基于感兴趣内容图像检索系统的核心代码

#include "stdafx.h"

#include "Retrieve.h"

#include "imagetrievalDlg.h"

#include "picture.h"

#include "math.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#undef THIS\_FILE

static char THIS\_FILE[] = \_\_FILE\_\_;

#endif

#define PI 3.1415926

CString\* temp[100];//检索库中图像路径

CString temp\_1[100];//临时存放检索库中图像路径

CString strfile;//待检索图像的路径

CString m\_strPath;//检索库路径

int tempi;//临时的检索库图像计数器

int counts;//检索库图像计数器

double pix[1000][1000];//当前分析图像的像素

double feature\_shape[8];//待检索图像的形状特征

double feature\_shape\_1[8]; //当前分析图像的形状特征

double feature\_color[3][12];//待检索图像的颜色特征

double feature\_color\_1[3][12]; //当前分析图像的颜色特征

bool open\_pic;//设置待检索图像标志位

bool dir;//设置检索库标志位

bool color;//各种颜色方法是否使用过标志位

bool shape;//形状方法是否使用过标志位

picture image;//基于颜色特征

picture image\_color\_temp;

picture image\_color\_1;

picture image\_color\_2;

picture image\_color\_3;

picture image\_shape;

picture image\_shape\_temp;

/////////////////////////////////////////////////////////////////////////////

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog

{

public:

CAboutDlg();

// Dialog Data

//{{AFX\_DATA(CAboutDlg)

enum { IDD = IDD\_ABOUTBOX };

//}}AFX\_DATA

// ClassWizard generated virtual function overrides

//{{AFX\_VIRTUAL(CAboutDlg)

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV support

//}}AFX\_VIRTUAL

// Implementation

protected:

//{{AFX\_MSG(CAboutDlg)

//}}AFX\_MSG

DECLARE\_MESSAGE\_MAP()

};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)

{

//{{AFX\_DATA\_INIT(CAboutDlg)

//}}AFX\_DATA\_INIT

}

void CAboutDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

//{{AFX\_DATA\_MAP(CAboutDlg)

//}}AFX\_DATA\_MAP

}

BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialog)

//{{AFX\_MSG\_MAP(CAboutDlg)

// No message handlers

//}}AFX\_MSG\_MAP

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CImagetrievalDlg dialog

CImagetrievalDlg::CImagetrievalDlg(CWnd\* pParent /\*=NULL\*/)

: CDialog(CImagetrievalDlg::IDD, pParent)

, ccmethod(\_T(""))

, ccper(\_T(""))

, ssper(\_T(""))

{

//{{AFX\_DATA\_INIT(CImagetrievalDlg)

//初始化选择的方法

ccmethod=("颜色直方图");

method=1;

c\_per=50;

s\_per=50;

open\_pic=false;

dir=false;

//}}AFX\_DATA\_INIT

// Note that LoadIcon does not require a subsequent DestroyIcon in Win32

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

void CImagetrievalDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

//{{AFX\_DATA\_MAP(CImagetrievalDlg)

//}}AFX\_DATA\_MAP

DDX\_CBString(pDX, IDC\_COMBO1, ccmethod);

CString str1,str2,str3;

str1=("颜色直方图");

str2=("累计直方图");

str3=("颜色矩");

if(ccmethod==str1)

c\_method=1;

if(ccmethod==str2)

c\_method=2;

if(ccmethod==str3)

c\_method=3;

DDX\_Text(pDX, IDC\_EDIT2, ccper);

c\_per=atoi(CT2CA(ccper.GetBuffer(ccper.GetLength())));

DDX\_Text(pDX, IDC\_EDIT3, ssper);

s\_per=atoi(CT2CA(ssper.GetBuffer(ssper.GetLength())));

}

BEGIN\_MESSAGE\_MAP(CImagetrievalDlg, CDialog)

//{{AFX\_MSG\_MAP(CImagetrievalDlg)

ON\_WM\_SYSCOMMAND()

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

ON\_BN\_CLICKED(IDC\_OPEN, OnOpen)

ON\_BN\_CLICKED(IDC\_PATH, OnPath)

ON\_BN\_CLICKED(IDC\_START, OnStart)

ON\_BN\_CLICKED(IDC\_RADIO1, &CImagetrievalDlg::OnBnClickedRadio1)

ON\_BN\_CLICKED(IDC\_RADIO2, &CImagetrievalDlg::OnBnClickedRadio2)

ON\_BN\_CLICKED(IDC\_RADIO3, &CImagetrievalDlg::OnBnClickedRadio3)

//}}AFX\_MSG\_MAP

ON\_EN\_CHANGE(IDC\_EDIT2, &CImagetrievalDlg::OnEnChangeEdit2)

ON\_EN\_CHANGE(IDC\_EDIT3, &CImagetrievalDlg::OnEnChangeEdit3)

ON\_CBN\_SELCHANGE(IDC\_COMBO1, &CImagetrievalDlg::OnCbnSelchangeCombo1)

END\_MESSAGE\_MAP()

/////////////////////////////////////////////////////////////////////////////

// CImagetrievalDlg message handlers

BOOL CImagetrievalDlg::OnInitDialog()

{

CDialog::OnInitDialog();

// 居中显示

CenterWindow();

// Add "About..." menu item to system menu.

// IDM\_ABOUTBOX must be in the system command range.

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != NULL)

{

CString strAboutMenu;

strAboutMenu.LoadString(IDS\_ABOUTBOX);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

// Set the icon for this dialog. The framework does this automatically

// when the application's main window is not a dialog

SetIcon(m\_hIcon, TRUE); // Set big icon

SetIcon(m\_hIcon, FALSE); // Set small icon

// TODO: Add extra initialization here

((CButton \*)GetDlgItem(IDC\_RADIO1))->SetCheck(TRUE);

((CButton \*)GetDlgItem(IDC\_RADIO2))->SetCheck(FALSE);

((CButton \*)GetDlgItem(IDC\_RADIO3))->SetCheck(FALSE);

((CComboBox\*)GetDlgItem(IDC\_COMBO1))->ResetContent();

CString str;

str.Format("颜色直方图");

((CComboBox\*)GetDlgItem(IDC\_COMBO1))->AddString(str);

str.Format("累计直方图");

((CComboBox\*)GetDlgItem(IDC\_COMBO1))->AddString(str);

str.Format("颜色矩");

((CComboBox\*)GetDlgItem(IDC\_COMBO1))->AddString(str);

ccper=("50");

ssper=("50");

UpdateData(FALSE);

return TRUE; // return TRUE unless you set the focus to a control

}

void CImagetrievalDlg::OnSysCommand(UINT nID, LPARAM lParam)

{

if ((nID & 0xFFF0) == IDM\_ABOUTBOX)

{

CAboutDlg dlgAbout;

dlgAbout.DoModal();

}

else

{

CDialog::OnSysCommand(nID, lParam);

}

}

// If you add a minimize button to your dialog, you will need the code below

// to draw the icon. For MFC applications using the document/view model,

// this is automatically done for you by the framework.

void CImagetrievalDlg::OnPaint()

{

//确定给定窗口是否是最小化（图标化）的窗口

if (IsIconic())

{

CPaintDC dc(this); // device context for painting

SendMessage(WM\_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

// Center icon in client rectangle

int cxIcon = GetSystemMetrics(SM\_CXICON);

int cyIcon = GetSystemMetrics(SM\_CYICON);

CRect rect;

GetClientRect(&rect);

int x = (rect.Width() - cxIcon + 1) / 2;

int y = (rect.Height() - cyIcon + 1) / 2;

// Draw the icon

dc.DrawIcon(x, y, m\_hIcon);

}

else

CDialog::OnPaint();

}

// The system calls this to obtain the cursor to display while the user drags

// the minimized window.

HCURSOR CImagetrievalDlg::OnQueryDragIcon()

{

return (HCURSOR) m\_hIcon;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

设置待检索图像

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::OnOpen()

{

// TODO: Add your control notification handler code here

CFileDialog fileOpenDlg(TRUE);

if (fileOpenDlg.DoModal () != IDOK) return;

open\_pic=true;//标志位设置为true，表示待检索图像已设置

CWnd\* pWnd = GetDlgItem(IDC\_VIEW);

CDC\* pDC = pWnd->GetDC();

pWnd->Invalidate();

pWnd->UpdateWindow();

POSITION pos = fileOpenDlg.GetStartPosition();//对于选择了多个文件的情况得到第一个文件位置

strfile = fileOpenDlg.GetNextPathName(pos);//得到待检索图像的路径

ShowPic(strfile,IDC\_VIEW); //显示待检索图像

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

显示待检索图像

pathfile为待显示的文件路径，idc为图像显示控件的ID号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::ShowPic(CString pathfile, int idc)

{

CBitmap hbmp;

HBITMAP hbitmap;

//将pStatic指向要显示的地方

CStatic \*pStaic;

pStaic=(CStatic\*)GetDlgItem(idc);

//装载资源

hbitmap=(HBITMAP)::LoadImage (/\*::AfxGetInstanceHandle()\*/NULL,pathfile, IMAGE\_BITMAP,

0,0,LR\_LOADFROMFILE|LR\_CREATEDIBSECTION);

hbmp.Attach(hbitmap);

//获取图像格式

BITMAP bm;

hbmp.GetBitmap(&bm);

//创建临时的内存DC对象

CDC dcMem;

dcMem.CreateCompatibleDC(pStaic->GetDC());

CBitmap \*poldBitmap=(CBitmap\*)dcMem.SelectObject(hbmp);

CRect lRect;

pStaic->GetClientRect(&lRect);

lRect.NormalizeRect();

//显示位图

pStaic->GetDC()->StretchBlt(lRect.left ,lRect.top ,lRect.Width(),lRect.Height(),

&dcMem,0 ,0,bm.bmWidth,bm.bmHeight,SRCCOPY);

dcMem.SelectObject(&poldBitmap);

pStaic->ReleaseDC(&dcMem);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

设置检索库路径

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::OnPath()

{

// TODO: Add your control notification handler code here

//打开通用对话框，BROWSEINFO结构中包含有用户选中目录的重要信息

BROWSEINFO browse;

ZeroMemory(&browse,sizeof(browse));//fills a block of memory with zeros.

browse.hwndOwner = NULL;

browse.pszDisplayName = m\_strPath.GetBuffer(MAX\_PATH);

browse.lpszTitle = "请选择一个图像目录";

//SHBrowseForFolder函数返回一个ITEMIDLIST结构的指针，包含了用户选择文件夹的信息

LPITEMIDLIST lpItem = SHBrowseForFolder(&browse);

if(lpItem == NULL) return ;

m\_strPath.ReleaseBuffer();

//SHGetPathFromIDList把项目标志符列表转换为文档系统路径

if(SHGetPathFromIDList(lpItem,m\_strPath.GetBuffer(MAX\_PATH)) == false) return;

m\_strPath.ReleaseBuffer();

dir=true; //标志位设置为true，表示待检索图像已设置

AfxMessageBox("您选择的目录为:"+m\_strPath,MB\_ICONINFORMATION|MB\_OK);

//扫描检索库，得到图像目录下文件的路径

CString tempath;

CString temps;

tempath=m\_strPath;

tempath.TrimRight();tempath.TrimLeft(); //去除前后多余

CString strfilepath=tempath;

tempi=0;

counts=0;//计数器清零

//检索库中图像个数放入counts中，其路径放入temp[100]中

StartDir(strfilepath);

temps.Format("该目录下共有%d幅图像!",counts);

AfxMessageBox(temps,MB\_ICONINFORMATION|MB\_OK);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

扫描检索库中的图像个数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::StartDir(const CString &strfile1)

{

BOOL yesno;

CFileFind find;

char tempFileFind[200];

sprintf\_s(tempFileFind,"%s\\\*.\*",strfile1);

RunDir(strfile1);//在当前目录中查找图像，不搜索子目录

yesno = (BOOL)find.FindFile(tempFileFind);

//查找当前目录下子目录中的文件

while(yesno)

{

yesno = find.FindNextFile();

if (find.IsDots() != TRUE)//过滤缺省目录

{

char foundFileName[200];

strcpy\_s(foundFileName,find.GetFileName().GetBuffer(200));

if((find.IsDirectory() == TRUE)) //判断是否为目录

{

char tempDir[200];

sprintf\_s(tempDir,"%s\\%s",strfile1,foundFileName);//获得子目录路径

StartDir(tempDir); // 递归调用，查找子目录中图像

}

}

}

find.Close();

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

扫描当前目录中的图像个数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::RunDir(const CString &strfile2)

{

BOOL yesno;

CFileFind find;

char tempFileFind[200];

sprintf\_s(tempFileFind,"%s\\\*.bmp",strfile2);

yesno = find.FindFile(tempFileFind); //在当前目录下查找BMP文件

while(yesno)

{

yesno = find.FindNextFile();

char foundFileName[200];//临时存储查找到的图像名

strcpy\_s(foundFileName,find.GetFileName().GetBuffer(200));//获取图像名

if(!find.IsDots()) //过滤缺省目录

{

char tempFileName[200];

sprintf\_s(tempFileName,"%s\\%s",strfile2,foundFileName);

CString strfilepath1;

strfilepath1.Format("%s",tempFileName);//获取图像完整路径

counts++;

temp[tempi] = new CString(strfilepath1);//保存图像完整路径

tempi++;

}

}

find.Close();

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

开始检索

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::OnStart()

{

//检查是否设置了待检索图像和检索库路径

if(open\_pic==false)

{

AfxMessageBox("请先设置待检索图像");

return;

}

if(dir==false)

{

AfxMessageBox("请设置检索库路径");

return;

}

if(method==1)

{

switch(c\_method)

{

case 1:

general(strfile,1);//提取待检索图像颜色直方图

break;

case 2:

succession(strfile,1);//提取待检索图像累计直方图

break;

case 3:

centerM(strfile,1);//提取待检索图像颜色矩

break;

}

Color\_SeekImage();//基于颜色特征的相似度计算函数

}

if(method==2)

{

torque(strfile,1);//提取待检索图像的不变矩

Shape\_SeekImage();//基于形状特征的相似度计算函数

sort();

}

if(method==3)

{

torque(strfile,1);

Shape\_SeekImage();

switch(c\_method)

{

case 1:

general(strfile,1);

break;

case 2:

succession(strfile,1);

break;

case 3:

centerM(strfile,1);

break;

}

Color\_SeekImage();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

基于形状特征的相似度计算函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::Shape\_SeekImage()

{

for(int count=0; count<counts; count++)

temp\_1[count] = \*temp[count];

if(image\_shape\_temp.strfile\_old==strfile&&image\_shape\_temp.m\_strPath\_old==m\_strPath)

shape=1;//此检索库下的此检索图像已经用此方法计算过

else

shape=0;

for(int pic=0; pic<counts; pic++)//逐个分析检索库中图像

{

if(shape==0)

{

torque(temp\_1[pic],2);//提取检索的第N幅图的不变矩

double dis=0;//第N幅图与待检索图像的距离

double dis\_temp=0;

//显示正在检索的第N幅图与待检索图像的距离

for(int i=0;i<8;i++)

dis+=fabs((feature\_shape\_1[i]-feature\_shape[i])/fabs(feature\_shape[i]));

image\_shape\_temp.tp[pic]=temp\_1[pic];//保存第N幅图路径

image\_shape\_temp.num[pic]=dis;//保存第N幅图与待检索图像的距离

}

ShowPic(temp\_1[pic],IDC\_VIEW1);

}

if(shape==0)//若本方法为第一次计算，则保存相似度计算结果

image\_shape=image\_shape\_temp;

else//若本方法已经计算过，则读取相似度计算结果

image\_shape\_temp=image\_shape;

//检索过的待检索图像及检索库路径更新

image\_shape\_temp.m\_strPath\_old=m\_strPath;

image\_shape\_temp.strfile\_old=strfile;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

基于颜色特征的相似度计算函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::Color\_SeekImage()

{

for(int count=0; count<counts; count++)

temp\_1[count] = \*temp[count];

switch(c\_method)

{

case 1:

if(image\_color\_1.strfile\_old==strfile&&image\_color\_1.m\_strPath\_old==m\_strPath)

color=1;//此检索库下的此检索图像已经用此方法计算过

else

color=0;

break;

case 2:

if(image\_color\_2.strfile\_old==strfile&&image\_color\_2.m\_strPath\_old==m\_strPath)

color=1;//此检索库下的此检索图像已经用此方法计算过

else

color=0;

break;

case 3:

if(image\_color\_3.strfile\_old==strfile&&image\_color\_3.m\_strPath\_old==m\_strPath)

color=1;//此检索库下的此检索图像已经用此方法计算过已经用此方法计算过

else

color=0;

break;

}

for(int pic=0; pic<counts; pic++)//逐个分析检索库中图像

{

if(color==0)

{

switch(c\_method)

{

case 1:

general(temp\_1[pic],2);//提取检索的第N幅图的颜色直方图

break;

case 2:

succession(temp\_1[pic],2);//提取检索的第N幅图的累计直方图

break;

case 3:

centerM(temp\_1[pic],2);//提取检索的第N幅图的颜色矩

break;

}

double dis=0;//第N幅图与待检索图像的距离

double dis\_temp=0;

//计算正在检索的第N幅图与待检索图像的距离

for(int i=0;i<3;i++)

{

for(int j=0;j<12;j++)

dis\_temp+=(feature\_color\_1[i][j]-feature\_color[i][j])\*(feature\_color\_1[i][j]-feature\_color[i][j]);

dis+=sqrt((double)dis\_temp);

dis\_temp=0;

}

image\_color\_temp.tp[pic]=temp\_1[pic];//保存第N幅图路径

image\_color\_temp.num[pic]=dis;//保存第N幅图与待检索图像的距离

}

ShowPic(temp\_1[pic],IDC\_VIEW1);

}

switch(c\_method)

{

case 1:

if(color==0)//若本方法为第一次计算，则保存相似度计算结果

{

image\_color\_1=image\_color\_temp;

sort();

}

else//若本方法已经计算过，则读取相似度计算结果

{

image\_color\_temp=image\_color\_1;

sort();

}

//检索过的待检索图像及检索库路径更新

image\_color\_1.m\_strPath\_old=image\_color\_temp.m\_strPath\_old=m\_strPath;

image\_color\_1.strfile\_old=image\_color\_temp.strfile\_old=strfile;

break;

case 2:

if(color==0)

{

image\_color\_2=image\_color\_temp;

sort();

}

else

{

image\_color\_temp=image\_color\_2;

sort();

}

image\_color\_2.m\_strPath\_old=image\_color\_temp.m\_strPath\_old=m\_strPath;

image\_color\_2.strfile\_old=image\_color\_temp.strfile\_old=strfile;

break;

case 3:

if(color==0)

{

image\_color\_3=image\_color\_temp;

sort();

}

else

{

image\_color\_temp=image\_color\_3;

sort();

}

image\_color\_3.m\_strPath\_old=image\_color\_temp.m\_strPath\_old=m\_strPath;

image\_color\_3.strfile\_old=image\_color\_temp.m\_strPath\_old=strfile;

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

提取不变矩函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::torque(CString pathfile,int mode)

{

CBitmap hbmp;

HBITMAP hbitmap;

int R,G,B;

//装载资源

hbitmap=(HBITMAP)::LoadImage (::AfxGetInstanceHandle(),pathfile,IMAGE\_BITMAP,

0,0,LR\_LOADFROMFILE|LR\_CREATEDIBSECTION);

hbmp.Attach(hbitmap);

//获取图像格式

BITMAP bm;

hbmp.GetBitmap(&bm);

//创建临时的内存DC对象

CDC dcMem;

dcMem.CreateCompatibleDC(GetDC());

dcMem.SelectObject(hbmp);

int width=bm.bmWidth;

int height=bm.bmHeight;

COLORREF clr;

for(int x=0;x<width;x++)

{

for(int y=0;y<height;y++)

{

//该函数检索指定坐标点的像素的RGB颜色值

clr=dcMem.GetPixel(x,y);

R=GetRValue(clr);

G=GetGValue(clr);

B=GetBValue(clr);

pix[x][y]=0.299\*R+0.587\*G+0.114\*B;

}

}

//大津法求阈值

int index\_max;//最佳阈值

double g\_max;

g\_max=0;

for(int index=0;index<256;index++)

{

double c0,c1,sum\_0,sum\_1;

double w0,w1,u0,u1,g;

c0=c1=sum\_0=sum\_1=w0=w1=u0=u1=g=0;

for(int x=0;x<width;x++)

{

for(int y=0;y<height;y++)

{

if(pix[x][y]<index)

{

sum\_0=sum\_0+pix[x][y];//统计背景灰度值

c0++; //统计背景像素值

}

else

{

sum\_1=sum\_1+pix[x][y];//统计前景灰度值

c1++; //统计前景像素值

}

}

}

w0=c0/(c0+c1);//背景像素所占比例

w1=c1/(c0+c1);//前景像素所占比例

if(c0==0)

u0=0;

else

u0=sum\_0/c0;//背景平均灰度值

if(c1==0)

u1=0;

else

u1=sum\_1/c1;//前景平均灰度值

g=w0\*w1\*(u0-u1)\*(u0-u1);

if(g>g\_max)//保存最佳阈值

{

g\_max=g;

index\_max=index;

}

}

//根据所求阈值进行阈值分割

for(int x=0;x<width;x++)

{

for(int y=0;y<height;y++)

{

if(pix[x][y]<index\_max)

pix[x][y]=0;

else

pix[x][y]=1;

}

}

//0+0阶矩

double m00=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m00+=pix[x][y];

//1+0阶矩

double m10=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m10+=(x+1)\*pix[x][y];

//0+1阶矩

double m01=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m01+=(y+1)\*pix[x][y];

//1+1阶矩

double m11=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m11+=(y+1)\*(x+1)\*pix[x][y];

//2+0阶矩

double m20=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m20+=(x+1)\*(x+1)\*pix[x][y];

//0+2阶矩

double m02=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m02+=(y+1)\*(y+1)\*pix[x][y];

//3+0阶矩

double m30=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m30+=(x+1)\*(x+1)\*(x+1)\*pix[x][y];

//0+3阶矩

double m03=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m03+=(y+1)\*(y+1)\*(y+1)\*pix[x][y];

//1+2阶矩

double m12=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m12+=(x+1)\*(y+1)\*(y+1)\*pix[x][y];

//2+1阶矩

double m21=0;

for(int x=0;x<width;x++)

for(int y=0;y<height;y++)

m21+=(x+1)\*(x+1)\*(y+1)\*pix[x][y];

//求图像的区域重心

double xbar,ybar;

xbar=(double)m10/m00;

ybar=(double)m01/m00;

//求中心矩

double eta11,eta20,eta02,eta30,eta03,eta21,eta12;

eta11=(m11-ybar\*m10)/(m00\*m00);

eta20=(m20-xbar\*m10)/(m00\*m00);

eta02=(m02-ybar\*m01)/(m00\*m00);

eta30=(m30-3\*xbar\*m20+2\*xbar\*xbar\*m10)/(m00\*m00\*sqrt(m00));

eta03=(m03-3\*ybar\*m02+2\*ybar\*ybar\*m01)/(m00\*m00\*sqrt(m00));

eta21=(m21-2\*xbar\*m11-ybar\*m20+2\*xbar\*xbar\*m01)/(m00\*m00\*sqrt(m00));

eta12=(m12-2\*ybar\*m11-xbar\*m02+2\*ybar\*ybar\*m10)/(m00\*m00\*sqrt(m00));

//HU不变矩

double phi[8];

phi[0]=eta20+eta02;

phi[1]=(eta20-eta02)\*(eta20-eta02)+(4\*eta11\*eta11);

phi[2]=((eta30-3\*eta12)\*(eta30-3\*eta12))+((3\*eta21-eta03)\*(3\*eta21-eta03));

phi[3]=((eta30+eta12)\*(eta30+eta12))+((eta21+eta03)\*(eta21+eta03));

phi[4]=(eta30-3\*eta12)\*(eta30+eta12)\*((eta30+eta12)\*(eta30+eta12)-3\*(eta21+eta03)\*(eta21+eta03)+

(3\*eta21-eta03)\*(eta21+eta03)\*(3\*(eta30+eta12)\*(eta30+eta12)-(eta21+eta03)\*(eta21+eta03)));

phi[5]=(eta20-eta02)\*((eta30+eta12)\*(eta30+eta12)-(eta21+eta03)\*(eta21+eta03))+

4\*eta11\*(eta30+eta12)\*(eta21+eta03);

phi[6]=(3\*eta21-eta03)\*(eta30+eta12)\*((eta30+eta12)\*(eta30+eta12)-3\*(eta21+eta03)\*(eta21+eta03))+

(3\*eta12-eta30)\*(eta21+eta03)\*(3\*(eta30+eta12)\*(eta30+eta12)-(eta21+eta03)\*(eta21+eta03));

//图像离心率

phi[7]=((eta20-eta02)\*(eta20-eta02)+4\*eta11\*eta11)/((eta20+eta02)\*(eta20+eta02));

if(mode==1)

for(int i=0;i<8;i++)

{

phi[i]=fabs(log(fabs(phi[i])));

feature\_shape[i]=phi[i];

}

else

for(int i=0;i<8;i++)

{

phi[i]=fabs(log(fabs(phi[i])));

feature\_shape\_1[i]=phi[i];

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

计算颜色直方图

pathfile为图像的路径，mode为模式状态位，为1时，表示计算的是待检索图像，

为2时，表示计算的是检索库中的图像

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::general(CString pathfile,int mode)

{

CBitmap hbmp;

HBITMAP hbitmap;

//装载资源

hbitmap=(HBITMAP)::LoadImage (::AfxGetInstanceHandle(),pathfile,IMAGE\_BITMAP,

0,0,LR\_LOADFROMFILE|LR\_CREATEDIBSECTION);

hbmp.Attach(hbitmap);

//获取图像格式

BITMAP bm;

hbmp.GetBitmap(&bm);

//创建临时的内存DC对象

CDC dcMem;

dcMem.CreateCompatibleDC(GetDC());

dcMem.SelectObject(hbmp);

int width=bm.bmWidth;

int height=bm.bmHeight;

int totalnum= height \*width;

//初始化

long m\_graph[3][12];

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

{

m\_graph[i][j]=0;

if(mode==1)

feature\_color[i][j]=0.0;

else

feature\_color\_1[i][j]=0.0;

}

COLORREF color;

double h=0,s=0,v=0;

//计算颜色直方图

for(long y=0;y<height;y++)

{

for(long x=0;x<width;x++)

{

color=dcMem.GetPixel(x,y);

//颜色空间转换

RGBToHSV(GetRValue(color),GetGValue(color),GetBValue(color),&h,&s,&v);

int result\_h=(int)(6\*h/PI);

if( result\_h ==12)

m\_graph[0][11]++;

else

m\_graph[0][result\_h]++;

int result\_s=(int)(s\*12);

if( result\_s ==12)

m\_graph[1][11]++;

else

m\_graph[1][result\_s]++;

int result\_v=(int)(v\*12);

if( result\_v ==12)

m\_graph[2][11]++;

else

m\_graph[2][result\_v]++;

}

}

//保存颜色直方图

if(mode==1)

{

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

feature\_color[i][j]=((float)m\_graph[i][j])/((float)totalnum);

}

else

{

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

feature\_color\_1[i][j]=((float)m\_graph[i][j])/((float)totalnum);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

计算累计直方图

pathfile为图像的路径，mode为模式状态位，为1时，表示计算的是待检索图像，

为2时，表示计算的是检索库中的图像

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::succession(CString pathfile,int mode)

{

CBitmap hbmp;

HBITMAP hbitmap;

//装载资源

hbitmap=(HBITMAP)::LoadImage (::AfxGetInstanceHandle(),pathfile,IMAGE\_BITMAP,

0,0,LR\_LOADFROMFILE|LR\_CREATEDIBSECTION);

hbmp.Attach(hbitmap);

//获取图像格式

BITMAP bm;

hbmp.GetBitmap(&bm);

//创建临时的内存DC对象

CDC dcMem;

dcMem.CreateCompatibleDC(GetDC());

dcMem.SelectObject(hbmp);

int width=bm.bmWidth;

int height=bm.bmHeight;

int totalnum= height \*width;

long m\_graph[3][12];

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

{

m\_graph[i][j]=0;

if(mode==1)

feature\_color[i][j]=0.0;

else

feature\_color\_1[i][j]=0.0;

}

COLORREF color;

double h=0,s=0,v=0;

for(long y=0;y<height;y++)

{

for(long x=0;x<width;x++)

{

color=dcMem.GetPixel(x,y);

RGBToHSV(GetRValue(color),GetGValue(color),GetBValue(color),&h,&s,&v);

int result\_h=(int)(6\*h/PI);

if( result\_h ==12)

m\_graph[0][11]++;

else

m\_graph[0][result\_h]++;

int result\_s=(int)(s\*12);

if( result\_s ==12)

m\_graph[1][11]++;

else

m\_graph[1][result\_s]++;

int result\_v=(int)(v\*12);

if( result\_v ==12)

m\_graph[2][11]++;

else

m\_graph[2][result\_v]++;

}

}

long temp[3][12];

for(int i=0;i<3;i++)

{

for(int j=0;j<12;j++)

{

temp[i][j]=m\_graph[i][j];

}

}

for(int i=0;i<3;i++)

{

for(int j=0;j<12;j++)

{

for(int k=0;k<j;k++)

{

m\_graph[i][j]+=temp[i][k];

}

}

}

if(mode==1)

{

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

feature\_color[i][j]=((float)m\_graph[i][j])/((float)totalnum);

}

else

{

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

feature\_color\_1[i][j]=((float)m\_graph[i][j])/((float)totalnum);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

计算颜色矩

pathfile为图像的路径，mode为模式状态位，为1时，表示计算的是待检索图像，

为2时，表示计算的是检索库中的图像

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::centerM(CString pathfile,int mode)

{

CBitmap hbmp;

HBITMAP hbitmap;

//装载资源

hbitmap=(HBITMAP)::LoadImage (::AfxGetInstanceHandle(),pathfile,IMAGE\_BITMAP,

0,0,LR\_LOADFROMFILE|LR\_CREATEDIBSECTION);

hbmp.Attach(hbitmap);

//获取图像格式

BITMAP bm;

hbmp.GetBitmap(&bm);

//创建临时的内存DC对象

CDC dcMem;

dcMem.CreateCompatibleDC(GetDC());

dcMem.SelectObject(hbmp);

int width=bm.bmWidth;

int height=bm.bmHeight;

int totalnum= height \*width;

long m\_graph[3][12];

float m\_graphf[3][12];;

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

{

m\_graph[i][j]=0;

m\_graphf[i][j]=0.0;

if(mode==1)

feature\_color[i][j]=0.0;

else

feature\_color\_1[i][j]=0.0;

}

COLORREF color;

double h=0,s=0,v=0;

for(long y=0;y<height;y++)

{

for(long x=0;x<width;x++)

{

color=dcMem.GetPixel(x,y);

RGBToHSV(GetRValue(color),GetGValue(color),GetBValue(color),&h,&s,&v);

int result\_h=(int)(6\*h/PI);

if( result\_h ==12)

m\_graph[0][11]++;

else

m\_graph[0][result\_h]++;

int result\_s=(int)(s\*12);

if( result\_s ==12)

m\_graph[1][11]++;

else

m\_graph[1][result\_s]++;

int result\_v=(int)(v\*12);

if( result\_v ==12)

m\_graph[2][11]++;

else

m\_graph[2][result\_v]++;

}

}

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

m\_graphf[i][j]=((float)m\_graph[i][j])/((float)totalnum);

float m1[3],m2[3],m3[3];

for(int i=0;i<3;i++)

{

m1[i] = 0.0;

m2[i] = 0.0;

m3[i] = 0.0;

}

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

m1[i] +=m\_graphf[i][j]/12;

for(int i=0;i<3;i++)

for(int j=0;j<12;j++)

{

m2[i] +=((m\_graphf[i][j] - m1[i]) \* (m\_graphf[i][j] - m1[i]))/12;

m3[i] +=((m\_graphf[i][j] - m1[i]) \* (m\_graphf[i][j] - m1[i])

\* (m\_graphf[i][j] - m1[i]))/12;

}

float zz=1/3;

for(int i=0;i<3;i++)

{

m2[i] = sqrtf(m2[i]);

m3[i] = (float)pow( m3[i], zz );

}

if(mode==1)

{

feature\_color[0][0]=m1[0]; feature\_color[0][1]=m2[0]; feature\_color[0][2]=m3[0];

feature\_color[1][0]=m1[1]; feature\_color[1][1]=m2[1]; feature\_color[1][2]=m3[1];

feature\_color[2][0]=m1[2]; feature\_color[2][1]=m2[2]; feature\_color[2][2]=m3[2];

}

else

{

feature\_color\_1[0][0]=m1[0]; feature\_color\_1[0][1]=m2[0]; feature\_color\_1[0][2]=m3[0];

feature\_color\_1[1][0]=m1[1]; feature\_color\_1[1][1]=m2[1]; feature\_color\_1[1][2]=m3[1];

feature\_color\_1[2][0]=m1[2]; feature\_color\_1[2][1]=m2[2]; feature\_color\_1[2][2]=m3[2];

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RGB空间至HSV空间的转换

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::RGBToHSV(int r,int g,int b,double \*h,double \*s,double \*v)

{

\*h=acos((r-g+r-b)/(2.0\*sqrtf((float)(r-g)\*(r-g)+(float)(r-b)\*(g-b))));

if(b>g)

\*h=2\*PI-\*h;

\*s=(mymax(r,g,b)-mymin(r,g,b))/(float)mymax(r,g,b);

\*v=mymax(r,g,b)/255.0;

}

int CImagetrievalDlg::mymax(int a,int b,int c)//寻找最大

{

int m;

if(a>b)

m=a;

else

m=b;

if(m<c)

m=c;

return m;

}

int CImagetrievalDlg::mymin(int a,int b,int c)//寻找最小

{

int m;

if(a<b)

m=a;

else

m=b;

if(m>c)

m=c;

return m;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

结果处理及显示

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CImagetrievalDlg::sort()

{

int picnum=1009;//图像显示控件的ID号

if(method==1)

{

CString tp;//图像路径

double num;//图像与待检索图像的距离

//图像相似度排序

for(int i=0;i<counts;i++)

{

for(int j=i+1;j<counts;j++)

{

if(image\_color\_temp.num[i]>image\_color\_temp.num[j])

{

tp=image\_color\_temp.tp[i];

num=image\_color\_temp.num[i];

image\_color\_temp.tp[i]=image\_color\_temp.tp[j];

image\_color\_temp.num[i]=image\_color\_temp.num[j];

image\_color\_temp.tp[j]=tp;

image\_color\_temp.num[j]=num;

}}}

//显示前十个相似度最高的图像

for(int i=0;i<10&&picnum<=1018;i++)

{

ShowPic(image\_color\_temp.tp[i],picnum);

picnum++;

}

}

if(method==2)

{

CString tp;//图像路径

double num;//图像与待检索图像的距离

//图像相似度排序

for(int i=0;i<counts;i++)

{

for(int j=i+1;j<counts;j++)

{

if(image\_shape\_temp.num[i]>image\_shape\_temp.num[j])

{

tp=image\_shape\_temp.tp[i];

num=image\_shape\_temp.num[i];

image\_shape\_temp.tp[i]=image\_shape\_temp.tp[j];

image\_shape\_temp.num[i]=image\_shape\_temp.num[j];

image\_shape\_temp.tp[j]=tp;

image\_shape\_temp.num[j]=num;

}}}

//显示前十个相似度最高的图像

for(int i=0;i<10&&picnum<=1018;i++)

{

ShowPic(image\_shape\_temp.tp[i],picnum);

picnum++;

}

}

if(method==3)

{

CString tp;//图像路径

double num;//图像与待检索图像的距离

//图像相似度排序

for(int i=0;i<counts;i++)//综合特征相似度

{

image.num[i]=(c\_per\*image\_color\_temp.num[i]+s\_per\*image\_shape\_temp.num[i])/100;

image.tp[i]=image\_color\_temp.tp[i];

}

for(int i=0;i<counts;i++)

{

for(int j=i+1;j<counts;j++)

{

if(image.num[i]>image.num[j])

{

tp=image.tp[i];

num=image.num[i];

image.tp[i]=image.tp[j];

image.num[i]=image.num[j];

image.tp[j]=tp;

image.num[j]=num;

}}}

//显示前十个相似度最高的图像

for(int i=0;i<10&&picnum<=1018;i++)

{

ShowPic(image.tp[i],picnum);

picnum++;

}

}

}

void CImagetrievalDlg::OnBnClickedRadio1()

{

// TODO: 在此添加控件通知处理程序代码

method=1;

}

void CImagetrievalDlg::OnBnClickedRadio2()

{

// TODO: 在此添加控件通知处理程序代码

method=2;

}

void CImagetrievalDlg::OnBnClickedRadio3()

{

// TODO: 在此添加控件通知处理程序代码

method=3;

}

void CImagetrievalDlg::OnEnChangeEdit2()

{

// TODO: 如果该控件是 RICHEDIT 控件，则它将不会

// 发送该通知，除非重写 CDialog::OnInitDialog()

// 函数并调用 CRichEditCtrl().SetEventMask()，

// 同时将 ENM\_CHANGE 标志“或”运算到掩码中。

// TODO: 在此添加控件通知处理程序代码

UpdateData();

s\_per=100-c\_per;

ssper.Format("%d",s\_per);

UpdateData(FALSE);

}

void CImagetrievalDlg::OnEnChangeEdit3()

{

// TODO: 如果该控件是 RICHEDIT 控件，则它将不会

// 发送该通知，除非重写 CDialog::OnInitDialog()

// 函数并调用 CRichEditCtrl().SetEventMask()，

// 同时将 ENM\_CHANGE 标志“或”运算到掩码中。

// TODO: 在此添加控件通知处理程序代码

UpdateData();

c\_per=100-s\_per;

ccper.Format("%d",c\_per);

UpdateData(FALSE);

}

void CImagetrievalDlg::OnCbnSelchangeCombo1()

{

// TODO: 在此添加控件通知处理程序代码

UpdateData();

}