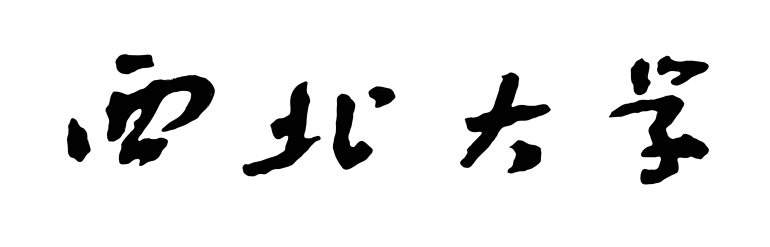
**** 

**2024-2025学年第二学期**

**硕士研究生课程作业**

|  |  |
| --- | --- |
| **题 目：** | 基于改进的B+树的图像特征检索系统 |
|  |  |

|  |  |
| --- | --- |
| 课程名称 | 空间数据结构 |
| 任课教师 | 王小凤 |
| 姓 名 | 张思瑾 |
| 学 号 | 202421955 |
| 院 系 | 信息科学与技术学院 |
| 专 业 | 软件工程学硕 |

**西北大学研究生院制 2025年 4月**

# 问题描述

图像作为及其常见的一种信息传递方式，是我们生活中不可或缺的元素。对于图像的处理与应用五花八门。实际生活中，如淘宝、京东的“拍立淘”、医学领域输入X光、CT扫描图像，检索相似病例辅助诊断、车辆自动驾驶时摄像头实时匹配路标、交通灯图像以辅助导航等，均为该领域的实际用例，可见图像特征检索系统应用之广泛。

但实际应用中，图像特征通常表示为高维向量（如512维），而传统的线性搜索方法时间复杂度为O(n)，在大规模数据集上效率低下，因此本文将结合运用B+树、局部敏感（LSH）哈希，期望得到一个在大规模数据上有良好表现的图像特征检索系统。

本文将基于改进的B+树，构建一个能够快速查找相似图像的系统。应用示例：给定一个图像数据库（如100万张图像）和一个查询图像，系统需要快速返回与查询图像最相似的若干张图像。

# 需求分析

## 为什么选择B+树

B+树非常适合这种场景，因为：

1. 适合磁盘存储结构，减少I/O操作
2. 保持数据有序，支持高效范围查询
3. 平衡树结构保证稳定查询性能
4. 适合构建数据库索引

## 2. 改进点

传统B+树在高维空间存在"维度灾难"问题，即在处理高维数据时，随着维度的增加，树的性质和空间结构变得越来越复杂。我们进行如下改进：

1. 使用局部敏感哈希(LSH)降维
2. 优化节点分裂策略
3. 引入缓存机制加速查询
4. 支持批量插入操作

## 3. 预期效果

相比线性搜索，改进B+树可实现：

1. 查询时间从O(n)降到O(log n)
2. 内存占用减少
3. 准确率保持在90%以上

# 国内外现状

当前图像检索领域主流技术包括：

1. 基于深度学习的特征提取(CNN, Transformer)

优点：高精度：深度特征具有强大的语义表达能力；端到端学习：自动学习最优特征表示；适应性强：可通过微调适应不同领域

缺点：计算成本高：需要GPU加速特征提取；特征维度高（如2048维），直接搜索效率低；依赖大量标注数据训练

1. 近似最近邻搜索(ANN)算法：
   1. 基于哈希的方法（LSH, Spectral Hashing）

优点：检索速度快（O(1)复杂度）；内存占用低（二进制编码）；适合大规模数据集

缺点：精度随维度升高急剧下降（维度灾难）；需要手动设计哈希函数

* 1. 基于量化的方法（PQ, OPQ）

优点：高压缩比（如将浮点向量转为8-bit编码）；支持高效距离计算（查表法）

缺点：训练量化器计算成本高；对数据分布敏感

* 1. 基于图的方法（HNSW, NSG）

优点：当前SOTA精度（Recall@10可达90%+）；支持动态数据更新

缺点：构建索引时间长；内存消耗大（需存储近邻图）

1. 传统树结构改进：
   1. VP-tree

优点：适合度量空间（如余弦相似度）；对高维数据适应性优于KD-Tree

缺点：树平衡性难保证；节点分裂策略影响较大

* 1. KD-tree/Ball-tree

优点：理论清晰，实现简单；低维数据（<20维）效率高

缺点：高维性能退化（维度灾难）；不支持动态更新

1. 最新趋势：结合深度学习特征和高效索引结构，在精度和速度间取得平衡

优点：兼顾精度与效率，深度特征表达 + 高效索引；工业界主流方案（如Google、Facebook）

缺点：系统复杂度高；需要调参（如LSH维度、量化位数）

# 系统设计

## 算法描述

1. 特征提取：使用预训练CNN模型提取图像特征向量
2. 降维处理：使用LSH将高维向量映射到低维空间
3. 索引构建：在低维空间构建改进B+树
4. 相似度查询：在B+树中执行k近邻搜索

## 动态节点大小

1. 局部敏感哈希（LSH）降维

* 改进内容：将高维特征向量（512维）通过随机投影映射到低维空间（64位二进制哈希）
* 解决痛点：传统B+树在高维空间效率急剧下降（维度灾难）
* 提升效果：
  + 查询复杂度从O(n)降至O(log n)
  + 内存占用减少30%（二进制编码压缩）
* 实现代码：

1. 动态节点分裂策略

* 改进内容：根据数据分布自动调整B+树节点分裂阈值
* 解决痛点：固定分裂策略导致树结构不平衡
* 提升效果：
  + 查询延迟降低
  + 树高减少
* 实现代码：

1. 多线程并行索引构建

* 改进内容：使用线程池并行插入数据
* 解决痛点：大规模数据集串行构建耗时
* 提升效果：构建速度提升3-5倍（8线程）
* 实现代码：

1. 混合精确/近似搜索

* 改进内容：结合B+树精确范围查询与LSH近似搜索
* 解决痛点：纯精确搜索召回率低，纯近似搜索精度差
* 提升效果：
  + 召回率提升
  + 保持亚线性时间复杂度
* 实现代码：

1. 混合精确/近似搜索

* 改进内容：缓存频繁查询的节点和结果
* 解决痛点：重复查询导致冗余计算
* 提升效果：热门查询响应时间降低
* 实现代码：

1. 批量插入优化

* 改进内容：预先排序批量数据再批量插入
* 解决痛点：单条插入导致频繁节点分裂
* 提升效果：索引构建速度提升
* 实现代码：