ImageJ 开发教程

（苑永超　整理，仅供参考，勿作商业用途）

目录

ImageJ 官网（http://rsb.info.nih.gov/ij/index.html）上有英文的用户手册和教程，以及一些例子。本教程主要是为看英文比较累的朋友提供一些快速的入门。如果想在 ImageJ 上开发自己的图象处理算法，建议先熟悉 java 编程知识。本教程基本不对 ImageJ 菜单中提供的各种文件操作、图象编辑、图象处理、图象分析等功能作详细介绍，请读者自行探索；也不准备介绍数字图象处理的各种算法和操作，本文假定读者是图象处理方面的专业人士，本教程的重点是如何进行二次开发，如果不特别指出，文中的部分内容和例子都为 ImageJ 软件包自带或采自相关书籍（如《数字图像处理-java 语言描述》），中文注释是后加的。

一、ImageJ 简述

图象处理的流程无外乎就是打开图象数据文件，将图象数据加载到内存，然后对该内存中的图象数据进行一系列处理（分割、检测、滤波、合成、识别、显示等等），最后可能还需要将处理结果保存成某种格式的文件。

对于一般的用户来说，类似 ACDsee 之类的傻瓜式的软件足够了。但是科学人士除了希望有广泛的、成熟的处理算法库可以直接调用外，一般还希望开发自己的特有的图象处理算法、特有的图象处理步骤、甚至特有的交互过程。ImageJ 就是这样的工具软件。

ImageJ 是基于 Java 的，ImageJ 在设计上实现了一个可以扩展的基本框架，开发人员可以通过其提供的接口来扩展图象处理功能。ImageJ 提供了很多现成的功能，这些功能可以通过菜单来调用，也可以调用相应的类的方法的 API。

用户只要按照接口要求开发好自己的处理模块，并按照要求部署和配置，Image 就可以自动加载和调用。

ImageJ 是完全开源和免费的，特别适合教学和科研。其关键的特征有：

1、 在菜单上集成了一系列的交互式工具，用于创建、加载、编辑、分析、处理、保存图象，支持常见的图象文件格式。

   目前，ImageJ 主菜单上的集成的主要功能：

   - **File:**　打开、保存、创建新的图象文件。
   - **Edit:**图象的编辑和绘制操作。
   - **Image**:图象的修改、转换、几何操作。
   - **Process**:图象的点运算、滤波器、以及多幅图象之间算法操作。
   - **Analysze:**对图象数据进行统计分析、用直方图或其他格式显示出来。
   - **Plugin:**编辑、编译、执行、管理用户自己定义的插件。

2、 提供简单的插件机制，帮助开发人员专注于自己的图象处理过程的开发，从而扩展 ImageJ 的功能。

3、 提供宏语言或 javascript 脚本以及解释器，可以通过组合现有的函数，来实现客户化的处理过程。这种方式不需要用户具有 Java 知识。此外还有一些使用其他脚本语言扩展的方式。

用户要想在 ImageJ 的基础上扩展自己的图象处理功能、进行二次开发，需要了解 Image 内部结构。

二、ImageJ 内部结构

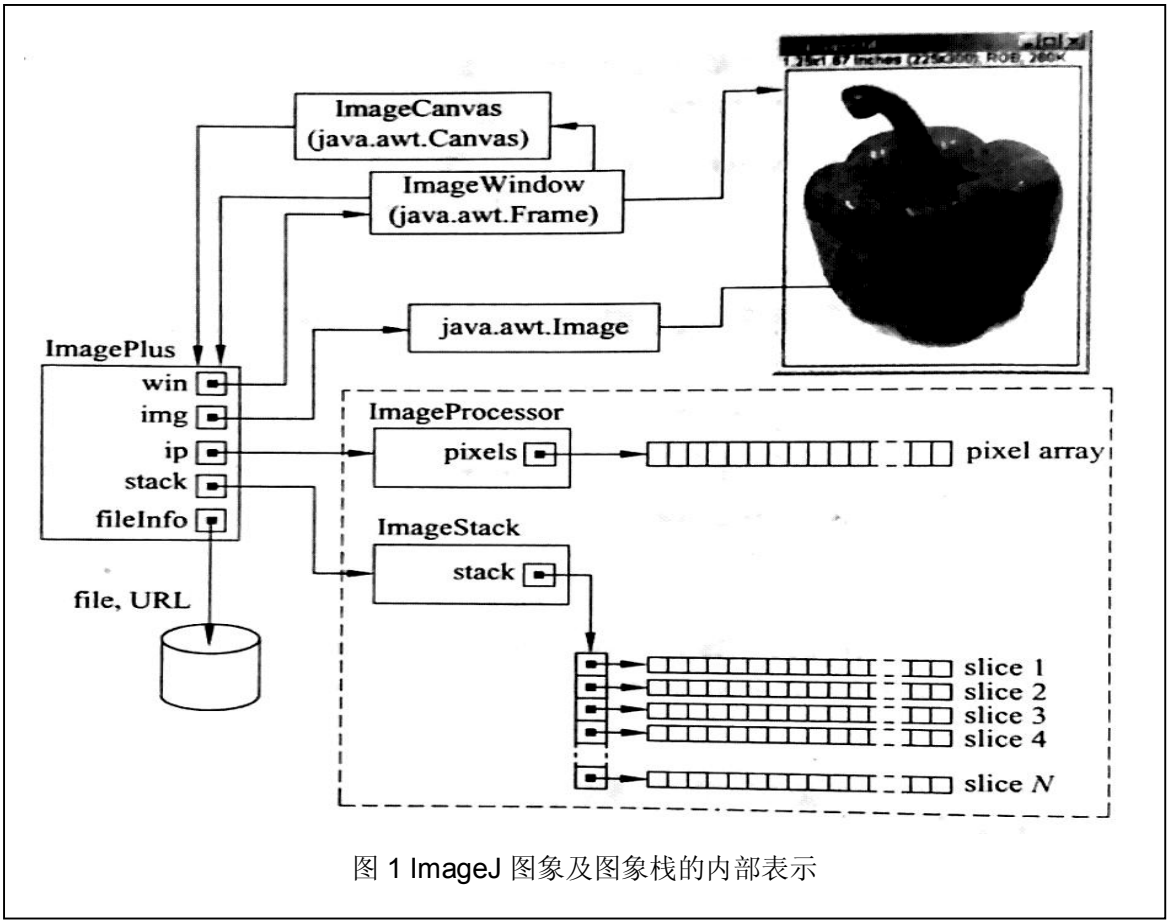

图 1 ImageJ 图象及图象栈的内部表示

上面是一个 ImageJ 的内部结构图，我们结合该图以“打开并显示一个图象”的功能为例说明一下 ImageJ 的工作原理：

1、首先创建一个打开文件类 FileOpener 对象，并调用其 Open 方法。

2、该 Open 方法首先从图象文件中读取像素数据，并放到数组 pixels[]中。

3、随后创建一个 ImagePlus 的对象。如图所见，这个对象成员中包含有一些指针，指向其他对象：

- 一个图象处理器对象 ImageProcessor 的子类：该对象主要是提供对当前图象数据的处理操作。（像素数据保存有对应的像素数组 pixels[]）。
- 一个图象栈对象 ImageStack：用来保存多幅图象数据或图象处理的中间数据。
- 文件信息类对象 fileInfo：存有图象的尺寸、位深度等的相关信息。
- AWT 的图象对象类 img：通过操作将图象数据映射到 Image 对象上，实现图象的显示。
- Frame 窗口类的子类对象 ImageWindow：实现对图象窗口的管理。

上述这些对象在 FileOpener 对象的 open 方法中创建，并赋予 ImagePlus 中的相应成员：ip,stack,fileInfo,img,win。

4、如果该图象需要显示，则调用 ImagePlus 的 show 方法。

该方法调用图象处理器对象 ip 的 createImage()方法创建和图象数组数据对应的图象类 Image 对象。并将 img 指向它。

然后创建 ImageWindow 窗口。设置窗口画布、窗口布局等。

最后通过 ImagePlus 的 draw（）方法调用完成绘制。

切记：

FileOpener 的 open()方法返回的是对应的 ImagePlus 对象。通过该对象，我们可以毫不费力地找到上述相关的对象，尤其是拥有众多图像处理操作和算法的 ImageProcessor，调用需要的方法，从而快捷地完成我们的工作。

三、ImageJ 通过插件扩展功能的方法

插件是一种小型的程序模块，该模块遵循简单的标准化接口，可以被集成到软件框架中，从而扩展宿主软件的功能。ImageJ 的许多内置的图象处理功能也是通过插件来实现的。这里所谓的简单的标准化接口其实就是 Java 的接口类。

ImageJ 提供如下三种不同的接口插件：

- **PlugIn**：启动该插件时不需要打开一幅图象
- **PlugInFilter**：启动该插件时，需要传递给该插件一幅打开图象的指针。该插件的操作将施加在该图象上。
- **PlugInFrame**：该插件可以扩展一个独立的操作交互界面。

很显然，通过 PlugIn 扩展的插件，如果要处理图象，则需要自己去处理获取图象数据，这可以通过获取一个已经被打开的图象文件的指针、自己打开或新创建一个图象文件并获取指针，或直接在内存中开辟图象数据的方式实现。

该方式的自由度很高。该接口的用法示例如下：

```
// 对话框打开并显示一个图像文件
import ij.plugin.*;
import ij.plugin.frame.*;
import ij.*;
public class Test_PlugIn implements PlugIn {
    public void run(String arg) {
        Opener xx = new Opener();
        xx.open() ;
 }
} // end of class
```

用户要实现该接口的 run 方法。

实际上，用的最多的是通过 PlugInFilter 扩展的插件。该接口的示例如下：

```
// 实现将一幅 8bit 位的灰度图取反
import ij.ImagePlus;
import ij.plugin.filter.PlugInFilter;
import ij.process.ImageProcessor;

public class My_Inverter implements PlugInFilter {
```

```
    public int setup(String arg, ImagePlus im) {
        return DOES_8G; // this plugin accepts 8-bit grayscale images
    }

    public void run(ImageProcessor ip) {
        int w = ip.getWidth();   // 获得图象宽度
        int h = ip.getHeight();   // 获得图象高度

        // iterate over all image coordinates
        for (int u = 0; u < w; u++) {
            for (int v = 0; v < h; v++) {
                int p = ip.getPixel(u, v);     // 取列为 u，行为 v 位置的像素的值
                ip.putPixel(u, v, 255 - p);   // 设置列为 u，行为 v 位置的像素值
            }
        }
    }
} // end of class
```

用户要实现该接口的两个方法。

public int setup(String arg, ImagePlus im)

系统执行 PlugInFilter 类型的插件时，首先调用 setup 方法获得插件本身的一些信息。该方法利用打开图象的 ImagePlus 对象 im 中包含的信息，进行版本校验，处理参数的设置等操作。

public void run(ImageProcessor ip)

该方法接收 ImageProcessor 类型的对象，其中包含待处理的图象及其相关信息。上面的程序段里利用了相关的方法获取图象尺寸、像素等信息。另外上述的 run()方法中还可以通过直接操作像素数组来实现类似的功能，但效率要高多了，程序段如下：

```
public void run(ImageProcessor ip) {
    int w = ip.getWidth();   // 获得图象宽度
    int h = ip.getHeight();   // 获得图象高度
    byte[] pixels = (byte[])ip.getPixels() ;   // 获得像素数组

    // iterate over all image coordinates
    for (int u = 0; u < w; u++) {
        for (int v = 0; v < h; v++) {
            int p = 0xff & pixels[v*w+u] ;          // 取列为 u，行为 v 位置的像素的值
            pixels[v*w+u] = (byte)(0xFF-p);       // 设置列为 u，行为 v 位置的像素值
        }
    }
}
```

如果用户要实现复杂的界面和交互，可以扩展 plugFrame 插件。该插件的示例如下：

```
import ij.*;
```

```
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import ij.plugin.frame.*;

public class TestFrame extends PlugInFrame {

    public TestFrame() {
        super("Plugin_Frame");
        TextArea ta = new TextArea(15, 50);
        add(ta);
        pack();
        GUI.center(this);
        show();
    }
}
```

系统创建该窗体插件，用户可以定义自己的用户界面、交互流程，当然前提要对 java 的 AWT 或 Swing 编程比较熟悉。

三、插件编辑、编译、运行与部署

1、 源码部署

用户的插件的源代码应该位于 ImageJ 安装目录的 plugins 目录或一个下一级目录。如果放到其他目录，ImageJ 将不能识别。

2、 源码编辑

用户可以选择 ImageJ 的菜单 Plugins->New，在选择对应的插件类型，来生成相应的插件原型源代码，用户可以在该代码的基础上编写自己的插件。但是 ImageJ 内置的 java 代码编辑器不能提供在线的语言帮助，不是很方便。用户可以通过 Eclipse 或其他集成编辑器开发插件。

3、 编译

ImageJ 自带了 java 编译器，可以在 ImageJ 的菜单中使用。选择 Plugins->Compile and Run… 选择要编译和执行的插件。这种方式，插件编译后会立即被加载和执行，如果是 PluginFilter 插件则一定要用菜单方式打开一幅可被处理的图像文件，否则插件执行会报错。

4、 菜单设置

要从菜单上执行该插件功能。选择 Plugins->Shortcuts->Install Plugins….在弹出的对话框中设置如下：

Plugin：从下拉列表中把你的插件选择出来（ImageJ 已经对部署到 Plugins 或下一级的目录下的插件自动识别出来料）。

Menu：选择要将菜单项放到那个子菜单项目下。（一般插件放到 plugins 或 plugins/shortcuts 下）。

Command：显示在菜单上的对应的菜单项的名称。

Shortcuts：对应的快捷键（一般是 F+数字，如 F4）。

Argument：该插件需要的参数，对应插件的 setup 方法中要传入的参数。

5、 从菜单中去除菜单项

选择 Plugins->Shortcuts->remove…，从对话框列表中选择要删除的菜单项名称。ImageJ 重起后生效。

6、 使用 Eclipse 开发插件

使用 Eclipse 的好处：a、随时可以得到语法帮助，b、相关类和方法的用法帮助，c、自动补全完成功能，d、调试和跟踪代码的执行。开发效率将大大提高。

使用 Eclipse 步骤如下：

- 从官网下载最新的 ImageJ 源码包（我下载的是这个链接的包 http://rsb.info.nih.gov/ij/download/src/ij145e-src.zip）。
- 用 eclips 建一个 java 的空的工程（我命名为 ImageJ），将 ij145e-src.zip 包解开，将其中的 source 下的子目录等内容拷贝到 ImageJ 下。
- 通过选择”project->clean”来编译 ImageJ 工程，Complier.java 会报错，这是因为使用的缺省的 java1.5 版本太高。具体做法是：将 ImageJ 安装包自带的 jre 目录拷贝到 ImageJ 的目录下，在工程浏览窗口的 ImageJ 工程上右键打开菜单，选择工程属性，打开 javaBuild Path->Libraries。先将原先的 1.5 版本的 jre 删除，然后选择 Add Libraries，在 libraries type 类型列表框中选择”jre system library”，再选择”alternate JRE”，按 Install JREs 按牛，在新的对话框中选择”Add”，在新的对话框中选择”Browse 按纽，选择 ImageJ 目录下的 jre 目录即可。确认后 jre1.4 就被添加进了了，选择 JRE 为刚添加的 jre 版本的就 OK 了。此时可以选择运行 ImageJ 的 main()方法，成功则显示 ImageJ 的主界面。
  注意，在 libraries type 类型列表框不要选择 J2EE1.4，而要使用 jre system library 去添加 jre1.4，否则会报 java/lang/NoClassDefFoundError: java/lang/Object）。
- 这时如果 ImageJ 下 Plugins 目录下有插件源代码，则浏览窗口提示有错误。提示的错误是“declare package “” not expect package XXX”。这是因为插件的 java 文件前面没有 package 语句，因为这些代码并不打到 IJ.jar 包里去，不影响 ImageJ 本身的编译。切记不要给插件加 package 声明语句，否则虽然错误提示没了，插件执行却会出错。
- 要开发插件，只需要在 Plugins 下创建新的类即可，要继承相应的接口，实现要求的方法，注意将自动添加的包声明去掉。
- 要编译插件，在 eclipse 菜单中选择 project->clean 即可。
- 要使用最新的 IJ.jar 可以通过 ImageJ 菜单的 help->ImageJ news 选择升级。大量的插件可以通过 ImageJ 菜单的 help->plugins 在线访问。

7、 使用宏程序

对于经常使用的一系列的插件调用，可以通过 ImageJ 的 Plugins->Macros->Record 记录成为宏程序，这样只要选择执行该宏就可以实现原先的一系列操作料。

要充分发掘 ImageJ 的功能，需要熟悉 Image 主要的类结构和相互关系，下面按照功能介绍以下主要的包的类图结构。

四、主要的包介绍

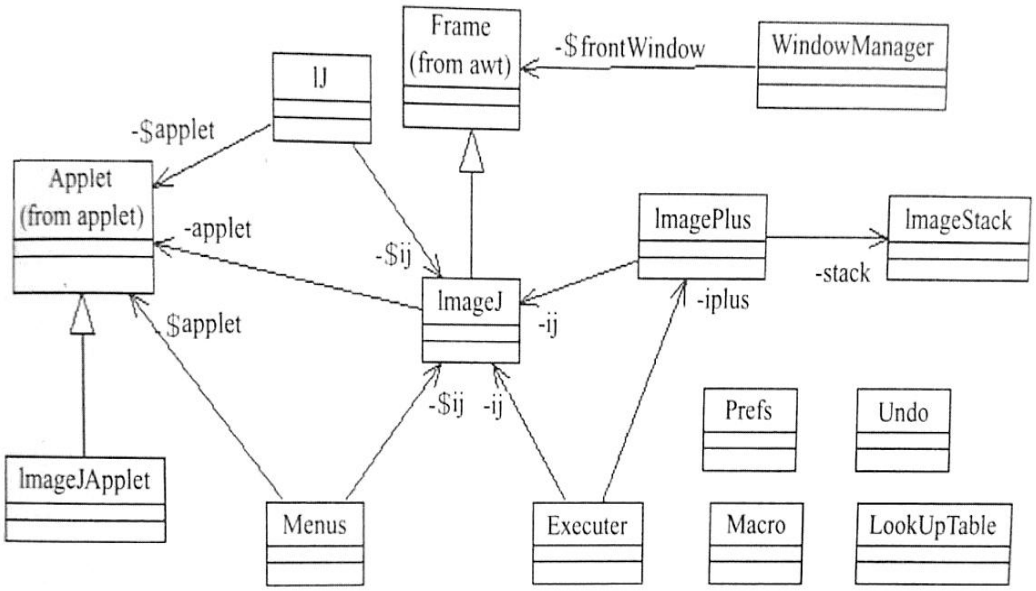

图 2 ImageJ 中 ij 包的类图

类型（所在包）	主要类	类说明
图象（包 ij）	ImagePlus	该类的一个对象和一幅要处理的图象（或图象序列）相对应。包含有图象处理类 ImageProcessor 抽象类。
	ImageStack	可扩充图象序列（栈），依附于 ImagePlus 对象
图象处理器（包 lj.process）	ImageProcessor	这是一个抽象类，有四个派生类
	ByteProcessor	用于 8 位（byte）类型的灰度图象和索引彩色图象的处理类。其子类 BinaryProcessor 实现了仅包含 0 和 255 两个像素值的图象。
	ShortProcessor	16 位灰度图象的处理类。
	FloatProcessor	32 位（浮点型）图象的处理类。
	ColorProcessor	32 位彩色（3X8 位 RGB 通道加上 8 位 α 通道浮）图象的处理类。
插件（包 ij.plugin）	plugin	包含 void run(String arg)方法。 不需要先打开图象。
插件（包 ij.plugin.filter）	PlugInFilter	要实现下面两个方法 int setup(String arg, ImagePlus imp) void run(ImageProcessor ip)
插件（包 ij.plugin.Frame）	PlugInFrame	用户要实现界面相关的消息处理

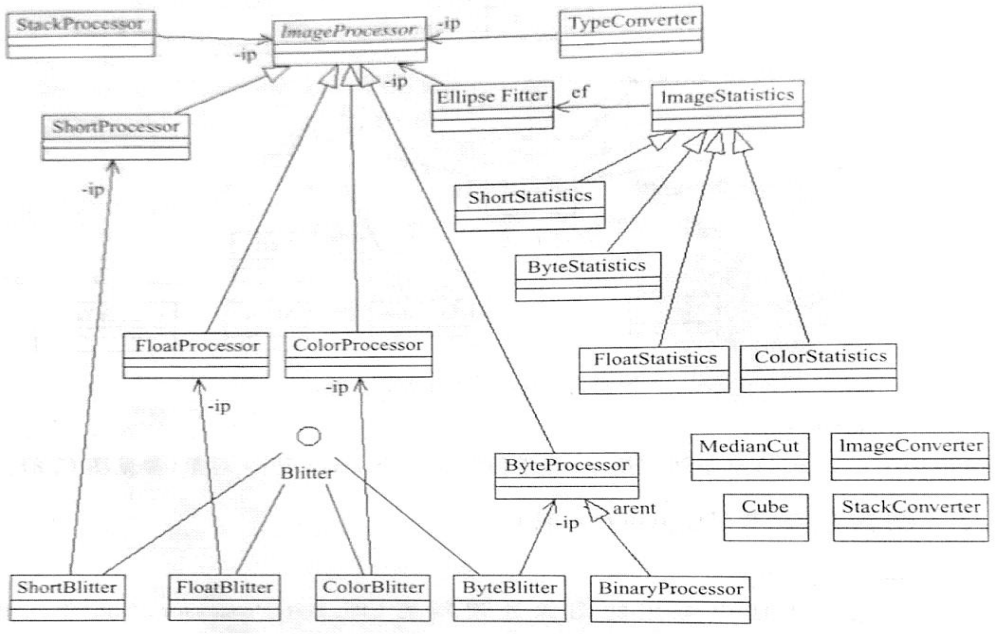

图 3 ImageJ 中 ij.process 包的类图

类型(所在包)	主要类	类说明
GUI（包 ij.gui）	ColorChooser	颜色选取对话框类
	NewImage	交互式创建新图象的类
	GenericDialog	提供一套标准交互字段可配置的对话框窗口类
	ImageCanvas	Awt 中 Canvas 的子类，描述窗口中显示图象的映射关系（源矩形框、缩放因子），也处理发送到该窗口的键盘和鼠标事件
	ImageWindow	Awt 中 Frame 的子类，用于显示 ImagePlus 类型图象的屏幕窗口。该类对象包含一个 ImageCanvas 类的实例，用于图象的实际显示。
	Roi	定义一个用户选取的感兴趣区域。是 ROI 类（Line,OvalRoi,PolygonRoi）等的超类
窗口管理（包 ij）	WindowManger	提供一套静态方法用于 ImageJ 窗口管理
工具（包 ij）		提供了一套静态工具方法，包括图象选取、创建、打开和保存以及操作环境信息获得的方法

输入输出（包 jj.io）		包含从文件度取（载入）和往文件写入各种格式和编码图象的类

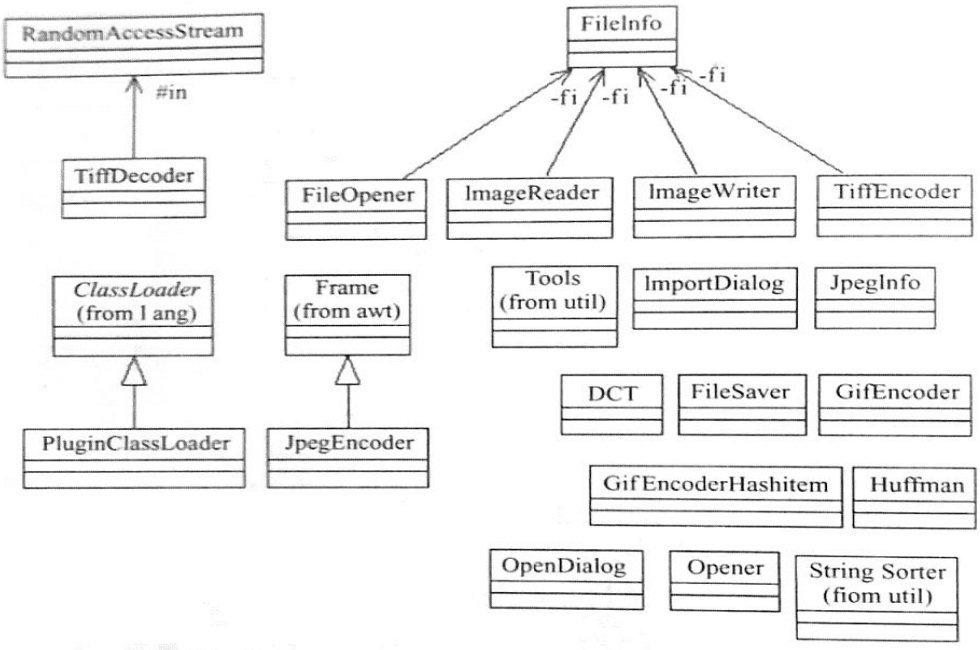

图 4 ImageJ 中 jj.io 包的类图

五、重要类方法介绍

1、创建图象和图象栈

ImageJ 中有多种方式创建图象。

ImagePlus 类中支持创建一个空的 ImagePlus 对象、从 URL 路径中创建、从给定的 Image 图象中创建、从 ImageProcess 对象中也可以创建、从给定的图象栈中可以创建、可以创建一个空的图象栈、可以返回与当前 ImagePlus 相关联的图象栈。

IJ 类支持创建一个给定参数的 ImagePlus 图象

NewImage 类用几个静态方法支持创建 ImagePlus 图象和图象栈

ImageProcessor 支持用 Image 的 CreateImage()方法创建当前图象的副本并以标准 Image 图象返回。

2、创建图象处理器

ImageProcessor 对象表示可以被创建、处理、销毁但一般不能在屏幕上显示的图象。
ImagePlus 的 getProcessor 返回指向图象的 ImageProcessor 对象的引用，如果没有，则创建一新的，如果没有 Image 则返回空。setProcessor 设置该对象

ImageProcessor 提供了创建 createProcessor,复制 duplicate。

子类中可以从 Image 图象创建或按照指定尺寸、色彩等参数构造。

3、载入和存储图象

IJ 类中提供了很多类似菜单功能的静态方法：
Run(“Open…”/”Revert”/”Save”)
可用于 IO 的方法 open(path)/openImage(path)/save(path)/saveAs(fmt,path)

Opener 类提供了重载的 open()/openImage()/openMultiple()方法
打开模式设置，以及针对特定格式的打开方法、保存方法。

FileOperner 类从该类保存的文件信息的位置上打开或保存图象

4、图象参数

ImageProcessor 中获得图象的宽度和高度、打开或关闭像素插值
ColorProcessor 提供彩色转灰度，权值获取/设置
getWeightingFactors()
set WeightingFactors()

5、操作像素

ImageProcessor 提供了很多坐标访问方法：
getPixel(x,y)/putPixel(x,y,val)
get(x,y)/put()x,y 是 2D 索引不验证参数的版本
get(i)/put(i)是按照 1D 索引访问
带 f 是浮点型的版本。
获取像素数组指针 getPixels

还可以整行/整列/整个数组的操作
彩色图象使用 getRGB()/setRGB()/getHSB()/setHSB()等进行

6、图象转换

ImageProcessor 实现了不同类型间图象相互转换的基本方法（非破坏性转换）
convertToByte/convertToShort/convertToFloat/convertToRGB

ImagePlus 类型可以通过 ImageConvert 类的方法进行转换（破坏性转换）
convertToGray8()/convertToGray16()/convertToGray32()/
convertToRGB()/converToHSB/convertHSBToRGB
convertRGBStackToRGB(将 RGB 栈转成单幅的 RGB 图象)
诸如此类

7、直方图与图象统计量

ImageProcessor 的方法
获取图象或感兴趣区的直方图
Int[] getHistogram()
Int[] getHistogramMax()
Int[] getHistogramMin()
Int[] getHistogramSize()
Int[] setHistogramRange()
Int[] setHistogramSize()
其他的统计量可通过 ImageStatictics 及其子类 ByteStatictics\ShortStatictics\
FloatStatictics\ColorStatictics\StackStatictics 实现。

8、点运算

对 ImageProcessor 类型的对象的整幅图或感兴趣区域的全体像素施加该运算。
abs() 给每个像素替换成其绝对值
add(val) /and(val)/or(val)/sqr()/sqrt()/xor/log()
applyTable(int []lut) // 查表影射
autoThreshold() // 转 2 值图象
gamma(g) // gamma 矫正
max(val)/ min(val) // 将所有大于/小于 val 的值设置成 val

multiply(val)乘/noise(val)加噪声随即量 val
threshold(int th) // <th →0, 否则 255
多幅像素运算，ImageProcessor 组合 2 幅图的方法：
copyBits(ImageProcessor, int, int, int mode)
mode 运算类型中在 Blitter 中定义：
COPY/ADD/AND/COPY_INVERTED/DIFFERENCE/DIVIDE/

9、滤波器

imageProcessor 类提供滤波方法：
void convolve(float []kernel, int w, int h)
对当前图象执行线形卷积，滤波 kernel 大小为 w*h。
void convolve3x3(int []kernel)
对当前图象执行线形卷积，滤波 kernel 大小为 3*3int 数组。
dilate()
使用 3*3 最小滤波器膨胀运算
erode()
使用 3*3 最小滤波器腐蚀运算
findEdges()
使用 3*3 边界滤波器运算（Sobel 算子）
medianFilter ()
使用 3*3 中值滤波器运算
smooth()
使用 3*3 均值滤波器运算（箱式滤波）
sharpen ()
使用 3*3 类似 Laplacian 的滤波核锐化处理

10、几何运算

ImageProcessor 类支持如下运算
ImageProcessor crop()
// 对当前图象或 ROI 区域创建一个新的 ImageProcessor 对象
void flipHorizontal() 当前图象或 ROI 区域破坏性水平镜像
void flipVertical()当前图象或 ROI 区域破坏性垂直镜像
ImageProcessor resize (int w, int h)
// 对当前图象或 ROI 区域产生一个缩放的新的 ImageProcessor 对象
Void rotate(double angle) // 破坏性按照一定角度顺时针地旋转 angle 角度
ImageProcessor rotateLeft()
// 按照一定角度逆时针地旋转 90’角度，
// 并返回一个新的 ImageProcessor

ImageProcessor rotateRight()
// 按照一定角度顺时针地旋转 90’角度，
// 并返回一个新的 ImageProcessor
Void Scale(double XScale, double yScale)
// 破坏性地比例缩放图象或 ROI 区域
Void setBackgroundValue (double value)
// 设置图象背景值，用于 rotate 或 scale
Boolean getInterpolate()
// 获取当前的双线性插值状态
Void setInterpolate(bool)
// 获取当前的双线性插值状态
Double getInterpolatedPixel (double x, double y)
//返回使用插枝得到的对连续坐标(x,y)得到的像素值
Double getInterpolatedRGBPixel (double x, double y)
//对彩色图象返回使用插枝得到的对连续坐标(x,y)得到的像素值

11、图形运算

ImageProcessor 提供了一系列的绘制/填充各种图形的函数。
Void drawXXX()形式
Void fill() / fillXXX()形式
Void setXXX ()设置对象属性，线宽等

12、显示图象和图象栈

只有 ImagePlus 对象可以使用下面的方法，ImageProcessor 必须通过得到关联的 ImagePlus 对象来显示的。
Void draw()/draw(x,y,wid,height)
Int getcurrentSlice // 获得当前图象栈片段的索引号
Int getID // 得到图象的 ID
String getTitle()得到图象的名字
ImageWindow getWindow() // 返回显示用的窗体
Void hide() // 关闭显示当前图象的任何窗体
Void repaintWIndow()
Void setSlice(index)
Void setTItle(String)
Void show() // show(statusString) //打开窗口显示此图象(状态条中显示文本)
Void updateAndDraw() // 根据 ImagPocessor 数据更新图象并显示
Void updateAndRepaintWindow() //当前图象数据重新绘制，并更新头信息（纬度/类型等）

13、图象栈上的操作

ImagePlus 类：
创建空的图象栈(具有和 ImagePlus 相同的宽度、高度、色表，新栈自动依附 ImagePlus)
    ImageStack createEmptyStack()
取得与 ImagePlus 相关联的图象栈，没有则调用 getStack()
    ImageStack getImageStack()
取得与 ImagePlus 相关联的图象栈,不存在则调用 createEmptyStack()生成单层栈
    ImageStack getStack()
取得栈大小
    Int getStackSize()
替换当前图象栈
    Void setStack(String title, ImageStack stack)

ImageStack 类：
将 IP 指定的图象加载到栈的尾部，并给新层分配 label
    Void addSlice(String label, ImageProcessor ip)
将 IP 指定的图象加载到栈的第 n 层之后，并给新层分配 label
    Void addSlice(String label, ImageProcessor ip, int n)
将由 pixels 指定的（像素数组）图象加入到栈的末尾
    Void addSlice(String label, Object pixels)
删除栈末尾的一层
    Void deleteLastSlice()
删除栈中指定的一层
    Void deleteSlice(int n)
取得栈中图象的高度
    Int getHeight()
取得栈中图象的宽度
    Int getWidth()
将整个栈作为一个一维像素数据的叔祖返回
    Object[] getImageArray()
返回栈的第 n 层图象的一维像素数组
    Object getPixels(int n)
创建一个 ImageProcessor 为栈的第 n 层，并返回它
    ImageProcesspr get Processor(int n)
返回栈的层数
    Int getSize()
返回第 n 层栈的标签
    String getSliceLabel(int n)
返回栈的所有层标签
    String[] getSliceLabel()

将像素数组分配给第 n 层
    setPixels(Object pixels, int n)
设置第 n 层标签
    setSliceLabel (String label, int n)

14、感兴趣的区域

所谓感兴趣的区域(ROI)就是为随后的处理选择一个特别的图象区域，通常由用户交互式来选定的。ImageJ 支持的 ROI 类型，包括：
矩形（Roi 类）、椭圆型（OvalRoi 类）、直线（Line 类）、多边形/折线（PolygonRoi 类、FreehandRoi 类）、点集（PointRoi 类）
ROI 对象一般和 ImagePlus 对象相关联。

ImagePlus 类：
Roi getRoi()
Void KillRoi()
Void setRoi(int x,int y, int width, int height)
Void setRoi(java.awt.Rectangle rect)
Void setRoi (Roi roi) // 给图象分配一个指定的 ROI 并显示，如果 ROI 为 null 或宽度或高度为 0，则删除所有现有的 ROI。
ImageProcessor getMask() // 对于矩形的 ROI，返回一个掩膜图象(ByteProcesspor 类型)；否则返回 null。

Roi\Line\OvalRoi\PointRoi\PolygonRoi 类
注意构造方法的参数

ImageProcessor 类：
ImageProcessor 对象也可有一个关联的 ROI，该机制类似于用于 ImagePlus 的机制，但又有所不同。特使是，一个非矩形 ROI 表示为一个包围矩形与由一个一维 int 数组指定的大小相同的掩膜的组合。
ImageProcessor getMask() // 同上
Byte[] getMaskArray() // 返回掩膜的 byte 数组，若此图象没有掩膜则返回 null
Rectangle getRoi()
Reset Roi() // 设置 ROI 包含整副图象
Void setMask(ImageProcessor mask) // 定义一个 byte 掩膜以限制对一个不规则 ROI 的处理。Mask 大小必须与当前 ROI 相同。
Void setRoi(int x,int y, int width, int height)
Void setRoi(java.awt.Rectangle rect)
Void setRoi (Roi roi)
Void setRoi (java.awt.Polygon poly) // 多边型 ROI

ImageStack 类:
只有矩形 ROI 可用语图象栈:

Java.awt.Rectangle　getRoi() / void setRoi (java.awt.Rectangel)
IJ 类：
下面的静态 ROI 方法作用于当前活动（用户选定）的图象
Static void makeLine(int u1,int v1,int u2,int v2) // 创建一个直线选择 ROI
Static void makeOval(int x,int y, int width, int height) // 创建一个椭圆型 ROI
Static void makeRectangle(int x,int y, int width, int height) // 创建一个矩形 ROI

15、图象属性

有时候需要从一个插件向另一个对性传递结果。但是 run()方法本身不提供返回值。解决办法：可以将要传递的值保存到对应的 ImagePlus 的对象的属性中。属性就是一个关键字/值对。属性必须是字符串，值可为任意 java 对象。ImageJ 中由哈希表支持该机制。
Java.util.Properties　getProperties() // 返回图象中所有的
Object　getProperty(String key)
Void　setProperty(String key, Object value)

16、用户交互

IJ 类
文本输出：
Static void　error (String msg) // 对话框标题为 Image
Static void　error (String title,String msg)　//对话框标题由 title 指定
Static void log (String msg) // 在日志窗口中显示一行 msg
Static void write (String msg) //　向日志窗口写入一行 msg

对话框：
Static double getNumber(String prompt, double defVal) // 读取用户输入的数字
Static String getString (String prompt, String defVal) // 读取用户输入的字符串
Static void noImage()　// 提示没有图象对话框
Static void showMessage(String msg) // 提示消息
Static void showMessage (String title,String msg)　//对话框标题由 title 指定
Static boolean showMessageWithCancel(String title,String msg) // 用户按取消则返回 false

进展条和状态条：
Static void showProgress(double prog) // 更新 ImageJ 中进度条, 0<=prog<=1
　　Proj>1 则进度条被删除。
Static void showProgress(int I, int n)　//　　0<=i<n
Static void showStatus (String msg) // 在状态条中显示一个消息

查询键盘：
Static Boolean altKeyDown()
Static Boolean escapePressed()
Static void resetEscape()
Static Boolean shiftKeyDown()
Static Boolean spaceBarDown()

其他：
Static void beep() // 发出蜂鸣声
Static ImagePlus getImage() // 返回当前活动图象的引用
Static void wait (int ms) // 等待 ms 毫秒

GenericDialog 类
该类提供了一种简单机制用于创建一个包含对多个不同类型字段的对话框窗体。这些对话框窗口的布局是自动建立的。

17、插件

插件接口：
Plugin：应用时不需要任何图象，可用于获取图象、显示窗体等。
PluginFilter:用于处理已有的图象。
其中 setup()返回插件的特征，有一些预定义的值
DOES_8G 8 位灰度图像
DOES_8C 8 位索引彩色图像
DOES_16 16 位灰度图像
DOES_32 32 位 float 图像
DOES_RGB 3*8RGB 图像
DOES_ALL 任何类型的图像
DOES_STACKS run 方法将作用于栈的所有层
DONE run 方法将不被调用
NO_CHANGES 插件不修改原始图像
NO_IMAGE_REQUIRED 插件不要求图像打开，这样穿给 run 的 ip 为 null
NO_UNDO 插件不要求复原
ROI_REQUIRED 插件要求 ROI 区域被明确指定
STACK_REQUIRED 插件要求一个图像栈
SUPPORTS_MASKING 对于非矩形 ROI，插件希望 ImageJ 自动地恢复图像位于包围矩形内却在 ROI 之外的部分。（极大地简化非矩形 ROI 使用）

插件的执行：
Static Object runPlugIn(String className, String arg)

18、窗口管理

WindowManager 类：
定义了一套静态方法用于操作 ImageJ 中的屏幕窗口：
Static Boolean closeAllWindows()
Static ImagePlus getCurrentImage()
Static ImageWindow getCurrentWindow()
Static int[] GetIDList() // 返回包含所有打开图象 ID 的数组，没有则返回 nulll
Static ImagePlus getImage(int ImageID)
Static ImagePlus getImage(String title)
Static int getImageCount()
Static ImagePlus getTempCurrentImage() // 返回临时设为当前图象（通过 setTempCurrentImage()）的图象
Static int getWindowCount()
Static void putBehind()
Static void repaintImageWindows()
Static void setCurrentWindow(ImageWindow win)
Static void setTempCurrentImage(ImagePlus im) // 使 im 为临时活动图象，并允许对当前没有显示在图象中的图象进行处理，如 im 为 null，则恢复以前的活动图象

19、其他函数

ImagePlus 类：
图象锁定和解锁：
Boolean lock()
Boolean lockSilently() // 锁定但不发出比比声
Void unlock()

内部剪贴板：
Void copy(Boolean cut)
Void paste()
Static ImagePlus getClipboard()

文件信息：
FileInfo getFileInfo() //返回包括保存图象数组等所需要的图象信息。
FileInfo getOriginalFileInfo() // 返回用于打开图象图象的信息

IJ 类：
目录信息：
Static String getDirectory (String target)

根据 target 的值（”home”,”startup”,”plugins”,”macros”,”temp”,”image”）返回对应的 ImageJ 的主目录、启动目录、插件目录、宏目录、临时目录、图形目录，如果 target 不是上述值，则弹出对话框让用户选择。

内存管理：
Static long currrentMemory() //返回 ImageJ 所使用的内存大小
Static String freeMemory()
Static long MaxMemory

系统信息：
Static String getVersion()
Static Boolean isJava2()
Static Boolean isJava14
Static Boolean isMacintosh()
Static Boolean isMacOSX()
Static Boolean isWindows()
Static Boolean versionLessThan(String version) //判断版本是否低于 version

六、学习资源

ImageJ 的源代码中包含了很多图象处理插件，是很好的学习材料。

IJ_Props.txt 是菜单和插件名称的对照表，可以边执行菜单上的功能项目边研究代码。

在 ImageJ 的安装包的 Plugins 目录下有一个 Examples 目录，有若干开发例子。

- Ip_demo.java 是一个综合性的基于 PlugInFrame 有独立截面的插件。
- Red_And_Blue.java 是一个 PlugIn 插件。
- Plasma2_.java 也是一个 PlugIn 插件。
- Image_Inverter.java 是一个基于 PlugInFilter 的插件。

官网上有很多比较大型的例子，可以参考，但不太适合学习。