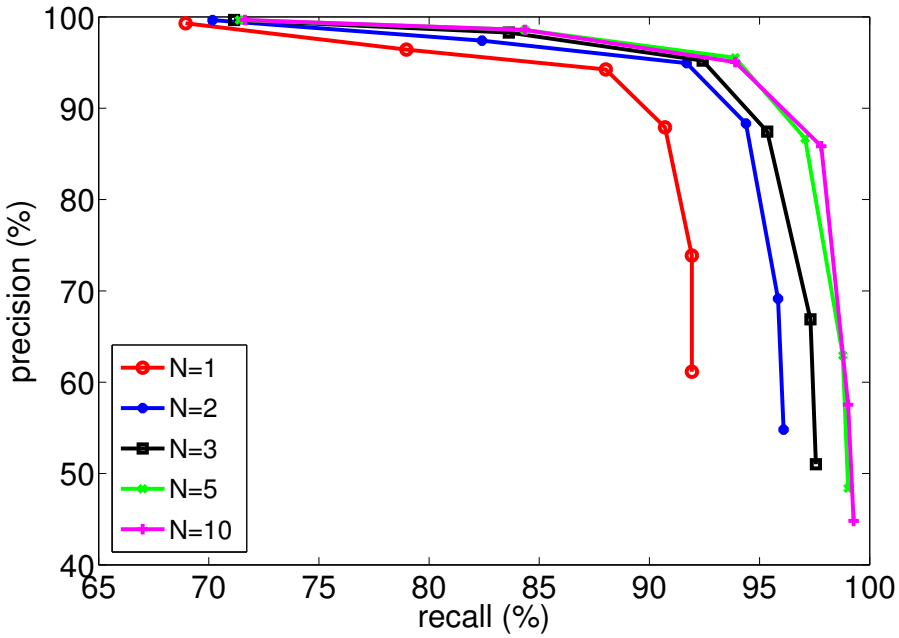


Figure 6.4: Precision-recall curves for the New College (left sequence) dataset. Each curve shows the precision vs. recall after verification on different number of candidates (N =1, 2, 3, 5 and 10).

number of candidates. As can be seen in Figure 6.4, with $N = 5$, we can achieve a similar performance to $N = 10$. As the number of matching candidates are reduced by half, we can expect a reduction in running time by half, as well. The total time of other main components has been measured at 0.08 second.

As in previous experiment, another approach to reduce the total processing time is speeding up the SGNNS-based vector-quantization by choosing graphs with smaller branching factors. Table 6.4 and Figure 6.5 show the results for different sizes of graphs. A 50-NN graph with VQ accuracy of 86% gives a 83% decrease in the processing time of VQ, while maintaining the same precision and recall in LCD.

Table 6.3: Time of different components of SLAM (in sec.) - New College

Vocabulary size	Feature detection	Key-frame detection	VQ with linear search	VQ with SGNNS	MVG	Others
C = 2K	0.1	0.0521	0.3327	0.1073	0.7678	0.08
C = 5K			0.8934	0.1423		
C = 10K			1.6946	0.1990		
C = 100K			15.7361	1.6358		

6.3.3 Google Street View Dataset

In order to evaluate the performance of LCD on a large dataset, we used the Google street view dataset with 12,556 images. We built a 10K-word vocabulary on a different dataset and a 400-NN graph over the vocabulary in an offline phase. Since the ground truth was not provided along with the dataset, we did not evaluate the LCD quantitatively. Based on GPS data and visual inspection, Liu and Zhang [2013] identified 2941 loop closure locations that are shown by green dots in Figure 6.6(b). We also visualize the 2611 loop closures that we detected correctly by red in Figure 6.6(a).

The processing times of different components of SLAM, when using 10K- and 100K-word vocabularies, are presented in Table 6.5. The average number of sampled SIFT features in each image is 257.

Figure 6.7 and 6.8 show some examples of the detected loop closures in the presence of appearance change, like lighting conditions and moved vehicles. Figure 6.7 shows a loop closure when the traversal direction of the vehicle was the same and Figure 6.8 shows one when the vehicle passes an intersection with the road it traversed before.

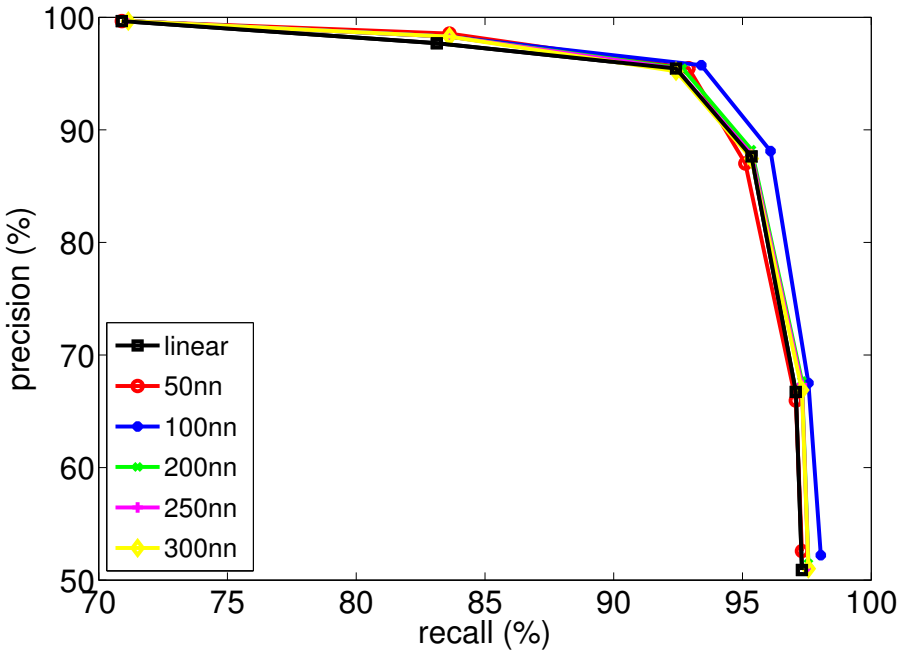


Figure 6.5: Precision-recall curves for the New College dataset, comparing the performance of BoW-based LCD when linear search is used for vector-quantization vs. when SGNNS is used. Different curves show the results when SGNNS is run on k-NN graphs with different branching factors (k = 50, 100, 200, 250, and 300).

Table 6.4: LCD results on New College dataset, with VQ using linear search vs SGNNS (5K vocabulary is used)

	Linear search	300nn graph	250nn graph	200nn graph	100nn graph	50nn graph
VQ Time(s)	1.2621	0.2073	0.1664	0.1396	0.0774	0.0455
VQ Acc.	1	0.9925	0.9890	0.9831	0.9452	0.8606
LCD recall	0.7090	0.7115	0.7115	0.7115	0.7090	0.7090
LCD precision	0.9966	0.9966	0.9966	0.9966	0.9966	0.9966

Table 6.5: Time of different components of SLAM (in sec.) - Google Street View

Vocabulary size	Feature detection	Key-frame detection	VQ with linear search	VQ with SGNNS	MVG	Others
C = 10K	0.1	0.0104	0.7907	0.1171	0.2336	0.06
C = 100K			7.7493	1.2740		

6.4 Map Optimization with g^{2}o

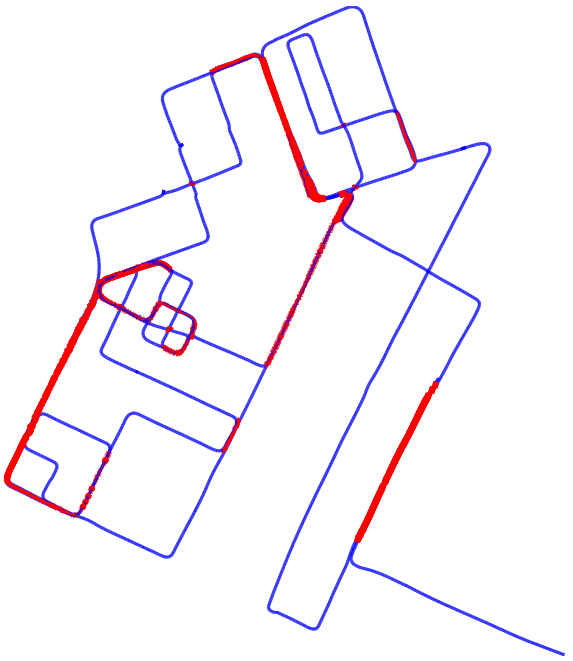
The intention of this section is to show that given odometry (metric) data and the detected loop closures from our SLAM front-end, we can optimize the map by using an optimizer like g^{2}o [Kümmerle et al. 2011]. We run experiments on one outdoor and one indoor dataset, as follows.

6.4.1 UofA Quad Dataset

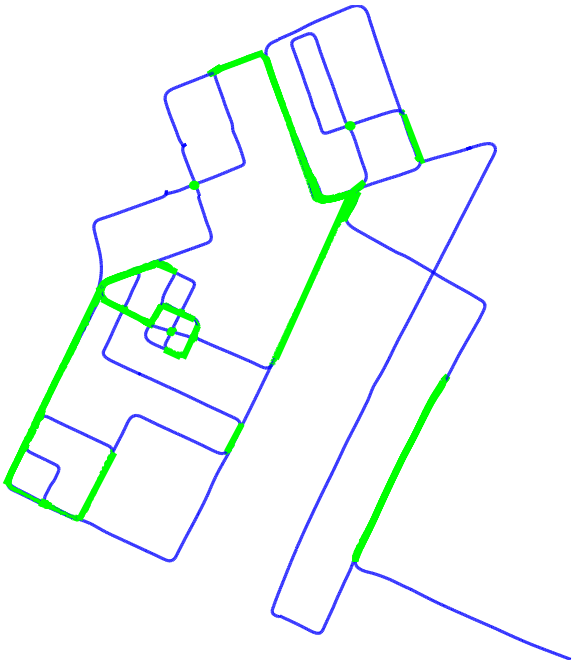
This dataset contains 1129 images. We extracted 451 SIFT features on average from each image. We also built a a 2000-word vocabulary on a training set and a 300-NN graph over the vocabulary in offline pre-processing. Based on our knowledge from the robot trajectory and visual inspection, we identified 465 loop closure locations, among which our SLAM system detected 300 correctly.

Figure 6.9 illustrates the general layout of the trajectory, manually marked over the aerial photo of the environment. The yellow lines show the trajectory while the red ones indicate areas with loop closure events. In the green area the robot traverses the path twice in opposite directions, and was not able to match the locations based on their appearance.

Figure 6.10(a) shows the robot trajectory based on odometry along with the loop closure constraints (matched locations are linked by purple lines). Figure 6.10(b) shows the trajectory optimized by g^{2}o. As the ground truth is not available, we can only provide a qualitative comparison of the maps. It is apparent that the map



(a)



(b)

Figure 6.6: Visualization of (a) detected loop closures, (b) correct loop closures - Google Street View Dataset

Figure 6.7: Loop closure at the same traversal direction

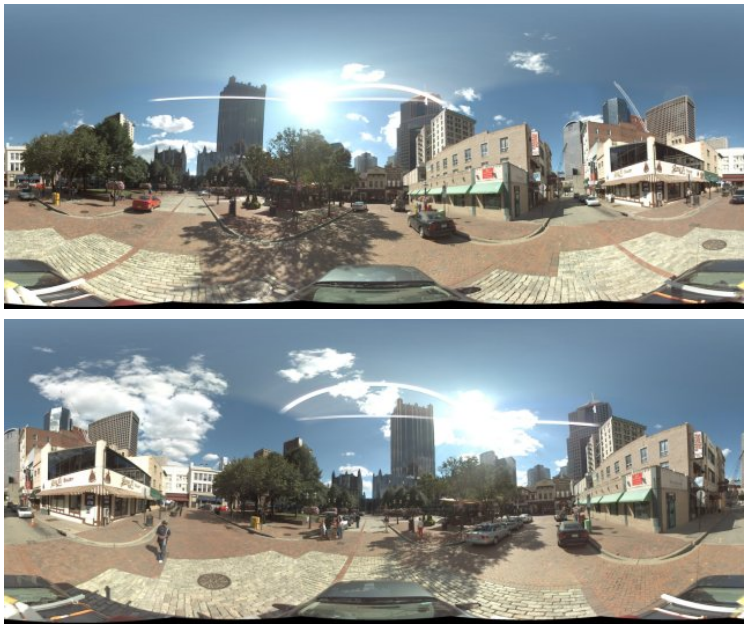


Figure 6.8: Loop closure at an intersection

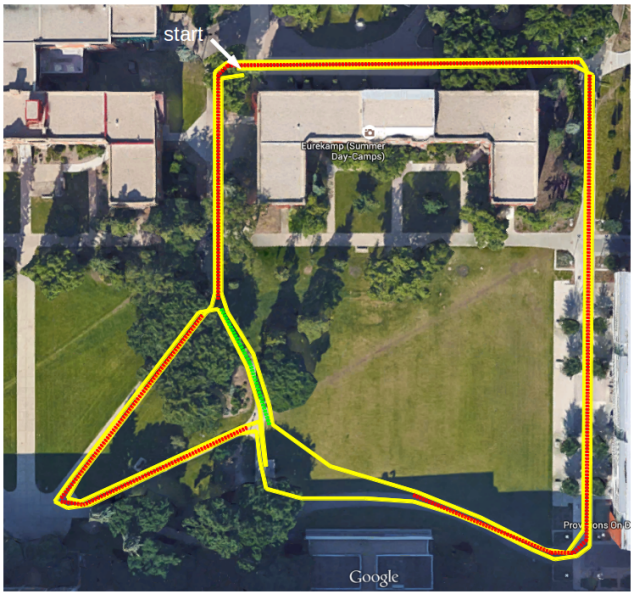


Figure 6.9: The sketched (non-exact) robot trajectory overlaid on the aerial photo of the UofA Quad. Yellow lines show the robot path; the red lines show the area of loop closures where the robot traversed twice in the same direction, and the green line shows the path the robot traversed twice but in opposite directions.

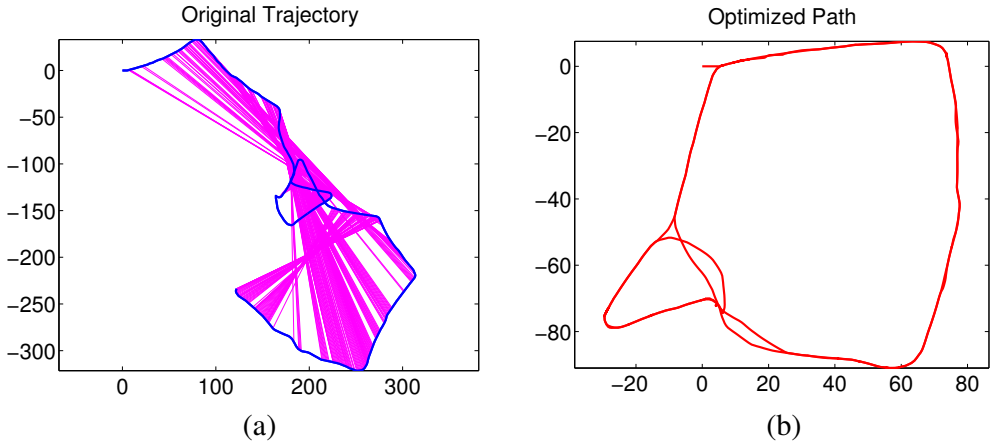
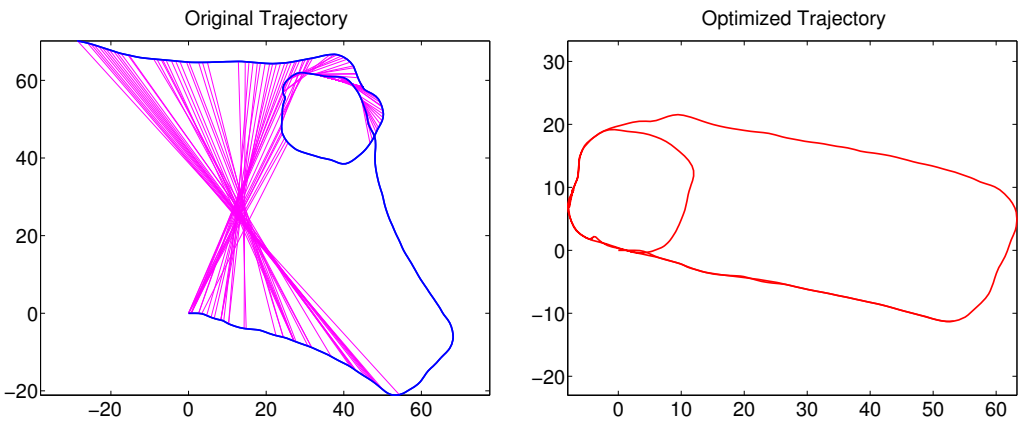


Figure 6.10: UofA Quad dataset. (a) Original odometry with loop closures; (b) final trajectory estimation after relaxation. Axes units are in meters.



(a) (b)

Figure 6.11: Mall dataset. (a) Original odometry with loop closures; (b) final trajectory estimation after relaxation. Axes units are in meters.

optimization correctly captures the general structure of the environment.

6.4.2 Mall Dataset

The Mall dataset contains 484 images. We extracted 399 SIFT features on average from each image. We built a a 2000-word vocabulary on a training set and a 300-NN graph over the vocabulary in offline pre-processing. The trajectory of the robot contains one big and one small loop. Visually inspecting the dataset, we identified 131 loop closure locations, among which our SLAM system detected 106 correctly.

Figure 6.11(a) shows the robot trajectory based on odometry along with the loop closure constraints. Matched locations are linked by purple lines. Figure 6.11(b) shows how the map is refined after relaxation by g^{2}o.

6.5 Summary

In this chapter, we evaluate the performance of our loop closure detection (LCD) method with different datasets and evaluation metrics, while using dissimilar training datasets. We use the City Center and New College datasets to test the

robustness of the system against strong perceptual aliasing and in dynamic environments. As the ground truth is available we are able to compute the precision and recall rates of the loop closure detection. We show that we can achieve high recall rates at 100% precision in real-time. We also compare the computational costs of different parts of our SLAM system and show that the vector-quantization is the computationally dominant part of the system when the size of the vocabulary increases. We demonstrate that our SGNNS algorithm can make VQ efficient and scalable. In another experiment, we study the impact of using smaller graphs, *i.e.* graphs with smaller branching factors, on the performance of our BoW-based loop closure detection method. We observe that by reducing the degree of the graph, we can substantially reduce the time of the VQ, while the degradation in the performance of LCD is insignificant.

We also use the Google street view dataset to test the robustness of the LCD in large-scale environments and show that we can successfully detect a high percentage of loop closures.

In the last part of experiments on UofA quad and shopping mall datasets, we integrate the metric data, *e.g.* odometry, into our appearance-based SLAM system and we optimize the metric map of the robot. As the ground truth is not available we could only qualitatively analyze the results. We show that resulting maps can capture the structure of the environment correctly.

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

In this thesis, we developed an appearance-based SLAM framework that can efficiently work in visually challenging environments. The key to our system's efficiency is the development and selection of schemes which are robust to changes in the environment and scalable to large maps. We first present a graph-based approximate nearest neighbor algorithm (GNNS) in Chapter 3, and experimentally show that it outperforms widely-used NN search algorithms such as KD-trees and LSH. GNNS constructs a k-NN graph as the search index and performs hill climbing starting from random node(s) at search time. The drawback of this method is the expensive construction time of the k-NN graph. However, in our application the graph build time is acceptable, as we construct the index in an offline pre-processing phase. Memory usage might be a concern in applications with limited memory.

The search index of GNNS has advantages over space partitioning tree structures which are used in KD-trees or HKM, and hash tables used in LSH. The complexity of accessing a new nearest neighbor candidate in k-NN graph is $O(1)$, while in tree structures, a sub-tree needs to be traversed to reach a NN candidate in

a leaf node. This leads to more time overhead for searching in trees. Another advantage is that the potential NN candidates are readily ordered as the neighbors of the current returned node. In neighborhood graphs, when a node is returned as a matching candidate to query, its nearest neighbors have a high probability of being good NN candidates to the query as well. In hashing based methods, the query point should be matched to the bucket with corresponding hash code, which contains a large number of matching candidates not ordered in an efficient way for searching. In trees, traversing potential candidates in depth-first or best-first search order is not as effective as that in k-NN graphs to guide the search toward true nearest neighbors. We verified the efficiency of GNNS as a standalone algorithm by running experiments on datasets of different sizes and dimensionality.

The bag-of-words method provides efficient image representation and retrieval in large-scale datasets, and we employed it in our SLAM framework. We demonstrated that vector-quantization of BoW can be a computationally expensive part of loop closure detection when the vocabulary is very large. We proposed to use the graph nearest neighbor search (GNNS) algorithm to speed up the vector-quantization task in BoW. We showed how GNNS can be integrated into the BoW and SLAM framework and improve the performance of vector-quantization by taking advantage of their properties. The first property is that the images in SLAM are acquired sequentially. This property can be exploited to reduce the cost of vector-quantization on the features that are repeated over subsequent frames. The other property that is inherent to the standard BoW model is the perceptual aliasing problem, forcing dissimilar visual features to map to similar visual words. This property allows GNNS to improve the speedup and search precision of VQ on "non-matched" features. Our experiments validated the effectiveness of our GNNS algorithm when applied to the problem of vector-quantization in the context of SLAM and its superior performance over the state-of-the-art algorithms.

As the graph construction of GNNS is computationally expensive, we investigated whether we can benefit from approximate methods. We observed that while using approximate k-NN graphs in GNNS might lower the accuracy of feature matching in vector-quantization, its impact on the accuracy of image matching in BoW is not noticeable. This allows using approximate methods to speed up the process of graph construction.

We finally presented our SLAM framework in Chapter 5, integrating the BoW model with a particle filtering scheme to estimate the likelihood of loop closures by exploiting the temporal coherence between sequential images. We experimentally showed that we can successfully detect loop closures with high precision and recall in real-time in different challenging environments.

7.2 Future Directions

There are several potential future research directions that can be explored to build upon the contributions in this thesis. We describe some of them below:

- **GNNS parameter tuning.** A future direction to our work can be parameter tuning for our graph search algorithm. The only parameter that we tune for VQ in the SLAM application is the branching factor of the k-NN graph. An optimum branching factor can depend on the desired accuracy and constraints on the graph construction time and memory usage. The size of the vocabulary and the query set can also influence the branching factor. As shown in Figures 4.2(b) and (c), by increasing the size of the vocabulary, a higher branching factor is required to maintain high accuracy and speedup. Figures 4.2(a) and (b) imply the dependence of the branching factor on the type of the query set. One can investigate how much each of these factors impact the branching factor of the graph.

– **GNNS search optimization.** The k-NN graph might experience dis-connectivity and sparsity when k, the branching factor, is small. This can make the hill-climbing on graph get stuck in a local minimum and lead to sub-optimal solutions. One potential solution to this problem is the addition of a small fraction of random edges to the k-NN graph that is used as the search index.

– **GNNS seed selection.** Standard GNNS initiates the search from a random node in the graph. In this thesis, we showed that by exploiting the sequential property of SLAM data we can select the starting seed more wisely in sequential GNNS and speed up the feature quantization process. However, we would like to remove the dependency of seed selection on the sequential assumption while maintaining the speedup of SGNNS. One approach can be choosing a random set of nodes in the graph and finding their nearest neighbor to the query feature. The 1NN node can then serve as the starting seed for GNNS. However, there is a trade-off between the number of seed candidates and the search time overhead.

– **Improving the theoretical analysis of GNNS.** The state-of-the-art theoretical results for KD-trees and LSH algorithms avoid the curse of dimensionality when the dataset is generated randomly. Theorem 1 is weaker in the sense that a term exponential in dimensionality appears in the bound. Improving Theorem 1 in this sense can be considered as future work. Much of the modern theoretical work on the KD-trees and LSH algorithms focuses on binary data, as it allows simpler analysis. We would like to follow the same approach and focus our theoretical work on binary data.

– **Bail-out strategies for vector-quantization.** It is always desirable to speed up the vector-quantization process. Although our proposed approximate NN

search algorithm can be efficiently employed in SLAM and provides sufficient speedup for VQ when large vocabularies are used, there are other approaches that can be utilized and combined with an efficient ANN search method to accelerate VQ. In [Hajebi and Zhang 2013], we proposed bail-out strategies for VQ in the appearance-based localization problem. We defined a hardness criterion for image search, considering how distinctive the query image is among all images in the dataset. Given this criterion, we show that the BoW search can be terminated earlier for easier queries. This means that, in such queries, mapping only a portion of features to visual words can be sufficient to yield a relatively good result, while saving a considerable amount of computational resources. Employing such bail-out strategies in BoW-based appearance SLAM can be investigated as future work.

Bibliography

Alok Aggarwal, Leo J. Guibas, James B. Saxe, and Peter W. Shor. A linear time algorithm for computing the voronoi diagram of a convex polygon. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 39–45, 1987.

Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, pages 1027–1037, 2008.

Kazuo Aoyama, Kazumi Saito, Hiroshi Sawada, and Naonori Ueda. Fast approximate similarity search based on degree-reduced neighborhood graphs. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 1055–1063, 2011.

Sunil Arya and David M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 271–280, 1993.

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.

Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, CVPR '97, pages 1000–1006, 1997.

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

Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23:214–229, 1980.

Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The x-tree: An index structure for high-dimensional data. In *Proceedings of the twenty-second International Conference on Very Large Data Bases*, VLDB '96, pages 28–39, 1996.

Marshall Bern, David Eppstein, and John Gilbert. Provably good mesh generation. *Journal of Computer and System Sciences*, 48(3):384–409, 1994.

Peter Biber and Tom Duckett. Experimental analysis of sample-based maps for long-term SLAM. *International Journal of Robotics Research*, 28(1):20–33, 2009.

Sergey Brin. Near neighbor search in large metric spaces. In *Proceedings of the twenty-first International Conference on Very Large Data Bases*, VLDB '95, pages 574–584, 1995.

Yang Cao, Changhu Wang, Zhiwei Li, Liqing Zhang, and Lei Zhang. Spatial-bag-of-features. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 3352–3359, 2010.

Jie Chen, Haw-ren Fang, and Yousef Saad. Fast approximate kNN graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10:1989–2012, 2009.

Kenneth L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, FOCS '83, pages 226–232, 1983.

Mark Cummins and Paul Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6):647–665, 2008.

Mark Cummins and Paul Newman. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems*, 2009.

Mark Cummins and Paul Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *International Journal of Robotics Research*, 30(9):1100–1123, 2011.

Matthew Dickerson and David Eppstein. Algorithms for proximity problems in higher dimensions. *Computational Geometry: Theory and Applications*, 5:277–291, 1996.

Matthew Dickerson, Robert L. (Scot) Drysdale III, and Jörg-Rüdiger Sack. Simple algorithms for enumerating interpoint distances and finding k nearest neighbors. *International Journal of Computational Geometry and Applications*, 2(3):221–239, 1992.

Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 577–586, 2011.

Tom Duckett, Stephen Marsland, and Jonathan Shapiro. Learning globally consistent maps by relaxation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4 of *ICRA'00*, pages 3841–3846, 2000.

David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytypes. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 64–73, 1993.

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

Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

John Folkesson and Henrik Christensen. Graphical SLAM - a self-correcting map. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 383–390, 2004.

Udo Frese, Per Larsson, and Tom Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2): 196–207, 2005.

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*, 3:209–226, 1977.

K. Fukunaga and P. M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers*, 24(7):750–753, 1975.

Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, 1991.

Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, SIGMOD '84, pages 47–57, 1984.

Kiana Hajebi and Hong Zhang. Stopping rules for bag-of-words image search and its application in appearance-based localization. *CoRR*, abs/1312.7414, 2013. URL `http://arxiv.org/abs/1312.7414`.

Kiana Hajebi and Hong Zhang. An efficient index for visual search in appearance-based SLAM. In *Proceedings of IEEE International Conference on Robotics and Automation*, ICRA'14, pages 353–358, 2014.

Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. Fast approximate nearest-neighbor search with k-nearest neighbor graph. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, volume 2 of *IJCAI'11*, pages 1312–1317, 2011.

A. Howard and N. Roy. The robotics data set repository (radish), http://cres.usc.edu/radishrepository/view-all.php [ualberta-csc-flr3-vision], 2003.

Piotr Indyk. Dimensionality reduction techniques for proximity problems. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, SODA '00, pages 371–378, 2000.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 604–613, 1998.

Herve Jegou, Hedi Harzallah, and Cordelia Schmid. A contextual dissimilarity measure for accurate and efficient image search. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 304–317, 2008.

Hordur Johannsson, Michael Kaess, Maurice F. Fallon, and John J. Leonard. Temporally scalable visual SLAM using a reduced pose graph. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 54–61, 2013.

Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM : Fast incremental smoothing and mapping with efficient data association. In *IEEE International Conference on Robotics and Automation*, pages 1670–1677, 2007.

Aram Kawewong, Noppharit Tongprasit, Sirinart Tangruamsub, and Osamu Hasegawa. Online and incremental appearance-based SLAM in highly dynamic environments. *International Journal of Robotics Research*, 30(1):33–55, 2011.

Jon M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 599–608, 1997.

B.J.A. Kröse and R. Bunschoten. Probabilistic localization by appearance models and active vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2255–2260, 1999.

Benjamin Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

Der-Tsai Lee. On k-nearest neighbor voronoi diagrams in the plane. *IEEE Transactions on Computers*, 31(6):478–487, 1982.

Fayin Li and Jana Kosecka. Probabilistic location recognition using reduced feature set. In *IEEE International Conference on Robotics and Automation*, pages 3405–3410, 2006.

Yury Lifshits and Shengyu Zhang. Combinatorial algorithms for nearest neighbors, near-duplicates and small-world design. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 318–326, 2009.

Yang Liu and Hong Zhang. Towards improving the efficiency of sequence-based SLAM. In *2013 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1261–1266, 2013.

David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, volume 2 of *ICCV '99*, pages 1150–1157, 1999.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

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

William P. Maddern, Michael Milford, and Gordon Wyeth. Continuous appearance-based trajectory SLAM. In *IEEE International Conference on Robotics and Automation*, pages 3595–3600, 2011.

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

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

Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In *Eighteenth national conference on Artificial intelligence*, AAAI '02, pages 593–598, 2002.

Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application*, VISSAPP '09, pages 331–340, 2009.

Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:2227–2240, 2014.

Paul M. Newman, David M. Cole, and Kin Ho. Outdoor SLAM using visual appearance and laser ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1180–1187, 2006.

David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2161–2168, 2006.

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.

Aude Oliva and Antonio Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, volume 155, pages 23–36, 2006.

Edwin Olson, John Leonard, and Seth Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA '06, pages 2262–2269, 2006.

Dimitris Papadias. Hill climbing algorithms for content-based retrieval of similar configurations. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 240–247, 2000.

Rodrigo Paredes and Edgar Chávez. Using the k-nearest neighbor graph for proximity searching in metric spaces. In *Proceedings of the 12th International Symposium on String Processing and Information Retrieval (SPIRE 2005)*, volume 3772 of *LNCS*, pages 127–138, 2005.

Rodrigo Paredes, Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Practical construction of k-nearest neighbor graphs in metric spaces. In *Proceedings of the 5th International Workshop on Experimental Algorithms*, volume 4007 of *WEA '06*, pages 85–97, 2006.

James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

Ankush Roy. Towards efficient search methods in object tracking: An evaluation and application to precise tracking. Master's thesis, University of Alberta, 2015.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16:187–260, 1984.

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

Thomas B. Sebastian and Benjamin B. Kimia. Metric-based shape retrieval in large databases. In *Proceedings of International Conference on Pattern Recognition*, volume 3, pages 30291–30296, 2002.

Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *Proceedings of the 2008 Conference on Computer Vision and Pattern Recognition*, CVPR '08, pages 1–8, 2008.

Gautam Singh. Visual loop closing using gist descriptors in manhattan world. In *in Omnidirectional Robot Vision workshop, held with IEEE ICRA*, 2010.

Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2 of *ICCV '03*, pages 1470–, 2003.

Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainly. *International Journal of Robotics Research*, 5(4):56–68, 1986.

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

Niko Sünderhauf, Peer Neubert, and Peter Protzel. Are we there yet? challenging SeqSLAM on a 3000 km journey across all four seasons. In *Proceedings of the International Workshop on Long-Term Autonomy*, ICRA '13, 2013.

Sebastian Thrun and Michael Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *International Journal of Robotics Research*, 25(5-6):403–429, 2006.

Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, 2000.

Iwan Ulrich and Illah R. Nourbakhsh. Appearance-based place recognition for topological localization. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, ICRA '00, pages 1023–1029, 2000.

Pravin M. Vaidya. An o(n log n) algorithm for the all-nearest-neighbors problem. *Discrete and Computational Geometry*, 4:101–115, 1989.

Fangyuan Wang, Hai Wang, Heping Li, and Shuwu Zhang. Large scale image retrieval with practical spatial weighting for bag-of-visual-words. In *MMM (1)*, volume 7732 of *Lecture Notes in Computer Science*, pages 513–523, 2013a.

Jing Wang, Jingdong Wang, Gang Zeng, Zhuowen Tu, Rui Gan, and Shipeng Li. Scalable k-nn graph construction for visual descriptors. In *Proceedings of the Twenty-fifth IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '12, pages 1106–1113, 2012.

Jing Wang, Jingdong Wang, Gang Zeng, Rui Gan, Shipeng Li, and Baining Guo. Fast neighborhood graph search using cartesian concatenation. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 2128–2135, 2013b.

Jingdong Wang and Shipeng Li. Query-driven iterated neighborhood graph search for large scale indexing. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 179–188, 2012.

Junqiu Wang, Roberto Cipolla, and Hongbin Zha. Vision-based global localization using a visual vocabulary. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4230–4235, 2005.

Xiaoyu Wang, Ming Yang, Timothee Cour, Shenghuo Zhu, Kai Yu, and Tony X. Han. Contextual weighting for vocabulary tree based image retrieval. In

Proceedings of the 2011 International Conference on Computer Vision, ICCV '11, pages 209–216, 2011.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes (2Nd Ed.): Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers Inc., 1999.

Zhong Wu, Qifa Ke, Michael Isard, and Jian Sun. Bundling features for large scale partial-duplicate web image search. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 25–32, 2009.

Hong Zhang. BoRF: Loop-closure detection with scale invariant visual features. In *IEEE International Conference on Robotics and Automation*, ICRA '11, pages 3125–3130, 2011.

Hong Zhang, Bo Li, and Dan Yang. Keyframe detection for appearance-based visual SLAM. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS '10, pages 2071–2076, 2010a.

Shiliang Zhang, Qingming Huang, Gang Hua, Shuqiang Jiang, Wen Gao, and Qi Tian. Building contextual visual vocabulary for large-scale image applications. In *Proceedings of the international conference on Multimedia*, MM '10, pages 501–510, 2010b.

Yimeng Zhang, Zhaoyin Jia, and Tsuhan Chen. Image retrieval with geometry-preserving visual phrases. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 809–816, 2011.

Liang Zheng, Shengjin Wang, Ziqiong Liu, and Qi Tian. Lp-norm idf for large scale image search. In *2013 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1626–1633, 2013.

Wengang Zhou, Yijuan Lu, Houqiang Li, Yibing Song, and Qi Tian. Spatial coding for large scale partial-duplicate web image search. In *Proceedings of the international conference on Multimedia*, MM '10, pages 511–520, 2010.

Wengang Zhou, Houqiang Li, Yijuan Lu, and Qi Tian. Large scale image search with geometric coding. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 1349–1352, 2011.

Wengang Zhou, Houqiang Li, Yijuan Lu, and Qi Tian. SIFT match verification by geometric coding for large-scale partial-duplicate web image search. *ACM Trans. Multimedia Comput. Commun. Appl.*, 9(1):4:1–4:18, 2013.