基于 MPEG-7 的图像颜色和纹理特征提取

计算机 0310　韦祎

指导老师：王天江教授

摘　要

MPEG-7 多媒体内容描述接口标准是 ISO 组织于 2001 年颁布，旨在为音频和视频等多媒体信息提供统一内容描述接口的标准。该标准中定义了一系列的图像视觉特征描述子，方便了基于内容的图像检索。本文首先简要介绍了 MPEG-7 标准中关于视觉特征描述子的规定，接着描述了如何实现基于 MPEG-7 标准提取静态图像的颜色和纹理特征的系统。目标系统能够生成 MPEG-7 标准中的主颜色描述子，颜色结构描述子，颜色布局描述子和边缘直方图描述子等四种描述子，并能将这些描述子以 XML 文档方式存放。同时系统还能进行基于样例图像视觉特征的本地图像检索。最后文章对本系统的实验结果进行了总结，并提出了改进的设想。

关键词：MPEG-7 标准，图像检索，视觉特征，描述子

Abstract

The MPEG-7 Multimedia Content Description Interface standard was released by the ISO in 2001, aiming at providing unified interfaces for audio and visual information description. This standard defines a series of image visual feature descriptors which can be applied in content-based multimedia retrieval. This paper starts with a brief introduction to the definition of visual feature descriptors set in MPEG-7 standard. Next we described how to build up a MPEG-7 based system to extract color and texture features from still images. This system can generate four descriptors defined in MPEG-7. They are dominant color descriptor, color structure descriptor, color layout descriptor and edge histogram descriptor. The content of these descriptors are saved in XML documents. Example image based local image retrieval can also be proceeded by this system. At the end of this paper, we analyzed the experiment results and gave some advices about future work.

Keywords: MPEG-7 Standard, Image Retrieval, Visual Feature, Descriptors

目 录

1 绪 论

1.1 课题的来源和国内外发展概况

自上个世纪九十年代以来，随着计算机的处理能力、存储能力和数据传输能力的进一步提高，人们可以接触到的音频、图像和视频等多媒体信息的数量也快速增加。与七、八十年代那些枯燥的文本信息相比，图形化的操作系统、带有丰富表现元素的网页、数字音乐、电影等多媒体信息大大丰富了用户的感官体验。不仅如此，大量的多媒体信息还被广泛应用于各行各业中，如数字图书馆、远程医疗、地理信息系统、网上购物等等。然而在以前，多媒体信息是存放在一个个文件中并由文件系统进行管理的，用户要使用这些信息或者要查找特定的信息时，只能逐个打开文件浏览。当文件数量大大增加后，这种检索方法的效率就会急剧降低。

为了解决这个问题，最初人们提出的方案是在多媒体信息中加入能与之相关的文本形式的注解，如音乐的作者、流派、电影的导演、主演、主要场景、图像中的主要景物等等，然后使用基于文本的数据库管理系统进行多媒体信息检索。这种注解方法虽然能在一定程度上以简洁的方式表达多媒体内容，但却存在着几个问题：第一，注解主要是以手工方式添加，效率较低，因此不适合海量的多媒体文件处理；第二，因为注解是人工添加的，而不同的人对于不同的多媒体内容可能有着不同的主观认识，所以文本注解受人的主观因素影响很大；第三，这些文本注解与组成多媒体内容的数据间并没有直接的联系，而且没有一套统一的标准来规范注解的格式，因此在进行多媒体信息检索时效率和准确率都较低。

理想的多媒体信息描述应该是能由计算机直接从多媒体数据中自动提取出来并按统一的格式存放。这些描述应与多媒体数据直接相关，表达方式应比较简洁，而且在提取和检索时计算复杂度比较合理。在过去十几年中，许多的公司、大学和实验室在这一方面做了大量的研究工作。典型的原型系统包括IBM公司的QBIC系统，MIT的PhotoBook系统、哥伦比亚大学的VisualSEEK系统以及中国科学院的MIRES系统等[1]。这些原型系统主要是对图像的颜色、纹理、形状和空间关系等底层图像特征建立特征矢量以表征图像的语义信息并根据此进行图像检索的。为了制定一个统一的多媒体内容描述标准，国际标准化组织ISO自 1998 年底开始了相关工作并在 2001 年 11 月正式颁布了ISO/IEC 15938 多媒体内容描述接口（Multimedia Content Description Interface）

标准，即人们后来熟知的MPEG-7 标准。MPEG-7 的目标是指定一组描述不同多媒体信息的标准描述符。这些描述符与信息内容相关以便能用来快速和有效地查询各种多媒体信息。[2]

MPEG-7 目前考虑的媒体主要包括：静止图像、序列图像、计算机图形、3D 模型、动画、语言、声音等。MPEG-7 的描述数据可以和相关的媒体结合在一起，也可以单独存放。在后一种情况下，需要能将 MPEG-7 描述与相应媒体信息关联起来的链接。MPEG-7 的描述与被描述的多媒体内容是否被编码和如何储存并没有关系。例如视觉信号仍可以用目前的各种编码方案（如 JPEG、MPEG-1、MPEG-2、MPEG-4）编码。

MPEG-7 标准是一个框架性的标准，多媒体特征的提取算法、多媒体内容的检索引擎以及用户如何使用 MPEG-7 描述数据都不包含在该标准中。MPEG-7 主要考虑的是与特征提取模块和检索引擎模块之间的接口，以及由不同系统产生的多媒体内容描述之间的互操作性。MPEG-7 被设计为既可以借助现有的特征提取技术，又可以兼容未来的技术发展。标准中没有规定统一的特征提取方法和检索引擎，这就使得业界各个厂商的产品可以相互竞争，产生最好的结果。

MPEG-7 目前讨论的应用可以分为三大类：第一类是索引和检索类应用，这也是MPEG-7 制定计划启动的最主要原因。MPEG-7 的最初构想就是让人们可以像检索文字信息那样方便的检索多媒体信息。事实上在互联网上已经有一些基于文字描述的多媒体搜索引擎，但由于缺乏统一的描述标准，其应用范围受到了限制。第一类应用也被成为“拉”（Pull）应用。本课题所要实现的系统正是对 MPEG-7 标准在静态图像检索上的“拉”应用。第二类应用是选择和过滤类应用，也被成为“推”（Push）应用，它与第一类应用相对，是指适合广播发送类型的信息服务应用；第三类是与传统的面向媒体内容不同的，与 MPEG-7 中要定义的“元”（Meta）内容相关的专业应用，如生物医学应用、遥感应用、地理信息系统等。

1.2 课题的目的和意义

本课题要处理的多媒体信息为静态图像。目的是实现一个能根据 MPEG-7 标准对图像进行视觉特征的提取和能进行基于内容检索的实验系统，以此来检验 MPEG-7 标准在多媒体检索方面的有效性。

1.3 课题要解决的问题

本课题要实现的系统的功能包括两个部分，第一部分为基于 MPEG-7 标准提取静态图像的视觉特征，要实现这部分需要解决以下问题：

（1）MPEG-7 标准中定义了几类图像视觉描述符

（2）需要从图像中提取哪几类视觉描述符

（3）每类视觉描述符的提取算法是什么

（4）视觉描述符以什么样的格式存放

（5）图像的描述符与图像之间如何关联起来

系统的第二部分功能为基于内容的本地图像检索，要实现这部分需要解决以下问题：

（1）如何获取指定路径下的所有图像文件

（2）如何从存放描述符内容的文件中读出有意义的描述符的值

（3）如何比较不同的图像以确定图像之间的相似程度

（4）如何实现 MPEG-7 标准中各个描述符的相似度算法

其中，描述符的提取、存放和匹配为实现该系统需要解决的核心问题。预计达到的功能目标是由用户提供样例图像和指定检索路径，系统接受输入后分析指定路径下的所有图像文件并逐个生成符合 MPEG-7 标准的视觉描述符，然后根据描述符相似度算法计算不同图像的描述符的相似程度并筛选出与样例图像在某一视觉特征上（如颜色或纹理特征）较为相似的若干幅结果图像，并输出显示。

2　MPEG-7 标准的相关定义

2.1　MPEG-7 标准中定义的视觉描述子（descriptor）

MPEG-7 标准视觉部分中定义了四种颜色特征描述子和三种纹理特征描述子，并支持六种颜色空间[3][4][5]。下面将简要介绍MPEG-7 标准中的颜色空间和各个视觉特征描述子：

2.1.1 颜色空间

MPEG-7 标准中支持的六种颜色空间为 RGB, HSV, YCrCb, HMMD, 单色（monochrome）和参考 RGB 的线性变换矩阵，以下介绍前五种颜色空间：

（1）RGB 颜色空间

RGB 颜色空间是最典型最常用的面向硬设备的颜色空间。RGB 颜色空间与人的视觉系统结构密切相连。根据人眼结构，所有的颜色都可看作是三个基本颜色—红（Red），绿（Green）和蓝（Blue）—的不同组合。在很多图像文件中所使用的颜色空间也都是RGB，但是，由于 RGB 空间在检索时的效率并不高，因此需要先进行颜色空间的转换，将 RGB 转换为其它的颜色空间再进行特征提取工作。

（2）HSV 颜色空间

与RGB颜色空间不同，HSV颜色空间是一种面向人类视觉感知的颜色空间。使用该颜色空间的目的是为了提供对颜色的直观表示并模拟人类感觉和处理颜色的方法[10]。RGB空间到HSV空间的变换是一个非线性但可逆的变换。色调（Hue）表达的是主要光谱元素—颜色的纯粹形式，如绿、红和黄。向纯色中加入白色会改变这种颜色，白色越少，颜色越饱和，这与饱和度（Saturation）相对应。值（Value）与颜色的亮度相对应。该空间的坐标系统呈圆柱形，而且经常用一个由倒立的六边金字塔定义的子空间来表示。金字塔顶对应V=1，中间代表白色。色调由与纵轴的夹角来量度，红色对应 0 度。饱和度S在金字塔中间为 0，逐渐变化直到金字塔表面为 1。

（3）YCrCb 颜色空间

YCrCb 颜色空间被应用于欧洲电视系统颜色编码中。其中“Y”表示明亮度（Luminance），也就是灰度值；“亮度”是通过 RGB 输入信号来创建的，方法是将 RGB 信号的特定部分叠加到一起。“色度”则定义了颜色的两个方面—色调与饱和度，分别用 Cr 和 Cb 来表示。其中，Cr 反映了 GB 输入信号红色部分与 RGB 信号亮度值之间

的差异。而 Cb 反映的是 RGB 输入信号蓝色部分与 RGB 信号亮度值之同的差异。

（4）HMMD 颜色空间

HMMD 是 MPEG-7 标准中定义的新的颜色空间。该空间中色调（Hue）的定义与 HSV 空间中的定义相同。最大值（Max）和最小值（Min）分别指 R，G，B 值中的最大值和最小值。差值（Diff）分量定义为最大值和最小值的差。这四个分量中只用三个就足以描述 HMMD 空间。在 MPEG-7 图像检索核心实验中，人们观察到 HMMD 颜色空间十分有效且相比于 HSV 颜色空间更为合适。

（5）单色（monochrome）空间

单色空间中存放的是亮度值，与 YCrCb 颜色空间中的 Y 分量值相等。在进行纹理特征提取时，彩色图像将被转换到单色空间中之后再进行处理。

2.1.2 颜色特征描述子

颜色在所有的视觉特征中可以说是最有表现力的。在过去的十几年中人们对将颜色应用于图像检索做了广泛的研究。MPEG-7 标准中定义了四种颜色特征描述子：

（1）可变颜色描述子（Scalable Color Descriptor）

可变颜色描述子是定义在 HSV 颜色空间上的颜色描述子。该描述子。该描述子反映了图像的全局颜色分布情况。它由一个直方图表达，并使用哈尔变换（Haar Transform）进行编码。由于直方图的级数可在 16，32，64…256 间取值，因此该描述子是“可变的”。

（2）主颜色描述子（Dominant Color Descriptor）

主颜色描述子的作用是描述那些在一幅图像或任意区域中占主导地位的颜色信息。在给定的图像区域，颜色信息将由少数具有代表性的色彩（即主颜色）来表示。虽然一幅图像中可能包含的颜色数量可以有很多，但通过特定的颜色聚类算法，可以得到那些出现次数最多的颜色。MPEG-7 标准规定，一幅图像至多可以提取八种主颜色。主颜色描述子的提取可以在 MPEG-7 所支持的各个彩色空间上进行，本系统中所使用的为 HSV 颜色空间。

（3）颜色结构描述子（Color Structure Descriptor）

颜色结构描述子定义在 HMMD 颜色空间上，它不仅能够反映图像的色彩信息（类似于颜色直方图），而且描绘了图像的局部颜色结构特征。在提取该描述子时，需要使用一定大小的结构单元来扫描整幅图像并进行直方图累加。根据不同的需要，可以将 HMMD 颜色空间非均匀量化为不同的级数，从而改变直方图的级数。

（4）颜色布局描述子（Color Layout Descriptor）

颜色布局描述子定义在 YCrCb 颜色空间上，并使用了离散余弦变换（DCT）和锯齿扫描来进行特征提取。它是在主导颜色描述子的基础之上产生的，可以反映图像中任意指定区域的颜色分布情况。利用图像的区域颜色分布特征可以实现基于结构（Sketch Based）的图像检索、基于图像索引的过滤等功能。颜色布局描述子不仅具有简练的表达形式，在性能上也同样的优异

2.1.3 纹理特征描述子

如颜色一样，纹理也是一个用于图像检索和获取的有用的低层描述子。MPEG-7 标准中定义了三种纹理特征描述子[7][8]。

（1）纹理浏览描述子（Texture Browsing Descriptor）

该描述子从类似于人类感知的角度对纹理的方向性（Directionality）、规则性（Regularity）和粗糙程度（Coarseness）进行了描述，适用于图像的浏览和根据纹理粗糙程度进行的分类。用户可以使用单一的或者组合的分量来对图像数据库进行查找。在进行描述子提取时，既可以使用自动方式，通过带通滤波进行计算，也可以手工对描述子各分量进行规定。

（2）均匀纹理描述子（Homogeneous Texture Descriptor）

均匀纹理描述子为基于相似度的图像匹配提供了定量的描绘。计算该描述子时，首先使用一系列对方向和灰度敏感的滤波器对图像滤波，再在频域中计算滤波输出的平均差和标准差。人们已经证明该描述子是健壮、有效且易于计算的。该描述子的计算在频域中使用 Gabor 变换完成，这样可以大大降低计算的复杂度。

（3）边缘直方图描述子（Edge Histogram Descriptor）

边缘直方图描述子获取边缘的空间分布，这与颜色布局描述子的任务有些相似。边缘分布在图像-图像匹配中是一个有用的纹理标记，即使基础纹理并不是均匀的。MPEG-7 中定义了五种边缘：垂直，水平，45 度，135 度和无方向边缘。

3 系统设计方案

在设计系统的时候，从计算和编码的复杂度出发，我选择了三种颜色特征描述子和一种纹理特征描述子作为本系统要提取的图像特征。三种颜色描述子是：主颜色描述子，颜色布局描述子和颜色结构描述子。三种纹理描述子中选择了边缘直方图描述子进行实现。这四种描述子的提取算法和特征匹配算法如下：

3.1 主颜色描述子

MPEG-7 标准规定每个主颜色描述子所包含的颜色个数最多可以为 8 种，但是现在一般计算机上存储的图片都是 24 位真彩色的，可表达的颜色数量达到了 2 的 24 次幂，即超过一千六百万种颜色，因此，提取主颜色描述子需要进行的第一步工作是将图像进行颜色空间的转换（如果原图像不是定义在 HSV 空间上则需要这一步），之后再将颜色的个数通过算法减少为一个较小的个数。

目前大部分的图像文件所使用的颜色空间都为 RGB 空间，从 RGB 空间到 HSV 颜色的转换是一个非线性但是可逆的转换，首先将颜色的 R、G、B 分量由[0，255]区间归一化到[0，1]区间上，之后进行如下变换：

```
value=max(R, G, B)
min=min(R, G, B)
if( Max ==0 ) then
    Saturation = 0
else
    Saturation = (Max-Min)/Max
if( Max == Min ) Hue is undefined (achromatic color)
otherwise:
if( Max == R && G > B )
    Hue = 60*(G-B)/(Max-Min)
else if( Max == R && G < B )
    Hue = 360 + 60*(G-B)/(Max-Min)
else if( G == Max )
    Hue = 60*(2.0 + (B-R)/(Max-Min))
```

else

Hue = 60*(4.0 + (R-G)/(Max-Min))

转换后完成后，Hue∈[0，360°]，Saturation∈[0，1]，Value∈[0，1]。

进行颜色空间转换后，颜色的数量还是很庞大，因此需要进行颜色量化。在进行量化级数（即需要量化为几种颜色）选择，需要考虑量化复杂度和颜色表示精度的问题。因为量化的本质就是将一组较为接近的颜色用一种颜色来表达，如果量化颜色数选的太小，那么可能不相似的颜色也被量化到了同一种颜色中，这将影响检索的效率。本系统中采用了符合人类视觉感知的 166 维颜色量化[6]。包括 4 级灰度和 162 级颜色。对于相应的HSV颜色c=（h,s,v），获取量化颜色q=（0,1,2…165）的公式如下：

$$q=\begin{cases}0, & v\le 0.1\\ g(h,s,v), & s<0.1\text{且}v>0.1\\ f(h,s,v), & \text{其它}\end{cases} \tag{3.1}$$

其中，g(h,s,v)的定义如下：

$$g(h,s,v)=\begin{cases}1, & s<0.1\text{且}0.1<v\le 0.1\\ 2, & s<0.1\text{且}0.4<v\le 0.7\\ 3, & s<0.1\text{且}0.7<v\le 1.0\end{cases} \tag{3.2}$$

f(h,s,v)则均匀划分剩余的 HSV 颜色空间，其中 h∈[0,360°]，s∈[0.1,1]，v∈(0.1,1]。色调 h 以 20°为间隔分为 18 份，饱和度 s 和亮度 v 以 0.3 为间隔分别分为 3 份。这样量化后整幅图像颜色可以用 166 维来表示。（18 色调×3 饱和度×3 亮度+4 灰度=166）

颜色量化工作完成后，即可对图像进行直方图累加，以得到每种量化颜色所包含的像素点个数，接着再对直方图进行归一化处理，就可以获得各个量化颜色所占图像的百分比。将这些百分比按升序排列并进行累积取值就可以获得至多 8 种主颜色了。具体方法为，从百分比最大的主颜色开始，如果该主颜色所占的百分比大于 0.6，则停止取值，否则取下一个主颜色的百分比并将其与前一个主颜色百分比相加，直至累积的百分比大于 0.6 或者主颜色个数等于 8。获取主颜色后，如果所有主颜色的百分比之和小于 1，应对百分比进行转换，将每个颜色的百分比除以所有百分比的总和，这样做的目的是使得各个主颜色描述子中的百分比之和均为 1，在计算描述子间的相似度时不会产生误差。在处理完成之后，即可得到一个主颜色描述子

$D=\{(C_i,P_i)\ i=1,2\ldots N,\ N\le 8\}$。

主颜色描述子的相似度计算方法：

在系统实现中采用了计算二次型距离的主颜色匹配方法，对于 HSV 空间上的两种颜色$C_i=(h_i,s_i,v_i)$和$C_j=(h_j,s_j,v_j)$。它们的欧几里德距离为：

$$d_{i,j}=[(v_i-v_j)^2+(s_i\cos h_i-s_j\cos h_j)^2+(s_i\sin h_i-s_j\sin h_j)^2]^{\frac{1}{2}} \tag{3.3}$$

由于进行颜色量化后色调分量 h 的范围变成了[0，17]，饱和度 s 和亮度 v 的范围了变成了[0，3]，因此不能将量化后的颜色分量直接带入式（2-3）进行计算，我采用的方法是对颜色分量进行变换后再带入计算，变换方式如下：

$$\begin{cases} H=h\times 20^{\circ} \\ S=s\times 0.3+0.1 \\ V=v\times 0.3+0.1 \end{cases} \tag{3.4}$$

转换后色调 H 的范围变成了[0，340]，饱和度 S 和亮度 V 的范围变成了[0.1，1]，这样就可以带入式（2-3）计算距离了。

对于两个主颜色描述子$F_1=\{(C_i,P_i),i=1,2\ldots N\}$和$F_2=\{(C_j,P_j),\ j=1,2\ldots M\}$，相似度可由以下公式得到：

$$D(F_1,F_2)=\sum_{i=1}^{N}P_i^2+\sum_{j=1}^{M}P_j^2-\sum_{i=1}^{N}\sum_{j=1}^{M}2a_{ij}P_iP_j \tag{3.5}$$

其中：

$$a_{ij}=1-d_{ij}/d_{\max} \tag{3.6}$$

上式中的d_{ij}为两个颜色间的距离，$d_{\max}$为颜色间的最大距离。通过计算两种相反的颜色（0°,1,0）和（180°,1,1）可得到$d_{\max}=\sqrt{5}$。

3.2 颜色结构描述子

该描述子定义在 HMMD 颜色空间上。HMMD 颜色空间是 MPEG-7 标准提出的一种新的颜色空间，它由色调分量 Hue，最大值分量 Max，最小值分量 Min，差值分量 Diff 和和值分量 Sum 组成。从 RGB 空间到 HMMD 空间的转换如下：

*Hue*分量转换方法与*HSV*空间相同

$$\begin{cases} Max = \max(R,G,B) \\ Min = \min(R,G,B) \\ Sum = (Max + Min)/2 \\ Diff = Max - Min \end{cases} \tag{3.7}$$

与主颜色描述子类似的是，颜色结构描述子在生成时也要进行颜色量化，量化过程的第一步是将 HMMD 颜色空间分为 5 个子空间。划分是根据 Diff 分量的值来决定的，Diff 分量的取值范围为[0，255]，子空间划分的分割点为 6，20，60 和 110。接着，对每个颜色子空间沿着 Hue 和 Sum 轴进行非均匀量化，本系统选择了 256 级量化方法，每个子空间的量化级数如下：

表 3.1　HMMD 子空间划分

子空间编号	Hue 分量级数	Sum 分量级数
0	1	32
1	4	8
2	16	4
3	16	4
4	16	4

颜色结构描述子生成过程的第二步是要使用 8×8 的结构单元来扫描整个图像并进行直方图累加。在累加时需要为每一个直方图级设置一个标志当前结构单元内是否存在该种颜色的指示变量，对于同一个结构单元内多次出现的同一种颜色，只进行一次直方图累加。在扫描完整幅图像后，对获得的直方图使用该图像中包含的结构单元的个数进行归一化。一幅图像中包含的结构单元个数为（图像宽度-7）×（图像高度-7）。归一化完毕后，即可得到颜色结构描述子。

本系统采用了欧氏距离来度量两个颜色结构描述子之间的相似度，设有两个颜色结

构描述子 $H_D(i)$ 和 $H_Q(i)$ $(i=1,2\ldots256)$，它们之间的距离可由以下公式进行计算：

$$S(Q,D)=\sqrt{\sum_{i=1}^{256}[H_Q(i)-H_D(i)]^2} \tag{3.8}$$

3.3 颜色布局描述子

MPEG-7 标准中定义的颜色布局描述子定义在 YCrCb 颜色空间上，从传统的 RGB 空间转化为 YCrCb 空间的方法如下：

$$\begin{cases} Y=0.299\times R+0.587\times G+0.114\times B \\ Cr=0.500\times R-0.419\times G-0.081\times B \\ Cb=-0.169\times R-0.331\times G+0.500\times B \end{cases} \tag{3.9}$$

在系统实现时，还需要将计算出的 Y，Cr，Cb 三个分量转化为整数。

完成颜色空间的转换后，第二步工作是将图像分为 64（8×8）块子图。对于每一块子图，计算子图中包含的所有像素的颜色的平均值作为该块子图的代表颜色值，这样就可以得到一个 8×8 的颜色矩阵。对这个矩阵进行 8×8 离散余弦变换（DCT）可得到一系列变换因子 $\{DY,DCr,DCb\}$。MPEG-7 标准中推荐的，也是本系统采用的 DCT 变换过程如下（以伪代码形式表示）：

```
int i, j, k;
double s;
double tmp[64];
for( i=0; i<8; i++ )
{
    for( j=0; j<8; j++ )
    {
        s = 0.0;
        for( k=0; k<8; k++ )
        {
            s += m[j][k]*d[i][k];
```

```
            }
            tmp[i][j] = s;
        }
    }
    for( j=0; j<8; j++ )
    {
        for( i=0; i<8; i++ )
        {
            s = 0.0;
            for( k=0; k<8; k++ )
            {
                s += m[i][k]*tmp[k][j];
            }
            c[i][j] = (int)trunc(s+0.499999);
        }
    }
```

其中 trunc 函数是对浮点数进行截断，矩阵 m[8][8]的定义如下：

```
    for( i=0; i<8; i++ )
    {
        for( j=0; j<8; j++ )
        {
            double s=(i==0||j==0) ? sqrt(0.125) : 0.5;
            m[i][j] = s*cos((M_PI/8.0)*i*(j+0.5));
        }
    }
```

完成 DCT 变换后，再对得到的因子进行锯齿扫描和量化，选出其中的低频因子，即可构成颜色布局描述子。锯齿扫描的顺序如下：

表 3.2　锯齿扫描顺序表

	i							
j	1	2	6	7	15	16	28	29
	3	5	8	14	17	27	30	43
	4	9	13	18	26	31	42	44
	10	12	19	25	32	41	45	54
	11	20	24	33	40	46	53	55
	21	23	34	39	47	52	56	61
	22	35	38	48	51	57	60	62
	36	37	49	50	58	59	63	64

MPEG-7 标准中规定该描述子所选择的 DCT 因子的个数是可变的，取值范围是[1,3,6,10,15,21,28,64]。在考虑了描述子简洁性和检索有效性的平衡后，本系统选择了 6 个 Y 因子以及 Cr 和 Cb 各 3 个因子构成颜色布局描述子。

颜色描述子的相似度计算可通过以下公式进行，设有两个颜色布局描述子 $L_Q=\{DY,DCr,DCb\}$ 和 $L_D=\{DY,DCr',DCb'\}$,则距离为：

$$D=\sqrt{\sum_{i=1}^{6}(DY_i-DY_i')^2}+\sqrt{\sum_{j=1}^{3}(DCr_j-DCr'_j)^2}+\sqrt{\sum_{k=1}^{3}(DCb_k-DCb_k')^2} \quad (3.10)$$

3.4 边缘直方图描述子

该描述子可以捕获图像中边缘的分布情况。由于纹理特征一般是对图像的灰度进行分析，因此如果分析的对象是彩色图像，首先需要进行由彩色到灰度的转换。转换公式如下：

$$G=0.299\times R+0.587\times G+0.114\times B \quad (3.11)$$

转换完成之后，按以下步骤生成边缘直方图描述子：

（1）将原始图像分割为 16（4×4）个子图像，编号为（0，0）到（3，3），每个子图像都对应有 5 种边缘，因此生成的直方图一共有 80 级；

（2）将每个子图像进一步分割为若干图块，在本系统的设计中将分割数目取 256。

这就要求输入图像的分辨率至少要是 128×128；

（3）对于每一个图块，将其视为一个分辨率为 2×2，即只有 4 个“像素”的图像，每个“像素”的灰度由该“像素”所对应的区域所有像素灰度的平均值。在每个图块上应用 MPEG-7 标准规定的 5 种边缘滤波器，具体形式如下：

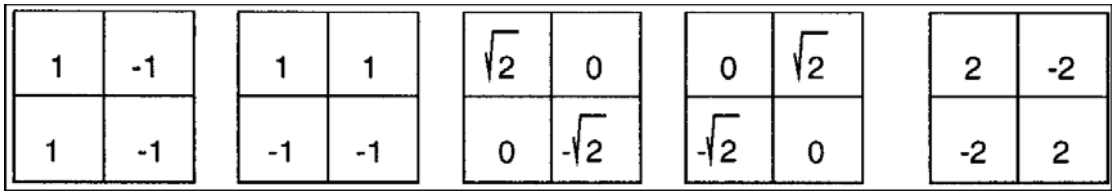

图 3.1　5 种边缘检测滤波器

（4）若 5 个滤波结果中的最大值大于设定的阈值，则认定该图块中含有边缘，并对相应的直方图级进行累加；

（5）对所有的子图完成直方图统计后，用一个子图包含的图块数对直方图进行归一化，然后使用下列表格对归一化直方图进行非均匀量化，即可得到边缘直方图描述子：

表 3.3　垂直边缘量化表

直方条长度	原始值
0	0.010867
1	0.057915
2	0.099526
3	0.144849
4	0.195573
5	0.260504
6	0.358031
7	0.530128

表 3.4　水平边缘量化表

直方条长度	原始值
0	0.012266
1	0.069934
2	0.125879
3	0.182307
4	0.243396
5	0.314563

6	0.411728
7	0.564319

表 3.5　45 度边缘量化表

直方条长度	原始值
0	0.004193
1	0.025852
2	0.046860
3	0.068519
4	0.093286
5	0.123490
6	0.161505
7	0.228960

表 3.6　135 度边缘量化表

直方条长度	原始值
0	0.004174
1	0.025924
2	0.046232
3	0.067163
4	0.089655
5	0.115391
6	0.151904
7	0.217745

表 3.7　无方向边缘量化表

直方条长度	原始值
0	0.006778
1	0.051667
2	0.108650
3	0.166257
4	0.224226
5	0.285691
6	0.356375
7	0.450972

两个边缘直方图描述子 $E_1(i)$ 和 $E_2(i)$ 之间的相似度可用以下公式进行计算：

$$D(E_1, E_2) = \sum_{i=1}^{80} \left| E_1(i) - E_2(i) \right| \tag{2.12}$$

4　系统的实现

4.1 系统用例分析

通过分析，可得到本系统的用例图如下（使用 Borland Together 2005 生成）：

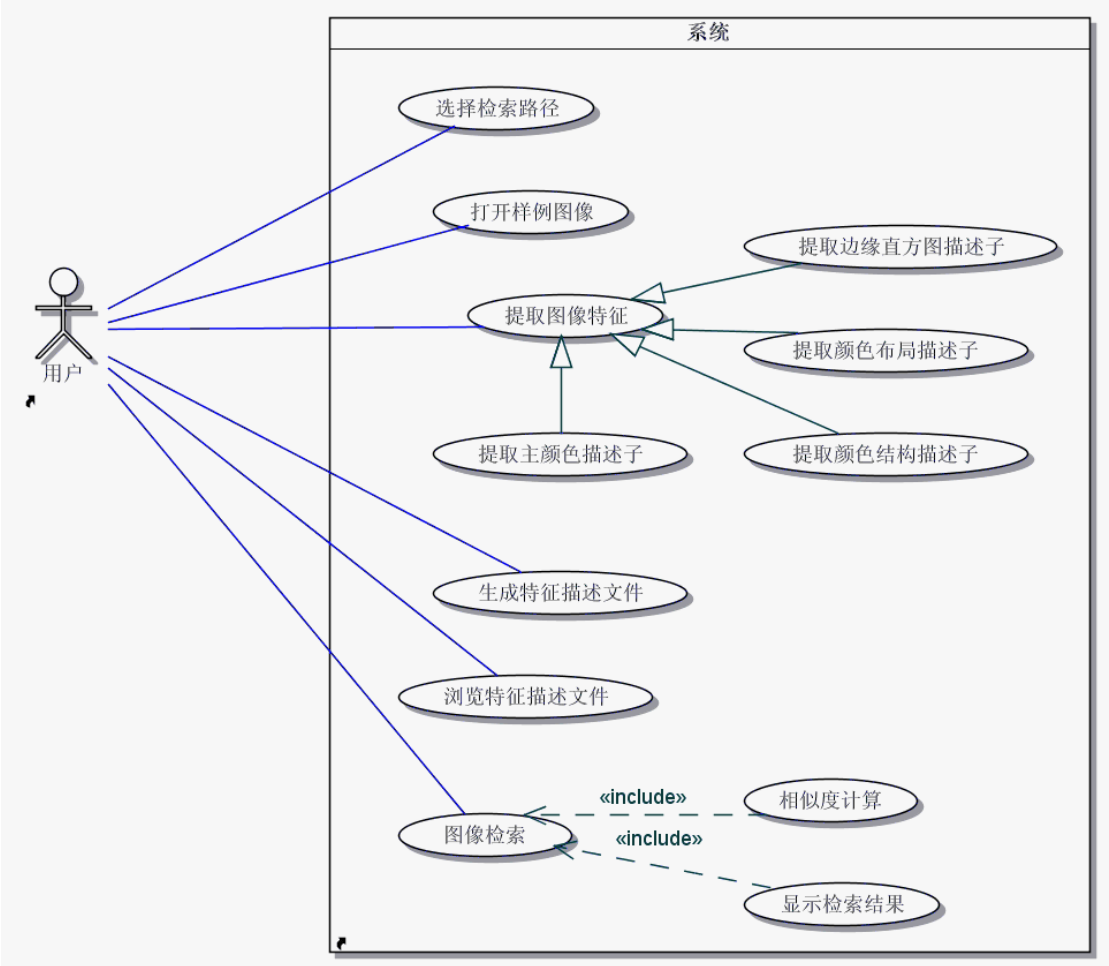

图 4.1　系统用例图

各个用例的描述如下：

（1）选择检索路径：用户可以在进行检索前指定检索路径文件夹。系统会将指定好的检索路径作为当前路径在图像检索和存放特征描述文件中使用。用户也可以在系统运行过程中改变检索路径。

（2）打开样例图像：系统打开用户选定的样例图像。

（3）提取图像特征：根据特征提取算法从图像中提取用户选定的图象特征，系统实现中提供了四种描述子作为选择。

（4）生成特征描述文件：对检索路径下的所有图像（包括子文件夹中的图像文件）进行特征提取并将提取出的描述子内容存放在 XML 文档中。

（5）浏览特征描述文件：打开相应的 XML 文档。

（6）图像检索：该用例功能的第一步是从特征文件中提取出描述子内容，并与样例图像的描述子内容进行匹配，再对匹配结果排序；第二步是根据排好序的匹配结果显示检索结果图像。

4.2 系统概要设计

4.2.1 系统总体流程

本系统的主要功能流程如下：

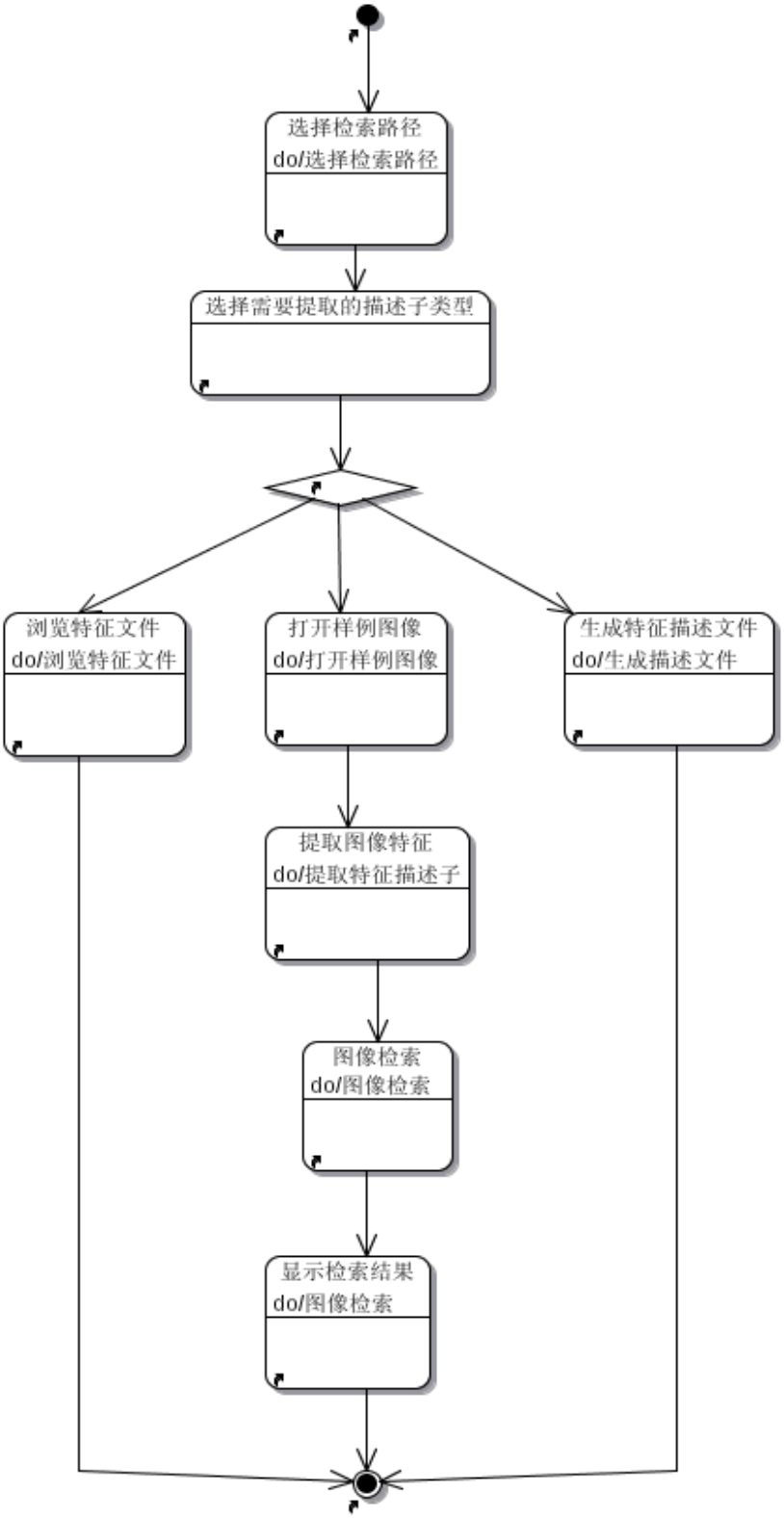

图 4.2　系统主要工作流程图

从图上可以看出，一次完整的检索过程分为以下几步：

（1）选择检索路径；

（2）打开样例图像；

（3）选择需要提取的描述子类型；

（4）从样例图像中提取特征描述子；

（5）从特征描述文件中提取检索路径下的图像文件的特征内容；

（6）根据相似度算法计算样例图像和被检索图像间的距离；

（7）将相似度距离排序并输出检索结果图像。

系统进行图像检索时并不再次打开图像文件进行特征提取，而是读取已存放在检索路径下的特征描述文件。因此系统必须提供生成特征描述文件的功能。在用户更换检索路径后，新路径下可能不存在特征描述文件，此时应该禁用图像检索功能，知道生成特征描述文件。

4.2.2 系统功能设计

根据用例图进行分析，可知系统应具有的主要功能为：

（1）记录检索路径；

（2）更改检索路径；

（3）读取并显示图像，包括样例图像和检索结果图像；

（4）提取特征描述子的值并显示；

（5）将描述子的内容写入 XML 文件；

（6）查找检索路径下所有的图像文件，系统支持的图像格式为.bmp 和.jpg；

（7）解析 XML 文件并读出描述子的值；

（8）根据不同描述子的特征匹配算法进行相似度计算。

4.2.3 开发工具和软硬件平台选择和配置

本系统使用了 Borland Together Designer（以下简称 Together）进行了用例获取、概要设计和详细设计。Borland Together Designer 是一个跨平台的 UML 建模解决方案，可为创建行业标准统一建模语言 Unified Modeling Language(UML)图表提供高级建模环境。可以使架构师、业务分析师和开发人员在无需任何成本的情况下，以更简便的方式获得建模利益。在系统分析过程中，我使用 Together 进行了用例图绘制，获取了

系统应具有的功能；在概要设计过程中，我使用 Together 绘制了系统的活动图、状态图等，进一步明确了系统的工作流程和控制流程；在详细设计阶段，我使用 Together 设计具体的类以及类的字段和函数，并可以对类的实例化过程进行说明。整个设计过程结束后，还可以使用 Together 自动的生成 HTML 格式或者 RTF 格式的设计文档，大大减轻了开发的劳动量。

在使用 Together 完成设计工作后，我选择了 Microsoft Visual Studio.NET 2003 作为系统开发的环境。开发语言为 C++.NET。Visual Studio.NET 集成开发环境为开发者提供了非常方便的编码、调试和程序发布等功能。C++.NET 语言在继承了 C++强大的功能的基础上，还添加进了.NET 关于托管代码的新概念，并将许多常用的控件以封装好的类的形式供开发人员使用。这种所见即所得的开发方式比起 MFC 也有了较大的进步，使得开发人员能够更快捷更简单的进行 Windows 窗体应用程序的开发。

由于需要对图像进行诸如颜色空间变换等的处理，因此我还选择了 Intel OpenCV 开源函数库加入系统之中以方便对图像进行操作。OpenCV 是由英特尔公司开发的一款免费的开源计算机视觉库。它由一系列 C 函数和少量 C++ 类构成，实现了图像处理和计算机视觉方面的很多通用算法。OpenCV 提供了超过 300 个能够跨平台使用的中、高层 API，并且不依赖于其它外部库。它提供了一系列与图像处理相关的数据结构和操作函数，方便了开发人员。要使用 OpenCV，需要在安装以后在 Visual Studio.NET 环境下进行配置，将 OpenCV 中相关的头文件和库文件的路径包含在 Visual Studio.NET 的查找路径范围中，这样就可以在程序中使用 include 宏来包含相应的头文件并在程序中调用相关函数了。本系统所使用的头文件有 cxcore.h 和 cv.h。

运行本系统需要有.NET 1.1 Framework 环境的支持。推荐的操作系统环境为 Windows XP sp2。

4.3 系统详细设计

4.3.1 功能模块设计

（1）选择检索路径模块

模块功能：提供文件夹浏览对话框，供用户进行检索路径的选择，并在用户确定检索路径后将之记录下来。检索路径的根目录设定为“我的电脑”，也就是说可供用户选择的文件夹只有“我的电脑”及其包含的所有子文件夹。如果用户没有选择检索路径，那么程序会在 C 盘根目录下创建一个名为 PicLib 的新文件夹作为默认的检索路径。

活动流程图如下：

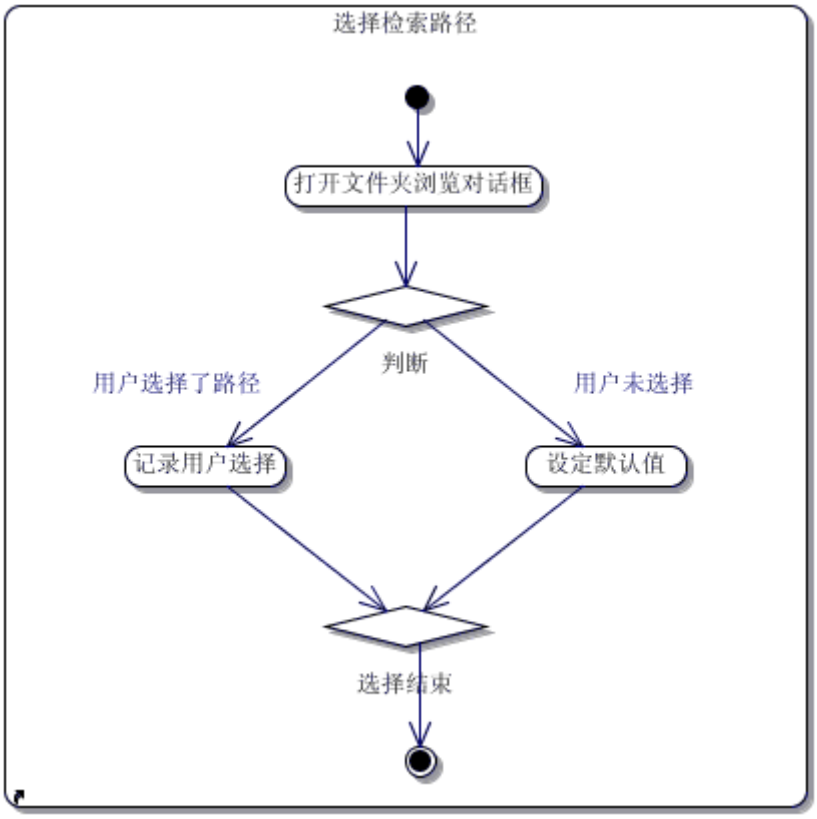

图 4.3　选择检索路径活动图

（2）特征提取模块

模块功能：系统向用户提供打开文件对话框供用户进行图像文件选择。目前实现的系统支持的图像文件格式为.bmp 和.jpg。用户选择要打开的样例图像后，系统将该图像文件读入内存，接着用户选择需要提取的图像特征描述子。系统记录下用户的选择，然后调用相应的函数对输入图像进行特征提取并获得描述子的值，最后系统将描述子内容以一定的格式显示。

活动流程图：

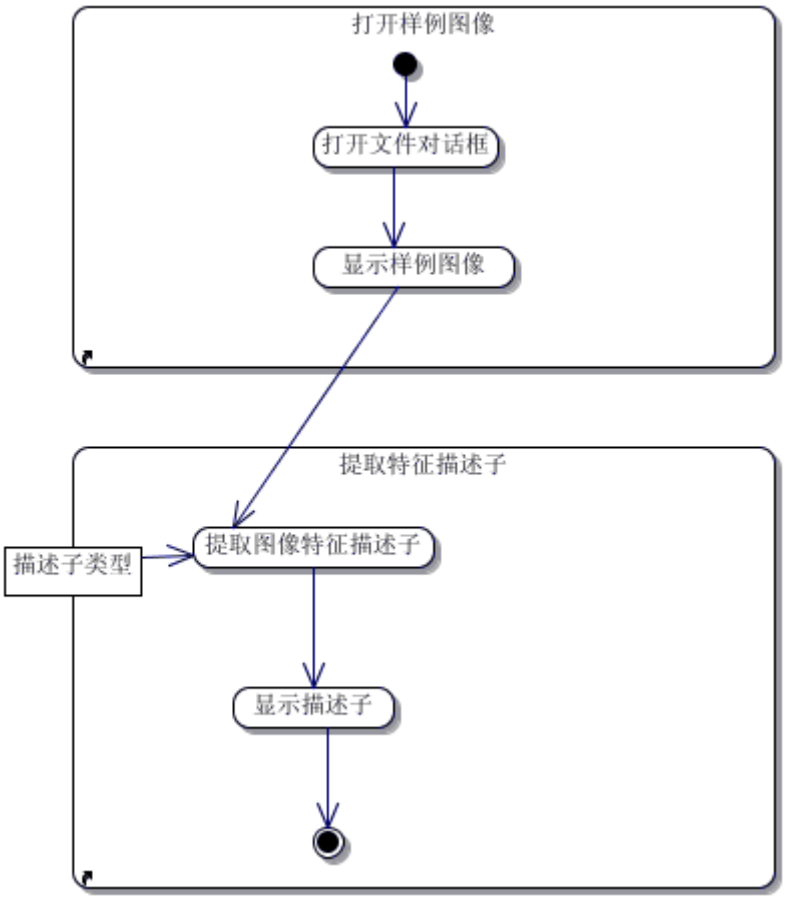

图 4.4 图像特征提取活动图

（3）生成和浏览特征描述文件模块

模块功能：首先系统判断描述文件是否已经生成并且包含路径下的所有图像文件。若没有生成则在用户选择的检索路径下进行查找，将所有未处理过的 BMP 和 JPG 图像文件找出逐个进行特征提取。并按一定的格式将提取出的特征描述子写入 XML 文件中。该模块还可以根据用户的选择打开已生成的特征描述文件，方便用户进行浏览。活动流程图：

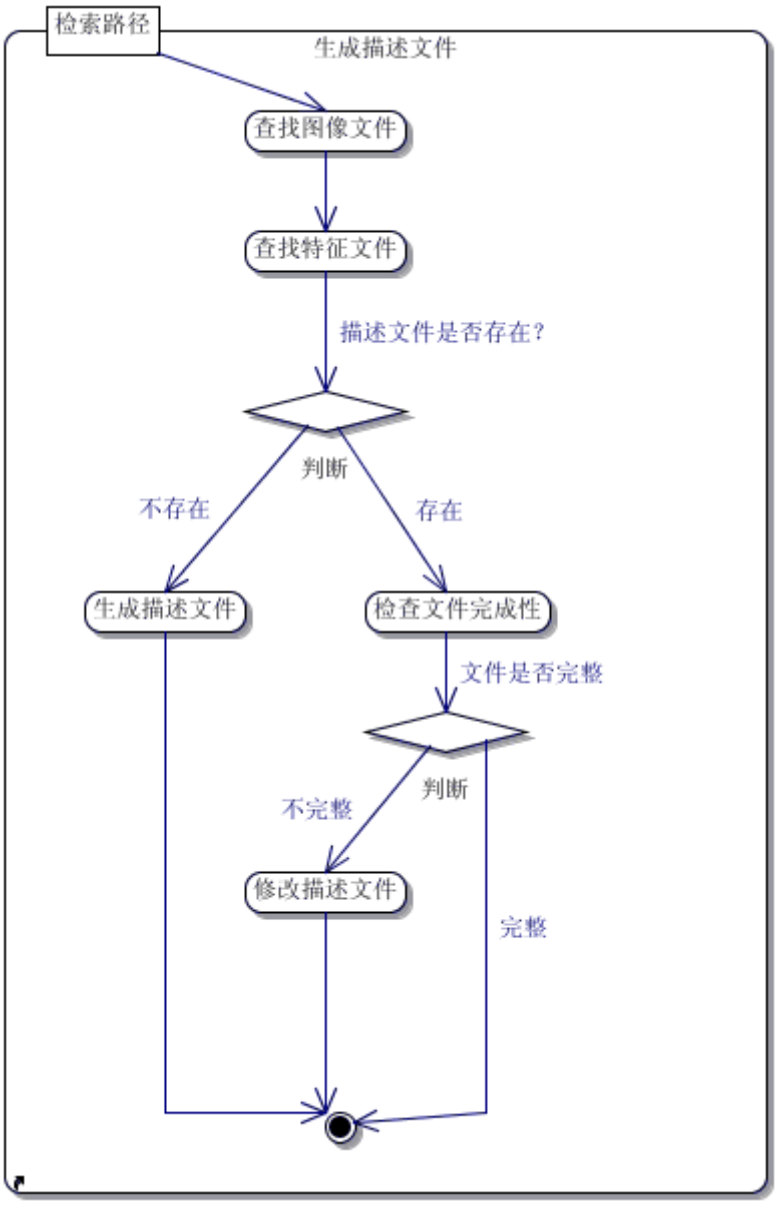

图 4.5　生成特征文件活动图

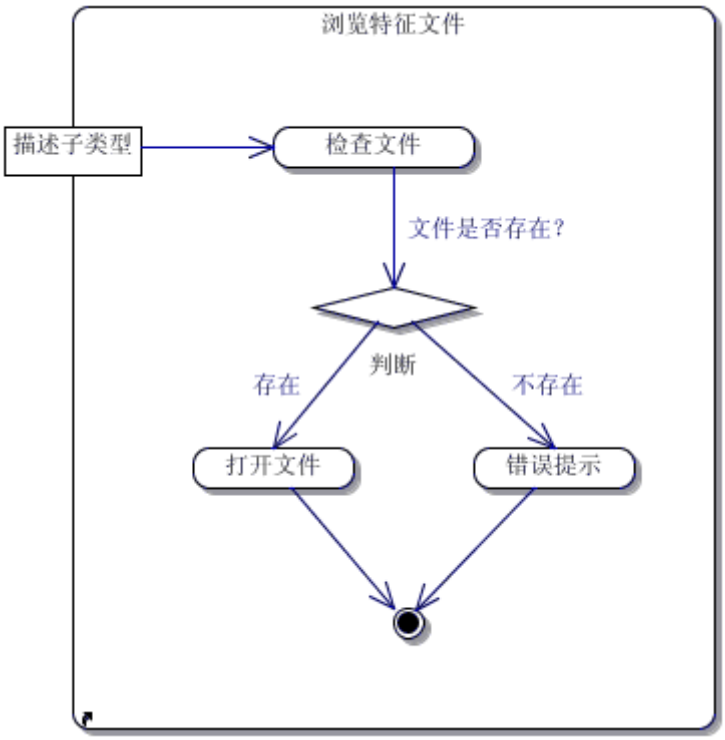

图 4.6　浏览特征描述文件活动图

（4）图像检索模块

模块功能： 根据用户选择的检索方式，系统首先检查是否已经生成了相应的特征描述文件，如果没有则报错，否则打开相应的特征描述文件，读出其中的描述子内容并根据相似性度量算法计算与样例图像的相似度。完成匹配计算后对计算结果进行排序并输出符合条件的结果图像。

活动流程图：

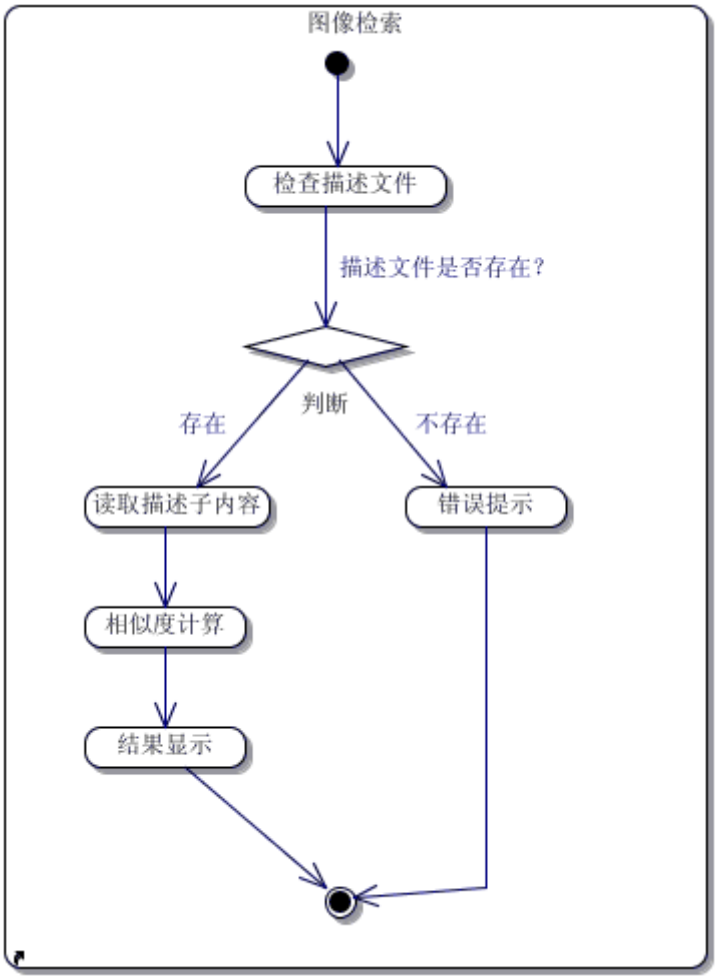

图 4.7　图像检索活动图

4.3.2 XML 文档格式设计

XML 是 eXtensible Markup Language 的缩写，意为可扩展的标记语言。与 HTML 相似，XML 是一种显示数据的标记语言，它能使数据通过网络无障碍地进行传输，并显示在用户的浏览器上。XML 是一种元标记语言。用户可以定义自己需要的标记。这些标记必须根据某些通用的原理来创建，但是在标记的意义上，也具有相当的灵活性。XML 对语法有严格的要求，所有 XML 的文件都必须经过严格的“验证”过程才算完成。XML 最大的优势在于对各种数据的管理。任何系统都可以通过 XML 的解析器来读取 XML 数据，因此它的数据可以通行各处，而不用担心系统不支持的问题。鉴于 XML 的这些优点，本系统在实现时选择了它来存放图像视觉描述子的相关信息。

XML 提供了一种结构化的组织数据的方式，有点类似于 Windows Explorer 中的文

件夹和文件。每个 XML 文档必须有单一的根元素，其中包含所有的元素和文本数据。如果在文档的顶级结点中有多个元素，那么文档就是不合法的 XML 文档。

.NET框架中提供了一组用于支持XML的命名空间。其中用于XML文档读写的类有XmlTextReader，XmlTextWriter，XmlValidatingReader和XmlDocument等。本系统中使用了System::Xml命名空间下的部分类和方法[12]。并使用了两种方法来处理XML文档，一种是单向向前的处理方式，为XmlTextReader类中使用的方法，该方法的优点是占用系统资源很小，缺点是很少缓存已经解析过的XML元素，不能重定向回文档中已经解析的元素；另一种方法是在XmlDocument类中使用的将整个XML文档都读入内存中再进行处理的方法。这种方法的好处是可以任意的对XML进行读，写，修改和删除，而缺点在于需要较大的内存空间用于载入整个XML文档，当XML文档很大时会较为耗费系统资源。

由于系统所能提取的描述子有四种，因此需要设计不同的文档格式来存放描述子信息。XML 文档的格式设计如下：

（1）主颜色描述子文件 DominantColorDes.xml

```
<?xml version="1.0"?>
<description>
    <descriptorType>Dominant Color Descriptor</descriptorType>
    <colorspace>HSV</colorspace>
    <MinNumber value="1" />
    <MaxNumber value="8" />
    <descriptor>
        <source filename="E:\sample images\leaflesstrees\image01.jpg" />
        <ColorNumber value="8" />
        <DominantColor vector="10 2 3" percent="0.664717768568759" />
                                    ⋮
        <DominantColor vector="10 2 2" percent="0.0346789700032781" />
    </descriptor>
          ⋮
</description>
```

整个 XML 文档以 description 为根元素，包含了许多的 descriptor 块，每块都是对一个图像文件进行主颜色描述子提取后所生成的。内容包括原始图像文件名，主颜色的数量、每个主颜色的 3 个分量的量化值以及所占的百分比。XML 文档的开头是主颜

色描述子的一些通用信息，包括颜色空间，最大/最小颜色数等等。

（2）颜色结构描述子文件 ColorStructureDes.xml

```
<?xml version="1.0"?>
<description>
    <descriptorType>Color Structure Descriptor</descriptorType>
    <colorspace>HMMD</colorspace>
    <histogramBin number="256" />
    <descriptor>
            <source filename="E:\experiment\image01.jpg" />
        <Histogram>……</Histogram>
    </descriptor>
          ⋮
</description>
```

与主颜色描述子文件类似，该 XML 文档中也包含了许多 descriptor 块。每块的内容包括图像文件名和直方图在各级上的值（两个 Histogram 标记之间的值有 256 个）。

（3）颜色布局描述子文件 ColorLayoutDes.xml

```
<?xml version="1.0"?>
<description>
    <descriptorType>Color Layout Descriptor</descriptorType>
    <colorspace>YCrCb</colorspace>
    <NumberofYCoff>6</NumberofYCoff>
    <NumberofCrCoff>3</NumberofCrCoff>
    <NumberofCrCoff>3</NumberofCrCoff>
    <descriptor>
        <source filename="E:\experiment\image01.jpg" />
        <YCoff>33 32 32 32 32 32 </YCoff>
        <CrCoff>63 32 32 </CrCoff>
        <CbCoff>9 32 32 </CbCoff>
    </descriptor>
          ⋮
```

</description>

颜色布局描述子中含有 Y 因子,Cr 因子和 Cb 因子三个部分，因此在描述文件的开头标明了各个因子的数量。descriptor 块中的内容是图像的文件名和 3 种因子的值。

（4）边缘直方图描述子文件 EdgeHistogramDes.xml

```
<?xml version="1.0"?>
<description>
    <descriptorType>Edge Histogram Descriptor</descriptorType>
    <colorspace>Gray Scale</colorspace>
    <histogramBin number="80" />
    <descriptor>
        <source filename="E:\experiment\image01.jpg" />
        <Histogram>……</Histogram>
    </descriptor>
        ⋮
</description>
```

该文件的头部加入了对直方图级数的说明。descriptor 块的内容有图像的文件名和直方图的内容，两个 Histogram 标记之间有 80 个值。

4.3.3 类设计

根据面向对象的设计思想，每个描述子应该设计一个类与之对应。由于生成描述子过程中有一些共同的操作，比如将描述子内容写入 XML 文档，对两个描述子进行相似度计算等，而且在对图像进行特征提取时也会重复用到一些数据，如图像的宽度和高度等，因此，有必要设计一个基类使得具体的描述子类为从这个基类中派生。基类的名称设计为 descriptor，带有一些子类会用到的方法和字段。类继承视图如下：

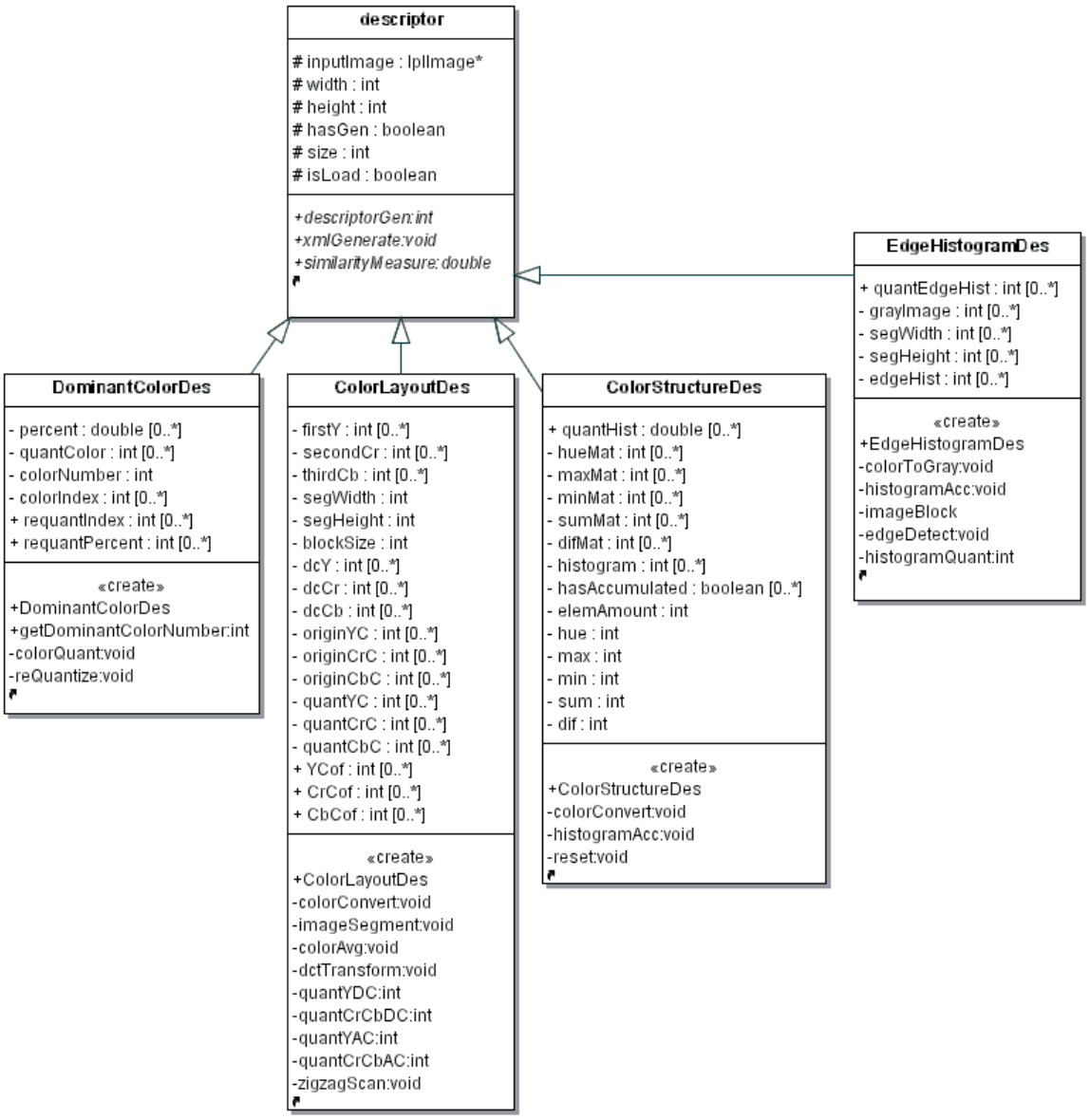

图 4.8　类层次图

基类和子类的字段和方法的描述如下：

（1）描述子基类 abstract class descriptor

将基类设计为抽象类，不能被实例化。

受保护的字段：

IplImage* inputImage

OpenCV 中规定的数据结构，用于指向待处理的原始图像。

int width

图像的宽度。

int height

图像的高度。

int size

图像的大小。

bool isLoad

指示图像是否被成功读入内存。

bool hasGen

指示是否成功生成了描述子。

公有方法：

virtual int descriptorGen()

描述子生成函数。子类在重写该函数时可根据不同的特征提取算法进行不同的实现。但在进行特征提取操作前应先通过 isLoad 字段判断图像是否已被读入内存中，在完成特征提取各步骤之后应将 hasGen 字段设置为 true。

virtual void xmlGenerate(XmlDocument* doc,XmlNode* root,String* sourceFile)

该函数接受 3 个参数，第一个 XmlDocument*型参数指向读入内存中的 XML 文档；第二个 XmlNode*型参数指向该文档的根元素；最后一个字符串指针型参数指向图像文件名。根据 XML 文档格式的设计，该函数应该将描述子内容进行转换，生成一个<descriptor>块并将其插入 XML 文档中。

virtual double similarityMeasure(XmlNode* descriptor)

该函数的功能为比较两个描述子的相似度。参数类型为 XmlNode*，指向一个 XML 块。实现该函数时首先应进行数据类型转换，然后根据不同描述子的匹配算法进行相似度计算。

（2）主颜色描述子类 class DominantColorDes

公有字段：

int requantIndex __gc[]

整型托管数组，用于存放经颜色选择后的主颜色索引。

double requantPercent __gc[]

双精度浮点型托管数组，用于存放经过重新量化后的颜色的百分比。

私有字段：

double percent __gc[]

未经重新量化前的颜色百分比

int quantColor __gc[]

存放量化后的颜色索引

int colorNumber

存放提取出的主颜色的数量

int colorIndex __gc[]

在进行百分比排序时用于对应相应的颜色索引

公有方法：

DominantColorDes(String*)

类的构造函数，参数为需要读入内存的图像文件的文件名。函数的功能为初始化各个字段并将需要处理的图像读入到内存中。需要注意的是因为该方法的参数为字符串指针,而 OpenCV 中的图像载入函数 cvLoadImage()接受的参数类型为 char*,而 C++.NET 中不能直接将 String*类型转为 char*类型。因此在实现时我先将 String 对象转换为宽字符型数组 wchar_t[],再使用 Windows API 函数 WideCharToMultiByte 将 wchar_t[]转换为 char*类型传递到 cvLoadImage 函数中。以下 descriptor 每个子类的构造函数中都必须进行这样的转换，同时，由于 char*无法存放中文字符，因此该系统不能处理中文文件名的图像。

~DominantColorDes()

类的析构函数，释放已分配的内存资源。

int getDominantColorNumber()

如果 hasGen 变量的值为 true，则返回主颜色的数量，否则返回 0。

int descriptorGen()

该函数首先调用 colorQuant 函数对每个像素进行 HSV 颜色进行量化，然后进行直方图累加，第二步对直方图进行归一化，得到各个颜色对应的百分比。然后使用 reQuantize 函数选择主颜色即可得到描述子。

私有方法：

void colorQuant(int row, int col)

该函数的作用的是对图像颜色进行 166 维量化。函数接受两个整型参数，分别为图像像素对应的行和列。colorQuant 首先将原始的 RGB 图像转换到 HSV 颜色空间上，接着使用颜色量化算法对 HSV 颜色进行量化，最后进行直方图累加以便其它函数计算百分比。

void reQuantize()

用于在量化生成的 166 种颜色中选择出不多于 8 种主颜色。该函数按照从大到小的顺序逐个选择主颜色，并对主颜色对应的百分比进行累加，选择过程持续到百分比之和大于 60%或者已经选出了 8 种主颜色为止。

（3）颜色结构描述子类 class ColorStructureDes

公有字段：

double quantHist __gc[]

用于存放归一化直方图各级的值。

私有字段：

int hueMat __gc[,]

int maxMat __gc[,]

int minMat __gc[,]

int difMat __gc[,]

int sumMat __gc[,]

由于OpenCV中不提供对HMMD颜色空间的支持，因此定义了以上五个二维数组来进行颜色空间转换。这五个数组分别对应HMMD颜色空间中的五个分量。

int histogram __gc[]

存放颜色直方图的值。

bool hasAccumulated __gc[]

指示在当前结构单元中，直方图的某一级是否已经进行过累加。

int elemAmount

存放图像中包含的结构单元的个数，该值可由（width-7）×（height-7）计算得到。

int max

int min

int dif

int sum

int hue

以上五个字段为HMMD颜色空间的五个分量，在颜色量化函数中会使用到。

公有方法：

ColorStructureDes(String*)

构造函数，负责进行图像载入，字段初始化等工作。

~ColorStructureDes()

类的析构函数。

int descriptorGen()

该函数从输入图像中提取颜色结构描述子。首先调用 colorConvert 方法对输入图像进行颜色空间转换；第二步使用一个 8×8 的结构单元扫描整幅图像并使用 histogramAcc 方法进行直方图累加。最后使用 elemAmount 字段对直方图进行归一化即可得到颜色结构描述子。

私有方法：

void colorConvert()

该函数将图像的颜色空间由 RGB 转换为 HMMD。

void histogramAcc(int,int)

该函数的功能是对完成颜色空间转换后的图像进行颜色量化和直方图累加。在直方图累加时需要注意的是同一个结构单元中重复出现的颜色只进行一次直方图累加。

void reset()

该函数用于将 hasAccumulated 数组中的值全部重置为 false。在对一个结构单元完成直方图统计后调用该函数以便进行下一个结构单元的统计。

（4）颜色布局描述子 class ColorLayoutDes

公有字段：

int YCof __gc[]

存放描述子的 Y 分量，数组大小为 6。

int CrCof __gc[]

存放描述子的 Cr 分量，数组大小为 3。

int CbCof __gc[]

存放描述子的 Cb 分量，数组大小为 3。

私有字段：

int firstY __gc[,]

int secondCr __gc[,]

int thirdCb __gc[,]

以上三个二维数组用于进行从 RGB 到 YCrCb 颜色空间的转换。

int segWidth

int segHeight

int blockSize

以上三个字段分别存放分割出的图块的宽度，高度和图块的大小。

int dcY __gc[,]

int dcCr __gc[,]

int dcCb __gc[,]

以上三个二维数组分别存放每个图块的平均颜色的三个分量。

int originYC __gc[,]

int originCrC __gc[,]

int originCbC __gc[,]

以上三个二维数组分别存放进行 DCT 变换后原始的因子值。

int quanYC __gc[,]

int quanCrC __gc[,]

int quanCbC __gc[,]

以上三个二维数组分别存放进行因子量化后的各分量因子值。

公有方法：

ColorLayoutDes(String*)

类的构造函数。

~ColorLayoutDes()

类的析构函数。

int descriptorGen()

函数首先调用 colorConvert 方法对输入图像进行颜色空间转换，然后调用 imageSegment 方法将图像分割为 64（8×8）个子图块。第三步调用 dctTransform 方法对每个子图进行 DCT 变换和量化并得到一系列的因子，最后调用 zigzagScan 方法选出低频因子构成颜色布局描述子。

私有方法：

void colorConvert()

将颜色空间由 RGB 转换为 YCrCb。

void imageSegment()

将输入图像分割为 64 个子图。

void colorAvg(int row,int col)

计算每个子图的平均颜色作为该子图的代表颜色。两个整型参数指示了子图在原图像中的行列位置。

void dctTransform()

按照 MPEG-7 中给出的算法对图像进行 DCT 变换，并对变换输出的原始因子进行量化。

int quantYDC(int)

int quantCrCbDC(int)

int quantYAC(int)

int quantCrCbAC(int)

以上 5 个方法为 MPEG-7 中规定的因子量化方法。

void zigzagScan()

对量化因子集合进行锯齿扫描，选出其中的低频因子构成颜色布局描述子。

（5）边缘直方图描述子 class EdgeHistogramDes

公有字段：

int quantEdgeHist __gc[]

存放进行量化后的边缘直方图各级的值。

私有字段：

int grayImage __gc[,]

存放进行灰度转换后的图像各个像素的灰度值。

int segWidth

int segHeight

以上两个字段分别存放子图像的宽度和高度。

int edgeHist __gc[]

存放原始的直方图值。

公有方法：

EdgeHistogramDes(String*)

类的构造函数。

~EdgeHistogramDes()

类的析构函数。

int descriptorGen()

该函数首先调用 colorToGray 方法将输入图像由彩色图像转换为灰度图像，接着调用 histogramAcc 方法进行边缘检测和直方图累加及量化，即可生成边缘直方图描述子。

私有方法：

void colorToGray()

如果输入是彩色图像，则将其转换为灰度图像。

void histogramAcc()

该函数对生成的灰度图像进行边缘检测和直方图累加与量化。

void imageBlock(int heightStart,int heightEnd,int widthStart,int widthEnd,int row,int col)

该函数将 histogramAcc 方法输入的子图进一步分割为 256 个图块，并在每个图块上调用 edgeDetect 方法进行边缘检测。前两个参数为子图的高度的起止坐标，第三、四个参数为子图宽度的起止坐标，最后两个参数为子图位在原图像中的行列位置。

int edgeDetect(int heiStart,int heiEnd,int widStart,int widEnd)

该函数使用 MPEG-7 中规定的五种边缘滤波器对输入的图块进行边缘检测。系统实现中所设置的阈值为 20，超过这个值的滤波结果将被视为在图块中含有边缘。

int histogramQuant(double origin,int index)

该函数根据 MPEG-7 标准中给出的量化表对归一化后的边缘直方图进行非均匀量化，得到一系列整数表达。

4.3.4 界面设计

本系统是 Windows 单一窗体应用程序。样例图像浏览，图像特征提取，特征描述文件生成，图像检索结果显示等工作都在这个窗口中完成。该窗口的截图和使用的主要控件及它们的功能设计如下：

界面截图：

图 4.9　系统界面截图

（1）MainMenu 控件

该控件为用户提供各种操作选项，主菜单包含两项：操作和帮助。操作菜单中包含以下选项：

打开样例图像；

更换检索路径；

生成特征文件；

浏览特征文件；

图像检索；

退出。

每一个菜单项都与相应的事件处理程序相关联。

帮助菜单中包含有关于选项，单击可显示作者信息。

（2）ToolTip 控件

使用该控件的目的在于当用户操作鼠标停留在界面的某部分之上时能够给出简短的提示，以此来提高系统的用户友好性。需要显示在 ToolTip 中的内容可以根据不同的

控件以及不同的事件使用 SetToolTip 方法来设置。

（3）OpenFileDialog 控件

该控件为用户提供了文件选择和打开功能。可以设置控件中的过滤器属性以控制对话框中显示的文件类型。本系统支持的图像类型为 BMP 和 JPG，因此过滤器可写成“BMP 文件(*.bmp)|*.bmp|JPG 文件(*.jpg)|*.jpg”的形式。这样在用户进行文件选择时，只会看见 BMP 文件或者 JPG 文件，防止了程序的异常输入。

（4）FolderBrowserDialog 控件

该控件的作用是提供给用户选择文件夹的功能。在系统实现中，我将该控件的 RootDirectory 属性设置为“My Computer”，这样在系统运行时，用户可以选择的文件夹范围为“我的电脑”中包含的所有子文件夹。

（5）PictureBox 控件

PictureBox 控件是界面设计时用到最多的控件了。我一共在界面上布置了 11 个 PictureBox 控件，其中一个用于显示样例图像，剩余 10 个用于显示检索结果。该控件需要设置的属性主要是 SizeMode 和 Image。前者用于确定当图像大小和控件大小不一致时控件将如何调整，我设置的值为“StretchImage”，意思是控件大小不变而改变图像的大小；后一个属性主要在程序中设置，用于在控件中显示图像。

由于界面大小的限制，每个 PictureBox 控件不能设计得太大，但这又会影响到用户观察图像的细节，因此我为每个 PictureBox 控件都加入了 doubleClick 事件，当鼠标在控件上双击，且控件中有图像显示时，即调用 Windows 默认的图片查看器显示图像。该功能可通过调用 System::Diagnostics 命名空间中 Process 类拥有的 Start 方法来实现。

（6）RadioButton 控件

该控件用于让用户直观的选择需要提取哪个特征描述子，界面上一共放置了四个这样的控件。

（7）Button 控件

用户单击这个控件即可进行样例图像的指定特征描述子提取。在打开样例图像之前，这个控件被设为不可用的。

（8）TextBox 控件

在提取完样例图像的特征描述子后，描述子的内容显示在这个控件中。将这个控件的 MultiLine 和 ReadOnly 属性都改为 true，可以实现只读的多行文本显示。

（9）ListView 控件

该控件用于在图像检索过程中列表显示样例图像和候选图像的距离。该控件分为两栏，分别显示候选图像的文件名和它们与样例图像的距离。在显示前需要将控件的 View

属性设置为“Details”才能正确显示表头。

（10）StatusBar 控件

该控件在系统运行时可以给用户提供一些信息，如系统就绪，正在提取特征文件等等。

（11）Panel 控件

该控件类似于一个容器，可以将其它控件添加到这个控件中，并能按索引访问添加进去的控件。在系统实现时，我将 10 个 PictureBox 控件以 System::Controls 为类型添加进了 Panel 控件中，这样就可以实现对控件的按索引访问，再使用<dynamic_cast>命令就可以将控件类型转换回 PictureBox，解决了检索结果数量不定，必须要按索引使用控件的问题。

4.4 图像检索系统测试

为了保证系统的正确性和可靠性，我对完成编码和调试的系统进行了一系列的测试，下面简要列出一些测试用例和系统运行结果截图：

（1）对样例图像进行主颜色描述子提取：

首先选择一幅样例图像打开，然后选择主颜色描述子作为要提取的描述子类型，并提取描述子的值进行显示。

图 4.10　显示样例图像

图 4.11　图 3.9 中的样例图像的主颜色描述子内容

（2）在检索路径下生成主颜色描述文件并使用“浏览特征文件”菜单项打开特征描述文件。

```
<?xml version="1.0" ?>
- <description>
    <descriptorType>Dominant Color Descriptor</descriptorType>
    <colorspace>HSV</colorspace>
    <MinNumber value="1" />
    <MaxNumber value="8" />
  - <descriptor>
      <source filename="E:\sample images\leaflesstrees\image01.jpg" />
      <ColorNumber value="8" />
      <DominantColor vector="10 2 3" percent="0.356568090199043" />
      <DominantColor vector="10 1 3" percent="0.0381708238851096" />
      <DominantColor vector="0 1 3" percent="0.033661921558747" />
      <DominantColor vector="0 0 1" percent="0.0236494289073654" />
      <DominantColor vector="1 1 1" percent="0.0227859662383472" />
      <DominantColor vector="9 1 3" percent="0.0222033257747543" />
      <DominantColor vector="10 1 2" percent="0.0207782186948854" />
      <DominantColor vector="10 2 2" percent="0.0186025027294869" />
    </descriptor>
```

图 4.12　特征描述文件 DominantColorDex.xml 片断

（3）在检索路径下进行基于主颜色匹配的图像检索：

图 4.13　检索结果

（4）在检索路径下进行基于颜色布局描述子的检索：

图 4.14　检索结果

5 结论

本文简要介绍了 MPEG-7 标准的形成，MPEG-7 标准的目标等概念。MPEG-7 标准是用于对多媒体内容进行描述以方便进行检索和筛选等应用的。在 MPEG-7 中一共定义了七种图像视觉特征描述子（四种颜色特征，三种纹理特征），本文中对这些描述子的功能、提取和匹配算法逐一进行了简要介绍。为了实验 MPEG-7 在图像检索方面的有效性，描述了一个基于图像视觉特征进行图像检索的系统的实现过程，包括用例分析，概要设计，详细设计等等。

经过对系统的测试发现，主颜色描述子的匹配结果比其它几种描述子要差一些。这可能与描述子的相似度算法选择有关。对于其它描述子，它们到自身的距离都为 0，而主颜色描述子到自身的距离并不为 0，这就导致了在检索过程中，与样例图像最相似的竟然不是样例图像本身。另外，在使用主颜色描述子检索较鲜艳的颜色诸如“红花”时，检索效果较好；而检索较暗淡的颜色诸如山脉的颜色时，检索效果明显会变差，这是今后工作需要改进的地方。

另外值得改进的地方还有在系统中加入未实现的三个描述子，还有就是对颜色结构描述子的提取算法进行改进，目前我使用的算法会耗费很多系统资源，影响了特征描述文件的生成速度，导致用户会长时间的等待。设计一个较有效率的索引结构也是今后工作中需要做的，目前系统使用的是线性查找方法，即打开特征描述文件后按顺序的逐个读入描述子内容。当图像数据库很大时，这种方法耗费的时间和资源就会较多，因此需要进行改进。

这次为期三个月的毕业设计过程是紧张而充实的，觉得自己收获最大的是学会了独立的分析问题和解决问题。以前在课堂上，总是老师提出问题学生去回答。而在毕业设计过程中，题目是需要实现的目标，但是具体怎么实现并没有人会告诉你。这就需要自己去思考，去发现有哪些问题需要解决，如何解决。经过毕业设计的锻炼，文献查找和阅读能力提高了不少，也熟悉了 C++.NET，Borland Together 等软件的使用。

致　谢

首先衷心地感谢我的毕业设计的指导老师王天江教授。在三个月的时间里，王老师通过言传身教让我学到了许多重要的思想，养成了良好的学习习惯。王老师对学生的严格要求，让我的毕业设计工作能够按质按量的完成。我还要感谢所在的视频处理小组的组长龚立宇博士。初进实验室的时候是他引导我开始了毕业设计的工作，并给了我不少的参考资料，让我逐渐形成了独立解决问题的良好习惯。另外，实验室的刘芳老师和尹光明、周柯、张丹等几位师兄也给了我许多的帮助，在此表示诚挚的感谢。最后还要感谢我的父母在大学四年里对我的支持和鼓励，没有你们我的学业和留学申请不可能这么顺利的完成，谢谢你们！

参考文献

[1] 李向阳 等，《基于内容的图像检索技术与系统》，Journal of Computer Research and Development，2001，Vol. 21，No. 3，p344-354

[2] 章毓晋，《基于内容的多媒体信息检索 CMIR 与国际标准 MPEG-7》，第九届全国图象图形学学术会议论文集，1998，p9-16

[3] ISO, IEC et al, Text of ISO/IEC 15938-3 Multimedia Content Description Interface-Part 3: Visual, 2001, p26

[4] B. S. Manjunath, et al，《Color and Texture Descriptors》，IEEE Transactions on Circuits and Systems for Video Technology，2001，Vol. 11，No. 6，p703-715

[5] 纪敏，《MPEG-7 颜色，纹理和形状描述子》，计算机工程与应用，2004, Vol. 24, p44-47

[6] 王海霞，《基于 MPEG-7 的图像检索技术》，硕士学位论文，2004

[7] Xu, Feng，Zhang, Yu-Jin，《Evaluation and comparison of texture descriptors proposed in MPEG-7》，Journal of Visual Communication and Image Representation, 2006, Vol. 172006, p701-716

[8] 帅勤 等，《基于 MPEG-7 边缘直方图描述符的图像检索系统的实现》，第十届全国青年通信学术会议论文集，2005，p567-573

[9] R. C. 冈萨雷斯，P. 温茨，《数字图象处理》，科学出版社，1983

[10] 章毓晋，《图像工程》第二版，清华大学出版社，2005

[11] Stanley B. Lippman，《C++ Primer》第三版，中国电力出版社，2005

[12] Julian Templeman et al，《Visual C++.NET 程序设计》，清华大学出版社，2002