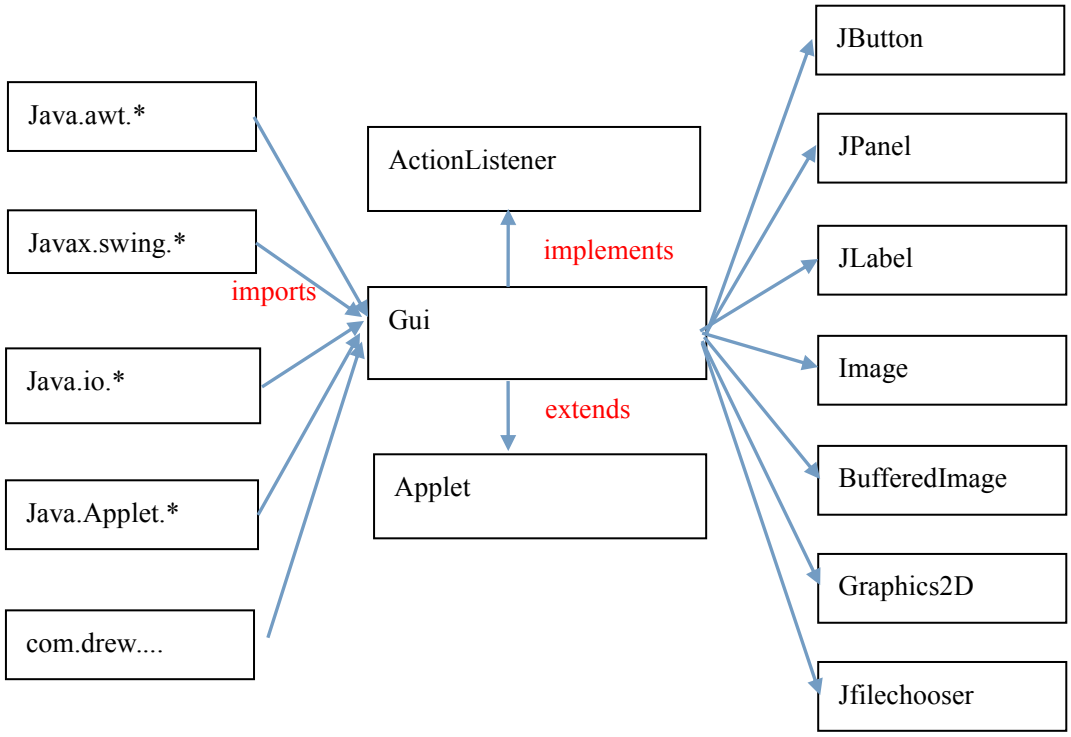
图片查看器

一.程序设计目的

1 设计简单的图片查看器，可以实现以下功能：
1）.指定后缀的文件的打开包括打开的时间
2）.读取图片的相关信息，包括分辨率
3）.对图片进行简单的操作，包括放大，缩小，读取下一张和上一张图片
4）.进行汽车和行人的识别（仅供参考功能）

二.程序设计思路

1.需要的 java 编程技术：主要就是图像的显示功能，其中运用到了双缓冲技术消除闪烁问题。
2.类图
本程序用到了基本类和接口的关系如下所示



三.程序设计

程序的总体设计思想是继承一个 Appplet 小程序类，在面板上加入各种组件构成图片查看器的基本界面。再对按钮进行监听，对事件确定事件源，实现不同的响应方法，包括 open(),Last()等方法。程序的主要难点也在于各种方法的实现。下面会介绍方法是怎么实现各自功能的。

1.程序中引用到的类

1）JPanel

程序中定义的 JPanel 对象主要为 pathPanel(顶部的显示区域),picPanel(图片信息及操作）区域,mainPanel（程序控制区域）

2）DrawPanel（JScrollPanel)

本程序用 DrawPanel 继承了 JScrollPanel，主要的 DrawPanel 对象为 showPanel（显示图片区域）

3) JLabel

主要的 JLabel 对象为：

pathInfo（路径标签）,picInfo（图片信息标签）,picSize（图片分辨率标签），picTime（打开时间标签）,picOper（图片操作标签）,memory(占用内存信息标签）,dateTime(拍摄时期标签);

4) JButton

主要的 JButton 对象为：

picLarger,picSmaller,picCarReco,openButton,lastButton,nextButton;

5) JTextField

主要的 JTextField 对象为：sizeText（显示分辨率）,timeText（显示打开时间），memoryText(占用内存信息),dateTimeText（拍摄时期）。

6) BufferedImage

此类为用于显示的缓冲区图像。在对图像进行操作前，都先将图片读进预设的缓冲区，然后再显示在用户前端，这样虽然速度慢了点，但很好的消除了图片闪烁效果。

7) Graphics2D

运用双缓冲技术消除闪屏问题

8) Filter

主要的 Filter 对象为 filterJpg。Filter 是一个元数据类，本程序实现了 FilenameFilter 接口的一个方法，建立过滤器，使得只显示 JPG 格式的图片。

2.主要方法如下

1).init()

在这里实现了 GUI 的布局，在 BorderLayout 布局中，将各 JPanel 添加到 BorderLayout 方式的 Container 中。

在 PicPanel 上使用了 GridLayout 布局 GridLayout g1=new GridLayout(12,1,5,5);其他面板上使用的是默认的 FlowLayout。

另外本方法初始化各按钮为 setEnabled(false)，因为初始时刻没有打开图片（HasPic=false);

2). 因为 Gui 实现了 ActionListener 接口，所以这里实现 interface 上的方法来处理事件。
首先确定事件源，然后进行不同的响应。

2）.open()
这里实现对 OpenButton 的操作反应，用户可以打开任意目录下的以 JPG 为后缀的图片，并跟踪此目录下的所有 JPG 图片，方法主要利用了过滤器和一个 JFileChooser 类。选定图片后，HasPic=true，同时上面所述按钮也 setEnabled(true)。

3）.last()，next()
这里实现对 LastButton 和 NextButton 的操作反应，用了 if{...}else{...}来循环浏览图片，因为在 open()方法中已经实现了对图片的跟踪，所以此时的文件名很容易知道，路径为 dir=AllFilePath+"\\"+Pics[index].

4). PicTrans()
如果 bufImage 为空则直接返回，之后进行过滤图像操作，2D 仿射变换，设置仿射变换的比例因子 transform.setToScale(scaleX, scaleY); 创建仿射变换操作对象，过滤图像，将目标图像存在 filteredBufImage 中，然后显示。

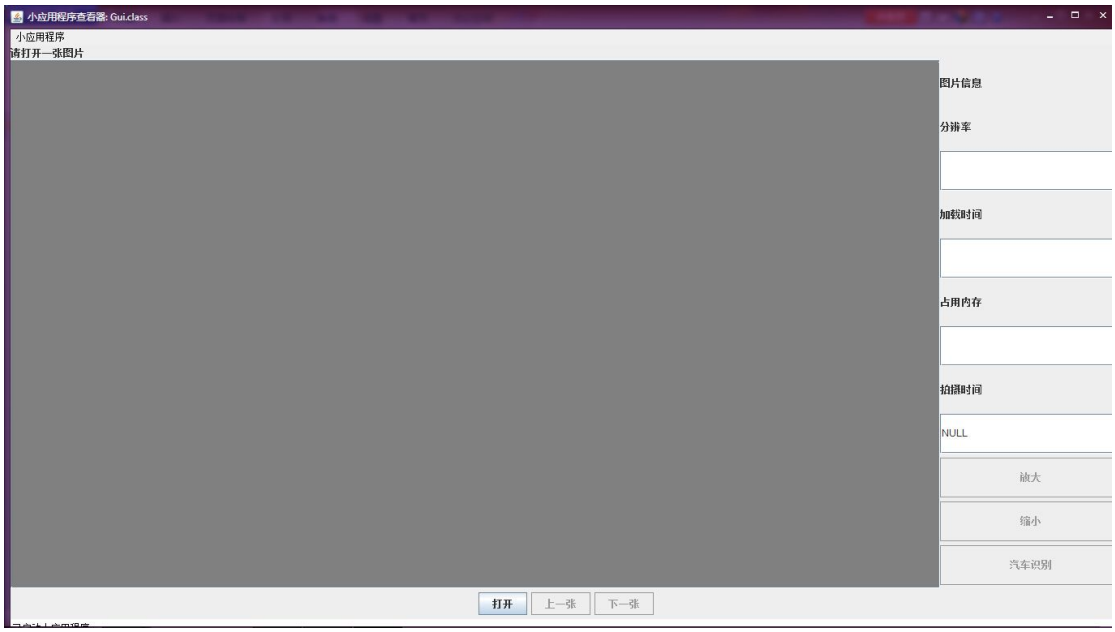
5).Larger(),Smaller()
当判断 scaleX>1.7 时不再放大图片，否则设置放射因子为 scaleX +=（-=） 0.15;scaleY +=（-=） 0.15;repaint()来重绘组件。

6）.loadImage（）
通过增加图像到加载器中，创建原始缓冲区图像，创建 bufImage 的图形环境：bufImageG = bufImage.createGraphics();传输源图像数据到缓冲区图像中一系列的操作，对图片进行了加载操作；
这里我引入了 Drew Noakes 写的 metadata-extractor 包，用来获取照片的 EXIF。由于原照片无拍摄时间信息，我随意添加了拍摄时间来以供演示。

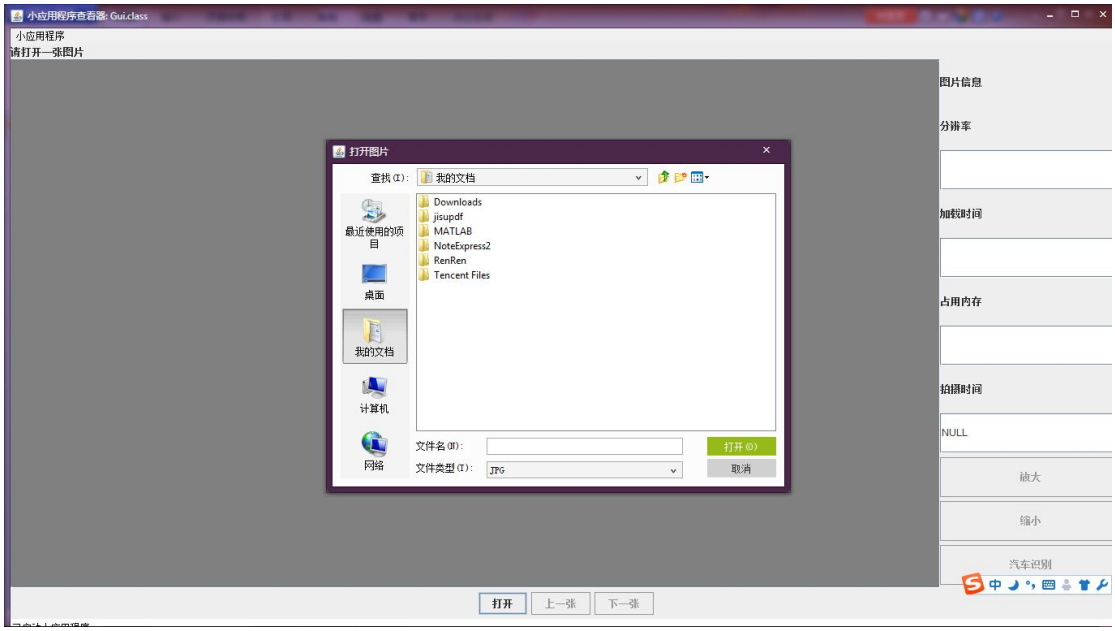
7).Reco()
该方法只是一个测试方法，用于标注当前画面中的汽车和行人。思想为先将 BufImage 灰度化，再二值化，这里可以设置阀值。然后将对应于白色区域标出，认为为汽车和行人，当然有一定误差。

四．程序演示

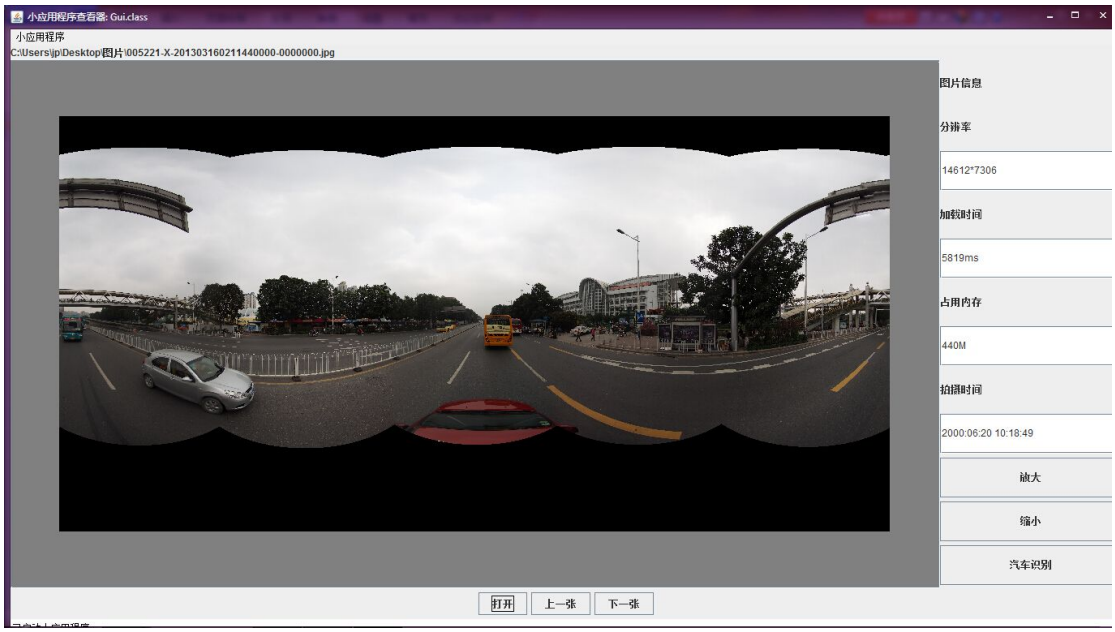
1.打开程序
使用 Eclipse 运行程序时，出现 Applet 小程序界面，显示程序已经启动。



2.单击打开按钮出现对话框：



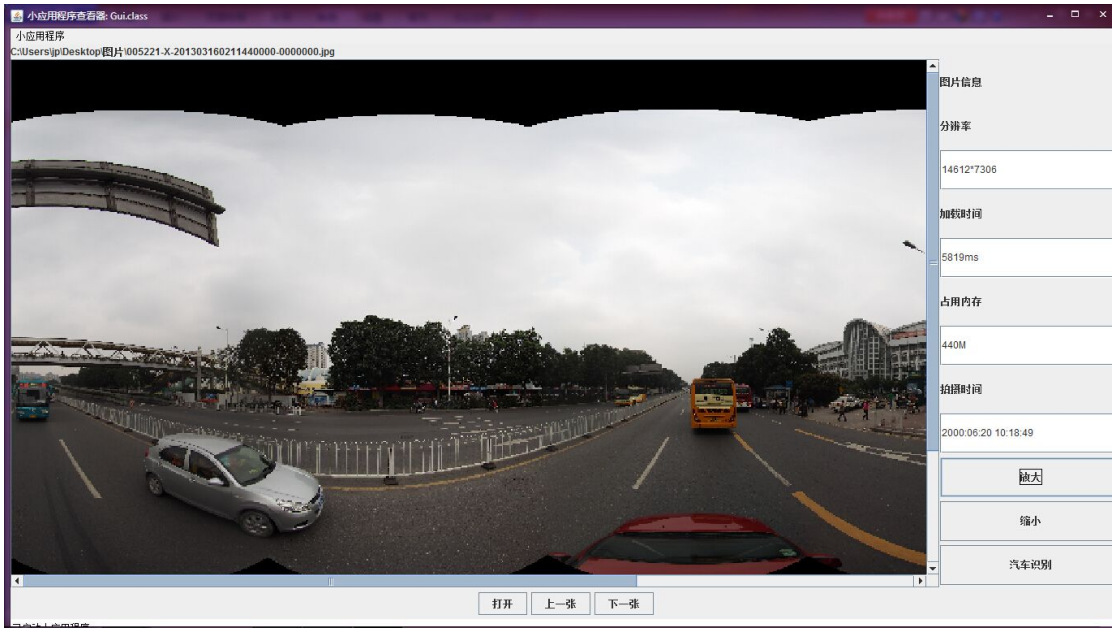
3.打开一张图片

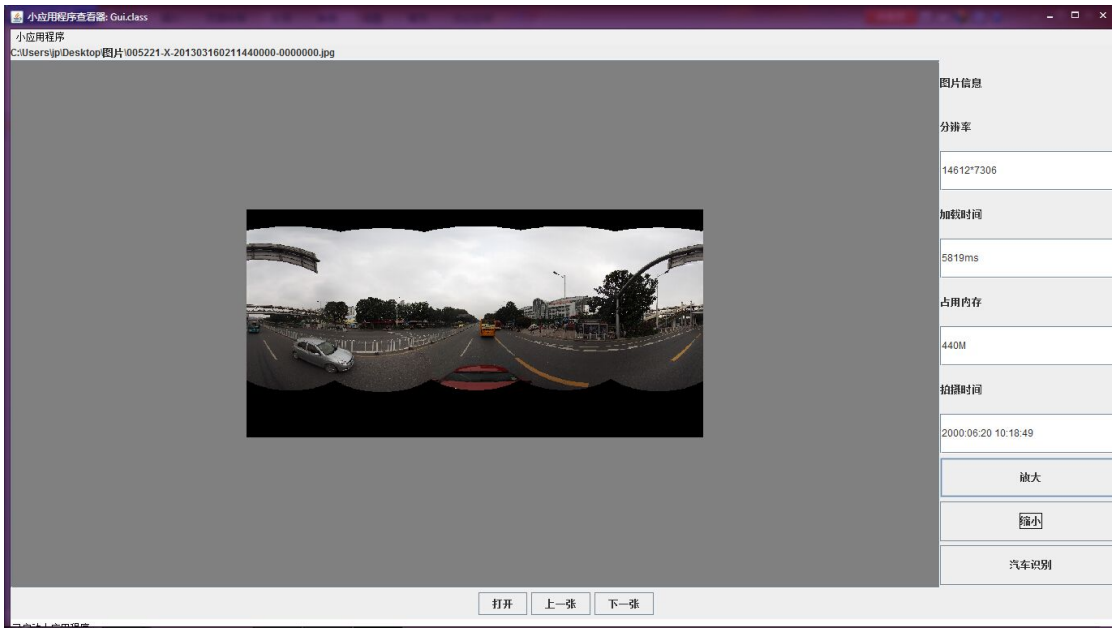


可以看到，在右边的面板上显示了分辨率，加载时间，占用内存情况和拍摄日期。顶部显示图片的路径。

4.进行放大和缩小操作

放大以后可以通过滚动条查看完整图像。





5.车辆识别功能（参考功能，未完善）

五．总结

1.遇到的问题

从技术角度来讲，程序设计过程中主要遇到两方面困难的问题。

1）超高分辨率的显示问题。我选择 java 作为此次的编程语言是因为 java 里面有各种各样的类，方便我处理很多问题。但是选择 java 有个缺点就是 JVM 的内存占用问题。小图像显示没有问题，但是从上面可以看到，超高分辨率的图像非常占内存，而且似乎没有很好的方法。有一种可选择的方法是修改 JVM 的内存，但这不是一种理想的方法，因为我不知道到底需要处理多大的图片。另外的方法设置缓存，我试了下也不明显。我觉得最理想的方式是实现自己的 ImageReader，分块处理数据文件，但是需要分析 jpg 的文件格式和写底层代码实现

优化，很困难！我认为这是 ImageIO.read(File input)方法的一个缺陷。最后我只能将图像缩小后绘制。

2.）车辆（这里主要讲车辆，行人的识别误差太大）识别

可以先对图像进行预处理，消除原始图像的噪声和畸变，消减无关特征加强图像感兴趣的特征。但由于时间原因，我没有对图像做这种处理。我的这个程序只是对图像进行了二值化，我的思路主要是首先观察图片（或者根据经验），0—width/2 部分的图像我没有考虑，因为这部分从图上来看是没有车的（要看本程序具体的应用场景即拍照地点等），对于剩余部分得到灰度值并二值化以后，可以增加车辆和路面的对比度，在识别出车辆。但是困难在于当光照较强时，会造成某些部分的亮度很高，具有迷惑性。这种情况我的处理方式是改变二值化的阀值，过滤掉这部分的影响，但会加大误差（过滤掉部分车辆）。

（二值化以后的图像，阀值 100）

（二值化以后图像，阀值 160）

（可以看出，过滤掉了路边栏杆和斑马线的影响，但部分车辆不能识别出来，其实可以动态确定阀值）

一般的处理方式我认为是首先使用 Robert 边缘算子强化车辆的轮廓信息，弱化非边缘信息。但是这种方法有一种缺陷就是边缘不是连续的曲线，并且精确度不高。

（基于 Roberts 算子的边缘检测）

通过查找文献，目前有一种彩色图像的边缘检测，将灰度边缘的模板算子扩展到彩色的边缘检测，使用多通道数据融合技术充分利用图像本身的色彩和梯度信息。另外结合滤波和边缘生长等方法抑制图像中的噪声和保证边缘的连续性。
确定边界以后，如果知道背景图的话，可以利用差影法去除背景，此时就可以得到车辆的轮廓，最后通过形态学运算能平滑图像的轮廓。
因为本问题没有背景图，所以我只是采取的简单的识别方法。

六．心得体会

短短的一周，期间还有几门考试，对于我来说时间真有点紧张。作为数学系的学生，以前很少用 Java 写过这么长的程序，通过这次实验，我发现设计模式对于 Java 的重要性和面向对象技术的灵活性。以前写程序都是只关注算法，比如 POJ 上面的那种题目，只要思路有了，实现起来很简单。现在发现，不仅要知道核心的算法，也要知道怎么编程把它实现，特别是对于规模比较大的程序。

七．参考文献

[1] 邵虹, 崔文成. 基于 BP 神经网络的人脸图像识别方法的研究[J]. 沈阳工业大学学报, 2000, 22(4): 346-348.

[2] 于烨, 陆建华, 郑君里. 一种新的彩色图像边缘检测算法[J]. 清华大学学报: 自然科学版, 2005, 45(10): 1339-1343.
[3] Liang Y. Java 语言程序设计[M]. 第三版. 机械工业出版社, 2004:204-340.
[4] Zhang H. 计算机图形学（Java2D 和 3D)[M]. 机械工业出版社, 2008:94-101.