****

**本科生毕业论文**



**题目：高维数据近似最近邻算法研究**

**学 院 智能与计算学部**

**专 业 计算机科学与技术(新工科试验班)**

**年 级 2018级**

**姓 名 杨鸣宇**

**学 号 3018216331**

**指导教师 赵来平**

**独创性声明**

本人声明：所呈交的毕业设计（论文），是本人在指导教师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本毕业设计（论文）中不包含任何他人已经发表或撰写过的研究成果。对本毕业设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在论文中作了明确的说明。本毕业设计（论文）原创性声明的法律责任由本人承担。

论文作者签名：

年 月 日

本人声明：本毕业设计（论文）是本人指导学生完成的研究成果，已经审阅过论文的全部内容。

论文指导教师签名：

年 月 日

**摘 要**

最近邻检索问题为数据库、数据挖掘和信息检索领域的一个基础性问题。随着机器学习、人工智能和数据挖掘等新兴技术的快速发展，数据特征也逐渐由高维向量数据表示。最近邻检索问题，则是找出和查询数据最相似的（距离最近）的数据。高维向量之间的相似性则由数据中的相似性则由欧式距离、汉明距离和余弦距离等度量方式进行计算。在数据规模极大的数据集中进行精确的最近邻检索的检索代价较大效率相对较低。一种可行的方案是使用近似最近邻检索（ANN）在牺牲小部分检索精度的代价下大幅提升检索效率。

近似最近邻检索有多种具体实现方式，其中量化技术和图索引技术应用较广，且性能远超其他近似最近邻检索技术。本文以目前的state-of-the-art算法HNSW和NSG算法作为图索引的具体实现方法。分析两者在SIFT1M等数据集下的检索效率、检索使用空间等性能参数。对于HNSW和NSG算法面对较大的原始数据所构建的索引使用空间过大这一问题，本文采用量化技术如标量量化、乘积量化等方式对图索引使用空间进行压缩。在此基础上，本文对量化后数据图索引距离计算的各种方式如非对称距离计算、对称距离计算方式进行性能分析，并引入了距离查询表加速非对称距离计算效率。

实验结果表明，使用量化技术可以在保证召回率要求的情况下将图索引使用空间进行大幅压缩，距离查询表可以在码本较小、数据访问较高量的场景下优化查询效率。该方法可以应用于推荐系统、视频去重等实际业务应用场景。

**关键词：**近似检索，乘积量化，图索引结构，相似度检索，近似最近邻，查询优化

**ABSTRACT**

Nearest neighbor retrieval problem is a fundamental problem in the fields of database, data mining and information retrieval. With the rapid development of technologies in scientific fields such as machine learning, artificial intelligence, and data mining, data features are mainly represented by high-dimensional vector data. he similarity between high-dimensional vectors is calculated by the similarity in the data, which is usually calculated by measures such as Euclidean distance, Hamming distance and cosine distance. Accurate nearest neighbor retrieval in very large datasets is expensive and relatively inefficient. A feasible solution is to use approximate nearest neighbor (ANN) retrieval to greatly improve retrieval efficiency under the trade off of small retrieval accuracy.

Approximate nearest neighbor retrieval can be achieved in the trade off of time and space in a variety of ways. Among them, quantization technology and graph indexing technology are widely used, and their performance far exceeds other approximate nearest neighbor retrieval technologies. In this paper, the current state-of-the-art algorithms HNSW and NSG are used as the implementation method of graph indexing. The performance parameters such as retrieval efficiency and retrieval space are analyzed under the SIFT1M and other datasets. It is worth noting that the index built by the HNSW and NSG algorithms uses too much memory when the original dataset is very large. Quantization techniques such as scalar quantization and product quantization can compress the graph index memory. On this basis, this paper analyzes the performance of various methods of calculating the distance of the quantized data graph index, such as asymmetric distance calculation and symmetric distance calculation, and then use a distance calculation table to improve the efficiency of asymmetric distance calculation.

The experimental results show that the use of quantization technology can compress the memory used by the graph index while ensuring the recall rate, and the distance calculation table can optimize the query efficiency in the scenario of small codebook and high data access. This method can be applied to practical business application scenarios such as recommendation systems and video deduplication.

**KEY WORDS:** Similarity Search, Product Quantization, Graph Index, Approximate Nearest Neighbor, Query Optimization

目 录

[第一章 绪论 1](#_Toc103791436)

[1.1 研究背景与意义 1](#_Toc103791437)

[1.2 问题定义 1](#_Toc103791438)

[1.3 研究现状 2](#_Toc103791439)

[1.4 研究内容 3](#_Toc103791440)

[第二章 相关理论 5](#_Toc103791441)

[2.1 基于哈希和倒排索引的方法 5](#_Toc103791442)

[2.1.1 哈希技术 5](#_Toc103791443)

[2.1.2 倒排索引 5](#_Toc103791444)

[2.2 基于树形结构的方法 5](#_Toc103791445)

[2.3 基于量化技术的方法 6](#_Toc103791446)

[2.3.1 标量量化 6](#_Toc103791447)

[2.3.2 迭代量化 7](#_Toc103791448)

[2.3.3 乘积量化 7](#_Toc103791449)

[2.4 基于图的方法 8](#_Toc103791450)

[2.4.1 单调搜索网络 8](#_Toc103791451)

[2.4.2 基于Delaunay图的方法 9](#_Toc103791452)

[2.4.3 基于KNN图的方法 10](#_Toc103791453)

[第三章 图索引相关技术研究 12](#_Toc103791454)

[3.1 图搜索算法 12](#_Toc103791455)

[3.2 图构建算法 13](#_Toc103791456)

[3.3 图优化算法 13](#_Toc103791457)

[3.4 性能分析 14](#_Toc103791458)

[3.4.1 时间复杂度分析 14](#_Toc103791459)

[3.4.1 内存使用 15](#_Toc103791460)

[第四章 量化优化方法 16](#_Toc103791461)

[4.1 标量量化优化 17](#_Toc103791462)

[4.2 乘积量化优化 18](#_Toc103791463)

[4.3 乘积量化数据距离计算 18](#_Toc103791464)

[4.4 图索引量化距离计算 18](#_Toc103791465)

[第五章 实验结果与分析 19](#_Toc103791466)

[5.1 数据集与实验配置 19](#_Toc103791467)

[5.2 实验结果 19](#_Toc103791468)

[5.3 索引构建空间 21](#_Toc103791469)

[5.4 结果分析 22](#_Toc103791470)

[第六章 总结 24](#_Toc103791471)

[参考文献 25](#_Toc103791472)

[附 录 27](#_Toc103791473)

[致 谢 28](#_Toc103791474)

# 绪论

## 1.1 研究背景与意义

数据的最近邻搜索（Nearest Neighbor Search）也称作最近点搜索，是指在不同尺度空间中搜索与距离查询点最近点的优化问题，同时也是数据库、数据挖掘和信息检索领域的一个基础性问题。随着人工智能、数据科学等学科的快速发展，数据特征也逐渐由维度在100以上的高维特征向量来表示。在实际应用中，从大量高维数据中进行近邻NN（Nearest Neighbor）搜索寻找数据实体之间的相似性在众多领域是不可或缺的过程。其在数据挖掘、地理信息检索、机器学习、数据库、多媒体、社交网络和计算机视觉等领域发挥着重要作用。在数据集中数据维数较低时，采用树形数据结构或哈希技术进行精确最近邻（NN）搜索在运行效率和索引空间使用上十分有效。当数据维度增加时，该类算法会随着维数的变化而严重受损。

然而，在许多实际应用场景中，数据最近邻检索精度的损失并不会给模型和任务带来影响。针对这一点，采用近似最近邻搜索技术ANN（Approximate Nearest Neighbor）可以在牺牲微小精度的情况下大幅提高计算效率。近似最近邻搜索技术（ANN）近期以图索引算法和乘积量化算法为最佳选择。其分别在检索效率和索引空间使用两个方面有着相对于传统空间划分、哈希和树形数据结构等方法的大幅提升。其在推荐系统、语义文档检索和多媒体数据查询等应用领域有着巨大应用潜力。因此，相对于传统方法，图索引算法和乘积量化方法更具有研究价值，在实际问题中的应用也更加广泛。

## 1.2 问题定义

ANN（Approximate Nearest Neighbor）近似最近邻搜索问题定义如下：代表了维空间中的共有条向量的向量数据集，代表查询向量。给定查询范围且，近似最近邻搜索（ANN）需要在向量数据集中搜索距离查询向量最近的向量即（）。近似K近邻检索（）则需要在向量数据及中查询距离查询向量最近的条向量，其中距离计算函数定义为，返回结果和计算过程形式化定义如下：

 （2-1）

返回结果且，在（1-1）中and保证。

本文重点关注在欧式空间的向量相似度计算，向量之间的相似度使用欧几里德距离进行计算。

## 1.3 研究现状

国内外针对数据的最近邻检索这一问题提出众多方法，这些方法可以分为以下几类：基于哈希（散列）的方法、基于树形结构（空间划分）的方法、基于量化技术的方法和基于图的方法。基于哈希的方法有多种实现方式，以局部敏感哈希[1]为例，其基本思想为对数据点使用特殊的局部感知哈希函数进行哈希处理，这样“相似”的数据点相对于“不相似”的数据点更有可能发生哈希碰撞（散列到同一个桶中）。该方法也存在查询与其绝对最近邻之间的距离相对较大时，哈希算法的搜索效率下降等限制。

基于树形结构的方法如KD-tree[2]，其在数据维度较低的场景上有着良好的表现，但是在高维数据集上表现不佳。其原因在于KD-tree的子树分支（子树对应的空间划分）随着数据维数的增大也会增多，这就意味着需要回溯判断的树分支增加，从而导致算法的查找效率大幅下降。一种解决方法时采用随机划分的KD-tree[3]、渐进KD-tree[4]等KD-tree改良结构，该类方法可以在损失部分精度的代价下提升搜索效率。另一种十分有效的树形结构（空间划分）方法为 K-means tree，与KD-tree对数据的维度进行划分不同。K-means tree[5]采用数据的内在结构对数据进行分组。该分组方式会给查询带来额外的开销，一般来说，树形结构方法的查询从近似最近邻树的顶部到底部的路径的计算效率较高。然而采用这种单一路径的方式会导致较低的精度。出于这个原因，回溯算法或者随机多条路径的方法可以大幅提高树形结构方法的最近邻检索精度，但其缺点则是回溯和多条路径导致访问节点增加，进而导致的查询代价上升。

基于量化技术的近似最近邻（ANN）检索方法以乘积量化PQ[6]（Product Quantization）方法为主，乘积量化（PQ）的基本方法为，将原始空间分解为有限数量的低维子空间的笛卡尔积。之后，对低维子空间的数据点进行聚类划分，每个点所对应的子空间通过聚类质心（centroid）表示。对聚类质心编码，原始向量即可用所有子空间编码和对应的码本（code book）表示。量化技术可以对高维数据所需内存空间进行大幅压缩，但量化过程本身也会导致数据点之前距离出现误差等情况。量化技术一定程度上缓解了海量高维数据索引构建对内存的大量需求，同时也加快了原始数据点之间距离的计算速度。

基于图（Graph）的高维数据近似最近邻（ANN）检索方法在检索效率和精度方面，到目前为止优于其他所有ANN检索方式。基于图的检索方式的主体思想为，将高维数据看作图中的节点，通过节点之间相互连边，构建成一个有向图或无向图。查询过程为，从图中的起始节点出发，每次贪心的搜索距离查询节点最近的邻居（neighbor），将ANN检索转化为图上的搜索问题。基于图的早期方法如NSW（Navigable Small World）[7]在高维数据上表现较差，其所构建具有导航性质的图节点出边和维度数正相关导致节点出度过高，进而导致检索速度下降。HNSW（Hierarchical Navigable Small World）[8]通过构建一个多层次的NSW图并应用类似RNG[9]的裁边策略对节点出边进行修剪，来达到提升检索速度的目的。在K临近图的基础上进行优化的NSG（Navigating Spreading-out Graph）[10]也应用了同样的裁边策略，这两种方法在SIFT[11]和GIST[12]等高维数据集上表现极佳，但在索引空间使用等方面还有优化空间。

## 1.4 研究内容

高维数据近似最近邻检索问题(ANN) 近邻搜索是在给定数据集中找到和查询数据（通常用向量表示）最相似的数据，相似程度的计算有多种方式，如向量之间的欧式距离或汉明距离[13]等。本文主要关注在欧式空间下的数据近似最近邻问题。

随着数据库规模达到数百万或数十亿个条目[14]，向量维度增长到数百维[15]，为了在准确性和效率之间取得更好的平衡，采用近似最近邻搜索是一个十分必要的选择。目前基于图的ANN检索技术可以做到在准确性极高的同时，仍能保持远超其他所有方法的检索效率。目前，ANN-Benchmark的研究表明[16]HNSW算法在检索效率和检索准确率两方面略高率其他基于图的算法，且远超其他基于树形结构和哈希技术的方法。NSG、SSG和Hierarchical SSG[17]虽然在某些数据集上有着略高于HNSW的性能，但其仍存在索引构建后不能新加入数据等缺点。从另一方面来来说，NSG算法在同等检索性能下相对HNSW等算法使用了更少的连边，其在空间使用上相对其他算法有着不小优势。因此，HNSW和NSG将作为本文算法的baseline。

基于图的ANN检索算法虽然能在效率和召回率上做到相对其他算法的最优，但其仍存在索引空间内存过大等问题。对于数十亿条高维数据，较大的图索引的计算可以使用Spark、Hadoop集群或128G内存的主机组上进行计算。然而，普通服务器或虚拟机上很难适配近100G的索引空间，导致基于内存的图索引算法在部署服务时受到多方限制。量化技术可以将高维数据重新编码，以部分相对原始数据的精度损失作为代价，进而大幅压缩索引构建所需空间。只使用量化技术进行ANN检索，仍不能避免检索效率过低这一缺点。本文主要关注量化技术和基于图索引技术的方法结合，使得在损失部分召回率的情况下（由量化失真导致）对图索引使用空间进行压缩。在此基础上，本文对量化数据在图索引上的多种距离查询方式进行了性能分析，并采用渐进构建量化距离表的方式对非对称距离计算进行了查询效率优化。

# 相关理论

## 2.1 基于哈希和倒排索引的方法

### 2.1.1 哈希技术

哈希技术在ANN检索应用上以局部敏感哈希[1]方法为主，该方法通过构建哈希表，哈希表将高维数据映射到一个低维的哈希码（hash code）。在使用过程中将查询数据的哈希码和哈希表中的哈希值进行比较，单个哈希表的检索精度和数据中元素的取值范围和数据分布相关。数据过于密集可能导致相同哈希码的元素过多，进而导致检索效率降低。

单个哈希表的查询结果往往不能满足ANN检索的精度需要，一种可行的解决方法为构建多个哈希表。查询过程为在每个哈希表中进行相同哈希值检索，哈希表使用的越多，查询数据的最近邻被映射到相同哈希值的概率越大，从而保证ANN检索召回率要求。

### 2.1.2 倒排索引

倒排索引技术[18]进行ANN搜索的方式同哈希方法类似，可以将倒排索引的桶看作某种哈希映射方法。倒排索引的基本思想和哈希算法类似，即查询数据的最近邻点更可能在距离其最近的几个桶中的点。将聚类算法应用于原始向量数据。将向量根据其聚类的簇类别进行编号，将原始向量加入到其编号所对应的倒排索引中，即将相同簇的数据点放入同一个“桶”中。查询过程为，计算倒排索引中的所有“桶”所对应簇的质心和查询向量的距离，进行升序排序，根据查询精度要求检索排名前K桶中的所有数据点。

## 2.2 基于树形结构的方法

基于树形结构[19]的方法的基本思想多为将空间划分为多个离散的子空间，所以也可以称为空间划分的方法。早期的树形数据结构如KD-tree[3]在低维数据上的检索效率极佳，并且可以返回精确的最近邻NN搜索结果。但是其在高维数据上的表现不佳，在维数高于100的高维数据上检索效率甚至可能退化到线性扫描的时间复杂度。一种类似局部敏感哈希构建多个哈希表的方法为构建多颗随机划分KD-tree，在多颗KD-tree上进行ANN检索以弥补单颗随机划分KD-tree召回率较低的问题。同时，该方法还可以用来构建精确的KNN图[20]。KD-tree还有多个改进变种，如贪婪近似KD-tree等优化方法，可以保证KD-tree的快速构建和在高维数据情况下ANN检索召回率保证。

另外一种基于近似空间划分的树形数据结构为SA-tree[19]，其对每个节点的应用的连边策略广泛应用于基于图进行ANN检索的方法中。SA-tree通过三角不等式和类似KD-tree的空间划分来判断是否访问到了最近邻点。相对于其他ANN检索方法而言，SA-tree可以返回精确的最近邻检索结构，且SA-tree使用的树形结构对内存占用极低。其缺点在于SA-tree检索方式为检索精确的最近邻导致其检索效率不高，且对于K近邻问题没有很好的适配方法。所以其在实际应用场景中的应用不是很广泛。

## 2.3 基于量化技术的方法

量化技术在信息论、数据压缩等领域有着十分广泛的应用，本文重点关注向量量化这一问题。量化是一个结构性解析的过程，其主要目的在于降低空间表示所需要的基数[6]。具体来说，一个量化器（quantizer）是一个将维向量映射到中的某个码字（codeword）的映射函数，其中索引集合为有限编号集。所构建的集合被称为码本。该映射函数可以表示为。在信息论中函数被称为编码器（encoder），函数被称为解码器（decoder）[21]，向量量化的失真函数定义如下：

 （2-2）

其中为数据集采样点的个数，代表l2-norm。给定一个码本，最小化失真 的量化器必须满足第一个Lloyd条件[6]。即编码器 应该将任何映射到其码本中最近的codeword c。两个向量之间的距离可以近似的通过其映射的码字c之间的距离表示。码本的构建和向量的编码均可以在离线的条件下提前计算，且码本的计算不一定依赖于整个原始数据集，可以在规模远小于原始数据集的训练集或从原始数据集中进行采样部分数据进行码本计算。码本的有多种计算方式，本章主要介绍以下三种常见的编码方式。

### 2.3.1 标量量化

标量量化[22]（Scalar Quantization）SQ主要针对高维数据每一维数据进行压缩，其基本思想是将较长的浮点数或长整数用较短的二进制码来表示。faiss[23]提出了一种基于标量量化的压缩方式。例如对于采用32位编码的浮点向量，faiss类似于正则化的方式将其压缩位8位二进制数组成的整数值。相较于原始向量，压缩后的数据在空间使用上只占原来的四分之一。其优势不仅在内存使用方面，压缩后的数据相较于原始浮点数向量而言有着更快的距离计算效率，可以一定程度的降低检索代价。但该种方法的缺点在于较大的数据精度损失，在近似最近邻ANN问题或K较小的近似K近邻问题上很难保证较高的召回率。

标量量化的另一种极端情况为将原始高维数据向量压缩为二进制01串[24]，将高维欧氏空间转换为高维汉明空间，距离计算也由欧式距离转化为汉明距离。该方法可以与局部敏感哈希和鸽笼原理结合起来进行精确的汉明空间中的最近邻检索[25]，进而避免扫描整个数据集。该方法在SIFT数据集上进行图像搜索有着良好的应用[24]，但是其检索效果受二进制压缩方法影响较大，有着较大的精度损失。其将欧式距离转换为汉明距离的量化方式可以看作是迭代量化的一种。

### 2.3.2 迭代量化

迭代量化[26]（Iterative Quantization）ITQ与SQ量化不同，迭代量化ITQ的码字c必须选取自一个可以旋转的超立方体的顶点，可以使用hash技术来到达失真（distortion）最小化。在空间中的维向量可以看作是一个轴对称以0为中心点的维立方体的顶点。ITQ的目标函数定义如下：

 （2-3）



其中是正交矩阵，为单位矩阵。

使用超立方体作为码本的好处在于，码字之间的欧几里德距离的平方等于其汉明距离，所以ITQ也属于二进制哈希算法的范畴。值得注意的是，在式（2-3）中数值的选取不会影响到ITQ的散列过程。可以根据数据集向量元素的规模选定适合的值，使得量化失真最小。

### 2.3.3 乘积量化

乘积量化[6]（Product Quantization）PQ，为目前在ANN搜索中应用最多且改进方法众多的向量量化方法。乘积量化的基本思想为，将原始的高维向量空间划分为多个低位向量空间的笛卡尔积。具体表述如下：

对于维空间的数据点，可以将其拆分为个子向量，即。一般来说，的选取应为的因子，这样可以保证所有子向量的维度都是。分别对每个子向量进行码本（codebook）和码字（codeword）的计算。原始向量对应的码本则由其所有子码本的笛卡尔积表示，即。对于每个向量，可以计算出其个子向量的子码字（sub-codeword）和其对应的子码本（sub-codebook）：，。由此可得PQ的目标函数计算：

 （2-4）



由于原始向量被划分成了多个子向量，所以目标函数的计算也可以划分成多个子问题进行计算。乘积量化的有点不仅在于可以加快向量之间距离的计算速度，同时也可以生成可以包含大量码字的码本。假设每个子向量可以生成具有条码字的子码本，其原始向量被划分成了个子向量。原始向量的码本为个子码本的笛卡尔积，这也就意味着原始向量的码本具有高达的状态空间基数。这是采用其他量化方式很难达到的。PQ还可以采用离线的过程加速计算，对于子码本中的条码字，可以预先计算出一个的距离矩阵来表示条码字中任意两个码字的距离。在使用过程中可以直接使用查表（look up）的方式进行距离计算，进而大幅提高计算效率。

基于乘积量化也有众多改良版本和优化方法，OPQ[27]（Optimal Product Quantization）认为经过特征提取算法提取的特征向量在某些维度有着很大的相关性，将原始向量对应的子向量去除相关性可以使乘积量化的量化效果更好。

## 2.4 基于图的方法

基于图的最近邻检索在检索精度和查询效率上有着很好的权衡。对于在维欧式空间的数据点集合，图被定义为连接数据点（node）的边（edge）集合。通常表示节点和节点之间的邻居(neighbor)关系。图的构建过程一些常见的图结构设计例如：相对邻域图（RNG）、Delaunay图和K邻近图等均会对边的选取有严格的限制，进而使图的结构更适合ANN搜索问题。这种图通常被称为邻近图[28]（Proximity Graphs），本节将介绍邻近图的构建和其对应的搜索算法。

### 2.4.1 单调搜索网络

单调搜索网络Monotonic Search Networks（MSNET）[29]保证从图的任意点出发到目的点的所有路径中，一定存在一条路径（Path）使得访问的中间节点距离越来越近，该路径被称为Monotonic Path。其具体定义如下：

给定在维欧氏空间中具有个数据点的数据集，和为数据集中任意两点，则定义为数据集的节点（node）和之间的边（edge）组成的图。设为图中的一条路径（Path），这条路径当且仅当条件成立时，为Monotonic Path。

同理，Monotonic Search Networks（MSNET）的具体定义如下：

给定在维欧氏空间中具有个数据点的数据集，为数据集的节点（node）和之间的边（edge）组成的图。为Monotonic Search Network当且仅当对于任意节点，之间至少存在一条Monotonic Path。

MSNET可以保证任意两个节点之间的路径是单调连通的。从MSNET的定义来看MSNET一定是强连通图，且在MSNET图的单调特性可以保证搜索过程中不需要回溯[29]。构建完美的MSNET通常需要极高的时间复杂度，其中FANNG[30]采用启发式的方式构建了一个近似MSNET，FANNG通过多个初始点和搜索回溯来应对启发式构建导致的MSNET精度缺失，从而提高检索精度。NSG[10]提出了MRNG（Monotonic Relative Neighborhood Graph）即在RNG图的基础上进行了加边，使得增强后的RNG满足MSNET的性质。

### 2.4.2 基于Delaunay图的方法

Delaunay图（或 Delaunay 三角剖分）被定义为 Voronoi 图[31]的对偶图，且可以证明Delaunay图在结构上满足MNSET的单调性质[7]。然而，随着空间维数的增加Delaunay图逐渐变得稠密，由高维数据构建的Delaunay图很容易变成完全图。针对这一问题，NSW（Navigable Small World）[7]通过构建一个近似的Delaunay图，对数据集中数据点采用顺序插入图中的方式，每次将插入数据点和图中距离插入点最近的K个数据点相连。这种方式可以使较早插入的节点和其连边对较后插入的节点具有导航性质，从而构建一个具有导航性质的小世界网络。NSW一定程度上解决了单纯采用Delaunay图进行近似最近邻搜索所具有的图构建过程复杂度过大、搜索效率较低等问题。但是，NSW节点出边数和数据维度成正相关关系，较高维度的数据同样会导致NSW搜索和图构建效率降低。

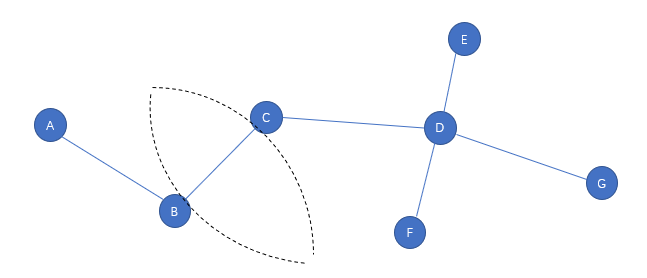


图 2-1 RNG图示例（无向图）

HNSW（Hierarchical Navigable Small World）[8]构建了一个多层次的NSW图，并且限制了每个节点的最大出边，层次结构的使用大大缓解了NSW在高维数据上表现不佳这一问题。同时HNSW所提出的基于候选集回溯的贪婪搜索方法也被NSG、SSG等基于图的ANN搜索算法所使用，其性能优于FANNG和EFANNA[20]等贪婪搜索算法，为目前的state-of-the-art。在选边策略上，HNSW认为NSW的选边策略可以进行优化，HNSW和FANNG使用了同样的方式对节点的出边数量进行裁剪，该选边策略和RNG图的构建方式类似。即，如果则。图2-1为RNG图的一个示例，RNG的选边策略在节点B上的具体表现为B和C之间的连边构建的虚线对应区域[10]（lune）内不能有节点连边存在。从选边策略这个方面进行分析，HNSW是对Delaunay图和RNG的近似。

### 2.4.3 基于KNN图的方法

KNN（K-Nearest Neighbor）图也称K邻近图，即每个数据点和其在度量空间中距离最近的K个数据点连一条有向边（从原数据点到各个近邻数据点）。与Delaunay图不同的是KNN图为有向图，即通过单向边相连的节点不一定互为K近邻。这也说明了其不能保证为强连通图，甚至在K较小的情况下整个图都有可能划分为独立的几个连通块。如图2-2所示，从节点D出发不能访问到节点C。

构建精确的KNN图不可避免的需要遍历整个数据集。在数据点都是高维数据的情况下，构建准确的KNN图需要极大的计算代价。一种可行方案如EFANNA[20]是采用多颗随机划分KD-tree对数据进行预处理，在多颗KD-tree上进行搜索进而保证KD-tree的精度。同时，EFANNA还应用了NN-descent的策略[32]，即邻居的邻居更不太可能是邻居这一策略，对KNN图进行优化。

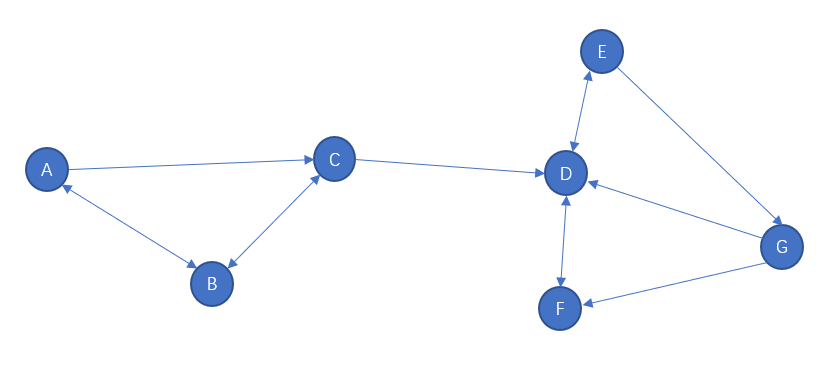


图 2-2 KNN图示例（有向图）

KNN图在ANN检索中有着明显的优势，其数据点和最近邻相连的特性十分贴合ANN搜索的需要。但由于其不是强连通图的特性导致其不能不能成为MSNET。同样由于其不强连通的特性，从单个出发节点进行搜索可能导致不能访问到目标节点。为解决这一问题，NSG在构建的KNN图的基础上采用DFS进行连通性增强，进而可以保证从给定初始节点可以访问到图中的所有节点。另一种可行方式为，通过随机多个初始节点进行搜索来增加访问到目标节点的概率，但其相对于单出发节点加图连通性增强的方式而言，并没有太大的效率和召回率提升。

# 图索引相关技术研究

## 3.1 图搜索算法

目前已知state-of-the-art的搜索算法[8]为如下**算法1**，同时也时HNSW和NSG图索引的搜索算法。该搜索算法在陷入局部最优解的情况下可以从候选集（candidate pool size）中选取未被访问过的新节点回溯继续进行搜索。对于查询数据点为数据集中给定点的情况下，多数网络搜索结构如HNSW、NSG和SSG[33]等满足MSNET性质的图甚至不需要保证候选集规模。因为MSNET的特性保证每个节点必定均在Monotonic Search Path中，即在搜索的过程中不需要回溯。但对于查询数据点不在数据集中的情况下，SSG给出了使用类似RNG选边策略的HNSW、NSG和SSG等结构在进行贪心搜索时，只有50%的概率保证搜索到下个节点在Monotonic Search Path上的理论证明。因此，使用候选集进行搜索回溯，对提升搜索召回率很有必要。

|  |  |
| --- | --- |
| **算法1**：图搜索算法 |  |
| 1：function Search-on-Graph(G, p, q,)  输入：  G邻近图  **p**出发节点  **q**查询节点  候选集大小  输出：  **q**的k近邻 |  |
| 2：, 候选集 |  |
| 3：将**p**加入候选集 |  |
| 4：**while**  **do** |  |
| 5： 中第一个未被标记的节点编号 |  |
| 6： 标记**pi**被访问 |  |
| 7： **for** **all** neighbor **n** of **pi** in G **do** |
| 8： 将**n**加入候选集 |  |
| 9： **end for** |  |
| 10： 将按到**q**的距离升序排序 |  |
| 11： **If**  **do**  12：  13： **end If** |  |
| 14：**end while** |  |
| 15：**return** 中的前节点 |  |

从算法1可以看出，检索效率和召回率均和候选集参数有关，该参数对应候选集的大小。参数如果过大会导致检索效率下降，过小则会导致召回率下降。同时，参数应大于检索的近邻数以避免检索结果精度的大幅下滑。

## 3.2 图构建算法

由**算法1**可知，算法的检索效率和节点的出边数量（出度）关联性很高。对于搜索过程中的每个点，其在访问下一个节点的过程中不可避免的要扫描该节点的所有出边，即所有邻居节点（neighbor）。过于如果图中节点出边数量过于稀疏则可能导致从初始节点到目标节点的搜索路径过长，进而使检索效率下降。图中节点出边数过多则会造成图过于稠密，在检索过程中每次移动下一节点的检索代价变大，从而也会导致检索效率降低。

为解决这一问题，HNSW使用了多层次结构和超参数来限制节点的最大出边数量。具体方法为，对数据点每层进行采样使每层数据点数量相对上一层数据点数量呈指数递增，最后的层数约为5-6层。其在搜索和构建过程中的精度通过参数来控制，构建过程中的参数为，该参数控制每个新插入节点的近邻候选集大小，在当前图中进行近邻搜索。同算法1进行最近邻检索相似，较大的参数可以构建一个更加精确的HNSW图，同时也意味着索引构建时间更慢。HNSW的搜索从分层图的顶层开始，每层的搜索算法如算法1所示。每层的搜索结果作为下层搜索的起使节点。搜索参数为，该参数和算法1的参数意义相同。HNSW在除底层外的数据层搜索参数通常设置为1，其目的在于找到下一层的出发节点，在底层搜索的参数设置通常和数据集规模和查询的近邻数，其大小设置一般要大于K近邻检索参数。

与HNSW算法不同的是，NSG需要在KNN图的基础上进行连通性增强和删除冗余出边。NSG图的出边数量可以通过控制KNN图的邻近数量来控制图出边数量。为解决KNN图很有可能不是强连通图这一问题（从出发节点不一定可以访问到所有节点）。NSG算法使用固定初始节点的方法，将距离所有数据质心最近的数据点作为出发节点，从出发节点进行DFS（深度有限搜索）。对于没有访问到节点将其和DFS树中最近的数据点相连，进而保证从出发节点可以访问到所有数据节点。

## 3.3 图优化算法

HNSW算法在插入过程中首先将数据集中所有数据点按上述方法插入图中。当前图的节点出边数量达到为时，如果由新的数据节点插入可能导致原来的节点出边数量超过限制。HNSW针对这一问题提出了两种解决方法，第一种方法为朴素的依靠距离限制节点出边的方法，其主体思想为使用节点的距离来限制每个节点的出边数量。该方法将数量超过限制的节点的所有邻居按照距离从大到小排序，取前k近的邻居节点连边其他邻居节点所连边直接删去。

另外一种方法，在兼顾距离的情况下更注重节点之间的散度（也就是RNG的选边策略），同时也被应用到NSG算法中。其方法为，对于节点的所有邻居和其邻居的邻居作为节点优化选取邻居的候选集中（过程中去除重复节点）。对于候选集中距离最近的节点，如果节点距离的距离小于的优化后邻居集合（初始为空）中的任意节点到的距离，则将节点加入中。如果中节点数量小于节点出边数量限制，则将候选集中未被加入的元素按照其到节点的距离排序，依次加入中直到中元素个数到达上限。最后节点的邻居为中所有元素，优化后的邻居相对朴素的第一种方法有着更高的节点散度。同时，第二种优化方法贴合MSNET的性质，即节点被优化后的邻居节点可以通过现有的邻居节点进行单调搜索访问到。

从构图角度来说，HNSW算法和NSG算法都是对RNG图的近似。但是HNSW算法同时也是Delaunay图的近似（可以保证MSNET性质），NSG在KNN的基础上不能保证单调搜索性质。为此，NSG对RNG图进行了增强，使其满足MSNET的要求。具体实现方式为判断两个节点之间的连边构成的lune区域是否仅含有一个节点的连边（连向判断节点），如果该区域有且仅有一个数据节点则不进行连接性增强。如果该区域没有数据点的连边，则从所有数据点中找出一个在该区域内的数据点进行连边，如果所有数据点均不在该区域内则不需要连接多余的边。

## 3.4 性能分析

### 3.4.1 时间复杂度分析

HNSW算法的时间复杂度分为索引构建时间复杂度和搜索时间复杂度，其中搜索时间复杂度如下：

HNSW各层节点数呈指数递增，整体搜索时间复杂度同在底层图进行搜索的时间复杂度相同，为。该时间复杂度的得出为通过对比实验[8]近似得出，其实际受搜索参数影响。NSG图索引的搜索算法和HNSW算法相同均为**算法1**，但NSG给出了其在SIFT1M数据集上搜索的具体时间复杂度为，以及NSG提出的理想模型MRNG搜索的具体时间复杂度为。，其中为MRNG节点平均出度，且在实验过程中可以看作是一个常数，不影响复杂度计算。

同理，HNSW的索引构建时间复杂度可以如下得出：

HNSW算法的索引构建是通过将数据集中所有元素进行迭代插入完成的，元素插入过程为在不同层的进行的启发式搜索（时间复杂度和参数相关）。相对于数据集中数据条目，HNSW的单个节点插入时间复杂度和搜索时间复杂度相同为，整体构建时间复杂度为。

NSG算法的索引构建时间复杂度分为两个阶段，阶段一为快速构建KNN图。使用EFANNA[20]或NN-decent可以在较小时间代价下构建出一个精确的KNN图。NSG算法使用EFANNA通过多个随机划分KD-tree快速检索数据集中所有节点的K近邻进而构建KNN图，对于KD-tree构建的原始KNN图进行NN-decent优化。总体时间复杂度受随机划分KD-tree数量、检索深度参数和NN-decent迭代轮数有关。NSG在KNN图的基础上进行了连通性增强和节点出边选择优化，总体时间复杂度为，在使用NN-decent[10]的情况下。

### 3.4.1 内存使用

HNSW和NSG算法主要的内存消耗为图索引所占空间和原始数据所占空间。由于两者均为内存依赖的方法，在检索前需要将整个数据集载入内存中，以便于距离计算和ANN检索。对于图索引所占空间主要和每个节点的连边数有关，设节点的平均出边为，图索引所占空间为。原始数据所占空间由数据条目数和数据维度两个参数所支配，其所占空间为（32位浮点数为4byte）。

HNSW和NSG算法在原始数据占用空间这一方面是相同的，两者均未使用量化方法或倒排索引方法进行空间压缩。对于图索引所占空间而言，NSG在和HNSW算法具有相同检索效率的情况下，有着较小的平均出边数量。具体来说，NSG图索引在空间使用上只占HNSW算法的二分之一到三分之一，在索引空间使用上NSG算法更有优势。

# 量化优化方法

上文分析了，HNSW算法的索引空间使用问题，本节将分析使用不同的量化方法对索引大小的优化程度，以及量化失真对搜索召回率的影响。量化方法和图索引结合有两种方式，其一为对原始数据进行图索引构建，图索引由原始数据的邻近关系构成，在查询阶段载入量化数据以达到整体索引空间的压缩。其二，为首先对原始数据进行量化压缩，在量化数据的基础上进行图索引构建。第一种方式相对方法二在图索引构建过程中需要较大得到内存空间（通常需要在主机组或Spark集群上进行模型构建），但由于其使用原始数据进行图索引构建，其在索引构建精度方面相对第二种方法有着较大优势。



图 3‑1 量化优化流程图

使用量化方法优化索引空间的流程如图3-1所示。量化训练过程中一般会在空间需求远小于原始数据规模的训练集上进行训练，如SIFT1M数据集提供的sift\_learn.ivecs文件。对于没有提供训练集的数据集GLOVE1M，本文使用原始数据中随机采样的方法进行乘积量化训练和码本的生成。使用训练集和采样方法的主要目的为减少乘积量化训练和码本生成时间。如果对原始数据集进行乘积量化训练，子空间划分和聚类会导致训练时间过长等问题，使用训练集和采样方法可以在几乎不损失精度的情况下快速构建乘积量化码本。

## 4.1 标量量化优化

标量量化SQ的优化目标为压缩索引构建过程中的原始向量数据，以实验所选用SIFT1M数据为例。SIFT1M原始数据为一百万条128维向量，向量中的每个元素用32位浮点数表示，表述数据所需byte数为。SQ将32位浮点数压缩到8位的整数，表示数据所需byte数为，在压缩后的数据集上进行图索引的构建，具体如图3-2所示。



图 3‑2 标量量化过程

该方法可以将原始数据压缩到原来的四分之一。SQ的量化损失由整形的数据压缩导致，其量化效果受数据分布影响过大。在数据存在离群点的情况下，可能会导致量化损失提升。将数据集中离群点分离原始数据并将其量化值设置为上限，可以有效解决这一问题。



图 3-3 乘积量化过程

## 4.2 乘积量化优化

乘积量化PQ同标量量化SQ的优化目标一样，都是对原始数据进行量化压缩。乘积量化将原始向量划分为不相交的子向量，以实验数据SIFT1M为例。乘积量化可以将原始的128维向量划分为32个维度为4的子向量，每个子向量聚256个簇。对每个聚类质心进行编号，将子向量用其聚类质心的编号表示。原始的128维向量可以用32个质心及其编号表示。由于每个子向量聚256个簇，簇编号范围可以使用8位整形表示，具体实验方式如图3-3所示。

乘积量化和标量量化相同，即和图索引结合有两种方式。第一种方式为在原始数据集上进行图索引构建，然后使用乘积量化方式对原始数据集进行量化编码和码本的生成。该方法的图索引连边关系表示的是原始数据的近邻关系。第二种方式为优先对原始数据进行量化压缩，在量化后数据集进行图索引的构建，该方法的图索引近邻关系由量化数据的近邻关系表示。第二种方式相对第一种方式在图临近关系的表示上可能存在有量化失真导致的偏差，但方式二可以使用数据分片等方式避免图索引构建阶段的巨大空间需求（对计算硬件如主机组、Spark集群的需求）。

## 4.3 乘积量化数据距离计算

乘积量化PQ数据之间的距离计算有对称距离计算SDC（Symmetric Distance Computation）[6]、非对称距离计算ADC（Asymmetric Distance Computation）[6]两种方式。其中对称距离计算方式将查询数据进行量化编码，对于距离计算则由量化编码之间对应的质心（centroid）之间的距离之和表示。具体表示如下：

 （4-1）

上式中向量和向量之间距离均由其子向量对应的质心（centroid）和表示，向量之间的距离也由其质心距离近似表示。

非对称距离计算方式为不对查询数据进行量化编码，将查询向量按乘积量化的划分方式进行划分，查询子向量到量化数据的距离则为查询子向量和对应量化数据质心（centroid）之间的距离，对于查询向量到量化数据的距离，则为其所有子向量到质心的距离相加。具体表示如下：

 （4-2）

上式中代表编号为划分子向量。

## 4.4 图索引量化距离计算

对于标量量化方法而言，标量量化后数据之间距离即为其数值欧几里德距离，对于查询数据无需做标量量化。对于乘积量化数据，查询数据可以使用对称距离计算和非对称距离计算两种计算方式。从以往数据来看[6]，非对称距离计算ADC相对于对称距离计算SDC有着更高的精度，因此本文中实验所使用距离计算方式为非对称距离计算。

在码本中码字数量较少的情况下，单个查询数据的子向量很有可能重复计算到其中的部分码字的距离。一种可行方法为在查询前计算查询数据的所有子向量到其对应的码字距离，并存入距离表中。在查询过程中可以使用直接查表（look up）的方式避免重复计算。另一种方法为在计算过程种渐进的构建距离查询表，对于每次距离计算优先访问查询表中是否有记录，如果没有记录则重新计算距离并将结果载入查询表中，反之则直接从插叙表中读取距离。具体方法如算法2所示：

|  |  |
| --- | --- |
| **算法2**：非对称渐进距离计算算法 |  |
| 1：function Progress-ADC-Calculate (T, p,q)  输入：  T 渐进距离查询表（初始为空）  p量化向量  q查询向量  输出：  查询向量到量化向量距离 |  |
| 2： |  |
| 3：**for** **all** sub vector in q **do** |  |
| 4： //对应划分向量的centroid |  |
| 5： **If**  **do** |  |
| 6： |  |
| 7： **else do** |
| 8： |  |
| 9： //保存历史结果 |  |
| 10： **end If** |  |
| 11：**end for** |  |
| 12：**return** |  |

上述算法中查询表的最大可能大小和乘积量化向量划分数以及每个子空间质心数量（编码范围）参数有关。由于查询向量可能和所有子空间的所有质心进行距离计算，查询表内含有最大元素数量为。

# 实验结果与分析

## 数据集与实验配置

本实验机器配置为Core i7-7700CPU，16G内存，查询测试均为单线程测试。实验数据集为图像检索数据集SIFT1M以及文本数据特征GLOVE1M数据集。SIFT1M数据集由1000000条128维向量构成，使用32位浮点数存储空间。GLOVE1M ，为1183514条100维向量构成，同样使用32位浮点数存储。本文HNSW算法为baseline的性能测试、量化优化等方式均使用开源框架faiss[23]实现，该开源框架提供HNSW、SQ和PQ等算法的python接口，底层使用C++实现。以NSG算法为baseline的量化优化、多种量化数据距离查询的实现方式均为C++实现，其中使用EFANNA进行KNN图的构建，在此基础上进行连通性增强和选边策略优化。

HNSW算法将训练数据和数据集中数据从原始的.ivecs和.hdfs5编码格式读取并转化为python 数据处理numpy矩阵。为消除数据格式对内存使用测量的影响，原始数据均以numpy的二进制文件编码格式.npy保存，其数据大小相对于所有向量理论占用byte数有些许误差（numpy会保存矩阵shape等参数）。NSG算法及其相关优化则将原始数据（.ivecs文件）作为二进制文件读入，量化数据和图索引数据均以二进制文件的形式保存。

具体实验内容为，在SIFT1M和GLOVE1M两个数据集上进行HNSW索引的构建并测试其性能。在此基础上，将原始数据进行多种量化方法的压缩，在量化数据的基础上进行HNSW索引的构建（方法二）。乘积量化方法受不同数据维度数的限制（子向量维数需为原始数据维数的因子），具体划分为，对SIFT1M使用32个4维向量的乘积划分方式HNSW\_PQ32，以及64个2维向量的划分方式HNSW\_PQ64和HNSW\_OPQ64。对于GLOVE1M数据集使用20个5维向量的划分方式HNSW\_PQ20以及50个2维向量的划分方式HNSW\_PQ50和HNSW\_OPQ50分别进行实验。对于索引的占用空间、检索性能进行不同规模的实验并分析其性能和理论代价。

NSG算法索引构建使用第二种构建方式，具体为对SIFT1Mhe数据集进行NSG图索引的构建，之后使用乘积量化和标量量化两种方式对原始数据进行压缩。具体划分方式为NSG\_PQ64、NSG\_PQ32和NSG\_PQ128，其中NSG\_PQ128可以看作是标量量化方式。NSG乘积量化的流程为，将128维的SIFT1M的训练数据集sift\_learn.ivecs（10000条向量）进行子空间划分，使用k-means对训练数据进行聚类（k=256），将聚类的256个簇的质心（centroid）作为PQ的子空间码本。之后，读取SIFT1M原始数据集sift\_base.ivecs将所有向量按照PQ的划分方式划分为子向量，对于每个子向量使用距离其最近的质心编号进行编码。

## 实验召回率结果

实验召回率定义如下：

 （4-1）

式（4-1）中，为搜索返回结果，为groundtruth数据。SIFT1M和GLOVE1M数据集提供了groundtruth数据，为每个向量提供了经过验证的近邻数据，对于数据集中每个向量groundtruth数据集包含了距离最近的100条向量（根据距离从小到大排序）。

HNSW索引及其量化优化方式在SIFT1M数据集测试结果如图4-1所示，分别在的参数控制下进行检索召回率和查询性能测试。

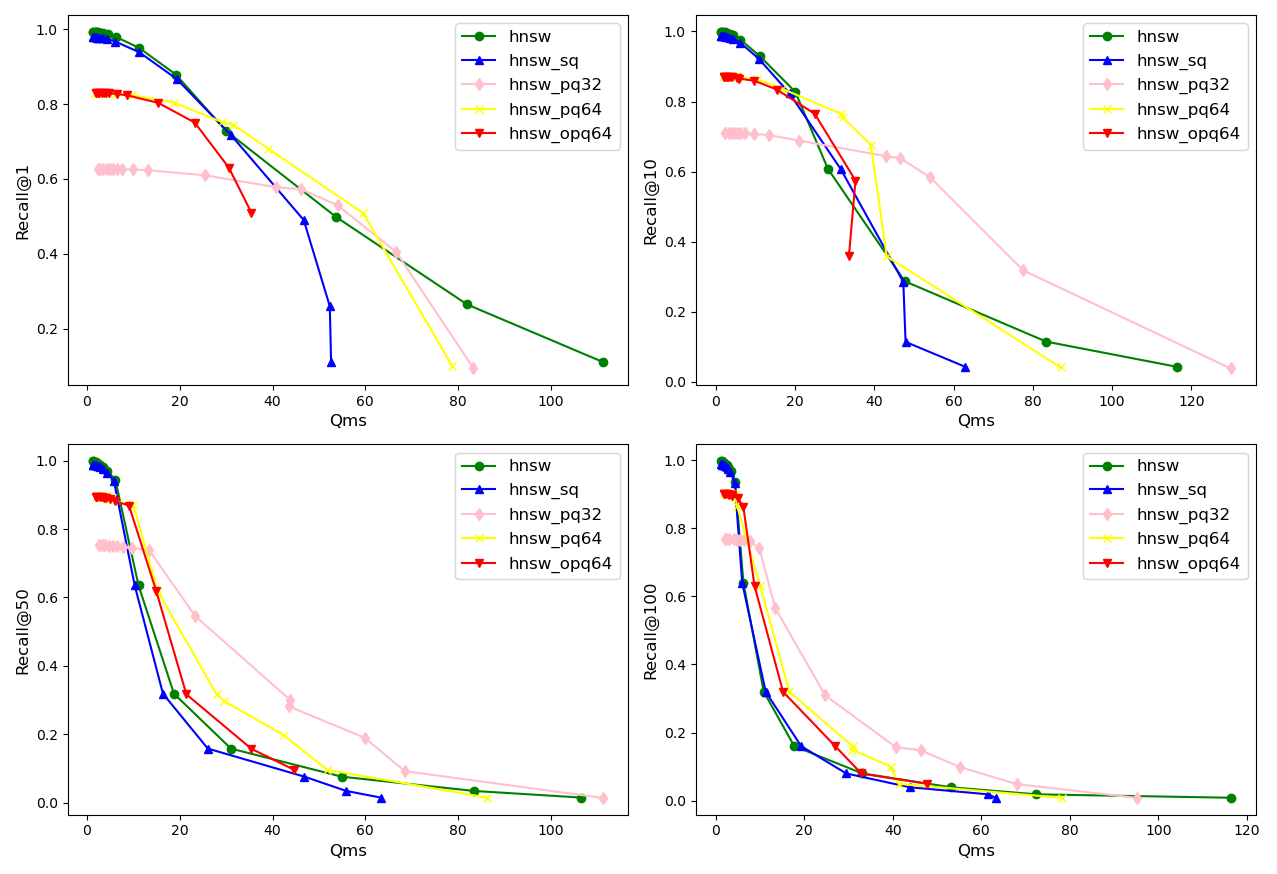


图 4‑1 SIFT1M数据测试结果（HNSW）

其中代表的检索结果，同理代表的检索结果。图4-1和图4-2中横坐标Qms为平均每毫秒处理查询请求数代表索引查询效率，纵坐标代表索引查询召回率，反映了索引查询的精度。

HNSW算法在GLOVE1M数据集测试结果如图4-2所示，分别在的参数控制下进行检索召回率和查询性能测试。

算法方法使用为原始HNSW算法、HNSW\_SQ（结合标量量化方法）、HNSW\_PQ（结合乘积量化的方法）等。其中HNSW\_PQ32和HNSW\_PQ64分别为乘积量化的不同划分方式，OPQ为优化原始向量之间相关性的乘积量化方法。

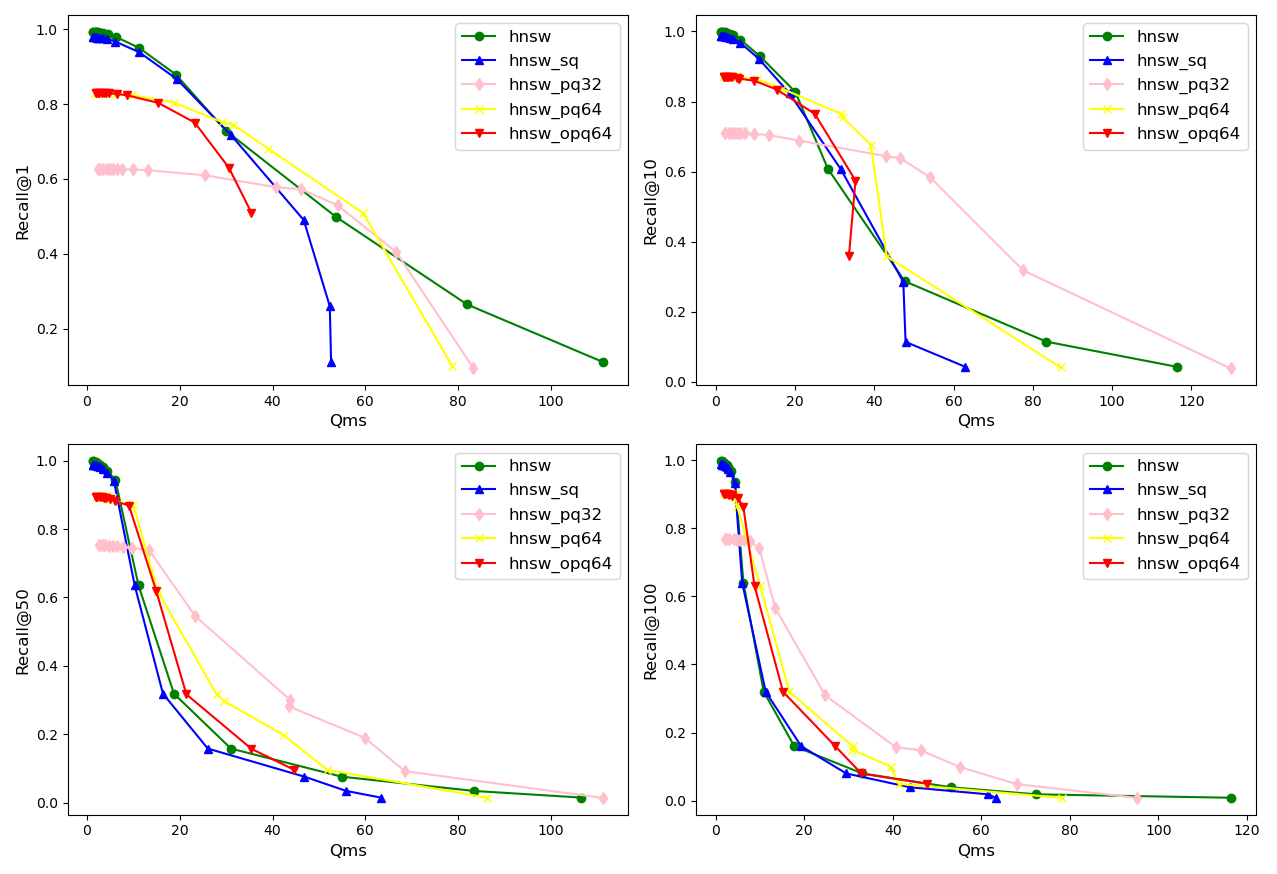


图 4‑2 GLOVE数据集测试结果（HNSW）

NSG算法及其量化优化方式如图4-3所示，其中NSG\_PQ128即子向量长度为1的划分方式，该种划分方式在距离计算和码字生成上等同于标量量化。

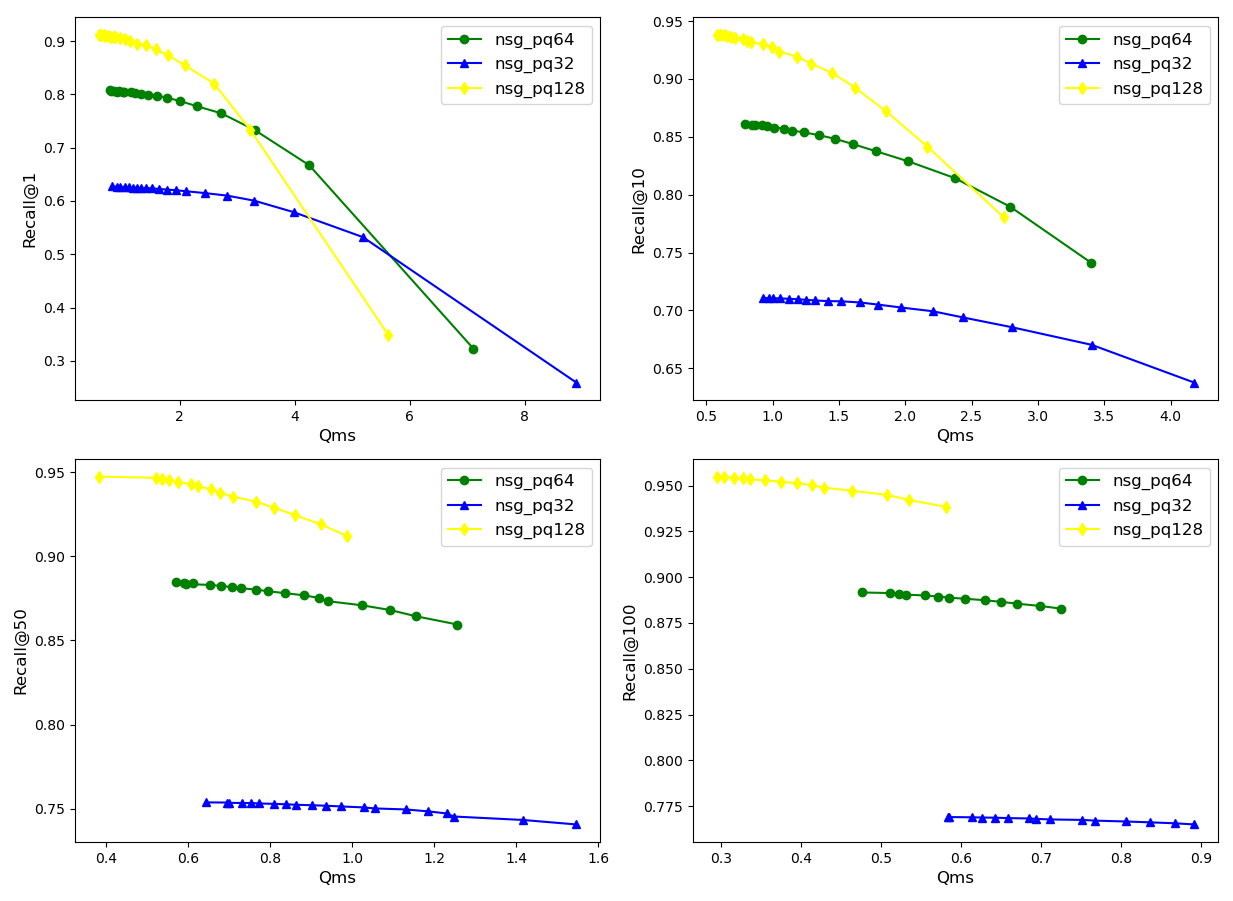


图 4-3 SIFT1M数据集测试结果（NSG）

## 索引构建空间结果

本文对HNSW算法、NSG算法、NSG结合乘积量化方法、HNSW结合标量量化算法、乘积量化算法以及优化维度相关性的OPQ量化算法进行了索引空间使用的分析。由于不同量化方式对原始数据集的编码方式不同导致的各个节点之间距离存在差异，最后导致所建立的图索引结构有所不同（所占空间也会略有差异）。

其中HNSW算法在SIFT1M数据集的空间使用状况表4-1所示。SIFT1M数据集由1000000条128维向量组成，向量采用32位浮点数表示。对于表中各个方法，HNSW为原始图索引构建方法，其中500MB为SIFT1M原始数据，在使用过程中载入内存由于距离计算。总使用空间为765MB，总使用空间出原始数据之外还包括图索引的空间使用（主要为节点之间连边）。HNSW\_SQ为标量量化优化方法，HNSW\_PQ32为将128维向量划分维32个4维向量的笛卡尔积的乘积量化方法，对于OPQ32和OPQ64分别对应32个4维向量和64个2维向量的划分方式。

表4-1 SIFT1M数据集各方法对应的内存空间使用（HNSW）

|  |  |  |
| --- | --- | --- |
| **实验方法** | **数据使用空间** | **总使用空间** |
| HNSW | 500MB | 765MB |
| HNSW\_SQ | 125MB | 390MB |
| HNSW\_PQ32 | 32MB | 297MB |
| HNSW\_PQ64 | 63MB | 328MB |
| HNSW\_OPQ32 | 32MB | 297MB |
| HNSW\_OPQ64 | 63MB | 328MB |

HSNW算法在GlOVE1M数据集对应的划分方法如表4-2所示，GLOVE1M数据集由1183514条100维向量组成，向量元素使用32位浮点数表示。表中方法HNSW为原始索引构建方法，HNSW\_SQ为标量量化方法，HNSW\_PQ20对应将100维数据划分为20个5维数据的乘积量化方法。

表4-2 GlOVE1M数据集各方法对应的内存空间使用（HNSW）

|  |  |  |
| --- | --- | --- |
| **实验方法** | **数据使用空间** | **总使用空间** |
| HNSW | 462MB | 776MB |
| HNSW\_SQ | 116MB | 430MB |
| HNSW\_PQ20 | 23MB | 337MB |
| HNSW\_PQ50 | 58MB | 372MB |
| HNSW\_OPQ50 | 58MB | 372MB |

NSG算法在SIFT1M数据集对应的划分方式和空间使用如表4-3所示，其中

NSG\_PQ32等为量化编码后的SIFT1M数据。其中NSG\_PQ32为使用32个码字对128维向量进行划分编码的方式。

表4-3 SIFT1M数据集内存空间使用（NSG）

|  |  |
| --- | --- |
| **数据内容** | **数据使用空间** |
| NSG邻近图 | 118MB |
| SIFT1M | 503MB |
| NSG\_PQ32 | 31MB |
| NSG\_PQ64 | 62MB |
| NSG\_PQ128 | 125MB |

## 距离计算优化结果

距离计算优化的实现方式为在NSG算法和PQ结合的基础上进行实验分析，具体内容为在不同规模的码本大小和检索召回率要求的情况下分析直接使用乘积量化非对称距离计算、使用渐进方法构建查询表进行距离计算以及预先构建完整查询表的距离计算三种方式。其中码本大小对三种距离计算方式影响较大，而质心数量（码字数量）受子空间划分数量和子空间质心数量影响。三种距离方式只在计算效率上有差异，在计算精度上则完全相同。本文以NSG\_PQ64的量化优化方式固定子空间划分数（为64个2维向量），在此基础上进行每个子空间256个质心编码、4096质心编码两种不同规模的编码方式进行距离计算优化相关效率实验。

## 5.5 结果分析

由图4-1可以看出，使用标量量化技术可以达到和使用原始数据几乎相同的召回率和精度。其原因在于SIFT1M数据虽然使用32位浮点数表示，其实际数值远小于32位浮点数可表示范围，因此是用标量量化将原始数据压缩为8位的整数表示没有太大的量化失真，进而可以保持相对较高的召回率。对于乘积量化方法而言，其检索召回率和划分的子空间数量这一参数较为相关。由图4-1和图4-2均显示，较多的子空间划分数量可以明显提高ANN检索召回率。由空间使用数据可得，较小地子空间划分数量意味着更小地空间使用，同时也意味着更大的量化失真导致的检索效率下降。

以图4-1和表4-1为例，HNSW\_SQ将1000000条128维32位浮点数据（原始数据）压缩到8位的整形数据，空间理论压缩为原来的1/4。HNSW\_PQ32将原始数据划分为32个子向量的编码，每个编码只使用8bit整形表示，使用该方法可以将原始数据压缩到原来的1/16（不考虑码本大小）。过度的压缩导致量化失真较大，直接的结果反应为召回率的大幅下降。使用维度相关性优化的量化算法OPQ进行索引优化的方法相对直接使用PQ算法而言性能没有明显提升，其原因在于OPQ对维度进行了重新排列使得量化损失降低，但对于输入向量的维度重新排列也会对性能造成一定影响，使得整体性能没有明显提升。

不同子空间划分数量的乘积量化方式对整体实验召回率有较大影响，其原因在于不同的划分方式的量化平均损失不同。子空间数量划分越多、编码使用的簇的质心数量越大可以构成更大的编码空间，同时也可以降低量化失真提升检索召回率。以NSG算法和乘积量化结合的方式在SIFT1M数据集上的应用为例，128个子空间划分方式NSG\_PQ128的平均量化失真（平方）为92.733，而NSG\_PQ64的平均量化失真为670.847，NSG\_PQ32的平均量化失真为3929.065。较多的子空间划分虽然可以提高检索精度，但也意味着较大的内存需要，在实际应用过程中可以按检索精度需求进行方式选取。

对于量化优化后的索引，其在较大的检索中有着相对较好的召回率，但在较小如近似最近邻（1-NN）的情况下召回率不高。其原因在于量化失真使得图索引搜索的近似最近邻和原始数据最近邻有着一定的数值差距，该数值差距在经过量化训练后对于每个节点而言在一个固定的范围。检索时的近邻平均距离随着的增大也随之变大，因此量化失真导致的数据差距对ANN检索的影响也会随之减弱。所以在的情况下HNSW\_PQ64和HNSW\_OPQ64的召回率达到了90%，可以满足一定实际应用的场景如推荐系统、视频去重和文本相似度查询等实际业务要求的需要。

# 总结

本文首先分析了哈希、倒排索引、树形结构、量化和图等高维数据近似最近邻检索技术的使用场景和其性能。对性能相对较好、应用场景较为广泛的量化和图技术和相关算法进行了理论分析。对于量化技术、主要分析了标量量化、乘积量化和迭代量化三种量化技术的构建算法和使用方式。对于图索引技术，重点研究了邻近图（Proximity Graph）构建的相关技术背景，如单调搜索网络MSNET的基本概念以及其在邻近图中的应用，以及MSNET概念的具体实现方式如Delaunay图、MRNG等图结构。构建完美的MSNET需要极高的时间复杂度，本文重点研究了Delaunay图（一种MSNET）的近似实现方式HNSW算法以及MRNG图（同样为MSNET）的一种近似实现方式NSG算法的索引构建算法和搜索算法，两者在ANN-benchmark的多个数据集性能排名第一（或具有相同性能），为目前的state-of-the-art且HNSW在插入节点和并行等方面有良好的适配性，而NSG在邻近图的内存空间使用上较HNSW算法稍有优势。

实际场景使用HNSW和NSG等图索引技术仍会受索引空间消耗过大这一严重问题影响。使用量化技术可以成倍压缩原始数据大小，在损失部分检索精度的条件下可以构建一个内存使用较低的图索引。本文应用量化技术优化图索引进行高维数据最近邻检索，分析了使用标量量化SQ、乘积量化PQ以及优化相关性的乘积量化方法OPQ和HNSW、NSG算法结合的实现方式，并进行了索引构建空间和检索召回率两个维度的实验分析。得出了使用量化技术优化图索引进行高维数据最近邻检索可以在部分精度损失的trade-off下，大幅优化索引使用空间。该方法在内存驱动的高维数据近似最近邻检索的实际场景如视频、文字推荐系统等应用中，有着减少内存空间使用的意义。

# 参考文献

[1] Slaney M, Casey M. Locality-sensitive hashing for finding nearest neighbors[J]. IEEE Signal processing magazine, 2008, 25(2): 128.

[2] Silpa-Anan C, Hartley R. Optimised KD-trees for fast image descriptor matching[C]. IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008: 1-82008: 1-8.

[3] Ram P, Sinha K. Revisiting kd-tree for nearest neighbor search[C]. Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining, 2019: 1378-1388.

[4] Jo J, Seo J, Fekete J-D. A progressive kd tree for approximate k-nearest neighbors[C]. 2017 IEEE Workshop on Data Systems for Interactive Analysis (DSIA), IEEE, 2017: 1-52017: 1-5.

[5] Wang X. A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality[C]. The 2011 International Joint Conference on Neural Networks, IEEE, 2011: 1293-12992011: 1293-1299.

[6] Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search[J]. IEEE transactions on pattern analysis and machine intelligence, 2010, 33(1): 117-128.

[7] Malkov Y, Ponomarenko A, Logvinov A, et al. Approximate nearest neighbor algorithm based on navigable small world graphs[J]. Information Systems, 2014, 45: 61-68.

[8] Malkov Y A, Yashunin D A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs[J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 42(4): 824-836.

[9] Toussaint G T. The relative neighbourhood graph of a finite planar set[J]. Pattern recognition, 1980, 12(4): 261-268.

[10] Fu C, Xiang C, Wang C, et al. Fast approximate nearest neighbor search with the navigating spreading-out graph[J]. arXiv preprint arXiv:1707.00143, 2017.

[11] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110.

[12] Oliva A, Torralba A. Modeling the shape of the scene: A holistic representation of the spatial envelope[J]. International journal of computer vision, 2001, 42(3): 145-175.

[13] Norouzi M, Fleet D J, Salakhutdinov R R. Hamming distance metric learning[J]. Advances in neural information processing systems, 2012, 25.

[14] Babenko A, Lempitsky V. Efficient indexing of billion-scale datasets of deep descriptors[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 2055-2063.

[15] Ke G, Meng Q, Finley T, et al. Lightgbm: A highly efficient gradient boosting decision tree[J]. Advances in neural information processing systems, 2017, 30.

[16] Aumüller M, Bernhardsson E, Faithfull A. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms[J]. Information Systems, 2020, 87: 101374.

[17] Zhang J, Ma R, Song T, et al. Hierarchical Satellite System Graph for Approximate Nearest Neighbor Search on Big Data[J]. ACM/IMS Transactions on Data Science (TDS), 2022, 2(4): 1-15.

[18] Baranchuk D, Babenko A, Malkov Y. Revisiting the inverted indices for billion-scale approximate nearest neighbors[C]. 2018: 202-216.

[19] Navarro G. Searching in metric spaces by spatial approximation[J]. The VLDB Journal, 2002, 11(1): 28-46.

[20] Fu C, Cai D. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph[J]. arXiv preprint arXiv:1609.07228, 2016.

[21] Gray R. Vector quantization[J]. IEEE Assp Magazine, 1984, 1(2): 4-29.

[22] Gersho A, Gray R M. Scalar Quantization II: Optimality and Design[M]. Vector Quantization and Signal Compression. Springer, 1992: 173-202.

[23] Johnson J, Douze M, Jégou H. Billion-scale similarity search with gpus[J]. IEEE Transactions on Big Data, 2019, 7(3): 535-547.

[24] Zhou W, Lu Y, Li H, et al. Scalar quantization for large scale image search[C]. 2012: 169-178.

[25] Qin J, Wang Y, Xiao C, et al. GPH: Similarity search in hamming space[C]. 2018 IEEE 34th International Conference on Data Engineering (ICDE), IEEE, 2018: 29-402018: 29-40.

[26] Gong Y, Lazebnik S, Gordo A, et al. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval[J]. IEEE transactions on pattern analysis and machine intelligence, 2012, 35(12): 2916-2929.

[27] Ge T, He K, Ke Q, et al. Optimized product quantization[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 36(4): 744-755.

[28] Jaromczyk J W, Toussaint G T. Relative neighborhood graphs and their relatives[J]. Proceedings of the IEEE, 1992, 80(9): 1502-1517.

[29] Dearholt D W, Gonzales N, Kurup G. Monotonic search networks for computer vision databases[C]. Twenty-Second Asilomar Conference on Signals, Systems and Computers, IEEE, 1988: 548-5531988, 2: 548-553.

[30] Harwood B, Drummond T. Fanng: Fast approximate nearest neighbour graphs[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 5713-5722.

[31] Aurenhammer F. Voronoi diagrams—a survey of a fundamental geometric data structure[J]. ACM Computing Surveys (CSUR), 1991, 23(3): 345-405.

[32] Dong W, Moses C, Li K. Efficient k-nearest neighbor graph construction for generic similarity measures[C]. Proceedings of the 20th international conference on World wide web, 2011: 577-586.

[33] Fu C, Wang C, Cai D. High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

# 附 录

# 致 谢

致谢应以简短的文字对课题研究与论文撰写过程中曾直接给予帮助的人员（例如指导教师）表示谢意。“致谢”两字之间空一个全角空格或两个半角空格，采用不编号章标题样式；内容应采用正文样式。