# 将软件参数记录在注册表中的框架

## 一、编写一个参数类。

编写一个参数类，用来记录参数和进行注册表读写操作，在这里我编写了一个记录算法参数的类，**代码见附录**，主要功能说明如下：

在类中，编写一个结构体，用来保存算法参数，参数类中定义一个该结构体对象，这样，在读写注册表时只需要一次读写就能完成多个算法参数的读写操作。但是记得重载等号运算符，方便类间数据交换：

const CAlgorithParam &operator=(const CAlgorithParam& InputParam);

编写读、写注册表函数：

BOOL SaveToRegistry(HKEY hKey, CString KeyName);

BOOL LoadFromRegistry(HKEY hKey, CString KeyName);

以上两个函数接收一个已打开的子键名称和一个键值名，主要用到RegQueryValueEx函数和RegSetValueEx函数，函数的使用不再赘述，请参考有关书籍。

代码如下：

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

函数名：SaveToRegistry

参 数：hKey 已打开的注册表键

KeyName 键名

说 明：把参数保存到注册表中，执行之前确保指定键已打开，执行完后请关闭该键

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

BOOL CAlgorithParam::SaveToRegistry(HKEY hKey, CString KeyName)

{

long setinfo = RegSetValueEx(hKey, KeyName, 0, REG\_BINARY, (LPBYTE)&Param, sizeof(Param));

if (setinfo != ERROR\_SUCCESS)

{

return FALSE;

}

return TRUE;

}

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

函数名：LoadFromRegistry

参 数：hKey 已打开的注册表键

KeyName 键名

说 明：从注册表中读取参数，执行之前确保指定键已打开，执行完后请关闭该键

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

BOOL CAlgorithParam::LoadFromRegistry(HKEY hKey, CString KeyName)

{

DWORD dwType;

DWORD dwLen;

long readinfo = RegQueryValueEx(hKey, KeyName, NULL, &dwType, NULL, &dwLen);

if (readinfo != ERROR\_SUCCESS)

{

return FALSE;

}

readinfo = RegQueryValueEx(hKey, KeyName, 0, &dwType, (LPBYTE)&Param, &dwLen);

if (readinfo != ERROR\_SUCCESS)

{

return FALSE;

}

return TRUE;

}

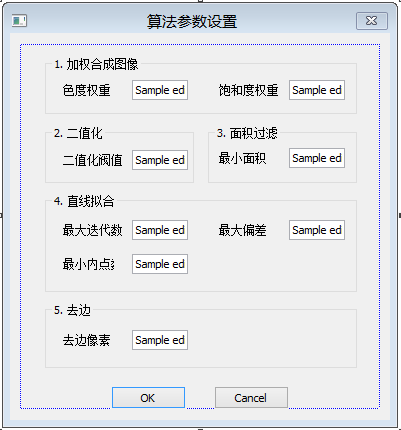
## 二、在主程序中打开注册表

由于参数类只负责实现从指定子健读取参数和将参数写入到指定子健，因此打开子健的操作放在主程序中进行。

无论是要读参数还是要写参数，首先需要打开指定键，主要使用到函数RegCreateKeyEx，函数用法不在赘述，请参考有关书籍和**附录代码**。打开子键后，使用参数类的SaveToRegistry和LoadFromRegistry即可实现参数写入和读取。

## 三、实现参数操作对话框

在这里，我设计如下的对话框：



生成对话框类，除了控件所关联的变量外，再定义一个参数类对象，方便与主程序的数据交换。

## 四、主程序中调用参数设置对话框进行参数设置的方法

实现了参数类后，设置参数、保存参数、读取参数变得很简单，代码如下：

void CVisionNavigationDlg::OnBnClickedAlgorithmssetting()

{

LoadParams();

CAlgorithmSetDlg dlg;

dlg.Algorithparam = Params;

if (dlg.DoModal() == IDOK)

{

Params = dlg.Algorithparam;

SaveParams();

}

}

这里要说明的是，由于上述代码中LoadParams函数可以会失败，例如在第一次运行的时候，但由于参数类的构造函数中对各参数进行了初始化，因此保证了参数合法有效。

## 五、其它说明

**5.1 在Windows8 中修改注册表要有管理员权限，否则会失败。**

**5.2 考察“四 主程序中调用参数设置对话框进行参数设置的方法”中的代码，在对话框弹出之前，将当前程序参数传给对话框中的参数类对象，由于对话框中的控件值是分别关联相应的值的，而不是关联整个参数结构体，所以如果不将参数结构体中的各个值传递给各个控件关联的值，那么，当对话框弹出的时候，控件中的值与参数结构体的值不符，因此，需要在设置对话框的OnInitDialog()中将参数结构体的值传给控件关联变量。注意，不能在对话框的构造函数中完成这些工作，因为构造函数发生在定义对话框过程中，即这段代码：**CAlgorithmSetDlg dlg;**，这时候还有将程序参数值传递到设置对话框类对象中，即还没有执行这一步：**dlg.Algorithparam = Params;例如，在我的程序中，我对设置对话框的OnInitDialog函数进行如下处理：

BOOL CAlgorithmSetDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

m\_nBinaryThreshold = Algorithparam.Param.m\_nBinaryThreshold;

m\_nMinArea = Algorithparam.Param.m\_nMinArea;

m\_nMaxTimes = Algorithparam.Param.m\_nMaxTimes;

m\_nMaxError = Algorithparam.Param.m\_nMaxError;

m\_nMinInnerPoints = Algorithparam.Param.m\_nMinInnerPoints;

m\_nSurroundPoints = Algorithparam.Param.m\_nSurroundPoints;

m\_nColorCenter = Algorithparam.Param.m\_nColorCenter;

m\_nColorWidth = Algorithparam.Param.m\_nColorWidth;

m\_ctrHueWeight.SetFloat(Algorithparam.Param.m\_dHueWeight);

m\_ctrSaturationWeight.SetFloat(Algorithparam.Param.m\_dSaturationWeight);

UpdateData(FALSE);

return TRUE;

}

## 附录：

### 1、参数类代码头文件

#pragma once

#include "StdAfx.h"

//算法参数类

struct AlgorithParamStruct

{

// 色度权重

double m\_dHueWeight;

// 饱和度权重

double m\_dSaturationWeight;

// 二值化阀值

int m\_nBinaryThreshold;

// 最小面积

int m\_nMinArea;

// 最大迭代次数

int m\_nMaxTimes;

// 最大偏差

int m\_nMaxError;

// 最小内点数

int m\_nMinInnerPoints;

// 除去周围环形区域宽度

int m\_nSurroundPoints;

};

class CAlgorithParam

{

public:

CAlgorithParam();

~CAlgorithParam();

BOOL SaveToRegistry(HKEY hKey, CString KeyName);

BOOL LoadFromRegistry(HKEY hKey, CString KeyName);

// 重载=运算符

const CAlgorithParam &operator=(const CAlgorithParam& InputParam)

{

if (this == &InputParam)

return \*this;

// 色度权重

Param.m\_dHueWeight = InputParam.Param.m\_dHueWeight;

// 饱和度权重

Param.m\_dSaturationWeight = InputParam.Param.m\_dSaturationWeight;

// 二值化阀值

Param.m\_nBinaryThreshold = InputParam.Param.m\_nBinaryThreshold;

// 最小面积

Param.m\_nMinArea = InputParam.Param.m\_nMinArea;

// 最大迭代次数

Param.m\_nMaxTimes = InputParam.Param.m\_nMaxTimes;

// 最大偏差

Param.m\_nMaxError = InputParam.Param.m\_nMaxError;

// 最小内点数

Param.m\_nMinInnerPoints = InputParam.Param.m\_nMinInnerPoints;

// 除去周围环形区域宽度

Param.m\_nSurroundPoints = InputParam.Param.m\_nSurroundPoints;

}

public:

AlgorithParamStruct Param;

};

### 2、参数类代码源文件

#include "stdafx.h"

#include "AlgorithParam.h"

CAlgorithParam::CAlgorithParam()

{

// 色度权重

Param.m\_dHueWeight =0.4;

// 饱和度权重

Param.m\_dSaturationWeight = 0.6;

// 二值化阀值

Param.m\_nBinaryThreshold = 150;

// 最小面积

Param.m\_nMinArea = 500;

// 最大迭代次数

Param.m\_nMaxTimes = 100;

// 最大偏差

Param.m\_nMaxError = 2;

// 最小内点数

Param.m\_nMinInnerPoints = 20;

// 除去周围环形区域宽度

Param.m\_nSurroundPoints = 10;

}

CAlgorithParam::~CAlgorithParam()

{

}

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

函数名：SaveToRegistry

参 数：hKey 已打开的注册表键

KeyName 键名

说 明：把参数保存到注册表中，执行之前确保指定键已打开，执行完后请关闭该键

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

BOOL CAlgorithParam::SaveToRegistry(HKEY hKey, CString KeyName)

{

long setinfo = RegSetValueEx(hKey, KeyName, 0, REG\_BINARY, (LPBYTE)&Param, sizeof(Param));

if (setinfo != ERROR\_SUCCESS)

{

return FALSE;

}

return TRUE;

}

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

函数名：LoadFromRegistry

参 数：hKey 已打开的注册表键

KeyName 键名

说 明：从注册表中读取参数，执行之前确保指定键已打开，执行完后请关闭该键

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

BOOL CAlgorithParam::LoadFromRegistry(HKEY hKey, CString KeyName)

{

DWORD dwType;

DWORD dwLen;

long readinfo = RegQueryValueEx(hKey, KeyName, NULL, &dwType, NULL, &dwLen);

if (readinfo != ERROR\_SUCCESS)

{

return FALSE;

}

readinfo = RegQueryValueEx(hKey, KeyName, 0, &dwType, (LPBYTE)&Param, &dwLen);

if (readinfo != ERROR\_SUCCESS)

{

return FALSE;

}

return TRUE;

}

### 3、主程序中载入参数和保存参数的代码

// 从注册表载入参数

BOOL CVisionNavigationDlg::LoadParams()

{

LPCTSTR lpPath = RegMainPath;

HKEY subKey;

DWORD dwExist;

//打开注册表

LONG result = RegCreateKeyEx(HKEY\_LOCAL\_MACHINE, lpPath, 0, NULL, REG\_OPTION\_NON\_VOLATILE,

KEY\_ALL\_ACCESS, NULL, &subKey, &dwExist);

DWORD dwError = GetLastError();

if (result != ERROR\_SUCCESS)

{

return FALSE;

}

else

{

if (!Params.LoadFromRegistry(subKey, RegKeyName))

{

RegCloseKey(subKey);

return FALSE;

}

else

{

RegCloseKey(subKey);

return TRUE;

}

}

}

// 将参数写入注册表

BOOL CVisionNavigationDlg::SaveParams()

{

LPCTSTR lpPath = RegMainPath;

HKEY subKey;

DWORD dwExist;

//打开注册表

LONG result = RegCreateKeyEx(HKEY\_LOCAL\_MACHINE, lpPath, 0, NULL, REG\_OPTION\_NON\_VOLATILE,

KEY\_ALL\_ACCESS, NULL,&subKey, &dwExist);

DWORD dwError = GetLastError();

if (result != ERROR\_SUCCESS)

{

return FALSE;

}

else

{

if (!Params.SaveToRegistry(subKey, RegKeyName))

{

RegCloseKey(subKey);

return FALSE;

}

else

{

RegCloseKey(subKey);

return TRUE;

}

}

}