清 华 大 学

# 综 合 论 文 训 练

题目:基于光场的 3D 对象检索系统的实现与研究


系　　别: 自动化系

专　　业: 自动化

姓　　名: 王泉

指导教师: 戴琼海　教授


2010 年 6 月 16 日

# 关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

**(涉密的学位论文在解密后应遵守此规定)**

签　名：王枭　导师签名：　　日　期：2010年7月1日

# 中文摘要

利用光场将 3D 对象表示为一系列多角度、多光照条件下的二维视图，能够很好地利用现有的基于内容的图像检索技术来实现基于内容的 3D 对象检索。随着光场数据量的不断积累，人们开始需要能够通过光场数据来对 3D 对象进行检索。因此，基于光场的 3D 对象检索正逐渐成为学术界研究的重点话题。

之前的研究中已有人利用对象的形状特征建立过一套基于光场的 3D 对象检索系统，并能得到较好的检索结果。然而，检索的准确率和计算的复杂度往往还不能令用户满意。因此，本文实现了一个集成了多种特征和多种距离度量算法的基于光场的 3D 对象检索系统，并研究了多种提高检索准确率以及降低计算复杂度的方法。

首先，本文介绍了一种基于实物的光场数据库以及我们实验室搭建的一套基于 3D 模型的光场数据库。在这两个数据库中，本文讨论并比较了三种用于描述 3D 对象视图的底层视觉特征，以及四种用于衡量 3D 对象相似度的距离度量算法。其次，我们通过实现一种按类进行标注的人机交互方式来提高检索的准确率，并研究了一种通过综合利用多种特征来提高检索准确率的多特征选择方法。然后，本文讨论了通过动态聚类将对象的视图数压缩以减小检索计算复杂度的方法。此外，我们提出了一种通过将数据库中的对象进行分级聚类并找出待查询对象的相似集来优化检索结果的方法。最后，本文介绍实验中我们搭建的 3D 对象检索软件，并描述了该软件的安装、使用和扩展方法。

**关键词**：光场；基于内容的 3D 对象检索；特征；距离度量；聚类

# ABSTRACT

By representing 3D objects as a series of 2D images obtained from multi-views and multiple light conditions with light field, we can make very good use of present Content-Based Image Retrieval methods for Content-Based 3D Object Retrieval. As the light-field datas on the Web keep increasing, people demand that light-field datas could be used for 3D object retrieval. Therefore, light-field-based 3D object retrieval is becoming more and more important in academics.

In previous studies, a light-field-based 3D object retrieval system has been established based on shape features, which also showed good performance. However, the precision of the retrieval results and the time cost of the retrieval are still unsatisfactory. Therefore, this article implemented a light-field-based 3D object retrieval system which integrated multiple features and multiple distances, and studied some promising approaches which help to improve the precision and lower the time cost.

Firstly, this article introduces a real-object-based light field database, and a 3D-model-based light field database that we established. Within these two databases, this article discusses three kinds of low-level visual features which are used to describe 3D object images, and compares four different distances which are used to measure the similarity between 3D objects. Secondly, we implement a human-computer interaction method in which the user can mark the retrieval results by classes to improve the precision of the retrieval, and studies a multi-feature selection approach which makes use of different features to improve the precision. Further, this article attempts to compress the number of object views by performing dynamic clustering to lower the time cost of the retrieval. Besides, we propose one approach which can help to optimize the retrieval results by performing hierarchical clustering on the whole database and finding out the similar object set for the input object. Finally, this article introduces the 3D object retrieval software that we established, and describes the installation, use and expansion of this software.

**Keywords：** light field; content-based 3D object retrieval; feature; distance measurement; clustering

# 目 录

# 第1章 引 言

随着当前 3D 对象数据库越来越丰富，3D 对象正成为一种非常重要的多媒体数据类型，并在计算机辅助设计、虚拟现实、分子生物学、军事和娱乐等领域有着越来越广泛的应用。因此，利用先进的检索技术来高效地利用当前庞大的 3D 数据库也正成为一种迫切的需求[2]。

基于内容的 3D 对象检索，其关键在于定义一种衡量不同的 3D 对象之间的相似度的方法，这种相似度的定义是实现一切高效检索的前提。在 3D 检索中，我们通常并不是在数据库中寻找与目标对象完全一致的精确匹配，而是根据目标对象给出一个相似对象的序列，排在序列前面的检索结果与目标对象有较高的相似度。

目前衡量 3D 对象之间的相似度主要有三种方法[7]。第一种是进行直接的几何变换，计算将一个 3D 对象转换为另一个 3D 对象所要花费的代价，此代价越小，说明两个 3D 对象的相似度越高。第二种是基于特征描述符的方法，从 3D 对象中提取特征描述符，并用描述符来表示 3D 对象，从而 3D 对象之间的相似度也可以通过描述符之间的相似度来衡量。以上两种相似度量都是基于整个 3D 对象的，而第三种相似度度量方法基于 3D 对象的局部区域，这种方法通过寻找 3D 对象之间相似的区域，来进行相似度度量。

本文所描述的基于光场的方法属于上述第二种方法，即基于特征描述符的方法。基于描述符的方法一般要求描述符具有对旋转变换、平移变换和尺度变换的不变性，以及一定的抗噪性。出于这种需求，从 3D 对象中获取特征一般有三种做法[7]：第一种是将 3D 对象看作无厚度的表面，从中可以获取该表面的微分特征；第二种是将 3D 对象看作有厚度有体积的表面；第三种便是本文所探讨的，将 3D 对象投影到多个平面上，从各个投影中获取特征。

# 第2章 实验数据库介绍

## 2.1 光场数据库简介

研究基于光场的 3D 对象检索以及检索系统的实现都需要一定规模的数据库，数据库需要具有良好的分类，足够的对象数量，以及清晰的视图。光场数据库一般分为两种类型：一种是基于实物的光场数据库，构建这种数据库需要大量的实物，并进行复杂的现场拍摄工作，成本较高；另一种是基于 3D 模型的光场数据库，构建这种数据库，可以从互联网上搜集各种模型，在软件中利用虚拟摄像机阵列模拟光场来进行拍摄，比较容易实现。在本文的研究工作中，利用到了苏黎世理工大学提供的基于实物的光场数据库 ETH-80，以及本人作为主要设计者参与搭建的基于 3D 模型的光场数据库 Sphere60。

## 2.2 ETH-80 数据库

ETH-80 数据库是由苏黎世理工大学提供的实物光场数据库[12]。该数据库的背景单一，能够很方便地从视图中提取出对象，并从对象中提取出各种特征。数据库中一共有 8 个类，每个类中有 10 个对象，总共 80 个对象。这 8 个类中，既有自然界的物体，又有人造物体。8 个类分别为：苹果、梨、西红柿、牛、狗、马、杯子和汽车。每个类的 10 个对象在采集时尽量选取了差异较大，但又明显属于该类的对象。搭建光场时，在对象实物的周围构造一个正八面体，然后对其进行三级细分，取上半部分，在得到的顶点上设置照相机进行拍摄，这样每个对象可以获取 41 个视图，80 个对象一共 3280 幅图片。ETH-80 数据库中，可以提取出对象的轮廓、颜色、纹理等各种特征。

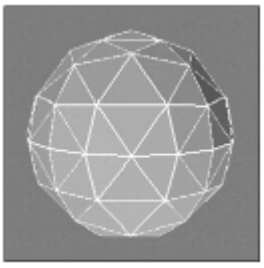

图2.1 正八面体的三级细分

ETH-80 数据库中 80 个对象的代表视图如下：

图2.2　ETH-80数据库对象代表视图

## 2.3 Sphere60 数据库

Sphere60 数据库是由本人作为主要设计者在实验室参与搭建的 3D 模型光场数据库。数据库一共有 8 个类，每个类有 10 个对象，总共 80 个对象。8 个类分别为自行车、水龙头、轮船、沙发、鸟、恐龙、鱼和树，其中 4 类为自然界的物体，4 类为人造物体。与 ETH-80 数据库相比，Sphere60 数据库中每个类中的对象差异性要大得多，比如鸟这一类中，既有展翅飞翔的鹰，也有奔跑的鸵鸟，还有收翅的企鹅，这无疑使检索难度大大增加。

Sphere60 数据库中，每个对象有 60 个视图。拍摄时在对象的周围构建一个球面，球面上按照 C60 分子 32 面体的结构选取 60 个顶点，在顶点上安置照相机进行拍摄。数据库的搭建过程中我们使用 3D max 软件，并利用 3D max 脚本语言编写拍摄程序。

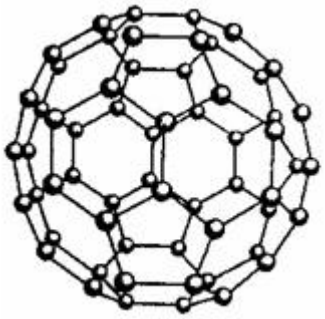

图2.3　C60分子32面体结构

Sphere60 数据库的构建过程如下：

1) 在互联网上收集 3D 模型，并按类整理，对于 obj 格式的文件可以直接保留用于拍摄，对于 max 等非 obj 格式的文件，需将对象统一导出为 obj 格式；
2) 将 obj 格式的 3D 模型导入 3D max 场景，置于场景的中心；
3) 设置球面半径和光照参数，使得既能拍摄到完整的模型，不出现对象超出图片的情况，又能使对象在图片中所占的面积比例较大；
4) 试拍单张图片进行效果测试；
5) 如果效果合适，则进行完整的拍摄，得到对象的 60 张图片；
6) 将图片格式转换为 jpg 格式。

由于网上搜集的 3D 模型很多都没有颜色、纹理等信息，因此拍摄时将光强设置为较大值，这样可以突出轮廓特征。

# 第3章 检索过程与算法

## 3.1 检索系统概述

我们在图 3.1 中给出整个检索系统的基本流程。可以看到，除了数据库部分之外，检索过程主要包括对象输入、特征提取、相似性度量、相关反馈和结果输出五大模块。

对象输入模块提供一定的输入方式，使得用户可以从数据库中选择要进行检索的 3D 对象，选择完成后，该对象被传递到特征提取模块。

特征提取模块让用户能够从输入的 3D 对象中提取不同的特征，这些特征被保存到特征文件中，并被相似性度量模块所调用。

相似性度量模块根据对象的特征，计算不同对象之间的距离，并进行从小到大的排序，将排序结果传递到输出模块。

而相关反馈模块使用户可以对输出结果进行标注，并将标注的信息返回到相似性度量模块，从而影响下一轮的输出。

在后面的几节中，我们将针对检索系统中的各个模块进行详细而深入的讨论。

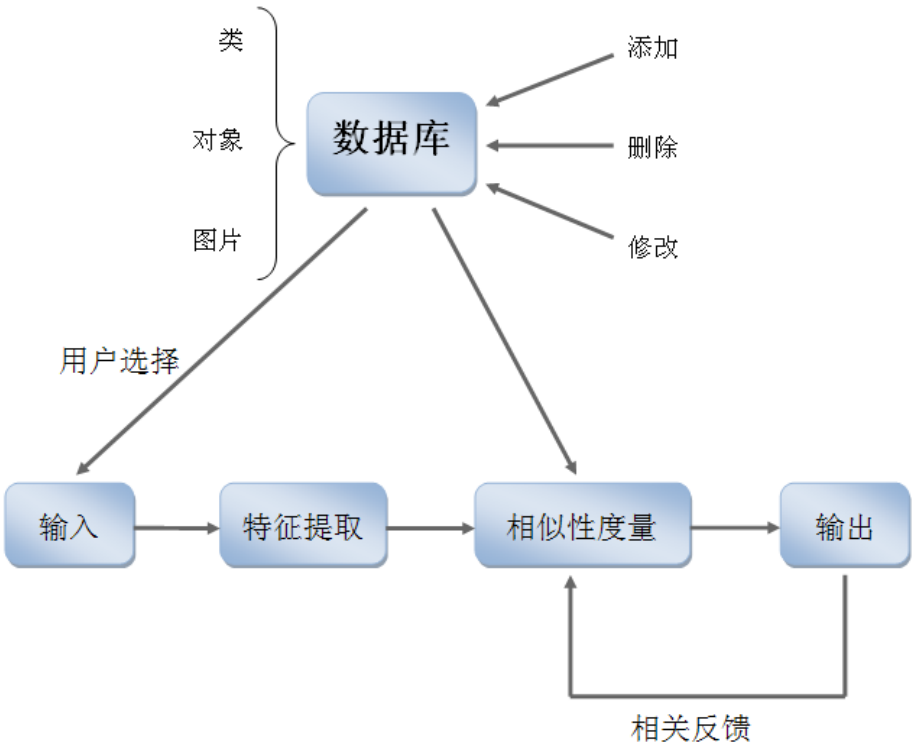


图3.1 检索系统基本流程

## 3.2 特征提取

### 3.2.1 概述

由于 3D 对象的光场数据库从实质上来说就是图像的数据库，因此 3D 对象的特征提取在这里转变成为 2D 图像的特征提取。按照图像的特征对图像内容的描述程度，可以将特征分为底层、中层和高层三个级别。底层特征描述的是最基本的视觉特征，如形状、颜色、纹理等；中层特征指图像中包含的逻辑对象；高层特征描述图像中的高层语义和抽象概念[1]。中层和高层特征的提取非常复杂，目前的图像检索技术大部分还是基于底层特征的，因此本文的研究也采用底层的特征来进行实验。

底层特征中比较常用的有形状特征、颜色特征和纹理特征，以下一一进行介绍。

### 3.2.2 形状特征

形状特征不仅需要先将物体或区域从图像中分割出来，而且还要满足对尺寸、旋转等变换的不变性，因此提取起来较为复杂。通常来说形状特征有两种，轮廓特征和区域特征。轮廓特征（例如傅里叶描述符）描述对象的外边界，区域特征（例如 Zernike 矩描述符）则关系到整个形状区域。

本文使用傅里叶描述符（Fourier Descriptor）来描述对象的轮廓特征。其基本思路是，先从图像中提取出对象，再从对象中提取出轮廓，沿着对象的轮廓取点，每个点可以表示成坐标*(x,y)*，那么这一系列的点可以表示成复向量

$$\tilde{U} = \begin{pmatrix} x_0 + iy_0 \\ x_1 + iy_1 \\ \cdots \\ x_N + iy_N \end{pmatrix}$$

再对所得的复向量使用离散傅里叶变换

$$\tilde{F}_\mu = FFT[\tilde{U}] = \sum_{k=0}^{N-1} \tilde{U}_k \exp\left(-\frac{2\pi i}{N} k\mu\right)$$

此时便可以用得到的 $\tilde{F}_\mu$ 的模 $\left|\tilde{F}_\mu\right|$ 作为对象的傅里叶描述符。

本文所实现的系统所用的傅里叶描述符为 120 维的实向量，并在程序中简写为 FD 特征。

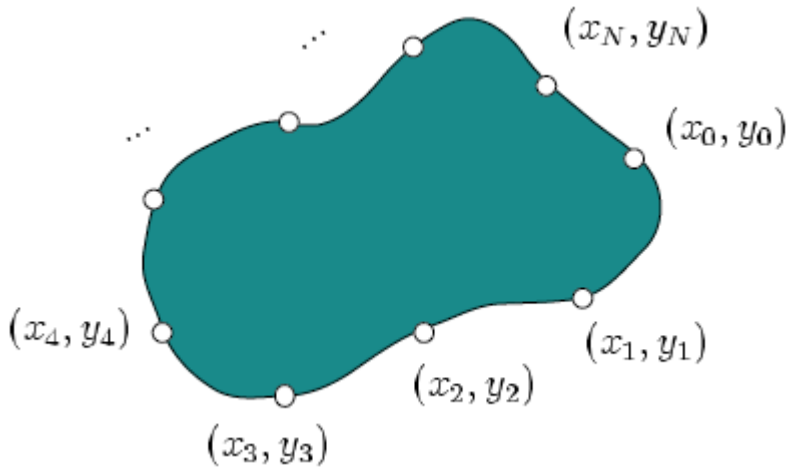


图3.2　沿对象轮廓取点示意图

### 3.2.3 颜色特征

然后是颜色特征，颜色特征中较为常见的有颜色直方图特征、颜色矩特征以及颜色聚合向量特征等。其中颜色直方图（color histogram）特征在图像检索中用的较多，其关注不同的颜色在图像中所占的比例，而与颜色在空间中的位置无关[5]。颜色直方图可以基于不同的颜色空间，例如 RGB 空间和 HSV 空间。虽然 RGB 空间非常常用，但 RGB 空间结构与人们对颜色相似性的主观感受并不是很相符，因此本文描述的系统采用的是 HSV 空间[1]。计算颜色直方图的方法很简单，将颜色空间划分为小的颜色区间，此过程称为颜色量化，然后计算每个颜色区间内的像素数量即得到颜色直方图。

本文所实现的系统用到的 HSV 空间颜色直方图为 256 维的实向量，并在程序中简写为 HSV 特征。

### 3.2.4 纹理特征

然后是纹理特征。纹理特征不依赖于图像的颜色和亮度，但能反映图像中的同质现象，包含物体表面的结构组织信息。然而由于物体的形状和纹理之间存在密切关系，加上纹理可能呈现嵌套式的分布，因此利用纹理特征进行图像检索也存在一定的困难。

纹理特征又包括了小波纹理、Tamura 纹理等。小波纹理根据用于纹理分析的小波变换分为两种，金字塔结构的小波变换（pyramid-structured wavelet transform, PWT）和树形结构的小波变换（tree-structured wavelet transform, TWT）。

本文所实现的系统用到的是树形结构的小波变换 TWT，为 104 维的实向量，并在程序中简写为 TWT 特征。

## 3.3 相似性度量

### 3.3.1 概述

有了每个视图的特征向量之后，非常重要的一点便是根据这些特征向量计算对象之间的相似性，也可以表述为计算对象之间的距离，因为可以认为两个对象之间距离越小，相似性便越高。

要计算对象之间的距离，首先要计算图像之间的距离。由于每个图像是由特征向量来表示的，因此图像之间的距离也就是特征向量之间的距离。特征向量之间的距离比较常用的是闵可夫斯基距离 $L_p(x,y)=\left(\sum_{i=1}^{d}|x_i-y_i|^p\right)^{\frac{1}{p}}$ ，其中 $p\geq 1$。本文实现的系统中取 $p=2$，即欧氏距离，计算起来比较简便[7]。

有了图像的距离之后，便要根据图像之间的距离来计算对象之间的距离了。由于每个对象有多个视图，每两个视图之间都有一个距离，因此两个对象之间就有一个距离矩阵。例如对象 A 有 n 个视图 $\{A_1,A_2,\cdots,A_n\}$，对象 B 有 m 个视图 $\{B_1,B_2,\cdots,B_m\}$，距离矩阵 $M$ 的第 $i$ 行第 $j$ 列的元素 $M_{ij}$ 便是 A 的第 $i$ 个视图与 B 的第 $j$ 个视图之间的距离。由于要根据目标对象与数据库中其他对象的距离大小来对数据库中其他对象进行排序以完成检索的过程，所以仅仅有距离矩阵是不够的，而要根据距离矩阵来算出一个实数作为两个对象之间的距离。

两个对象的距离也可以看做是两个点集的距离，每个对象是一个点集，而对象中每一个视图可看作多维空间中的一个点，而视图的特征向量便可以看做是视图在多维空间中的坐标。

两个点集之间的距离度量有很多种方法，这里介绍四种距离：最近邻距离，豪斯道夫距离，改进的豪斯道夫距离，以及最优匹配。

### 3.3.2 最近邻距离

最近邻距离（Nearest Neighbour）是点集与点集之间最简单的一种距离，点集 A 中的点到点集 B 中的点的距离的最小值便是 A 到 B 的最近邻距离。用矩阵语言描述，也就是 $d_{NN}(A,B)=\min_{i,j} M_{ij}$。用最近邻距离来衡量两个对象之间的相似度，也就是将两个对象的最相似视图的相似度作为两个对象的相似度。

### 3.3.3 豪斯道夫距离与改进的豪斯道夫距离

豪斯道夫距离（Hausdorff Distance）的定义则比较不同，它先求点集 A 中的所有点到点集 B 中一点的所有距离的最小值，再在 B 中求所有这些最小值的最大值，然后再互换 A 与 B 的地位进行相同的计算，最后在两个最大值之间取较大者作为 A 与 B 的距离。用矩阵语言描述为 $H(A,B)=\max(h(A,B),h(B,A))$，其中

$$h(A,B)=\max_i\left(\min_j M_{ij}\right),\quad h(B,A)=\max_j\left(\min_i M_{ij}\right)$$

此外还有改进的豪斯道夫距离（Modified Hausdorff Distance，简写为 MHD），其与豪斯道夫距离的不同之处在于，不再是对最小值求最大值，而是对最小值求平均值[3]，也就是 $h(A,B)=\operatorname{mean}_i\left(\min_j M_{ij}\right)$。后面的实验将证明改进后的豪斯道夫距离检索正确率远远优于改进前的豪斯道夫距离。

### 3.3.4 最优匹配与 KM 算法

最优匹配（Optimal Matching）与上面三种距离的定义形式差别较大，它试图找到两个点集之间点与点的一种一一对应，使得在这种一一对应下，所有点对的距离之和最小。这种定义的前提是两个点集中点的个数相同。如果两个点集中点的个数不同，可以将其转换为点的个数相同时的情况，这将在后面讨论。最优匹配用矩阵语言来描述，也就是在 $n\times n$ 的矩阵 $M$ 中，找到 $n$ 个两两不同行且不同列的数，使得这 $n$ 个数的和最小。这种定义显然非常适合于对象间的距离度量，因为相当于先假定了两个对象是相关的，然后去寻找不同视图间的对应，这样的定义很好的满足了对 3D 对象旋转变换的不变性。但是最优匹配的算法实现起来并不容易。如果用最简单的穷举法，算法复杂性将达到 $O(n!)$，当 $n$=60 时，为 $10^{81}$ 量

级的计算量，显然不可能用于检索。因此本文实现的系统中用于最优匹配的算法是 Kuhn - Munkras 算法，简称 KM 算法。

KM 算法的描述要涉及到图论的语言，首先需要定义二分图以及二分图的最大权匹配。

设 $G=(V,E)$是一个无向图，如果顶点集 $V$ 可分割为两个互不相交的子集 $A$ 与 $B$，并且图中的每条边 $(i,j)$所关联的两个顶点 $i$ 和 $j$ 分别属于这两个不同的顶点集（即 $i$ 是 $A$ 中的顶点，$j$ 是 $B$ 中的顶点），则称图 $G$ 为一个二分图。如下图所示，图 $G$ 分为子集 $A$ 与 $B$，$A$ 由点 1、2、3、4 组成，$B$ 由点 5、6、7、8、9 组成，每一条边的两个顶点都分别位于 $A$ 与 $B$ 中。

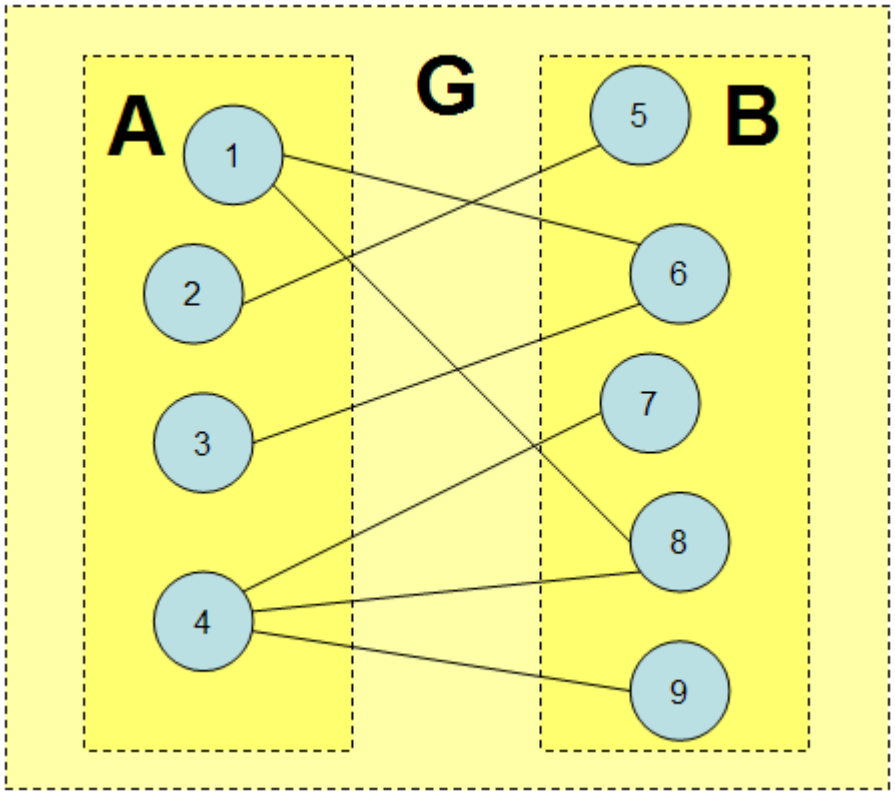


图3.3　二分图示例

给定一个二分图 $G$，设 $M$ 是 $G$ 的一个子图，如果 $M$ 的边集中的任意两条边都不与同一顶点相关联，则称 $M$ 是 $G$ 的一个匹配。如果一个匹配中的每个顶点都和某条边相关联，则称此匹配为完备匹配。所有边的权值之和最大的匹配称为最大权匹配。

KM 算法通过对每个顶点进行标号来把求最大权匹配的问题转化为求完备匹配的问题。不妨设二分图中两个顶点集分别为 $X$ 与 $Y$，顶点 $X_i$ 的标号为 $A[i]$，顶点 $Y_j$ 的顶标为 $B[j]$，顶点 $X_i$ 与 $Y_j$ 之间的边权为 $w[i,j]$。在算法执行过程中的任一时刻，对于任一条边$(i,j)$，$A[i]+B[j] \geq w[i,j]$ 始终成立。

KM 算法的正确性基于以下定理：若由二分图中所有满足 $A[i]+B[j]=w[i,j]$ 的边$(i,j)$构成的子图（称为相等子图）有完备匹配，那么该完备匹配即为二分图的最大权匹配。

KM 算法的实现流程如下：

(1) 初始时令 $A[i]$为所有与顶点 $X_i$ 关联的边的最大权，$B[j]$=0。这是为了满足 $A[i]+B[j] \geq w[i,j]$。若当前的相等子图没有完备匹配，则按以下的方法修改标号，使相等子图扩大，直到相等子图有完备匹配。

(2) 当前相等子图没有完备匹配是由于对于某个 $X_i$ 顶点，我们找不到一条从它出发的增广路。这时我们得到了一棵交错树，它的叶子结点全部是 $X$ 中的顶点。然后我们将交错树中 $X$ 中的顶点标号全都减小某个值 $d$，$Y$ 中的顶点标号全都增加同一个值 $d$，则：

1）如果边$(i,j)$两端都在交错树中，则 $A[i]+B[j]$的值没有变化。也即它原来属于相等子图，现在仍属于相等子图。

2）如果边$(i,j)$两端都不在交错树中，则 $A[i]$和 $B[j]$都没有变化。也即它原来属于（或不属于）相等子图，现在仍属于（或不属于）相等子图。

3）如果边$(i,j)$的 $X$ 端不在交错树中，而 $Y$ 端在交错树中，则 $A[i]+B[j]$的值增大。也即它原来不属于相等子图，现在仍不属于相等子图。

4）如果边$(i,j)$的 $X$ 端在交错树中，$Y$ 端不在交错树中，则 $A[i]+B[j]$的值减小。也即它原来不属于相等子图，现在可能进入了相等子图，从而使相等子图得到了扩大。

为了使 $A[i]+B[j] \geq w[i,j]$ 始终成立，且至少有一条边进入相等子图，设交错树为 $T$，则应该取

$$d = \min\{A[i]+B[j]\text{-}w[i,j] \mid X_i \in T \cap Y_j \notin T\}$$

按照以上的方法，由于需要寻找 $O(n)$ 次增广路，每次增广最多需要修改 $O(n)$ 次顶标，每次修改顶标时由于要枚举边来求 $d$ 值，时间复杂度为 $O(n^2)$，因此 KM 算法的时间复杂度为 $O(n^4)$。通过引入“松弛量”，KM 算法的时间复杂度可以做到 $O(n^3)$，因此 KM 算法适用于检索。

因为 KM 算法求得的是最大权匹配，因此用 KM 算法计算矩阵 $M$ 的最优匹配时，需要先将 $M$ 中所有元素取相反数，然后再求最大权匹配。

如果两个点集中点的个数不相同，也即 $M$ 不是方阵，可以通过如下方法将 $M$ 转换为方阵。设 $M$ 是 $n\times m$ 的矩阵，不妨设 $m>n$，则对于 $M$ 的这 $m$ 个列，计算每列的 $n$ 个元素的平均值，从这 $m$ 个平均值中找出最小的 $n$ 个，这 $n$ 个平均值所对应的列组成新的方阵 $M'$，然后用 $M'$ 代替原来的 $M$ 来计算最优匹配即可。

## 3.4 性能评价

在信息检索中，最常用的两个性能评价指标为准确率（precision）和召回率（recall）[7]。不妨假设在一次检索中，检索返回结果中与输入对象相关联的对象数为 $n$，检索返回结果中所有对象数为 $N$，数据库中所有与输入对象相关联的对象数为 $M$，则

$$precision=\frac{n}{N},\quad recall=\frac{n}{M}$$

也就是说，准确率反映系统检索到的对象中相关对象所占的比例，召回率反映系统从数据库中检索到相关对象的能力。有了这两个性能评价指标之后，我们可以给出另外两个衡量系统检索性能的方法：R-准确率（R-precision）和 P-R 曲线（precision versus recall plots）[7]。R 准确率指的是当 $N$=$M$ 时的准确率。由于 R 准确率是一个单一的实数，因此可以用来衡量或比较不同检索算法的效果。P-R 曲线是指 $N$ 取从小到大的一系列不同的值时，准确率和召回率在坐标空间形成的曲线。$N$ 增大时，$n$ 也会相应地增大或保持不变，因此横坐标召回率 recall 是随着 $N$ 的增大而增大的，而 $n$ 的增大速度肯定不会快于 $N$ 的增大速度，因此纵坐标准确率 precision 是随着 $N$ 的增大而减小的。因此，在 P-R 曲线中，准确率也是随着召回率的增大而减小的。对于相同的召回率，准确率越高，检索性能便越好，此时曲线靠近坐标轴的右上方。

为了比较不同距离度量算法的性能，我们搭建了一个混合数据库，数据库中包含 11 个类，每个类有 10 个对象，这些类的来源不一，因此不同类中的对象的视图数量也不同。11 个类的类名和类中对象的视图数如下表所示：

表3.1 混合数据库

| 类名 | bird | bottle | chair | dinosaur | guitar | plane | wheel | pear | apple | car | dog |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 视图 | 60 | 60 | 60 | 60 | 60 | 60 | 60 | 41 | 41 | 41 | 41 |

利用该数据库的傅里叶算子特征，可以计算四种距离度量算法的总体 P-R 曲线。

计算总体 P-R 曲线不同于计算某一次检索的 P-R 曲线。计算总体 P-R 曲线需要对数据库中所有的对象分别执行一次检索，然后将返回次数 *N* 从小到大进行取值（例如从 1 取到 30），计算 *N* 取不同值时，数据库中所有对象检索结果的准确率和召回率的平均值，作为数据库总体的准确率和召回率。

四种距离度量算法的总体 P-R 曲线如下图所示：

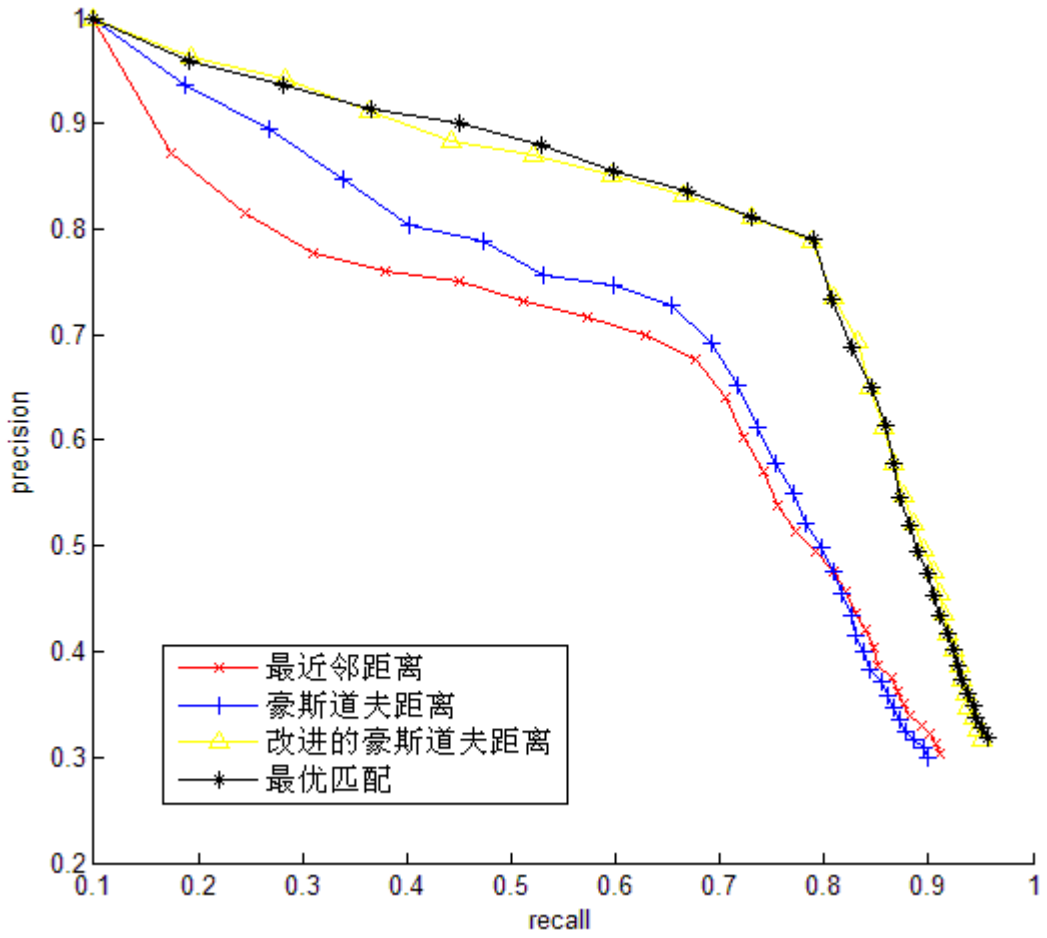


图3.4　四种距离度量算法的总体P-R曲线

从图中可以看到，最近邻距离的 P-R 曲线比较靠左下方，性能不是很好。豪斯道夫距离比最近邻距离有明显的改进，但也不是很好。改进的豪斯道夫距离和最优匹配的 P-R 曲线靠近右上方，性能最好。

## 3.5 相关反馈

### 3.5.1 基本方法

尽管 3D 检索系统中已有多种特征和多种距离度量方式可以使用，但是检索的结果还是常常不能令人满意。这是因为图像底层的视觉特征与图像所表示的高层语义之间还存在很大的差异。为了解决这个问题，一种有效而简单的方法便是通过人机交互来建立底层视觉特征和高层语义之间的联系。这里我们使用相关反馈（Relevance Feedback）技术。相关反馈技术的基本思想是，进行检索时，检索系统根据用户的查询返回一系列检索结果，然后用户对这些检索结果进行手工标注，这些标注的信息会反馈到系统中，系统进行下一次查询时，会利用反馈的信息进行学习，返回新的结果[9]。由于用户的标注是人工建立起的联系，因此新的返回结果通常更符合用户的需求。

相关反馈机制主要分为两类，即短期的相关反馈（short-term relevance feedback）和长期的相关反馈（long-term relevance feedback）。短期的相关反馈可以看做一个循环检索的过程，用户对某对象进行查询后，根据返回结果进行标注，标注结束后，检索系统直接根据标注信息该对象优化检索结果。而长期的相关反馈在每次标注之后，都会将用户反馈的信息记录下来形成一个语义库，从中学习一些高层的语义信息，并进行知识的积累，用于指导今后所有的检索过程[1]。短期的相关反馈与长期的相关反馈最大的区别就在于是否记录下用户的标注信息进行知识的积累。长期的相关反馈普遍比短期的相关反馈更能有效地将底层视觉特征与高层语义联系起来。但考虑到短期的相关反馈实现起来更加容易，并且每次检索不受以前操作的影响，因此为了便于研究，本文实现的系统采用的是短期的相关反馈。

而用户对检索结果的标注有很多种方法。比较常用的方法是对每个返回对象进行相关性评价。例如检索系统返回相似度最高的 30 个对象，用户可以手工对每个对象评分，比如可以设置“非常相关”“比较相关”“比较不相关”“非常不相关”等评分等级，评分后这些评分信息会保存在电脑中，下一次检索的时候，在距离度量算法结束后，根据用户的评分对距离值进行调整，用户标注相关的对象将距离值减小（例如乘以 0.9），用户标注不相关的对象将距离值增大（例如乘以 1.1），这样在新一轮的检索中，相关的对象会排到返回序列的更前面，不相关的对象会排到更后面，从而整个检索结果得到优化。

### 3.5.2 按类标注

本文实现的系统采用的是另一种标注方式，即按类进行标注。每次检索结束后，系统返回 30 个相似度最高的对象，并考察这 30 个对象属于哪些类别，这些类别中哪些类别包含这 30 个对象中最多的对象，也就与输入对象相关性越高。标注时，系统给出与输入对象相关性最高的 3 个类别，并显示其中包含的对象的代表视图，然后用户从这 3 个类别中进行选择，选择哪个类别也就表示用户将输入对象标注为哪个类别。在下一次的检索中，该类别中的对象与输入对象之间的距离都会减小（乘以一个小于 1 的数），从而使该类别中的对象能够排在返回序列的更前面。

相比于按对象进行标注，按类进行标注的方法能够更好地利用数据库中已有的分类信息，同时大大降低用户手工标注的工作量。

我们不妨在 ETH-80 数据库中比较一下不同输入对象的检索结果在不同反馈次数下的 R-准确率值的变化。我们用如下 3 个对象来进行实验：一个 dog 类中的对象，采用轮廓特征；一个 car 类中的对象（黄色的小汽车），采用颜色特征；一个 cup 类中的对象（白色的矮茶杯），采用纹理特征。距离度量方法统一用豪斯道夫距离。

由于 dog 类中的对象和其他四足动物如马、牛等很难通过轮廓特征区分，car 中的对象颜色互不相同，且我们所用的是一辆黄色的汽车颜色跟 pear 类中的对象更为接近，cup 类中的对象之间材质纹理也有很大的差别，所以我们所选的这三个对象和对应的特征检索效果是不大好的，我们可以看到通过多次相关反馈能够使 R-准确率逐渐上升：

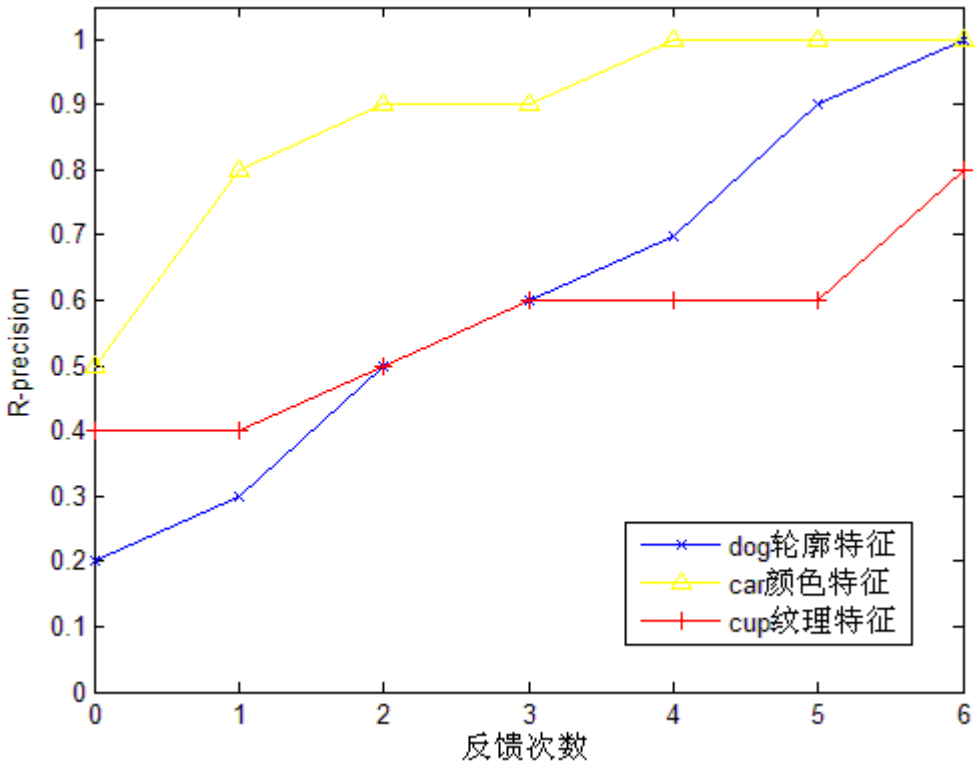


图3.5　多次相关反馈的R-准确率

## 3.6 多特征选择方法

对于某些类别，往往很难用单一的特征将其与其他类别区分开，例如四足动物马、狗、牛之间往往难以用轮廓特征区分开。因此我们非常希望能够结合多种特征，来区分单一特征无法区分开的类别。这里我们实现了一种多特征选择的算法，在 ETH-80 数据库中，综合利用轮廓特征、颜色特征和纹理特征三种特征，来实现类别之间更有效的区分。

多特征选择算法在 ETH-80 数据库中的实现方法如下[6]：

(1) 先将数据库中的每个类都分为两部分，一部分为训练集，另一部分为测试集。我们不妨先假设每个类中的前 6 个对象属于训练集，后 4 个对象属于测试集，也即 *训练集:测试集*= 6:4 。

(2) 求取每一特征的相关性矩阵 $R$。相关性矩阵用于描述在某特征下不同类别之间的相关性。由于 ETH-80 数据库有 8 个类，因此 $R$ 是一个 $8\times8$ 的 0-1 矩阵。$R$ 的第 $i$ 行第 $j$ 列元素为 0 表示第 $i$ 类和第 $j$ 类在该特征下无关，为 1 表示相关，对角线上的元素没有意义，可以都设为 1。$R$ 的求法具体如下：初始时设 $R$ 中元素

全为 0，选取某一特征后，在训练集中对每个对象进行检索（例如用最近邻距离），取前 $m$ 个检索结果作为相关对象，不妨假设 $m$=6，这样由于每个类有 6 个对象，因此每个类得到 $6\times6=36$ 个相关对象，考察这 36 个相关对象所属的类，将其认定为被检索对象所属类的相关类，将 $R$ 中对应的元素设为 1。

(3) 得到了所有特征的相关矩阵后，我们考察每个类至少需要哪几个特征才能跟其他类区分开。对于每一个类，我们取出其在不同特征的 $R$ 矩阵中对应的行向量，由于 ETH-80 数据库有轮廓、颜色、纹理三种特征，因此我们对于每个类有 3 个长度为 8 的行向量，然后我们去掉 3 个行向量中表示该类与自己是否相关的那个元素（也即原 $R$ 矩阵中位于对角线位置的元素），此时我们有 3 个长度为 7 的行向量，不妨假设这 3 个行向量组成一个向量集 $A$。然后用如下方法确定该类需要哪些特征才能区分开，不妨设所需的特征集为 $B$：

1) 如果 $A$ 中有元素全为 0 的行向量，则把该向量对应的特征加入到 $B$ 中，此时的 $B$ 就是区分该类所需的特征集；

2) 如果 $A$ 中没有元素全为 0 的行向量，则从 $A$ 中寻找 1 元素最少的行向量，将该行向量对应的特征加入到 $B$ 中，再用该行向量去与 $A$ 中所有其他行向量进行“与”运算，用运算后的结果代替 $A$ 中原来的行向量，最后将该行向量从 $A$ 中删除，回到步骤 1)。

(4) 通过上面的计算，我们知道了每个类需要哪些特征才能与其他类区分开，然后我们在测试集中进行测试。我们对测试集中的每一个对象进行检索，检索过程中计算该对象与其他对象的距离时，分别用特征集中不同的特征（区分该对象所属类与其他类所需的特征）进行计算，再求平均值（由于不同特征的绝对大小之间不具有可比性，不能直接平均，因此需要将不同特征下的距离归一化，即用当前距离减去最小距离除以最大距离减去最小距离，归一化之后再求平均，便能得到有意义的多特征距离）。如此可以绘出测试集下的总体 P-R 曲线。

由于训练集和测试集的比例可以改变，在训练集中检索返回的相关对象个数 $m$ 也可以改变，因此我们给出三种情况下的特征选择结果和总体 P-R 曲线。

a) 训练集为 6，$m$=6 时，区分每个类所需的特征如下：

表3.2 训练集为6，$m$=6时的特征集

| 类名 | 区分该类所需特征 |
|---|---|
| apple | 轮廓，颜色，纹理 |
| car | 轮廓 |

| | |
|---|---|
| cow | 轮廓，颜色，纹理 |
| cup | 轮廓，颜色 |
| dog | 轮廓，颜色，纹理 |
| house | 轮廓，颜色，纹理 |
| pear | 轮廓 |
| tomato | 轮廓，颜色，纹理 |

测试集总体 P-R 曲线如下：

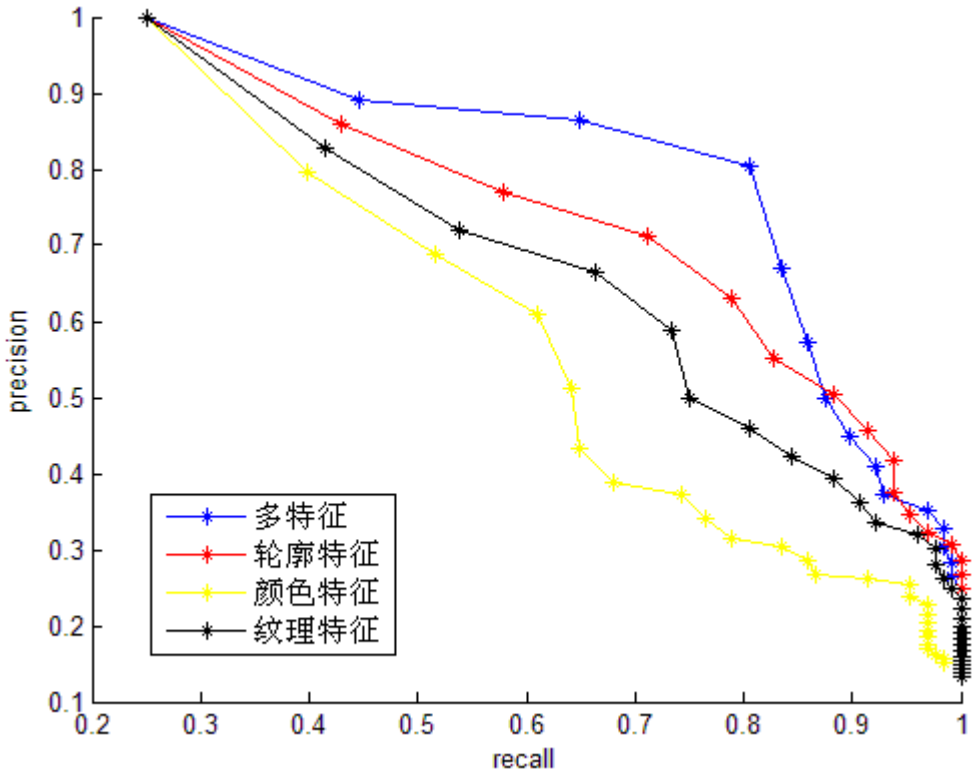


图3.6　训练集为6，*m*=6时的总体P-R曲线

b) 训练集为 6，*m*=4 时，区分每个类所需的特征如下：

表3.3 训练集为6，*m*=4时的特征集

| 类名 | 区分该类所需特征 |
|---|---|
| apple | 轮廓，颜色，纹理 |
| car | 轮廓 |
| cow | 轮廓，颜色，纹理 |
| cup | 颜色 |

| | |
|---|---|
| dog | 轮廓，颜色，纹理 |
| house | 轮廓，颜色，纹理 |
| pear | 轮廓 |
| tomato | 颜色 |

测试集总体 P-R 曲线如下：

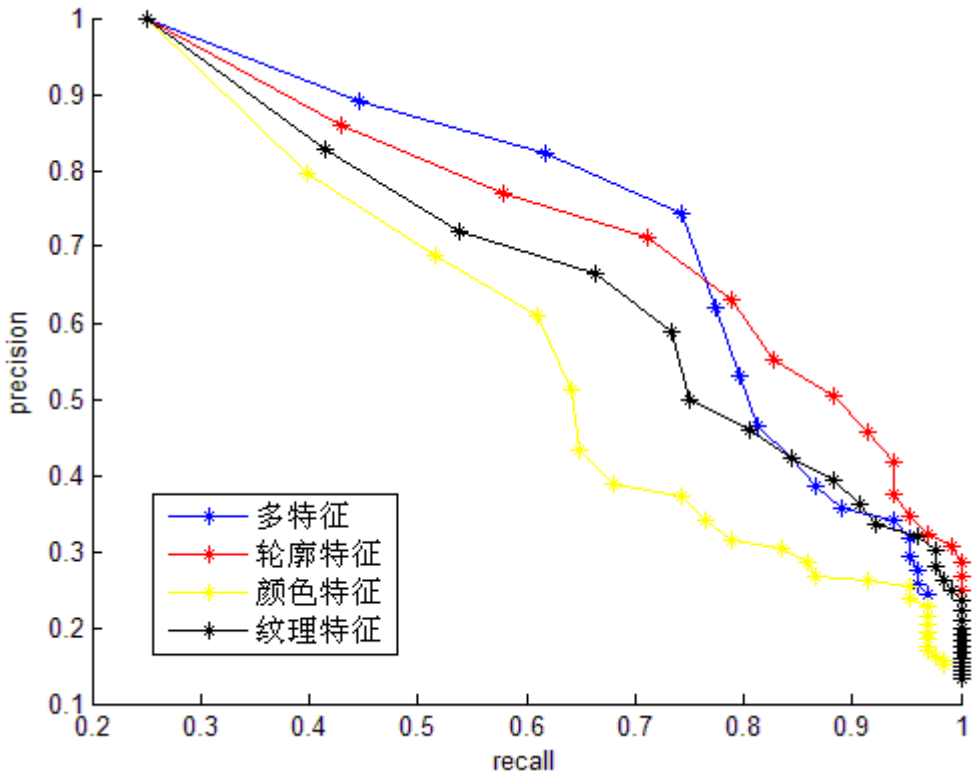


图3.7　训练集为6，*m*=4时的总体P-R曲线

c) 训练集为 7，*m*=6 时，区分每个类所需的特征如下：

表3.4 训练集为7，*m*=6时的特征集

| 类名 | 区分该类所需特征 |
|---|---|
| apple | 轮廓，颜色，纹理 |
| car | 轮廓 |
| cow | 轮廓，颜色，纹理 |
| cup | 轮廓，颜色 |
| dog | 轮廓，颜色，纹理 |
| house | 轮廓，颜色，纹理 |

| | |
|---|---|
| pear | 轮廓 |
| tomato | 轮廓，颜色，纹理 |

测试集总体 P-R 曲线如下：

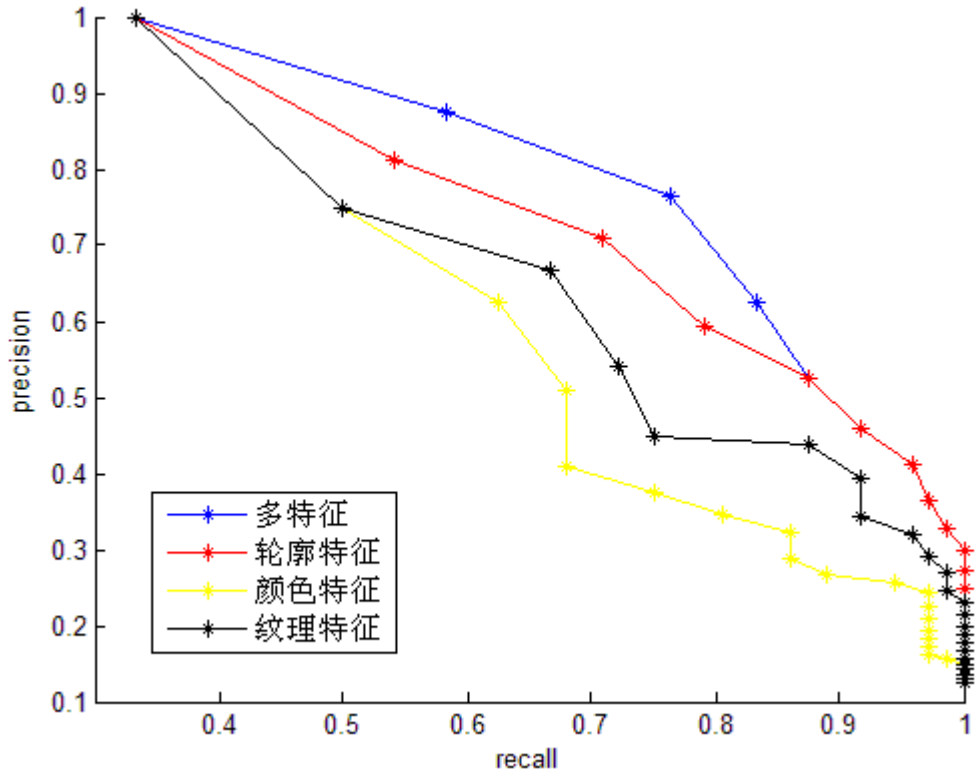


图3.8　训练集为7，*m*=6时的总体P-R曲线

由以上结果可见，多特征选择方法的检索结果明显优于单独使用任何一种单一特征时的结果。并且用训练集进行训练时，返回的相关对象越多，使用到的特征也就越多，因此多特征选择方法的优势也越明显。

## 3.7 特征简化方法

由于数据库中包含大量对象，每个对象又包含大量视图，因此检索时需要进行大规模的计算，特别是在进行距离度量的时候。假设每个对象包含 $n$ 个视图，那么距离矩阵的生成就要进行时间复杂度为 $O(n^2)$ 的计算，KM 算法又要进行时间

复杂度为$O(n^3)$的计算。如果能够在不对检索效果产生较大影响的前提下有效减小 $n$ 的值，那么完成检索所需的时间将能大大降低。

ETH-80 数据库中用 41 个视图来代表对象，Sphere60 数据库用 60 个视图来代表对象，我们不妨使用聚类的方法，来减少每个对象的视图数，从而简化对象的特征。

我们这里采用的是动态聚类方法中的 C-均值算法。我们把某个对象的每个视图看做多维空间中的一个点，视图的特征向量看做该点在多维空间中的坐标，然后对这多个点进行动态聚类。例如我们将某个对象的 $n$ 个视图（$n$ 个点）聚为 $m$ 类（$m<n$），然后我们可以在得到的每个类中，找到其中心点，我们以该中心点代替该类中的所有点，这样我们就将 $n$ 个点变为了 $m$ 个点。在检索的后续过程中，用这 $m$ 个点去代替原先的 $n$ 的点，便能有效缩减检索所需的时间。

对视图进行聚类的实际意义是，同一对象从比较接近的角度进行拍摄，可能会得到非常接近的视图，这些视图中包含大量的重复信息，却只有较少的新的信息，因此可以用其中的一幅视图来代替相似的多幅视图。

我们再次使用本章第 4 节性能评价部分使用过的混合数据库，特征采用轮廓特征，距离度量算法采用最优匹配，将每个对象的视图聚为 5、10、15、20 类，绘出总体 P-R 曲线，并与不聚类时的总体 P-R 曲线进行对比，如下图所示：

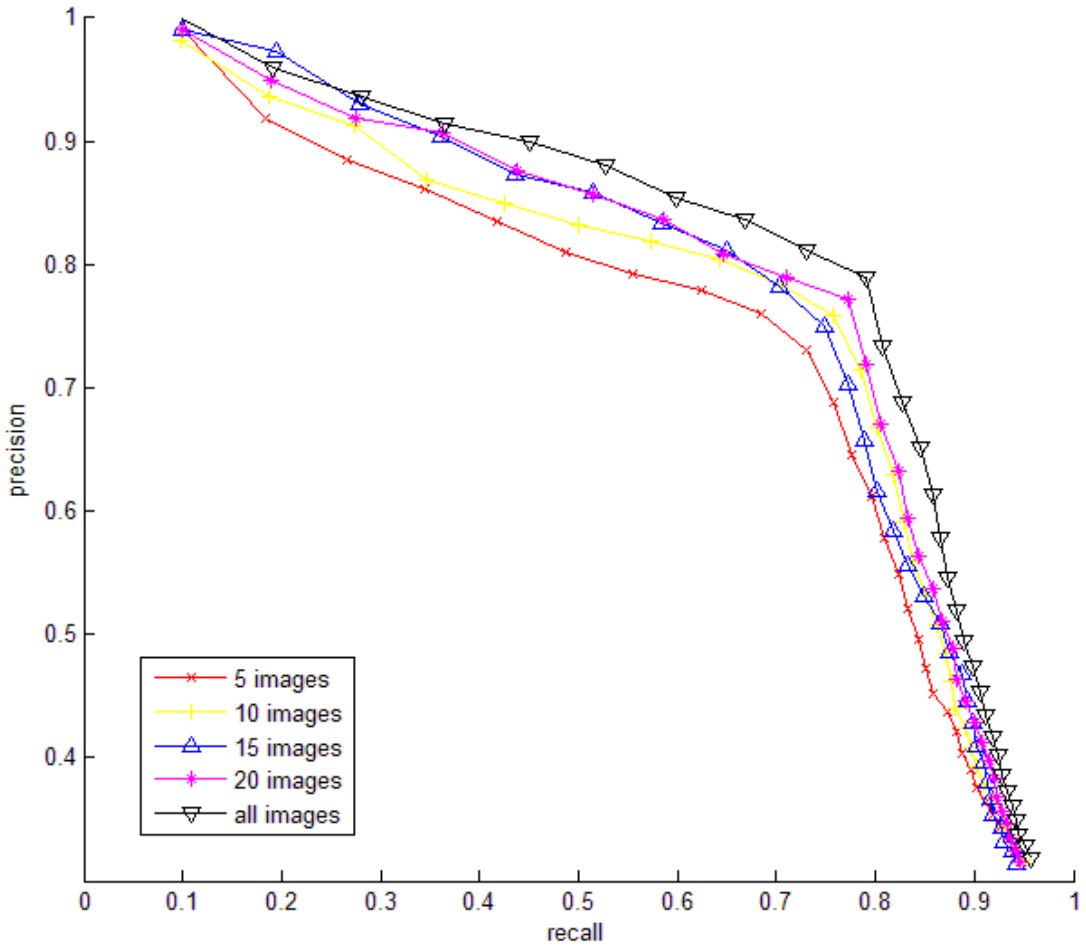


图3.9　不同聚类数下的总体P-R曲线

可见，聚类数为 15 或 20 时，检索的性能与不聚类时相差不大。假设聚类数为不聚类时的视图数的$\frac{1}{3}$，考虑到 KM 算法的时间复杂性为$O(n^3)$，则 KM 算法的执行时间便可以减小为不聚类时的$\frac{1}{27}$。可见通过聚类对特征进行简化是一种有效的方法。

## 3.8 聚类检索方法

之前我们所用的检索方法，都是将数据库中其他对象按照其与输入对象的相似度从高到低进行排序，返回一个对象序列，序列中靠前的对象被认为是较好的检索结果。

由于用户有时需要查询的只是与输入对象最相似的一些对象，而这些对象中哪些相似度更高可能并不重要，因此我们除了直接按照对象间的距离进行排序外，还可以有其他的方法来根据对象之间的距离确定与输入对象最相似的对象，我们这里就讨论通过分级聚类的方法来进行检索。

我们再次使用本章第 4 节性能评价部分使用过的混合数据库，特征采用轮廓特征。我们现在假设已通过最近邻距离、豪斯道夫距离和最优匹配三种方法计算出整个数据库中所有对象两两之间的距离，现在我们要利用这些距离来找出与输入对象相似度最高的对象集（我们不妨称之为相似集）。

利用之前直接根据相似度进行排序的方法，我们可以假定排在前 10 位的检索结果为输入对象的相似集。我们再定义这种检索机制下的准确率为相似集里与输入对象相关的对象在相似集中所占的比例。由于数据库中每个类有 10 个对象，因此这时的准确率也就等于之前定义的 R-准确率。

如果使用分级聚类，则将数据库中全部的 110 个对象依据两两之间的距离聚为 11 类（混合数据库中一共有 11 个类），输入对象位于哪个类，该类中的全部对象便组成输入对象的相似集。同样，我们定义这种检索机制下的准确率为相似集里与输入对象相关的对象在相似集中所占的比例。在分级聚类中，有了对象之间的两两距离之后，还需要定义两个聚类之间的距离度量。我们在这里采用如下 5 种距离进行实验：

1) 最近距离：聚类 A 中的点到聚类 B 中的点的距离的最小值；

2) 最远距离：聚类 A 中的点到聚类 B 中的点的距离的最大值；

3) 平均距离：聚类 A 中的点到聚类 B 中的点的距离的平均值；

4) 中心距离：聚类 A 的中心点到聚类 B 的中心点的距离；

5) 方差增量距离（Ward 距离）：聚类 A 和聚类 B 合并后的方差增量，定义如下式所示

$$d_{rs} = \frac{n_r n_s}{n_r + n_s} \cdot \left\| \bar{x}_r - \bar{x}_s \right\|^2$$

其中 $r$ 和 $s$ 表示两个聚类，$n_r$ 和 $n_s$ 表示两个聚类中的点的个数，$\bar{x}_r$ 和 $\bar{x}_s$ 是两个聚类的中心点，$\|\cdot\|$ 表示欧氏距离[4]。

我们给出在混合数据库中用不同的对象距离度量算法和聚类距离度量算法所得到的总体准确率如下表所示：

表3.5 新的检索机制下的准确率

| 对象距离度量算法 | 检索方式 | | 准确率 |
|---|---|---|---|
| 最近邻距离 | 按相似度排序检索 | | 0.6764 |
| | 分级聚类检索 | 最近距离 | 0.3545 |
| | | 最远距离 | 0.6545 |
| | | 平均距离 | 0.6091 |
| | | 中心距离 | 0.5364 |
| | | 方差增量距离 | 0.7636 |
| 豪斯道夫距离 | 按相似度排序检索 | | 0.6918 |
| | 分级聚类检索 | 最近距离 | 0.3727 |
| | | 最远距离 | 0.5455 |
| | | 平均距离 | 0.4545 |
| | | 中心距离 | 0.4545 |
| | | 方差增量距离 | 0.7818 |
| 最优匹配 | 按相似度排序检索 | | 0.7900 |
| | 分级聚类检索 | 最近距离 | 0.4545 |
| | | 最远距离 | 0.6273 |
| | | 平均距离 | 0.4545 |
| | | 中心距离 | 0.4545 |
| | | 方差增量距离 | 0.8818 |

从结果中可以看到，在上述的准确率定义下，按相似度进行排序检索的准确率高于多数分级聚类检索方法的准确率，但是如果分级聚类检索时采用方差增量距离，其准确率却明显高于按相似度排序的检索结果，这给我们带来一种新的检索思路，即利用分级聚类检索的结果来优化常规检索的结果。

例如我们要进行常规的检索，在进行检索之前，可以先根据方差增量距离对数据库中的全体对象进行分级聚类，这样便可以获得每个对象的相似集（事实上我们只需要进行一次聚类操作）。然后我们进行常规检索，在距离度量算法中，把每个属于输入对象的相似集的对象与输入对象之间的距离都减小（例如乘以一个小于 1 的数），那样我们得到的检索结果会比不进行聚类时要好。

再次利用本章第 4 节性能评价部分使用过的混合数据库，我们来对比利用分级聚类检索结果优化常规检索结果之前和之后的总体 P-R 曲线。特征采用轮廓特征，对象间的距离度量采用最优匹配，聚类间的距离度量采用方差增量距离，优化方法为将相似集中的对象之间的距离乘以 0.8，此时总体 P-R 曲线如下：

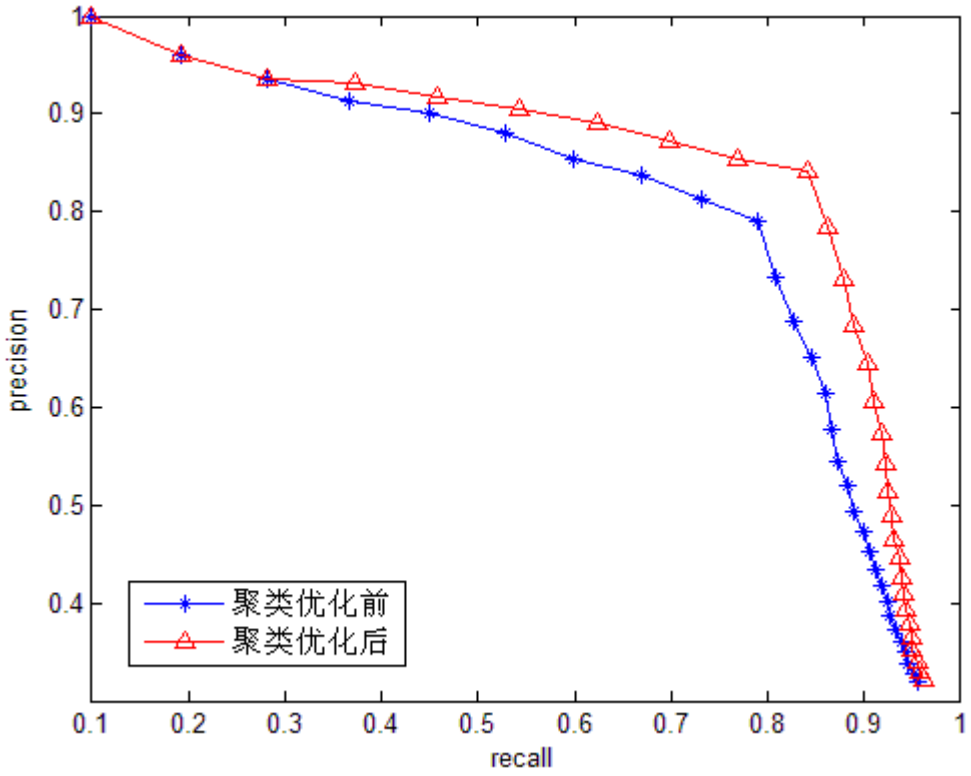


图3.10 聚类优化之前与之后的总体P-R曲线

由图可见，利用分级聚类检索的结果来优化常规检索的结果之后，总体 P-R 曲线大幅向右上方移动，也就是说，检索性能得到了明显提升，可见这种方法是非常有效的。

这种方法的有效性可以有如下直观的理解：如果直接按照数据库中的对象与待检索对象之间的距离来排序进行检索，那么只利用到了数据库中很少的距离信息；但如果先进行聚类，相当于是利用到了整个数据库中所有对象两两之间的距离，也即更充分地利用了整个数据库的内部结构信息，因此用这些信息来优化检索的结果，会使检索性能得到改善。

# 第4章 LF3DR 检索系统

## 4.1 基本功能

本章介绍我们在研究中所编写的检索系统软件，软件名称为 LF3DR，意为基于光场的 3D 检索系统（Light-Field-Based 3D Retrieval System）。该系统将光场数据库、特征数据库、距离度量算法和相关反馈进行集成，能够实现 3D 对象的在线的检索功能。以下介绍软件的安装方法、使用方法、扩展方法，并对程序中的类与函数进行说明。

## 4.2 安装方法

本软件的安装程序为 LF3DR Setup.exe 文件，可以在 Windows 平台下直接双击进行安装。安装文件图标如下所示：


图4.1 安装图标

双击图标后，按照提示即可将软件安装到指定路径。安装后，打开对应路径下的 LF3DR 文件夹，得到文件列表如下所示：


图4.2 LF3DR文件夹列表

其中 data 文件夹为程序运行所要用到的数据，jre1.6.0_01 文件夹为程序运行的 Java 环境，用户不要改动这两个文件夹中的内容。eth80 database 文件夹和 sphere60 database 文件夹分别是两个数据库，其中有包含光场数据库的 database 子文件夹，包含轮廓特征的 FDfeatures 子文件夹，包含颜色特征的 HSVfeatures

子文件夹，以及包含纹理特征的 TWTfeatures 子文件夹。LF3DR.exe 为程序的入口，双击即可打开程序。

## 4.3 使用方法

### 4.3.1 标签页概述

这里我们通过一些实例来介绍本软件的使用方法。

双击软件图标后，首先出现欢迎界面，等待程序加载，加载完成后便进入程序主页面。程序的主页面有三个标签页，如下所示：

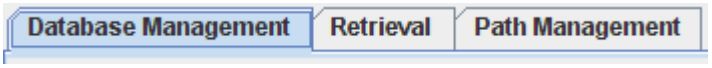


图4.3　标签页

三个标签页分别用于管理程序的三个不同功能。Database Management 标签页用于查看以及管理当前的数据库，Retrieval 标签页用于完成检索功能，Path Management 标签页用于管理数据库路径以及改变当前数据库。

### 4.3.2 Database Management 页面

Database Management 标签页的页面布局如下所示：


图4.4　Database Management标签页

页面左侧为数据库目录，目录右侧为数据库管理按钮，再往右的大部分区域为对象视图查看区。在数据库目录中，用户可以查看数据库中的类与对象，用户点击某个对象之后，对象视图查看区就会随机显示该对象的 24 个视图。

要删除某个类或者对象时，先选中该类或对象，然后点击相应的删除按钮，则该类或对象会从数据库目录中被删去。如果要将数据库中的类加入到目录中，则点击 Add Class 按钮，弹出添加类的对话框；如果要将数据库中的对象加入到目录中，则要在哪个类中加入对象就先点击该类，然后点击 Add Object 按钮，弹出添加对象的对话框。假如我们要添加一个目录中没有的类 dog，添加类的对话框如下图所示：

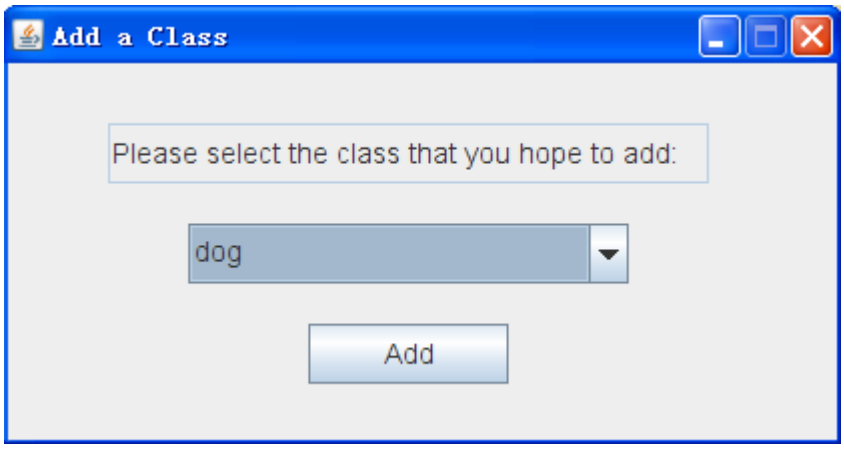


图4.5　添加类对话框

选中 dog 后单击 Add 按钮，则程序回到主页面，此时 dog 类成功添加到了目录中。添加对象的操作与此类似。

### 4.3.3 Retrieval 页面

Retrieval 标签页的页面布局如下图所示：

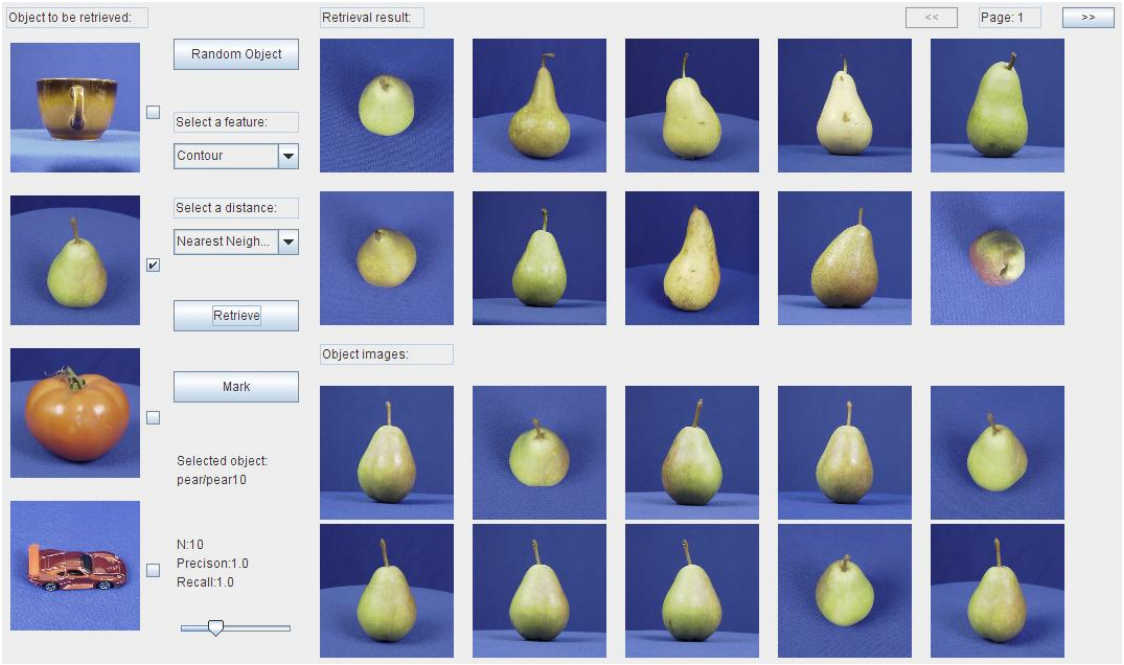


图4.6　Retrieval标签页

页面最左侧为待查询对象显示区，其右侧为操作区，操作区右上方为检索结果显示区，右下方为对象视图显示区。

在操作区点击 Random Object 按钮，则待查询对象显示区会随机出现数据库中的对象，单击其中的对象视图或其右方的选择框，可以选中某个对象作为待查询对象，此时对象视图显示区会显示该对象的 10 个代表视图，操作区 Mark 按钮下方会显示当前所选择的对象名称以及其所属类的名称。

用户还可以在操作区通过下拉框选择用于检索的特征和距离度量算法，如下图所示：

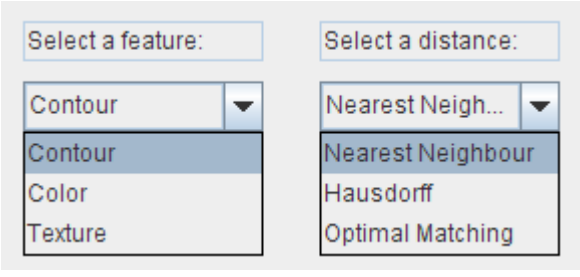


图4.7　特征选择与距离度量算法选择下拉列表

选择好待查询对象、特征和距离度量算法后，点击 Retrieval 按钮，就可以进行检索了。检索完成后，结果显示区会按序显示检索返回结果，第一排从左至右为相似度最高的 5 个结果按相似度从高到低排序，同理，第二排从左至右为相似

度次高的 5 个结果按相似度从高到低排序。结果显示区的右上角有翻页按钮，可以用于查看排在更后面的结果，但是检索总共只返回前 30 个结果，所以一共只有 3 页可供查看。检索结束后，还可以在操作区的底部查看当前检索的准确率和召回率，如下图所示：

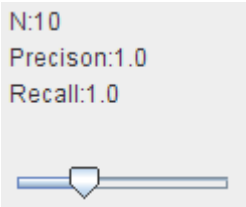


图4.8　准确率与召回率查看栏

通过拖动滑块，可以改变 N 的值，此时准确率和召回率会相应改变。

检索完成之后，用户可以对检索结果进行按类标注。点击操作区的 Mark 按钮，会弹出标注页面如下图所示：

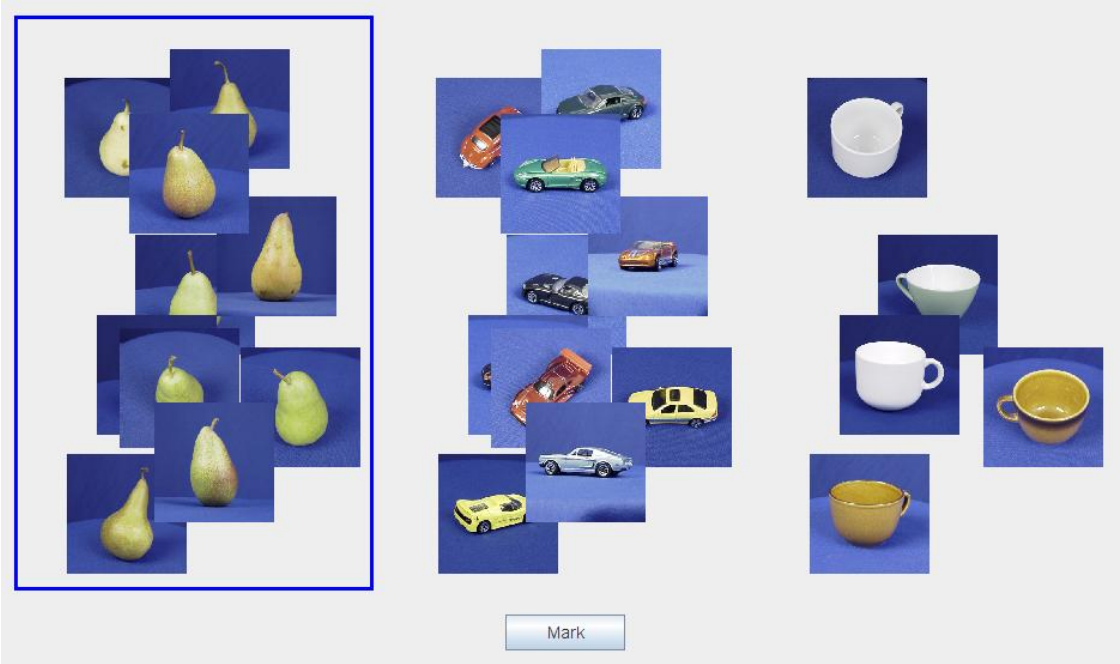


图4.9　标注页面

检索结果中出现最多的 3 个类会显示在标注页面中，用户通过点击该类的对象视图可以选择该类，然后点击 Mark 按钮进行标注。点击 Mark 按钮后，程序会回到主页面，如果标注正确，此时再次进行检索，返回结果会更优。

### 4.3.4 Path Management 页面

Path Management 标签页的页面布局如下图所示：

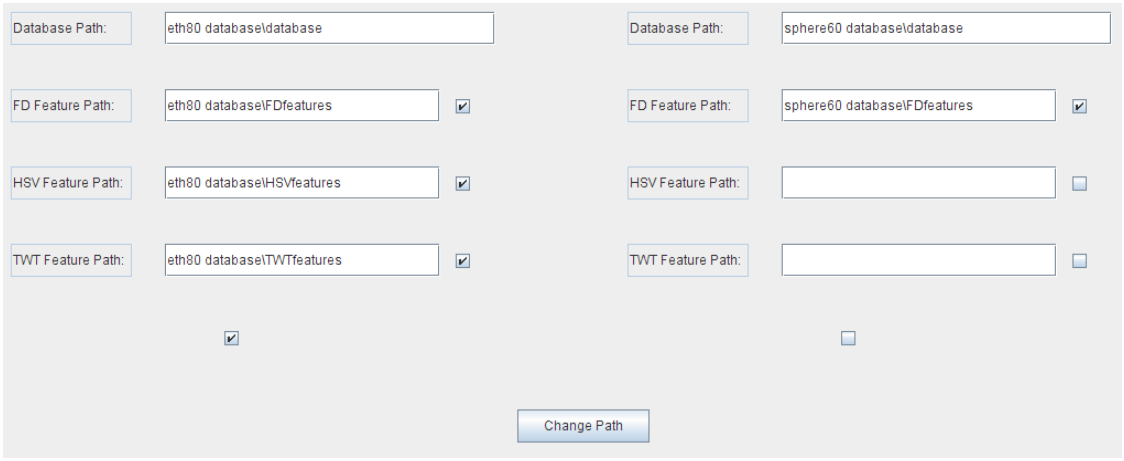


图4.10 Path Management标签页

页面中显示的是两个默认的数据库的路径以及其特征文件的路径，用户可以通过改变这些路径来配置新的数据库，在 Path Management 标签页进行修改后单击 Change Path 按钮就可以完成配置，具体做法见下一节扩展方法部分。

## 4.4 扩展方法

### 4.4.1 数据库的扩展

程序本身设置了数据库扩充的功能，因此扩展数据库不需要修改源代码即可实现。

新的光场数据库应当按照 database->class->object->image 的层次配置好，其特征文件按照 feature database->class->object features 的层次配置好。光场数据库的图片格式限定为 jpg 格式或者 png 格式，特征文件限定为 txt 格式，且文件名应该与光场数据库中对应的 object 的名称一致。

例如 ETH-80 数据库的文件夹结构如下所示：

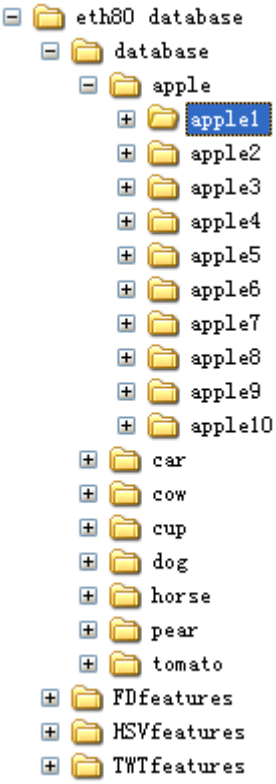


图4.11 ETH-80数据库的文件夹结构

apple1 为 apple 类中的一个对象，其内部文件列表如下图所示：

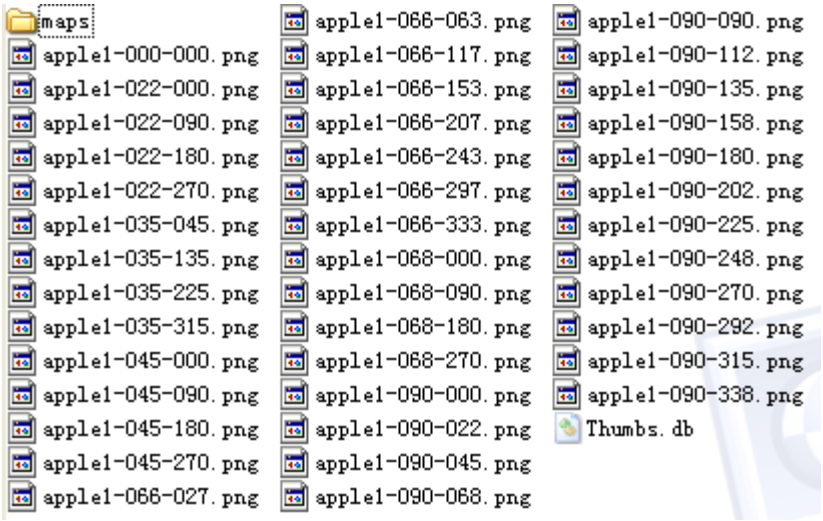


图4.12 对象内部文件列表

每一个 png 文件都是一个视图。

特征数据库（例如轮廓特征）的文件夹结构如下所示：

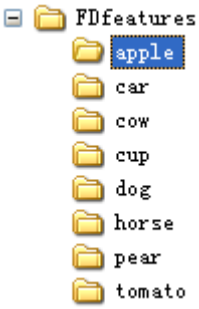


图4.13 特征数据库的文件夹结构

其中 apple 是一个类，其内部文件如下所示：

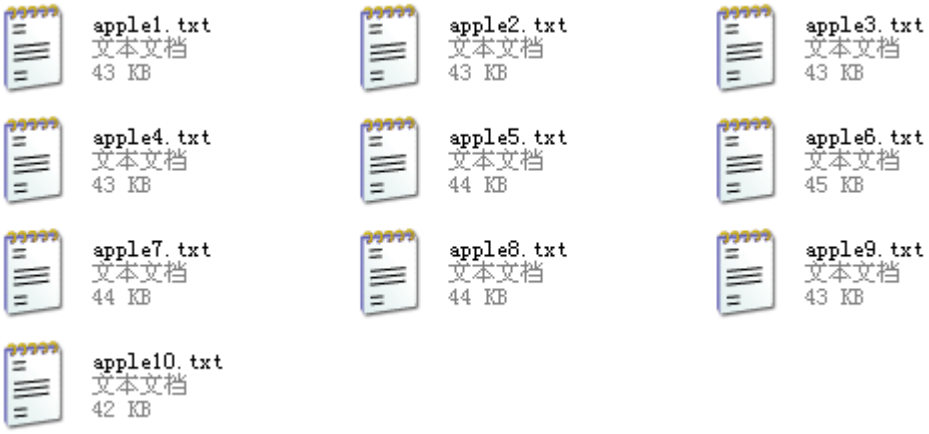


图4.14 类的特征文件夹结构

每一个 txt 文件是一个对象的特征文件。

特征文件需要按照一定的格式来生成。特征文件的数据以空格（ASCII 码为 10）相分隔，第一个数据为该对象的视图个数，之后分别为每个视图的特征。例如一个视图的轮廓特征为 120 个实数，颜色特征为 256 个实数，纹理特征为 104 个实数。一个对象的特征文件格式简要表示为：

视图数 N 视图 1 特征 视图 2 特征 …… 视图 N 特征

配置好光场数据库和特征数据库之后，打开 Path Management 标签页，就可以将程序和新的数据库相连接了。

在左右两个设置栏中任选一个，填写对应的光场数据库路径和特征数据库路径，在对应的特征后面打钩表示该特征数据库存在。最后在总栏下打钩表示选择该设置，并单击 Change Path 按钮，此时该数据库设置将成为程序的当前设置和默认启动设置。

### 4.4.2 特征的扩展

特征的扩展比较复杂，需要修改程序多处源代码。

1、首先按照类似 FD 特征、HSV 特征或者 TWT 特征的格式生成新的特征数据库。

2、打开 Interface 类，打开 Retrieval 标签页，在 getFeatureSelectBox 函数中，添加语句 featureSelectBox.addItem("特征名")，则检索界面中增加了新的特征选项。

3、打开 ObjectDistance 类，在 objectDistance 函数中，增加一个 else if 模块，按照其他特征的格式，设置路径和特征维数即可。例如颜色特征的模块代码如下：

```
else if(selectedFeature.equals("Color"))
{
    object1FilePath=index.HSVFeaturePath+"\\"
        +object1.className+"\\"+object1.objectName+".txt";
    object2FilePath=index.HSVFeaturePath+"\\"
        +object2.className+"\\"+object2.objectName+".txt";
    featureSize=256;
}
```

4、为了能够在 Path Management 标签页下配置该特征，还需要在 Interface 类中增加新的组件。打开 Interface 类的 Path Management 标签页，类似其他特征，增加提示文本栏，特征文件路径文本栏，以及选择框，并为这些组件设置相应的属性和动作。

5、此外，由于加入新的特征，索引文件也会发生相应的变化，应在 IndexObject 类，IndexGenerate 类和 IndexInitialize 类中做相应的修改。IndexObject 类中，仿照其他特征，增加该特征数据库是否存在以及特征数据库路径这两个相应的属性，可参考已有特征的代码如下：

```
public boolean hasFDFeatures;
public String FDFeaturePath;
public boolean hasHSVFeatures;
public String HSVFeaturePath;
public boolean hasTWTFeatures;
public String TWTFeaturePath;
```

然后在 IndexGenerate 类和 IndexInitialize 类中，对新加的属性进行赋值，具体参见其他特征的代码。

### 4.4.3 距离度量算法的扩展

距离度量算法的扩展也需要修改程序多处源代码。

1、打开 Interface 类，打开 Retrieval 标签页，在 getDistanceSelectBox 函数中，添加语句 distanceSelectBox.addItem("算法名")，则检索界面中增加了新的距离度量算法选项。

2、打开 MatrixDistance 类，在 matrixDistance 函数中，增加一个 else if 模块，类似其他距离度量算法，加入新的算法即可。matrixDistance 函数中定义了变量 result 作为计算结果，函数的最后即返回 result 的值，而函数的参数为距离度量算法名称 selectedDistance 和距离矩阵 matrix（例如 matrix 的第 i 行第 j 列元素表示第一个对象的第 i 个视图和第二个对象的第 j 个视图的特征向量的欧氏距离）。例如 Nearest Neighbour 算法的模块代码如下：

```
if(selectedDistance.equals("Nearest Neighbour"))
{
    for(int i=0;i<matrix.length;i++)
    {
        for(int j=0;j<matrix[0].length;j++)
        {
            result=Math.min(result,matrix[i][j]);
        }
    }
}
```

由于 KM 算法的代码较复杂，因此 KM 算法的实现单独由一个名为 KM 的类来完成，在 MatrixDistance 类 matrixDistance 函数的相应 else if 模块中只是调用了 KM 类。

## 4.5 类与函数说明

### 4.5.1 工程概述

本软件用 Java 语言编写而成，开发工具为 Eclipse 3.2，运行环境为 jre1.6，工程结构如下所示：

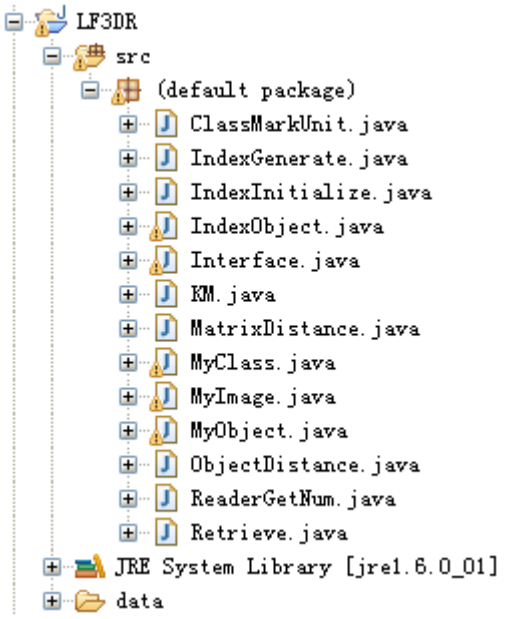


图4.15 软件工程结构

工程中总共有 13 个类，每个类的基本属性如下表所示：

表4.1 类基本属性

| 类名称 | 含main函数否 | 功能类型 | 主要函数 | 功能概要 |
|---|---|---|---|---|
| Interface | 是 | 界面 | main | 界面的布局以及各组件的功能 |
| ClassMarkUnit | 否 | 数据结构 | 构造函数 | 定义标注单元 |
| IndexObject | 否 | 数据结构 | 构造函数, writeToFile | 定义索引文件 |
| MyClass | 否 | 数据结构 | 构造函数 | 定义数据库中的类 |
| MyObject | 否 | 数据结构 | 构造函数 | 定义数据库中的对象 |
| MyImage | 否 | 数据结构 | 构造函数 | 定义数据库中的视图 |
| Retrieve | 否 | 算法 | retrieve | 检索功能的实现 |
| ObjectDistance | 否 | 算法 | objectDistance | 对象之间的距离计算 |
| MatrixDistance | 否 | 算法 | matrixDistance | 距离度量算法的实现 |
| KM | 否 | 算法 | sum, km, init | KM算法的实现 |
| IndexInitialize | 是 | 算法 | main | 初始化索引文件 |
| IndexGenerate | 否 | 算法 | indexGenerate | 生成程序使用的索引文件 |
| ReaderGetNum | 否 | 算法 | getDouble, getInt | 从特征文件中读取整数或实数 |

其中 Interface 是可视类，可以用 Visual Editor 打开，其余类都只能用文本编辑器打开。IndexInitialize 类用来生成初始的索引文件，是软件开发者使用的类，软件使用者在使用过程中不会涉及到此类。

以下重点描述程序的数据结构，以及一些重要的函数的实现。

### 4.5.2 数据结构部分

程序与数据库最直接的联系通过 IndexObject 类实现。IndexObject 类是整个数据库的索引，可以通过 writeToFile 函数进行保存。IndexObject 类中包含了光场数

据库的文件夹路径，特征数据库的类型与文件夹路径等基本信息。此外，IndexObject 有一个名为 classList 的链表结构，链表的每一个节点是一个 MyClass 类。

MyClass 类用来表示数据库中的类，除了有一个 className 的属性用来表示类的名称之外，还有一个名为 objectList 的链表结构，链表的每一个节点是一个 MyObject 类。

MyObject 类用来表示数据库中的对象，除了有 objectName 和 className 属性分别表示对象的名称和上级类的名称之外，还有一个名为 imageList 的链表结构，链表的每一个节点是一个 MyImage 类。

MyImage 类用来表示数据库中的视图，只有一个属性 imagename，用来表示视图的名称。

数据结构如下图所示：

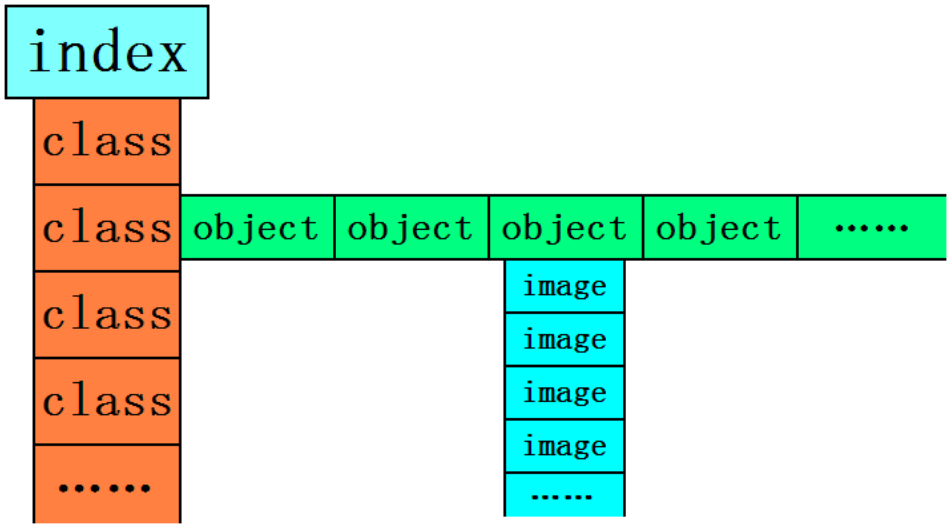


图4.16 数据结构示意图

以上为基本的数据结构。此外，还有用于标注的类 ClassMarkUnit，其属性有被标注的对象名，被标注的对象的上级类名，以及为其标注的类名。标注功能在 Retrieval 类的 retrieval 函数中用到。Interface 类中定义了一个名为 classMarkList 的链表，用来记录所有的标注信息。

### 4.5.3 函数部分

(1)Retrieval.retrieval 函数：参数有 5 个，被检索的对象 objectRetrievedSelected，索引文件 index，用于检索的特征名 selectedFeature，用于检索的距离度量算法

selectedDistance，用于标注的信息列表 classMarkList。函数返回一个 MyObject 数组，为有序的检索结果，排在越前面的结果与待检索对象相似度越高。默认的是返回前 30 个结果。Retrieval.retrieval 函数需要调用 ObjectDistance 类的 objectDistance 函数。

(2)ObjectDistance.objectDistance 函数：参数有 5 个，用于检索的特征名 selectedFeature，用于检索的距离度量算法 selectedDistance,第一个对象 object1，第二个对象 object2，以及索引文件 index。函数返回一个实数作为两个对象之间的距离。函数的基本功能是生成两个对象的距离矩阵，然后调用 MatrixDistance 类的 matrixDistance 方法计算相应的距离。

(3)MatrixDistance.matrixDistance 函数：参数只有 2 个，用于检索的距离度量算法 selectedDistance 和距离矩阵 matrix。函数返回一个实数作为计算结果。

以上三个函数的调用关系如下图所示：

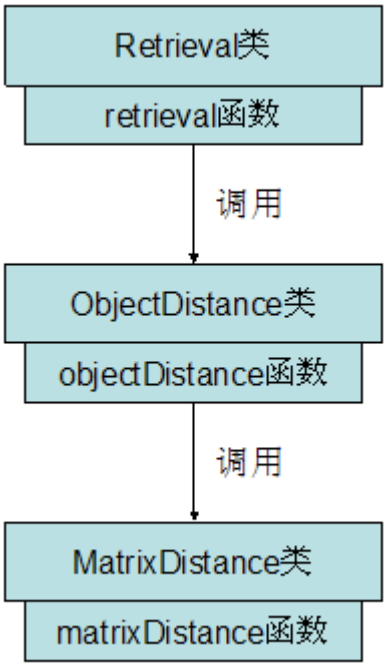


图4.17 函数调用关系示意图

# 第5章 结 论

本文从数据库搭建、检索方法和检索系统的实现三个方面对基于光场的 3D 对象检索进行了研究。

在数据库搭建方面，本文建立了一套基于 3D 模型的光场数据库 Sphere60，提取出了其中对象的轮廓特征，并集成到了 3D 检索系统之中。该数据库中既有自然界的物体也有人造物体，并且同一类别中对象间的差异性较大，适合用于 3D 对象检索的研究。

在检索方法方面，本文通过傅里叶描述符、HSV 空间颜色直方图和树形结构小波变换三种特征，来研究形状、颜色和纹理三种底层视觉特征在 3D 对象检索中的检索效果。为了更好地利用不同特征对不同类别的区分能力，本文实现了一种多特征选择算法，能够通过一定的训练集确定出每个类别需要哪些特征的组合才能与其他特征互相区分开，再综合利用这些特征来更好地区分每一个类别。本文实现了最近邻距离、豪斯道夫距离、改进的豪斯道夫距离和最优匹配四种对象间的距离度量算法，并在一定规模的混合数据库中讨论了这四种算法的检索效果，发现改进的豪斯道夫距离和最优匹配这两种距离度量算法有较好的检索性能。本文在检索系统中实现了一种短期的相关反馈方法，用户可以按类对检索结果进行标注，实验发现这种标注能够很好地提高检索的准确率和召回率。本文讨论了通过动态聚类将对象的视图数压缩以减小检索计算复杂度的方法，我们发现，在我们的混合数据库中，将对象的视图数通过动态聚类压缩为原视图数的三分之一左右，能够很好地减小检索过程中的计算复杂度，并且对检索性能的影响并不大。本文利用分级聚类来实现一种新的检索机制，发现在使用方差增量距离（Ward 距离）来作为聚类之间的距离度量方式时，能够得到很高的准确率，因此用这种分级聚类检索的结果来优化常规检索的结果，在实验中有效地提升了检索性能。

在检索系统的实现方面，本文搭建了一个用户友好的基于光场的 3D 对象检索系统。通过该系统，用户可以根据需求选择特征和距离度量算法来进行 3D 对象的检索，并能通过按类标注来提高检索准确率。此外，用户还可以自行管理数据库，对数据库中的类或对象进行添加和删除操作，并能够对该系统进行扩展，向其中加入新的数据库，新的特征，或者新的距离度量方法。

对于本课题，未来的改进方向可以从以下一些方面进行：

(1) 可以进一步扩充数据库：建立更大规模的实物光场数据库和 3D 模型光场数据库，使得数据库中包含更加丰富的类别，这样的数据库不仅能够更好的用于测试我们的算法，还能够更好的满足用户查询 3D 对象的需求；

(2) 可以扩展特征的种类：目前我们只用到了 3 种特征，分别属于形状特征、颜色特征和纹理特征这三类，但每一类中其实还包含了很多其他的特征，例如区域特征，RGB 空间下的颜色直方图特征等等，这些特征非常值得研究，并可以集成到检索系统中；

(3) 可以研究其他的距离度量算法：两个点集的距离度量算法非常多样，本文只对其中四种进行了探讨，在今后的研究中，可以尝试使用更多的距离度量算法，并将其检索性能与这四种距离进行比较；

(4) 可以建立更多样更完善的相关反馈机制：本文的系统只实现了短期的按类标注反馈机制，今后可以在系统中加入长期的相关反馈，并提供多种标注方式，例如用户对返回结果评分等等；

(5) 可以实现一个真正实用的网络在线检索系统：本文实现的系统其特征提取部分还是离线的，并且所有操作都是在本地完成的，因此只适合用于研究，并不具有真正的实用性，今后可以开发网络版的检索系统，将特征提取部分做成在线实现，并将 3D 模型与光场中的 3D 对象相联系，使得用户能够通过输入 3D 对象来查询网络数据库中的相似对象并能够下载检索结果。

# 插图索引

# 表格索引

# 参考文献


[1] 蒋薇. 基于内容的图像检索系统中基于学习的检索机制的研究[D]. 北京: 清华大学自动化系, 2005.

[2] 路瑶. 光场数据库的搭建与检索示范系统的实现[D]. 北京: 清华大学自动化系, 2007.

[3] 刘晓雯, 章毓晋, 谭华春. 基于 Hausdorf 距离的相似性和对称性度量及在人脸定位中的应用[J]. 信号处理: 2008, 24(1): 118-121.

[4] 吴彪, 许洪国, 张文会. 基于 Ward 聚类法的道路交通事故多元统计分析[J]. 黑龙江工程学院学报: 自然科学版, 2009, 23(3): 4-6.

[5] 巩艳华, 朱爱红, 代凌云. 基于颜色直方图的颜色特征提取[J]. 福建电脑: 2007, 5: 96-97.

[6] 戴琼海, 路瑶, 尔桂花. 一种三维对象检索方法: 中国发明专利, CN101599077[P].

[7] B. Bustos, D. Keim, D. Saupe, and T. Schreck. Content-Based 3d Object Retrieval. *IEEE Computer Graphics & Applications*, 2007.

[8] D. Y. Chen, M. Ouhyoung, X. P. Tian, and Y. T. Shen. On visual similarity based 3D model retrieval. *Computer Graphics Forum*, 223–232, 2003.

[9] K. Chakrabarti, M. Ortega, K. Porkaew, and S. Mehrotra. Query refinement in similarity retrieval systems. *IEEE Data Engineering Bulletin*, 24(3): 3 – 13, September 2001.

[10] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The Amsterdam library of object images. *Int. J. Comput. Vision*, 61(1):103–112, 2005.

[11] A. Jaimes, M. Christel, S. Gilles, R. Sarukkai, and W. Y. Ma. Multimedia information retrieval: what is it, and why isn't anyone using it? *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, 3–8, 2005.

[12] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. *CVPR'03*, 2003.

[13] B. Bustos. Feature-Based Similarity Search in 3D Object Databases. *ACM Computing Surveys*, 37(4): 345-387, 2005.

[14] H. Sundar. Skeleton Based Shape Matching and Retrieval. *Proc. Shape Modeling International 2003 (SMI 03)*, 130-142, 2003.

[15] D. Y. Chen and M. Ouhyoung. A 3D Object Retrieval System Based on Multi-Resolution Reeb Graph. *Proc. of Computer Graphics Workshop*, 16, June 2002.

[16] D. S. Zhang and G. Lu. An Integrated Approach to Shape Based Image Retrieval. *Proc. of 5th Asian Conference on Computer Vision (ACCV)*, 652-657, Jan. 2002.

# 致 谢

感谢清华大学自动化系对我的培养，感谢导师戴琼海教授和尔桂花副教授对我的细心指导，感谢路瑶师姐在我学习和研究过程中提供的大量帮助，感谢实验室中其他师兄师姐以及同学们的支持。

# 声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：王泉 日 期：2010年7月1日