第 3 卷 第 9 期
1998 年 9 月

中国图象图形学报
Journal of Image and Graphics

Vol. 3, No. 9
Sep. 1998

一个可视化集成图象处理环境

范成法　叶秀清　顾伟康

(浙江大学信电系,杭州　310027)

摘　要　提出了图象处理的一个可视化集成环境,提供了图象和图象分析的可视化以及图象处理算法的集成。在此基础上研究图象处理算法或构造图象理解系统将大大提高工作效率。

关键词　图象处理环境,图象处理可视化,图象表示,混合编程

1　引　言

从图象处理被确立为一门学科以来,关于这一方面的研究成果日新月异。然而在实际工作中我们发现,缺乏一个有效的工具将图象处理和图象分析的各个过程综合起来。于是在一段处理程序中,不得不重复进行读入、存储和显示图象等繁琐的基本操作;如果希望在当前处理的基础上再使用其它一些处理,又要退出去运行另外一个处理程序来然后再运行显示程序,显示处理结果。如此凌乱和繁琐的过程,势必影响工作效率。如果能将繁琐的基本操作包装起来,将各种图象处理方法集成起来,并且使图象和图象分析可视化,将是十分有益的,也符合软件发展的潮流。

图象处理和编辑已经有了许多辅助工具,常见的有 PhotoStyler、PhotoShop 等。我们可以利用这些工具做一些工作,也应该吸收其中的优点,但要利用这些来开发用户的处理程序却不适合,因为没有源代码可利用。因此,需要用某种编程语言来开发可扩展的可视化集成处理环境。Delphi 语言正好满足这个要求。这是一种 Windows 下的强大的可视化编程开发工具,从一出现就风靡全球。其包装新颖、组件齐全、考虑周全,真正做到了可视化;风格优美,使用扩充的面向对象的 Pascal 语言,功能和风格上接近于 C++,而编译速度远非 C++可比。再加上其完整的数据库开发功能,深受编程人员喜爱。Delphi 良好的特性使得用它来实现整个环境是很合适的。特别是 Delphi 中提供了 TImage 组件(Component),非常适合于显示图象,减少了许多不必要的麻烦。对于习惯于使用 C++来编程的用户,通过 DLL(动态链接库)技术,也能保证图象处理的核心代码完全用 C++来书写。

收稿日期:1997-09-01

2　图象的表示

图象处理系统中常用的图象有灰度图象和彩色图象两种,另外一种常见的二值图象属于灰度图象的特例,下面主要考虑彩色图象。在深入讨论图象的表示之前,先给出用 Pascal 代码写的彩色图象的定义,灰度图象和彩色图象类似,只是少了 2 个分量。

```
type
  TByteArray = array [0..2047]of Byte;
  PByteArray = ^ TByteArray;
  T2DByteArray = array [0..2047]of PByteArray;
  P2DByteArray = ^ T2DByteArray;
  TClrPhoto = class // class of color photo
public
  RowSize,ColSize:Word;
  pRed,pGreen,pBlue:PByte;
  R,G,B:P2DByteArray;
  TheBitmap:TBitmap;
  pDispBuffer:PByte;
  constructor Create(Row,Col:Word);
  destructor Free;
  function LoadFromFile(FileName:PChar):Boolean;
```

```
    function SaveToFile(FileName:PChar):Boolean;
    procedure CreateDispBuffer;
    function GetBitmap(Sender:TForm):HBitmap;
      ... ...
end
```

从上述定义中已经可以看到，一幅图象用一个类来表示非常合适，将一些繁琐而雷同的操作包装起来之后，用起来就方便多了。一幅图象，首先要知道其大小，在彩色图象类中，以 RowSize，ColSize 来表示。图象的三基色存储实体指针以 pRed，pGreen 和 pBlue 来表示。在创建彩色图象类的实例时，首先将 RowSize 和 ColSize 记住，然后为图象存储实体指针 pRed，pGreen 和 pBlue 分配 RowSize * ColSize 大小的空间。为了能以数组的方式调用各个象素值，避免对 pRed，pGreen，pBlue 指针的直接操作，特别定义了 R、G、B 3 个变量。每一变量是指向数组的指针，而此数组中的每一元素是指向另一 Byte 型数组的指针。在 Delphi 中，数组和指针不象 C++语言中那样可以互换，但在 Delphi2.0 以上的版本中，可以将指向某一类型变量的指针赋给一个指向某一类型数组的指针，而后一指针可以当作数组来用，从而可以实现对指针分配空间的数组操作。在上述代码中，P2DByteArray 就是为此而定义的相当于二维数组的指针类型。当然，为了能让指针指向正确的位置，还要在构造函数中加入以下代码：

```
R:=AllocMem(RowSize * SizeOf(PByteArray));
G:=AllocMem(RowSize * SizeOf(PByteArray));
B:=AllocMem(RowSize * SizeOf(PByteArray));
for I:=0 to RowSize-1 do
begin
  R[I]:=Pointer(Longint(pRed)+I * ColSize);
  G[I]:=Pointer(Longint(pGreen)+I * ColSize);
  B[I]:=Pointer(Longint(pBlue)+I * ColSize);
end
```

在创建好图象类后，就可以用 **R**[I][J]或 **R**[I,J]等调用象素点了，在 Delphi2.0 以上的版本中，^**R**[I]^[J]与 **R**[I]^[J]或 **R**[I][J]或 **R**[I,J]等价。

表示图象，除了上述的变量以外，还要一些操作的函数，或称方法(Method)。图象读入和存储的方法必不可少。图象的格式多种多样，这里选用最常用的 DAT 格式和 Windows BMP 格式，作为读入、处理和存储的格式。其它格式的图象可用例如 Hijaak 之类的图象转换工具转化为 BMP 格式。DAT 格式的图象事先知道大小，由 2 个函数 LoadFromFile 和 SaveToFile 来完成读入和存储的操作。BMP 格式的图象事先不知大小，所以先读入一个 Bitmap 中，知道大小后再构造类的实例。细致的工作还包括很多其它的方法，这里不再一一列举。

图象可视化的关键在于与主窗口上的一个 TImage 组件建立关联，关联的核心是 TImage 组件的 Picture.Bitmap 属性(Property)。为此，在彩色图象类中定义一个 TheBitmap 变量，需要显示时用 API 函数 CreateDIBitmap 创建图象的位图，将句柄赋给 TheBitmap.Handle，再将 TheBitmap 赋给 TImage 组件的 Picture.Bitmap 属性，即可正确显示图象。

3　可视化集成环境的建立

图象处理的可视化集成环境，不仅应该将图象处理的各种方法集成在一起，具有很好的扩展性，而且应该提供图象、图象处理过程和图象分析的可视化，以及图象各种处理方法的可视化。

3.1　建立图象和信息显示控件

图象处理程序的主体是图象，多数中间和最后处理结果也以图象的形式存在，所以显示图象是最为重要的。Delphi 提供了 TImage 组件，我们在窗口面板上放置了 2 个大的和 4 个小的 TImage 组件。大的用来显示原始图象和处理结果图象，小的用来显示缩小的原始图象及其 R、G、B 分量。在程序的内部，同时定义了一个 BWOrgPhoto 和一个 ClrOrgPhoto 变量，用来存放灰度原始图象和彩色原始图象，原始图象处理的依据，而与 TImage 组件关联显示时，就实现了可视化。结果图象可能不只一个，所以要为它们建立一个翻页的机制，TTabSet 组件满足这一要求，并且结果图象及其位图可以存放在 TTabSet 组件的 Tabs.Objects 数组中，可供同时处理多幅图象的情况下当作原始图象使用。

图象处理过程中的文字信息可以直接写到一个 TMemo 组件上，在与 C++语言混合编程时，可以通过消息传送方法间接写入文字信息。

一行的信息也可以显示在 TPanel 组件上，可以将一些简短的图象信息例如图象的某一位置的 R、G、B 象素值显示到某一 TPanel 组件上，而另一 TPanel 组件上可以显示当前的工作时间。可视化的方法是多种多样的，不必把思路完全集中在主窗口

上。为了显示一个简短消息，可以弹出一个消息框；输入一个数值可使用输入框；对于复杂的情况可以再设计一个窗体(Form)，例如必要时设计一个统计、显示和分析直方图的窗体。

3.2 建立多种图象处理方法

为了达到集成的目的，系统提供了大部分关于图象的基本操作，省去了进一步开发可能遇到的繁琐的工作。同时集成了许多常用图象处理算法，使得图象进一步分析更加简便。这些操作和处理算法的内核是一个个函数或过程，可供继续开发调用，而操作则通过菜单、快捷键和弹出式菜单来进行。菜单中File，Edit，View 和 Help 为系统菜单。File 提供图象的文件操作以及系统退出；Edit 菜单与 Windows 剪贴板相关，实现与其它相应软件的资源共享；View 包含窗口切换和系统选项。另外一些菜单提供常用图象处理算法或用户开发算法的入口。这样，用户在这样一个集成环境里，可以调用已有的算法进行调试，不必退出程序，并且处理结果能马上显示，真正实现了集成和可视化。

为一些常用操作建立快捷键，给常用组件建立弹出式菜单，给菜单项加上有意义的图标，都使得集成环境可用性进一步提高。

4 混合编程的实现

只用 Delphi 完全能编写图象处理程序，但实际上许多已有的图象处理程序是用 C++写成的，同时 C++由于其灵活性也深受人们喜爱，所以一个有效的图象处理环境不能排斥 C++的编程。现在类似于 Delphi 的 C++Builder 已经推出，原则上可以取代 Delphi 而成为更有效的编程工具，但其过慢的编译速度一时还难让人接受，所以一个以 Delphi 为主，C++语言为辅的混合编程模式是合适的。实现混合编程的关键在于使用 DLL 技术。将C++语言书写的函数文件编译成 DLL 文件，在 Delphi 中对这些函数做相应的说明，就可以像其它函数一样调用。Delphi 中的基本类型与 C++类似，对应方便。为了说明具体问题，现举一个例子。假设 Delphi 的相关文件为 MainProg.pas；C++的相关文件为 Process.cpp，要编译生成 Process.dll。现在希望把二值化操作 ThresholdSegment 交给 C++模块来处理，则应在 Process.cpp 中定义如下：

```
extern "C" {
        void export pascal ThresholdSegment(BYTE
 * pGrey,int RowSize,int ColSize,BYTE Threshold);
    }
void_export pascal ThresholdSegment(BYTE * pGrey,int
RowSize,int ColSize,BYTE Threshold)
    {
      // main block
      ... ...
    }
```

而在 MainProg.pas 中，应包含以下说明：

```
procedure ThesholdSegment(pGrey:PByte;RowSize,
ColSize:Integer;Threshold:Byte);stdcall;
external'Process';
```

在C++源代码中，输出函数必须加_export 和 pascal 指示。另外将函数说明写入 extern "C"{}的花括号内也必不可少，否则此函数会被当作普通的C++函数而不被输出。

DLL 的链入是通过 Delphi 中的指示 external 'process'来实现的，这一指示指出函数存在于 Process.dll 中，将在运行时调入。另一 stdcall 指示在 Delphi2.0 以上版本中必要，原因是其使用了更快速的函数调用 register 或称 fastcall 方式，而 DLL 输出函数的缺省调用方式是 stdcall。加入这样一个指示，才能保证调用无误。利用这样一种混合编程方法，使得完全可能用 C++来编写图象处理程序的主体。

5 实验结果

笔者在研究图象处理算法和构建一个道路图象理解系统的过程中，深感建立一个可视化集成图象处理环境的重要性，这不仅能大大提高工作效率，而且也是软件发展的潮流。于是在工作过程中逐渐开发了这样一个图象处理环境，不断加以完善，并以此为基础构建了一个基于知识的道路图象理解系统。

图1描述了系统主窗口的视图。可以看到，图象和图象的分析以多种多样的方式得到了可视化的处理，多种处理方法也通过这一环境集成在一起。这样一个可视化集成处理环境具有较好的扩展性，提高了进一步研究图象处理算法的效率。

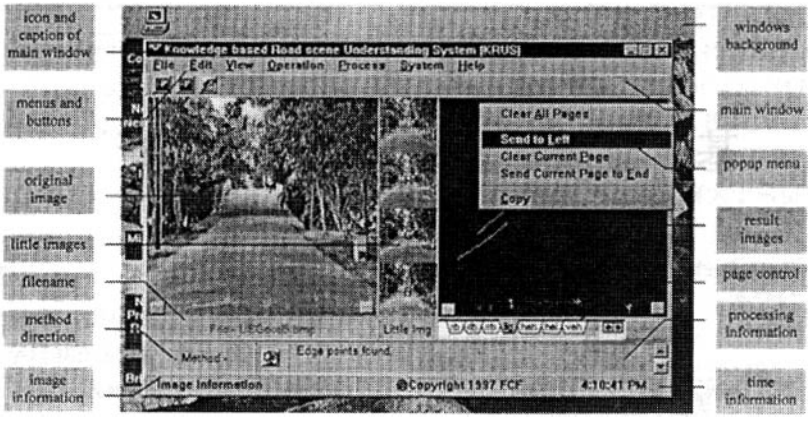

图 1　可视化集成图象处理环境主窗口及其说明

参 考 文 献

1　Smeulders Koelma A . A visual programming interface for an image processing environment. Pattern Recognition Letters 15 (1994) 1099～1109.

2　Delphi OnLine Help.

3　Cornell G, Strain T. Delphi NUTS & BOLTS: for Experienced Programmers. McGraw-Hill, 1995.

范成法　1994 年获浙江大学信电系学士学位,现为浙江大学硕博一贯制研究生。主要研究方向包括计算机视觉和电子系统设计。曾经参加"八五"重大课题"军用智能机器人"中的视觉系统的研究,并为"九五"后续课题的主要研究人员之一。

叶秀清　浙江大学信电系教授,浙江大学信息与智能研究所 ALV 课题组主要负责人之一。主要研究方向为图象处理、机器人。曾参与领导一系列重大课题的研究。

顾伟康　教授,博士生导师,浙江大学副校长,智能与信息系统研究所,研究方面为模式识别,视觉智能机器人。

A Visualized and Integrated Image Processing Environment

Fan Chengfa, Ye Xiuqing, Gu Weikang

(*Dept. of Information and Electronic Engineering, Zhejiang University, Hangzhou* 310027)

Abstract　An integrated visual image processing environment is proposed in this paper. Visualization of image and image analysis and integration of image processing algorithms are available in this environment. Working efficiency will be improved considerably when developing image processing algorithms or building image understanding systems based on this environment.

Keywords　Image processing environment, Visualization of image processing, Image representation, Multiple language programming