文章编号:1671-4598(2016)02-0307-04 DOI:10.16526/j.cnki.11—4762/tp.2016.02.085 中图分类号:TP319 文献标识码:A

图像处理应用组态开发系统的设计与实现

刘泽桂，李 迪，王世勇，张春华
(华南理工大学 机械与汽车工程学院，广州 510640)

摘要：传统的文本编程方式效率低下，无法满足图像处理应用迅速增长的需求，针对这个问题，提出一种基于图形化组态编程的开发系统；从功能和结构上对系统进行分析，着重讨论编程环境的设计与实现；采用组合模式管理应用程序、迭代器模式实现遍历，进一步讨论了编程环境的仿真运行机制；基于MFC设计界面，通过一般—特殊结构实现界面复用，利用策略模式实现程序树的重建；使用OpenCV和动态链接库技术设计预定义函数集；最后，对运行环境的数据模型做出了探讨；实例验证，该系统能减少开发时间，降低开发难度，为图像处理相关人员提供了一个简单友好的二次开发环境。

关键词：组态；图形化编程；图像处理；组件技术

Structure Design and Implementation of a Configuration System for Image Processing Application Development

Liu Zegui, Li Di, Wang Shiyong, Zhang Chunhua
(School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou 510640, China)

Abstract: Given the fact that traditional programing method falls behind demand for image processing programs, this paper proposes a graphic-based developing system for image processing programing. Function and structure of the software are analyzed, as a part of which, programming environment is mainly discussed. Composite pattern is employed for program management, iterator pattern for traversal operation, and simulation mechanism is given afterwards. MFC, generalization-specialization structure and strategy pattern are adopted for interface design, interface reuse and program tree reconstruction respectively. OpenCV library and Dynamic Link Library technology are taken as support for predefined function set. At last, data model of the running environment is discussed. Examples show that the software reduces development time and lowers its difficulty, presenting a friendly secondary development platform for image processing.

Keywords: configuration; graphic programing; image processing; component technique

0 引言

随着计算机技术的发展，图像处理技术得到越来越广泛的应用，图像处理应用程序的需求量随之增加。然而，图像处理是一个复杂的过程，图像处理任务往往需要由多种算法协同完成工作[1]，传统的文本式编程要求开发人员了解算法库函数的接口，编写连接代码，以形成完整正确的程序。这对初学者入门形成一定难度，同时也增加了开发者的开发难度。

针对市场需求和发展现状不协调的矛盾，本文提出一种通用的面向图像处理应用的组态开发系统，将图像处理算法封装成图形控件，通过图形化组态的编程方式，形成图像处理程序。图形化编程具有程序结构清晰、编程出错率低、组态较为灵活等优点，通过这样的方式，软件能降低初学者入门门槛，减少开发人员的开发时间和开发难度，从而提高开发效率。

收稿日期:2015-09-21； **修回日期**:2016-01-06。
基金项目:国家科技支撑计划项目(2015BAF20B01);广东省科技计划项目(2015B010101005)。
作者简介:刘泽桂(1991-),男,广东潮州人,硕士研究生,主要从事嵌入式系统、计算机应用方向的研究。
李 迪(1965-),女,山东青岛人,教授,博士生导师,主要从事嵌入式系统、自动控制和机器视觉方向的研究。

1 功能和结构分析

1.1 系统组成

组态开发系统通过图形化编程的方式形成应用程序，输出特定格式的数据文件，实现应用程序脱离编程环境运行。在这种模式下，应用程序的编辑一般通过编程环境的交互界面实现，应用程序的运行则依赖于运行环境对数据文件的正确解析。系统的结构组成及模块间的相互联系如图1所示。

图1 系统结构组成

编程环境需要实现的部分功能如下：

1）应用程序编辑：拖放图元形成程序，利用交互界面配置算法参数。

2）应用程序仿真：程序编辑过程中实时仿真运行，通过显示界面使当前配置参数下的运行结果可视化。

3）程序运行监视：在监视面板中显示输入图像及自定制监视数据项的当前结果。

4）数据文件生成：输出图形化编程形成的应用程序，提

供特定格式的数据文件，实现应用程序脱离编辑平台运行。

5）工程存储：支持编程环境的工程存储/打开等功能。

数据文件主要由算法的配置参数、算法间的数据联系、应用程序的结构组成等信息构成。

运行环境则用于解析数据文件、运行应用程序、执行图像处理任务。针对不同的运行平台，可定制相应的运行环境，确保系统的可移植性。

1.2 编程环境的功能结构划分

图形化编程基于数据流可视化程序语言，利用预定义函数库及图形化语法，通过连接功能模块图元形成应用程序[2]。一个典型的数据流可视化程序语言提供必需的数据结构和预定义函数库，以支撑应用程序设计时可能涉及的复杂计算[3]。在此基础上，从功能和结构上对编程环境进行划分[4]，如图 2 所示。

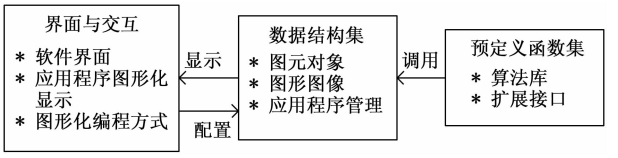

图 2 软件的功能结构

软件运行过程中，用户通过界面和交互编写程序、配置工具，软件内部则根据用户操作修改数据结构集中的相关数据；图元对象调用预定义函数集的图像处理函数进行仿真运行，并将结果在界面中显示出来，实现编程过程中当前结果的实时显示。

2 关键技术及实现

2.1 数据结构集

数据结构集定义了软件必要的数据结构，描述图元对象、图形图像在软件中的存在形式，并以合理的方式管理应用程序，为软件提供数据基础。

2.1.1 应用程序的管理

通常情况下，应用程序呈树形结构，设计时，将分支、循环等复杂程序结构抽象成逻辑控制图元，程序流程视为逻辑控制图元的子分支，则组件对象组成分支辅助图元，分支辅助图元组成逻辑控制图元，逻辑控制图元本身也是一个组件，可以组成更大的结构。这样一来，形成一种典型的“部分－整体”的层次结构，采用组合模式[5]进行设计，使客户区对单个对象和组合对象的使用具有一致性。

组合模式的关键在于组件对象的抽象基类，此处为 CImgTool 类，简单的算法图元以及复杂组合对象都派生于 CImgTool。设计时，在基类中定义组合对象和单个对象共同的操作接口，如串行化接口函数 Serialize（）等。实现时，组合对象调用其子部件的接口，通过递归实现接口操作。

图元对象派生于 CImgTool 基类，统一使用基类指针 CImgTool * 进行管理，充分发挥面向对象的多态特性，通过动态绑定实现具体接口的调用。对于指针的管理，则采用 MFC 模板类 CTypedptrArray，为 CObjeet 类或其子类的对象提供类型安全保证，防止由不匹配的指针类型引起的错误。定义如下：

```
CTypedPtrArray<CObArray,CImgTool * > m_Branch;//分支辅助图元的子图元对象列表
CTypedPtrArray<CObArray,CAuxNode * > m_Branchs; //控制图元的子分支列表
CTypedPtrArray<CObArray,CImgTool * > m_ImgToolArr; //任务类最外层图元对象列表
```

2.1.2 应用程序的遍历

应用程序的数据存储在文档类中，文档类提供相应的访问列表元素的接口，实现不同的遍历方式。采用迭代器模式，将对列表的访问和遍历从列表对象中分离出来，放入迭代器中。在迭代器类中定义访问列表元素的接口，负责跟踪当前的元素。迭代器基类的定义如下：

```
class CIteratorIf{
public:
  CIteratorIf(const List * pList);
  ~CIteratorIf(void);
  virtual void First(void) = 0;
  virtual void Next(void) = 0;
  virtual CImgTool * CurrentItem(void) const = 0;
  virtual bool IsDone(void)const = 0;
private:
  const List * m_pList;
  stack<CImgTool * > m_Stack;
};
```

迭代器类与列表紧密耦合，实例化迭代器之前，需要提供待遍历的列表，类中使用指针 pList 引用列表数据。First 操作初始化迭代器，指向第一个元素；Next 操作将当前指针推进一步，指向下一个元素；IsDone 操作检查是否遍历结束；CurrentItem 操作返回当前元素。

将遍历算法封装在迭代器内部，不同类型的迭代器有不同的实现方式。以完全遍历迭代器为例，First 操作将 m _ ImgToolList 的元素依次压入栈中；Next 操作判断栈顶元素是否为控制图元，若是，则当前元素出栈，再依次将 m _ Branchs 中辅助图元的子组件列表 m _ Branch 中的元素压入栈中，否则，则直接弹出当前元素。使用迭代器时，伪代码为：

```
CIterator it = CreateTraverslIter();
for (it.First(); ! it.IsDone(); it.Next()){
  do something with CurrentItem();
}
```

2.1.3 仿真运行机制

仿真运行操作从文档中读取数据，解析出应用程序的逻辑结构，进行图像处理。软件的仿真运行机制如图 3 所示。

其中任务列表进行最外层遍历，控制图元子流程则通过组合模式的 Process（）接口进入。

2.2 界面与交互

界面与交互定义了应用程序的可视化形式及编程形式，提供清晰明了的图形用户界面，使应用程序直观可视，编程操作简单方便。

2.2.1 软件框架及主界面

在可视化编程环境下进行人机界面的开发，能简化设计工作，显著提高开发效率[6]。VS2012 平台及 MFC 框架提供了

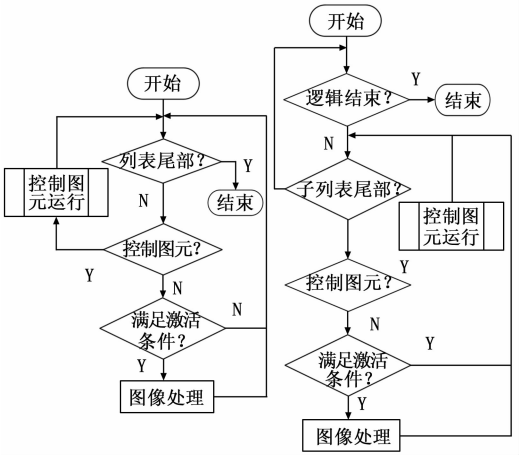

图 3　程序仿真运行流程图

一个这样的编程环境，在此基础上进行开发。结合 MFC 的文档/视图架构和基于对话框程序两种框架的优势，基于对话框进行设计，在应用程序类中引入文档类，在文档类中实现数据的管理，从而既实现数据和数据显示的分离，又能以“所见即所得”的方式设计界面，简便直观。

软件主界面基于框架窗口 CFrameWndEx 进行搭建，使用工具栏、窗口分割、停靠窗口等技术形成主界面，使用抽屉式菜单分类显示图像处理工具库，主界面布局如图 4 所示。

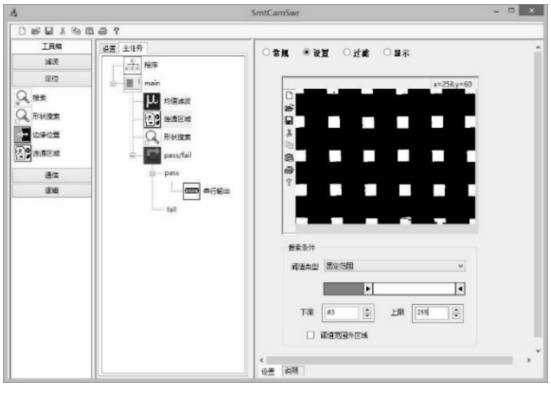

图 4　软件主界面

2.2.2　一般－特殊结构实现界面复用

软件中，某多个界面具有许多共同特征，设计时，采用一般－特殊的结构，将共同属性和操作抽象成具有一般性的基类，特殊类继承一般类，从而实现界面的复用，减少重复代码。

以配置界面为例，结合泛型编程的思想，将配置界面抽象成通用容器类，通过构造函数传参实例化具体的配置界面对象。抽象时，需要消除配置界面间的差异，主要解决单选按钮的显示文本和切换面板消息回路这两个问题。通用容器类部分定义如下：

```
class CDispPaneContainer : public CMyFormView
{
public:
    CDispPaneContainer(CImgTool * pImgTool, const std::vector
<UINT> &IDList); //构造函数
    virtual ~CDispPaneContainer();
private:
    std::vector<UINT> m_IDList;//单选按钮项显示文本字符串ID组
    CTypedPtrArray<CObArray, CButton * > m_pRBtn; //单选按钮组
    CMyFormView * m_pChildPane;//当前配置面板
    void OnRBnClk(); //自定义 BN_CLICKED 消息响应函数
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void * pExtra, AFX_CMDHANDLERINFO * pHandlerInfo); //消息路由函数
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct); //设置窗口
    ……
};
```

用参数 IDList 在资源视图的 String Table 中预定义单选按钮的字符串 ID 组，容器类在 OnCreate（）函数中通过字符串 ID 和 LoadString（）函数获得按钮相应显示文本；改写 OnCmdMsg（）消息路由函数，捕捉单选按钮的 BN_CLICKED 消息，并分发给自定义响应函数 OnRBnClk（），通过简单工厂函数实例化相应的配置面板对象，从而创建了切换面板的消息回路。

2.2.3　策略模式实现程序树的重建

文档通过指针管理组件对象，重建时，当前指针指向对象的类型不同，添加节点的策略有所差异，主要有普通算法结点、辅助结点、控制图元结点 3 种不同的结点添加情况。

采用策略模式，将添加节点的方式封装成一系列的策略，策略彼此间可以相互替换，客户区以相同的方式调用不同的策略。策略模式类如图 5 所示。

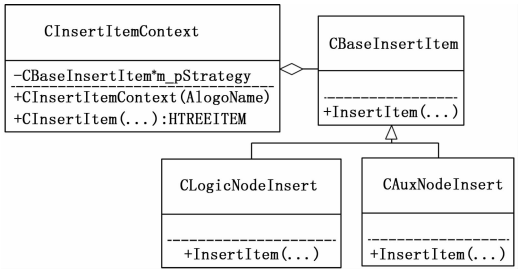

图 5　策略模式类图

定义一个上下文类 CInsertItemContext，类中用策略类基类指针 m_pStrategy 管理具体策略类，利用简单工厂方法在上下文类的构造函数中实例化具体策略类。对于基本策略类 CBaseInsertItem，添加结点时，若前驱兄弟结点及兄弟结点的父结点不为空，则添加为后继兄弟结点；不然，则以程序树当前最后结点为父结点添加新结点。对于逻辑结点添加策略 CLogicNodeInsert，则先按 CBaseInsertItem 类添加逻辑结点，再依次添加辅助结点及相应分支。辅助结点添加策略 CAuxNodeInsert 遍历分支辅助图元子组件列表 m_Branch，再使用 CInsertItemContext 类，依次添加结点。

2.3　预定义函数集

预定义函数集定义了通用、全面而高效的图像处理算法

库，预留了良好的扩展接口，在算法层面为系统及应用程序提供强大的支持。

OpenCV（Open Source Computer Vision Library）是一个开源的跨平台计算机视觉库，提供了图像处理和计算机视觉方面的很多通用算法，在此基础上定制图像处理算法库，可以缩短软件开发周期，并保证图像处理算法的通用性和高效性。在VS2012平台中通过动态链接库技术（Dynamic Link Library，DLL）调用OpenCV的库函数前，需要在开发环境中对项目进行配置，若OpenCV安装路径为D：/Program Files，配置如下：

1）在属性管理器的包含目录项中添加OpenCV文件的include、opencv和opencv2三个文件夹的路径：

D:/ Program Files/ opencv/ build/ include

D:/Program Files/opencv/build/include/opencv

D:/Program Files/opencv/build/include/opencv2

2）在库目录项中添加引入库文件文件夹的路径：

D:/Program Files/opencv/build/x86/vc11/lib

3）在链接器的输入中添加附加依赖项，使用OpenCV3.0时，附加依赖项为opencv _ world300d. lib、opencv _ world300.lib等。

编程环境使用隐式链接的方式调用链接库，方便库函数的使用，并且在软件界面初始化时载入DLL，避免初次调用时因载入DLL导致操作迟滞而影响用户体验。

3 运行环境的数据模型

运行环境的数据模型定义了数据的流动形式，主要涉及两个方面：算法结点以及结点间的数据联系。有两种不同的解决方案：

1）面向过程：算法结点作为应用程序的执行步骤，封装成函数，应用程序在主函数中调用子函数，并对结点所依赖、所产生的数据统一进行管理。

2）面向对象：将算法结点抽象成类，在类的内部独立管理参数、算法间数据联系等相关数据。应用程序实例化结点对象，通过调用接口函数管理对象。

采用面向过程的方案，则主函数几乎承担了程序的全部职责，工程只由极少数的源文件以及相应的头文件和库文件所构成，构建十分方便。然而，此方案中，主函数职责重大，代码耦合度高，不利于程序的调试和维护。

采用面向对象的方案，则数据管理的职责分散在各个算法类中，高内聚低耦合，程序的封装性、复用性、可维护性能得到更好的保障。然而，采用该方案进行设计，每个算法类都有各自的头文件和源文件，工程较为庞大，缺少集成开发环境时，构建过程相对比较麻烦。

对于图像处理应用而言，算法间存在数据联系，尤其是图像方面的联系，前驱算法的输出往往作为后继算法的输入。若采用面向过程的方案，对于每个输出，主函数中都有相应的数据缓冲区，算法步骤或算法内部输出较多时，数据缓冲区十分散乱，数据难以管理。综合考虑，采用面向对象的方案进行设计，并在PC机上开发运行环境，利用PC机资源丰富的特点，使用各种集成开发环境和工具辅助开发，解决工程构建复杂的问题。

4 应用示例

以LED固晶过程为例，某工序查找输入图像中LED晶元的个数，若查找成功，则输出目标形状的个数及位置；查找失败，则不做处理，等待下一次触发。针对该任务，首先对输入图像进行滤波去噪；然后对图像进行二值化处理，提取连通区域，去除无关灰度信息；最后根据输入模板进行形状匹配。在编程环境中输出应用程序的数据文件，通过运行环境执行任务，程序运行结果正确，且与仿真结果完全一致。运行时一次触发的结果如图6所示。

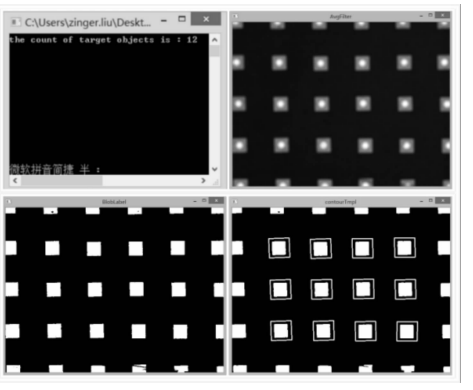

图6 图像处理任务执行结果

5 结束语

本文基于组态图形化编程开发了面向图像处理应用的二次开发系统。该系统将图像处理算法封装成图形控件，用户通过拖放的方式编辑程序树，通过交互界面配置算法参数，最终输出应用程序的数据文件，配合运行环境执行图像处理任务。软件操作简单，功能完备，界面友好，具有良好的工程实用价值，值得进一步推广应用。

参考文献：

[1] 王建新，陆炜妮，王伟平. 基于组件的数字图像处理仿真系统的设计与实现 [J]. 系统仿真学报，2004，16（6）：1213－1216.

[2] 徐小良，刘阳 等. 图形化编程平台的结构设计及实现 [J]. 计算机工程与应用，2001，37（4）：4－5，8.

[3] 程 钊，张天序，卢海风. 基于图形语言的图像处理算法开发环境模型 [J]. 华中科技大学学报：自然科学版，2010：82－85.

[4] 鲍贤捷，陈卫东，曹其新. 机器人图标化编程环境的设计及实现 [J]. 机器人，2006，28（6）：617－622.

[5] 殷 飞，丁维明. 组态软件设计中的模式研究 [J]. 计算机测量与控制. 2005，13（3）：298－300.

[6] 邵维忠，刘 昕. 可视化编程环境下人机界面的面向对象设计 [J]. 软件学报，2002，13（8）：1494－1499.