# Indexing Images for Content Based Retrieval


蒋梦青
2014013443
jmq14@mails.tsinghua.edu.cn

张子昭
2014013430
zzz_14@126.com

叶佩
2014013456
yep14@mails.tsinghua.edu.cn

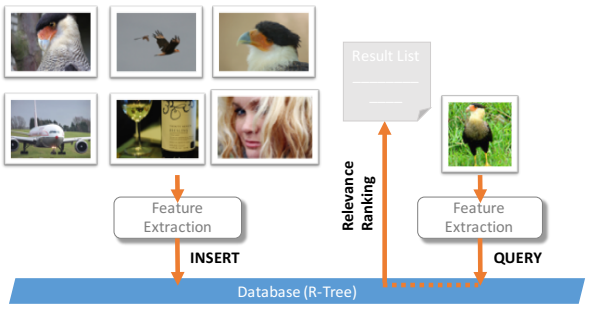

## 1. 引言

搜索引擎不断发展，“以图搜图”，即基于内容的图片检索技术也逐渐成熟。本文主要研究以 R 树为索引的基于内容的图片检索系统中，不同的图片特征向量种类、维度、数据库大小、R 树的类型等因素对检索效率和正确率的影响。

本文的架构如下：第 2 部分介绍基于内容的图片检索系统的构架，第 3 部分介绍我们对 R 树的研究，第 4 部分介绍实验中用以比较的图片特征，第 5 部分展示我们的实验过程和结果分析，第 6 部分为结论。

## 2. 基于内容的图片检索系统

该系统的基本架构如下图所示，分为两个部分：构建索引部分和查询部分。




构建索引：提取数据集图片特征向量建立索引，存到 R 树。

查询：提取查询图片特征向量，在 R 树中进行范围查询，并将查询结果根据相似度函数排序后返回结果列表。

## 3. R 树

R 树是 GUTTMAN 于 1984 年提出的最早支持有序扩展的对象存取方法之一，也是目前应用最为广泛的一种空间索引结构。

一棵 R 树满足如下的性质：

1.除根结点之外，所有非根结点包含有 m 至 M 个记录索引。根结点的记录个数可以少于 m。通常，m=M/2。

2.对于所有叶子中存储的记录，I 是最小的可以在空间中完全覆盖这些记录所代表的点的矩形。

3.对于所有非叶子结点上的记录，i 是最小的可以在空间上完全覆盖这些条目所代表的点的矩形。

4.所有叶子结点都位于同一层，因此 R 树为平衡树。

### 3.1 不同节点形状的 R 树

对于普通的 R 树（记为 Regular R-Tree），我们分别研究了它的节点形状为高维球（记为 Sphere R-Tree）和高维长方体（记为 Rect R-Tree）的情况。

对于 Sphere R-Tree，输入特征值时，以特征值表示的坐标为球心，某一设定的值为半径储存。将两个球合并为一大球时，应使大球包含两个小球且大球的体积最小。

对于 Rect R-Tree，输入特征值时，以特征值表示的坐标为长方体的中心作一个边长为给定值的高维立方体。将两个长方体合并为一个大长方体时，应使大长方体包含两个小长方体且大长方体的体积最小。

### 3.2 其它类型的 R 树

除了普通的 R-Tree 外我们研究了它的一个变种：Hilbert R-Tree。[1]

Hilbert 曲线是一种空间填充曲线，空间的每一点都有其 Hilbert 值，它将多维空间位置关系映射成为一维顺序关系，能较好地表达空间聚集的程度。Hilbert-R-Tree 总体结构类似于 R-tree，但是在节点单元中，增加了 LHV(lagerest hilbert value)字段来记录该单元 MBR 所包含的子 MBR 的 Hilbert 值中的最大值。此外，对于索引树的每一层节点，所有单元均按 LHV 值大小从左向右有序排列。

Hilbert R-Tree 的搜索算法与 R-Tree 类索引相似。插入算法和溢出处理算法如下：

插入算法：将 hilbert 编码为 h 的对象插入到索引中
S1：令 N 等于树的根节点
S2：如果 N 为叶子节点，则 N 即为所求，退出返回节点 N
S3：如果 N 不是叶子节点，则在节点单元中选取 E，其 LHV 值大于 h，而其左边的单元的 LHV 小于 h
S4：令 N 等于单元 E 所指的子节点，从 S2 开始重复

溢出处理：
S1：设 C 为一个单元的集合。将溢出节点 N 中的所有单元（M+1）和其 S-1 个兄弟结点中的单元极爱入到 C 中
S2：（如果 N 的兄弟节点都是满节点，则生成一个新的节点）将 C 中的单元按 LHV 的顺序，均匀的分配到这 S 或 S+1 个节点中

Hilbert R-Tree 通过兄弟节点链表推迟节点分裂，使空间利用率达到 100%，这样在搜索时能减少磁盘访问次数。同时，根

[1] 参考 https://github.com/atoader/HilbertRTree，并对它的代码进行 bug fix 和改进之后用于实验对比

据 LHV 值对子节点进行排序，分裂节点时将其一分为二，而非反复计算比较 MBR 的增量，提高了插入的效率。

## 4. 图片特征的提取与处理

我们提供了 7 种图片特征，分别为：Color moment features[2], color histogram features 及它的改进版本, local binary pattern (LBP) features[3], histogram of oriented gradients (HOG) features[4], Gabor features[5] and Gist descriptor features[6].

### 4.1 特征提取

这里具体介绍 Color histogram features 及它的改进版本。Color histogram 是计算 RGB 颜色落在每个颜色小区间(bin)内的像素数量可以得到颜色直方图，正则化之后得到该特征。

但是可能产生颜色相近但是落在不同的 bin 的情况，这样会使本来相似的图片但是特征的欧式距离变大，优化方法是对颜色直方图事先进行平滑过滤，即每个 bin 中的像素对于相邻的几个 bin 也有贡献。

LBP 和 HOG 都是对图片局部区域纹理进行统计提取特征；Gabor 函数是一个用于边缘提取的线性滤波器，它的频率和方向表达同人类视觉系统类似；Gist 则是基于自然度、开放度、粗糙度、膨胀度、险峻度的场景特征描述。

### 4.2 特征处理

由于通过特征提取函数得到的特征向量维度很大，但是高维向量作为索引会大幅提高内存消耗，减小查询效率等（我们在实验过程中也证实了这一点，在维度大于 256 时内存溢出，程序无法正常运行）

所以我们引入了 Principal Component Analysis (PCA)[7]主成分分析方法来降维，在尽可能代表原特征的情况下，将原特征进行线性变换、映射到低维空间。

## 5. 实验

### 5.1 实验环境

操作系统：windows 10 教育版

IDE: Visual Studio 2012 release 模式

编程语言：C++

### 5.2 实验过程

用了 10 个类别共 5613 张图片作为实验数据，分别对上述三种 R-Tree，通过控制变量法对查询的节点访问次数和正确率进行了四组实验。

**实验一：**探究特征值的维度与节点访问次数的关系。

从 5613 张图片中分别提取出维度为 4、8、12、16、20 的 color histogram(improved) feature，分别构建 R 树。然后使用相同的 20 张图片（每个类别两张）依次查询，记录这 20 次查询的访问节点数量，取平均值。最后比较特征值的维度与平均节点访问次数的关系。

**实验二：**探究数据量与节点访问次数的关系。

从 5613 张图片中分别选取 1000，2000，3000，4000，5000 张图片作为实验数据（比如：若总共选 1000 张，就从每个类别中选前 100 张），提取特征值（本次实验特征值类型全部选择的是 Gabor_PCA_16），构建 R 树。然后使用相同的 50 张图片（每个类别的前五张）依次查询，记录这 50 次查询的访问节点数量，取平均值。最后比较数据量与平均节点访问次数的关系。

**实验三：**探究特征值种类和维度与正确率的关系。

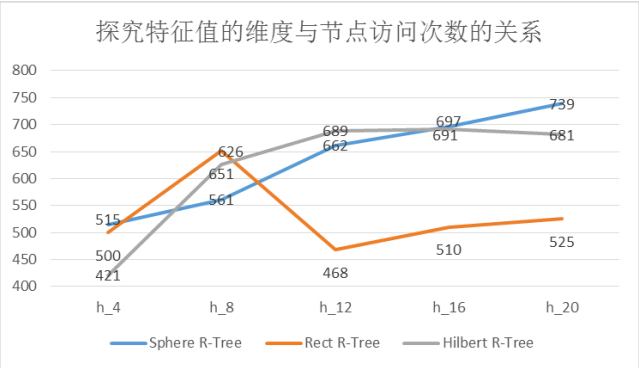
我们用 color moment feature、维度为 12 的 gabor feature、gist feature、color histogram feature、color histogram (improved) feature、HOG feature、LBP feature 和维度分别为 4、8、12、16、20 的 color histogram (improved) feature 分别构建 R 树，然后使用相同的 20 张图片（每个类别两张）依次查询，每次取查询的前 40 张计算正确率（和搜索的图片为同一类别的图片数量在 40 张图片中所占比例）并记录，取平均值后比较。

**实验四：**探究特征值种类和维度与相似度排序的准确性的关系。

我们用维度分别为 4、8、12、16、20 的 color histogram (improved) feature 和维度为 12 维的 gabor feature、gist feature、color histogram feature、color histogram (improved) feature、HOG feature、LBP feature 分别构建 R 树。然后使用相同的 20 张图片（每个类别两张）依次查询，分别计算出相似度排名前 10、20、30、40、50 的正确率，并求正确率关于排名数量的直线斜率，斜率的绝对值越大，说明相似度排序越准确。

### 5.3 实验结果

**实验一：**




对于 Sphere R-Tree，在合并两个球形节点时，能保证合并后的大球体积最小，但如果对大球的要求是包含两个球形节点的所有子节点时，其实可以获得体积更小的大球。所以在一次次合并的过程中，球形节点存在着过度扩充节点体积的问题，这一问题随着维度的增高而加剧，导致了特征值维度越高，磁盘访问次数越多的问题。而 Rect R-Tree 则没有这个问题。

[2] 特征值由助教提供

[3] 特征提取函数由 Matlab 提供

[4] 特征提取函数由 Matlab 提供

[5] (C)Mohammad Haghighat, University of Miami, haghighat@ieee.org

[6] http://people.csail.mit.edu/torralba/code/spatialenvelope/

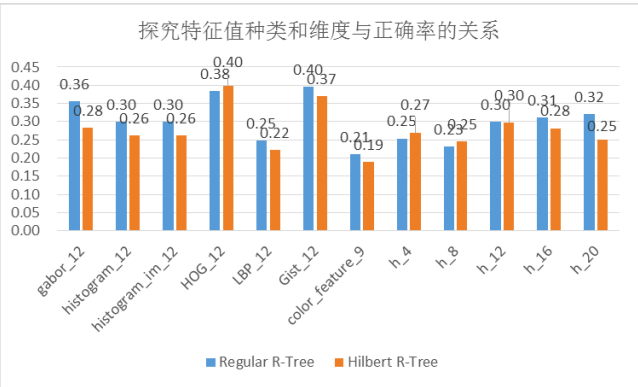
[7] PCA 函数由 Matlab 提供

Hilbert R-Tree 相对于 Regular R-Tree 并没有明显地降低节点访问次数，说明 Hilbert R-Tree 并不能有效地降低节点覆盖和交叠，直接影响 R-树的查询效率。我们可以在之后继续研究对 Hilbert R-Tree 树节点进行聚类的索引算法，以解决相邻数据的聚类存放，使叶节点 MBR 面积减小，内部节点交叠降低。

**实验二：**

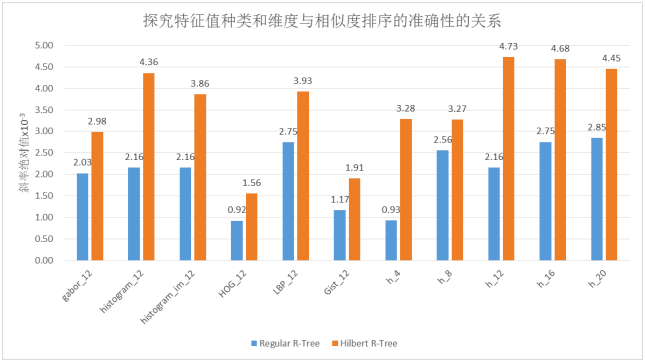

三种 R-Tree 的节点访问次数都随数据量的增加而增大，其中，相同条件下，Sphere R-Tree 的节点访问次数最多，与实验一中的分析吻合；Hilbert R-Tree 能有效地降低节点访问次数，而且数据量越大，优化性能越明显。本次实验只使用了 5613 张图片，规模比较小，Hilbert R-Tree 适合在数据量较大时使用。

**实验三：**




从实验结果可以看出，HOG feature 得出的结果正确率最高。总体来说，Regular R-Tree 得出的正确率随特征值维度的增加而增高，Hilbert R-Tree 得出的正确率没有表现出与特征值维度的关系。大部分测试中 Hilbert R-Tree 的正确率没有 Regular R-Tree 高，除了实验一中分析的原因，还有一部分原因是实验使用了 Chris Hamilton 的计算 hilbert 值的代码，但是阅读源码发现，其数据结构 HilbertValue 和比较大小的算法都存在一些问题。该作者已经在他的主页中给出了较新版本的代码，可以在之后的实验中使用新版的代码，观察正确率是否有所提高。

**实验四：**




由图可以看出，关于特征值种类的测试，color histogram 得出的结果相似度排序的准确率最高；关于特征值维度测试结果表明相似度排序的准确率与特征值维度没有直接联系。在这里 Hilbert R-Tree 得出的结果明显优于 Regular R-Tree，说明 Hilbert R-Tree 在相似度排序有很大的优势。

## 6. 结论

我们共实现了三种不同的 R 树，Rect R-Tree，Sphere R-Tree 与 Hilbert R-Tree（Rect R-Tree 与 Sphere R-Tree 只有节点形状不同，在某些时候统称为 Regular R-Tree），除给定特征向量外提取了 6 种不同的特征值，对特征值的维度与节点访问次数的关系、数据量与节点访问次数的关系、特征值种类和维度与正确率的关系与特征值种类和维度与相似度排序的准确性的关系进行了探究。

我们发现，正常情况下，节点访问次数与特征值的维度关联不大，与数据量成正相关；在不同特征值得出的正确率的比较测试中，所有另外提取的特征值得出的正确率都高于 color moment feature，其中 HOG feature 得出的正确率最高；在不同特征值维度得出的正确率的比较测试中，Regular R-Tree 得出的正确率随特征值维度的增加而增高，Hilbert R-Tree 得出的正确率没有表现出与特征值维度的关系；在特征值种类和维度与正确率的关系与特征值种类和维度与相似度排序的准确性的关系的测试中，我们发现 color histogram 得出的结果相似度排序的准确率最高。Hilbert R-Tree 作为优化后的 R 树，在降低搜索时的节点访问次数与相似度排序方面占有很大优势。

## 7. REFERENCES


[1] Kamel I, Faloutsos C. Hilbert R-tree: An improved R-tree using fractals[J]. 1993.

[2] 何小苑, & 闵华清. (2009). 基于聚类的 Hilbert R-树空间索引算法. *计算机工程*,*35*(9), 40-42.

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