**车票序列号检测与识别**

——图像处理课程设计报告

1500012741 陈小康

1500012797 赵浩然

**一、任务说明**

1. 检测车票的位置，将它裁剪出后摆正。

2. 在裁剪后的图片中标注出21位码和7位码，并将其分割。

3. 对每张图片，输出对21位码和7位码的识别结果。

**二、算法原理**

**1. 车票票面定位**

这部分主要目的是检测车票的轮廓，并进行较为精确的轮廓贴近。

（1）考虑到车票票面与背景颜色不同，首先对整张图像进行二值化。由于四周噪声比较多，二值化后会产生线装条纹，可能影响后期操作。因此，在此基础上使用大小 9\*9 的中值滤波去噪。二值化和滤波后的二值化图像分别见下图。可以明显地看出，去噪后，除了票面区域外几乎无噪点。

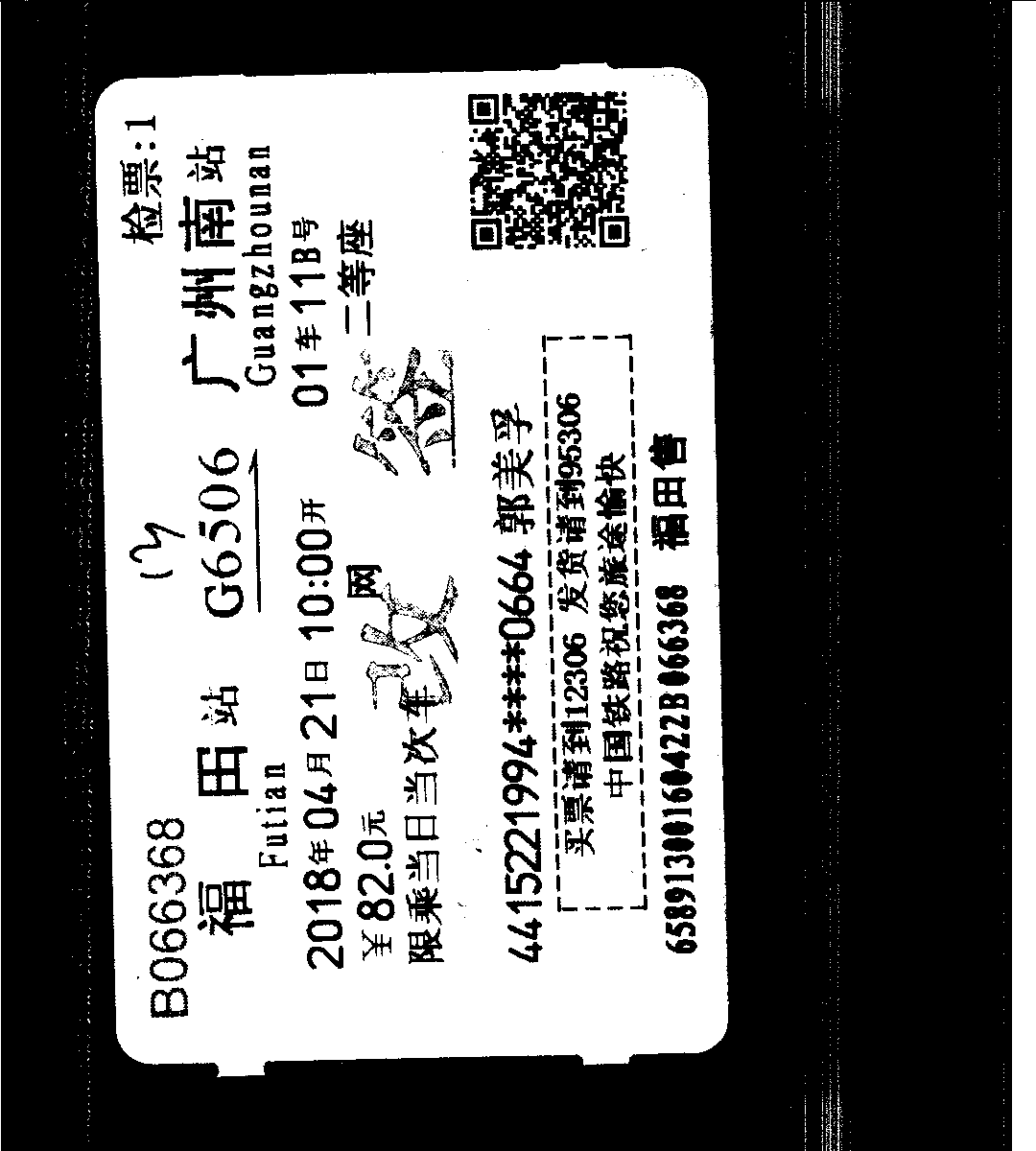


图 1 二值化图像



图 2 二值化并去噪的图像

（2）形态学操作

为了能准确检测车票边缘的线，而不是检测出其他乱七八糟的线条，我们需要对车票中间的黑色进行“填充”。最适合的办法莫过于开操作和闭操作了。首先用闭操作填充中间的黑色，然后用开操作“精修”，去除车票四周突出的小矩形（如图2车票下方一排两个小矩形）。处理后的图像如下。

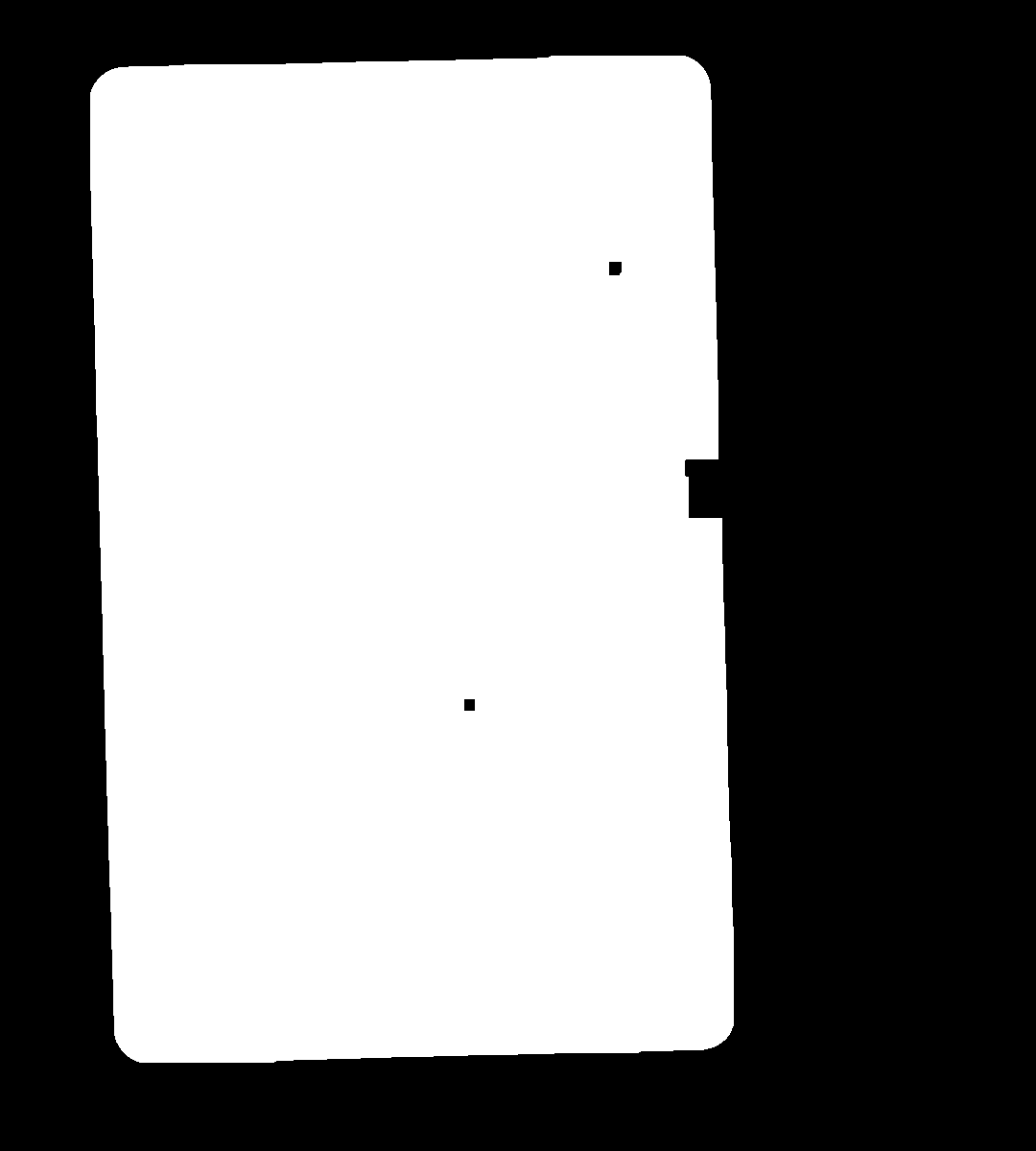


图 3 形态学操作后的车票票面

（3）轮廓检测

先使用 Canny 算子进行边缘检测，然后利用霍夫变换检测直线。之后利用检测到的直线，计算一个最小拟合矩形，以及矩形的四个顶点的位置。由于图中可能存在其他干扰情况，所以最后可能得到多个矩形框，但只保存周长最长的矩形框，作为车票票面。将矩形框画在原图上效果如下：



图 4 车票票面轮廓检测结果

（4）旋转与裁剪

上一步骤已经得到矩形四个顶点，此时可以用顶点计算边的斜率，然后用斜率计算矩形倾斜角，旋转图像使矩形摆正，然后裁剪出来。为了方便后续操作，裁剪后的图像需成为“横长竖短”这样的形状。

对于已经旋转成横长竖短的车票，我们检测其右下角和左上角的灰度值。如果左上角灰度值小于右下角，说明此时二维码在左上角，车票放置反了，我们将其旋转180°回到正常位置。同时如果车票有缺损，我们通过检测二维码区域（右下角）的平均灰度值，如果偏高说明缺损的是右侧带二维码区域，否则缺失的是左侧区域，我们按照车票平均长宽比，将缺损的区域用白色补全。由于训练集中只有长边缺损的数据，我们只补全长边的缺失。

**3. 在图片中标注21位码**

Num21是检测21位码的类，输入图片为分割好并摆正的车票。

二值分割。我们首先将输入图像二值化，去除背景和噪音的影响。21位码附近是空白背景，21位码本身是黑色字迹，我们采用固定阈值分割，阈值灰度设为20。分割结果如下：

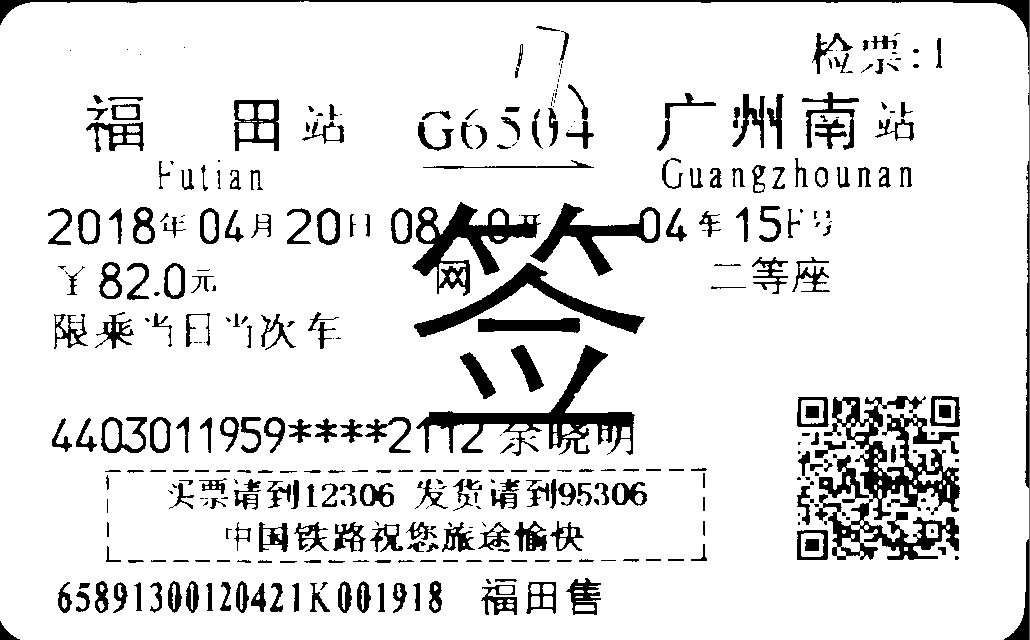


图 5 21位码二值化结果

定位二维码。由于二维码和21位码的相对位置和长宽比比较固定，我们先检测出二维码的高度和位置，以其为标准定位21位码。这一功能封装为函数findBitcode()。该函数输入和Num21相同。对输入的图像首先进行阈值为20的二值分割，以去除背景和噪音的影响。接下来截取右下角二维码部分，取反并做闭操作，使得二维码成为连通区域。由于训练数据中有一张车票右边缺损的图片，只取左半边的二维码。然后我们根据该区域图片行和，删除底部车票裁剪不精确产生的边缘。接下来将操作后的左半边二维码放回车票对应区域，剩余部分用黑色模板掩膜，随后调用DetectRectangle(). getRectangle()函数，获得二维码最小矩形闭包的顶点，并将其返回。

获取21位码区域。经过对图片的测量，我们确定了二维码高度和像素的映射关系，二维码高度和二维码左边缘到21位码左侧、顶部的长度比，以及与21位码长宽的比例。由此根据findBitcode的返回值，可以确定21位码的大致范围。我们对这一区域先取反，然后做闭操作和膨胀操作，让21位码形成连通区域，同时避免框得太紧。然后将处理后的结果放回车票对应区域，其余区域用黑色背景掩膜。接下来调用DetectRectangle(). getRec-tangle()，获得最大连通区域的最小矩形闭包的四个顶点，将其作为一个返回值。另一个返回值是画好了21位码区域的车票，这个值主要是在编程过程中查看中间结果用。

**4. 在图片中标注7位码**

Num7是检测7位码的类。Num7和Num21的操作过程类似。

二值化处理。和基本为纯黑色的21位码不同，7位码灰度值较大，我们采用双阈值分割，将60 ~ 120灰度值之间的像素值设为255，其余区域设为0，相当于在二值化的同时做了取反处理。

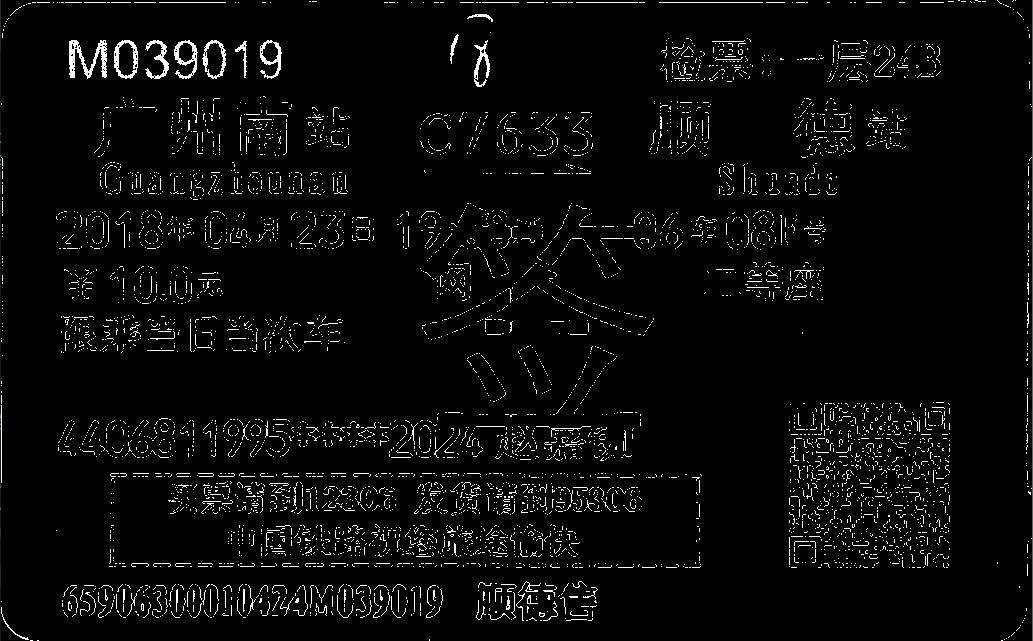


图 6 7位码二值化结果

接下来我们参考21位码部分的操作，调用findBitcode函数获取二维码闭包顶点，根据二维码高度和与7位码距离、7位码长宽等信息裁出7位码大致区域。

与21位码不同，7位码的印制区域误差比较大，有些还和下面的车站信息重叠到一起。在二值化并裁减之后，保留的是7位码和车站名汉字的部分轮廓。

我们首先尝试用均值滤波和阈值分割去除汉字，这一步骤后汉字轮廓大部分被去除，对残留的类似盐噪声的白点，再用中值滤波将其去除。然后同样用闭操作好膨胀操作使得7位码区域连通，同时防止框得太紧。接下来参考Num21，对处理后的区域掩膜并放回原图，调用DetectRectangle(). getRectangle()，获得最大连通区域的最小矩形闭包的四个顶点，和画出了7位码区域的图片一起作为返回值。



图 7 均值滤波结果

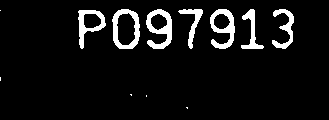


图 8 中值滤波结果

**5. 票面标注与分割**

NumAll根据以上两个类返回的矩形闭包顶点在车票图片中对应位置框出两组码并分割。输入也是分割好并摆正的车票，另外还有21位码和7位码的最小矩形闭包顶点。首先将输入的灰度图像转化为RGB彩色图像。然后调用两次drawSegLines函数先后画出21位码和7位码的区域，并分割各位数字/字母。最后将用红线标出分割结果的图片返回。

drawSegLines函数输入的是裁剪好的车票图片，以及闭包顶点和数字+字母总数。首先判断输入的是灰度图还是RGB彩色图，如果是灰度图转换为彩色图。然后将四个顶点按照左下角、左上角、右上角、右下角的顺序排列。此处由于车票已经摆正，矩形长边的斜率一般都很小，所以将横纵坐标之和最小的点作为左下角，然后根据与左下角横纵坐标之差依次定位其它点，按顺序生成新的顶点集合box\_new。然后计算长边斜率，调用DetectRectangle(). drawLine()画出外轮廓。

接下来我们根据分割数在内部同样用红线分割码字。由于字母和数字的宽度不同，我们要判断是7位码还是21位码，对其中字母部分额外增加宽度。我们分别以box\_new[0]和box\_new[1]为起点，在横轴上每次增加相同的宽度，取各自长边上对应位置的点，用cv2.line连线。这样就把21位码和7位码根据位数不同分割了。



图 9 21位码分割结果



图 10 7位码分割结果

**6. 识别21位码**

**7. 识别7位码。**

**三、实验结果及其分析**

**四、结论及存在的问题**

**五、实验分工**

陈小康：

赵浩然：负责21位码和7位码的标注与分割，将横长竖短的车票摆正并补全，对裁切出来的码字区域删除多余边缘。

**六、参考文献**