

一维码识别工程实验报告

13354372 熊铠能

1. 一维码简述;

一维条码是一种能用于信息编码和信息自动识别的标准符号,是由一组宽度不同的黑白符号按一定规则交替排列编码组成的图形符号,用于表示一定的信息。

码制指条码符号的类型,不同的类型有不同的编码规则。我们本次实验是基于 EAN-13 码制。EAN-13 码主要由起始符(3)、左侧数据符(42)、中间分割符(5)、右侧数据符(42)、校验符、终止符(3)组成,一共 95 个模块,表示 13 个字符。条表示 1,空表示 0;只能表示 0-9 这十个数字;每个字符的宽度为 7 个模块,交替由两个条和两个空组成,每个条或者空的宽度不超过 4 个模块。起始符 101,中间分割符 01010,终止符 101。

我完成的这个识别程序能解析的条码类型包括标准、受噪声污染以及倾斜的一维码图像。

2. 解码方法(分为图像处理和译码两个部分);

2.1 图像处理

2.1.1 用 `imread()` 方法载入需要验证的一维码图像;



2.1.2 将载入的 RGB 三通道图像转化为灰度图像,每个像素点取值范围为 0-255,共有 256 个灰度级别。用 `rgb2gray()` 函数可得到灰度图:



2.1.3 用大津法求阈值进而从灰度图像得到二值图,二值图的像素点取值范围不是 0 就是 1,利于我们后续的译码操作.求阈值用 `graythresh()` 函数,求二值图用 `im2bw()` 函数:



2.1.4 接下来可以对图像进行滤波去噪以及图像校正，这一部分将在后文详细描述。这里先只讨论标准一维码的图像。

2.2 译码

2.2.1 获取条和空的宽度:这里的思路是遍历图像的每一个像素点，在一行中，当遇到像素值与其后一个点像素值不等的时候，记录其位置；后面的位置减去前面的位置，既可以得到条或空的宽度。

对于一张标准的一维码图像，边界区域有 60 个，所以每一行应该有 59 个条/空的宽度值，当某一行的宽度值不等于 59 时，忽略该行。同时在这一步做了一个优化操作：由于得到的二值图中的条码的边界可能会出现锯齿和毛刺等现象，这就导致每次计算的宽度可能不一样，减少这个误差的方法是将所有有效行（59 个宽度）的宽度相加后取平均值。相关代码如下：

```
[m,n]=size(A);
number=0;
for i=1:m
    pos_cnt=1;width_id=1;
    for j=1:n-1
        if A(i,j) ~= A(i,j+1)
            pos(i,pos_cnt)=j;
            if pos_cnt>1
                width(i,width_id)=pos(i,pos_cnt)-pos(i,pos_cnt-1);
                width_id=width_id+1;
            end
            pos_cnt=pos_cnt+1;
        end
    end
    if width_id==60
        number=number+1;
        for k=1:59
            %将所有条/空的宽度都存储在total_len这个二维数组里
            total_len(number,k)=width(i,k);
        end
    end
end
end
```

```

[mm,nn]=size(total_len);
for i=1:nn
    tmp=0;
    for j=1:mm
        tmp=tmp+total_len(j,i);    %该宽度的所有值求和
    end
    final_width(1,i)=tmp/mm;        %求均值
end

```

2.2.2 获取单位模块宽度以及条空比例：前文已经提到，一维码图像包括 95 个图像，将上一步得到的全部宽度求和，除以 95 即可得到单位模块长度。然后将每个条/空的宽度除以单位模块宽度，即可得到条/空比例。这一步比较简单就不贴代码了。

2.2.3 对条和空区域进行 0/1 标注：将条码区标注成 1，空白区标注成 0；这里需要注意的是，一个单位模块只能标注一种符号，条码和空白区域可能占据三四个单位模块。标注完成后，检查起始符（101）、中间分割符（01010）、终止符（101）是否符合 EAN-13 的条件，不符合则输入相应的判断信息，否则进行下一步：

```

index=1
for i=1:59
    if mod(i,2)==1
        for j=1:1:round(proposition(1,i))
            mat95(1,index)=1;
            index=index+1;
        end
    else
        for j=1:1:round(proposition(1,i))
            mat95(1,index)=0;
            index=index+1;
        end
    end
end
isCheck=0;
if(mat95(1,1)==1&&mat95(1,2)==0&&mat95(1,3)==1&&mat95(1,46)==0&&mat95(1,47)==1&&mat95(1,48)==0&&mat95(1,49)==1&&mat95(1,50)==0&&mat95(1,93)==1&&mat95(1,94)==0&&mat95(1,95)==1)
    isCheck=1;
end
if isCheck==0
    msgbox('不满足EAN-13码的条件! '); %不满足则弹出msg框，同时终止程序
    return
end

```

2.2.4 查表译码:

计算左侧和右侧数据栏的十进制数:

```
j=1;
for i=4:7:39
    left(1,j)=bin2dec(num2str(mat95(1:1,i:i+6)));
    j=j+1;
end
k=1;
for i=51:7:86
    right(1,k)=bin2dec(num2str(mat95(1:1,i:i+6)));
    k=k+1;
end
```


查表得到左边和右边各 6 个字符对应的 0-9 字符, 同时根据表格创建一个 Map: 根据左边数据用 AB 字符集序列得到前置位. 部分代码如下:

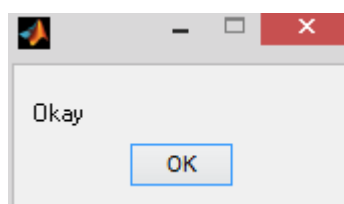
```
checkLeft=[13,25,19,61,35,49,47,59,55,11,39,51,27,33,29,57,5,17,9,23];
num_bar='';
AB_check='';
%以下求得左边序列以及AB序列
for i=1:6
    for j=0:19
        if left(i)==checkLeft(j+1)
            if j>9
                AB_check=strcat(AB_check,'B');
            else
                AB_check=strcat(AB_check,'A');
            end
            num_bar=strcat(num_bar,num2str(mod(j,10)));
        end
    end
end
%以下根据Map得到对应的前置位
preMap =
containers.Map({'AAAAAA','AABABB','AABBAB','AABBBA','ABAABB','ABBAAB',
'ABBBAA','ABABAB','ABABBA','ABBABA'},...
{'0','1','2','3','4','5','6','7','8','9'});
pre=preMap(AB_check);
num_bar=strcat(pre,num_bar);
```

接下来就是检查校验位是否正确:将前面的 12 个数字的奇数位相加,得到一个数 oddSum,偶数位相加得到 evenSum, 令 $c = \text{oddSum} + 3 * \text{evenSum}$, 若 c 的个位数为 0, 则校验位为 0; 否则校验位为 $10 - c \% 10$. 这里判断两个数是否相等时稍微注意一下是否是同一类型的。相关代码如下:

```
oddSum=0;evenSum=0;
for i=1:12
    if mod(i,2)==1
        oddSum=oddSum+str2num(num_bar(i));
    else
        evenSum=evenSum+str2num(num_bar(i));
    end
end
c=oddSum+3*evenSum;
if mod(c,10)==0
    checkBit=0;
else
    checkBit=10-mod(c,10);
end
%如果checkBit和13位的最后一位相等,则识别正确,否则错误。弹出相应信息
if num2str(checkBit)==num_bar(13)
    msgbox('Okay')
else
    msgbox('Failed');
end
```

对应上文中的那张一维码图,检验结果如下:

 num_bar '6937526503743'



3. 所做的额外工作;

3.1 对于倾斜一维码图像的校正:



对于像上图这样的一维码图像，我们在遍历一行试图求条/空的宽度时，是无论如何也得不到正确结果的，因为图像倾斜后宽度都变长了。所以较好的做法是将这个图像摆正，摆正的关键是找到**偏离角度**。这里选用的 **hough 直线检测** 方法。hough 变换的主要思想是将该方程的参数和变量交换，对于直线 $y=kx+b$ ，即用 x, y 作为参数， k, b 作为变量，所以在直角坐标系中的直线 $y=kx+b$ 在参数坐标上表示为点 (k, b) ，而直角坐标上的点 $(x1, y1)$ 则在参数坐标下表示为一条直线。此外，为了计算方便，将参数控件的坐标转换成极坐标进行运算。

所以，先将图片进行边缘检测，然后对图像上每一个非零像素点在参数坐标下变换为一条直线，然后根据统计方法找到聚集点即可。边缘检测可以使用 `edge()` 方法，这里使用的是 `canny` 边缘检测：



以上算法，`matlab` 都帮我们封装好了。这里还有一个小技巧：因为我们需要验证的是一维码图像，一维码图像的特点是所有条/空都是两两平行的，所以我们根本没有必要找出所有的直线，而仅仅需要找出最长的那一条直线（其实无论哪一条都无所谓，对结果没什么影响）即可：

```
[H,T,R]=hough(BW);
P=houghpeaks(H,4,'threshold',ceil(0.3*max(H(:))));
```

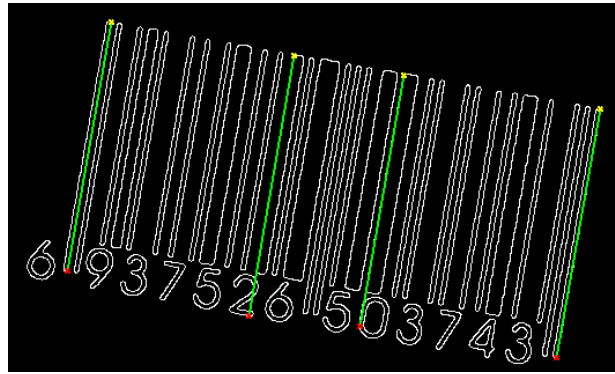
这一句选取了4个峰值，即聚集点，所以对应到参数坐标上是四条直线；`H`对应的是 θ 和 ρ 的关系矩阵，两个参数分别代表极坐标中的夹角和到原点的距离。

```
lines=houghlines(BW,T,R,P,'FillGap',50,'MinLength',10);
```

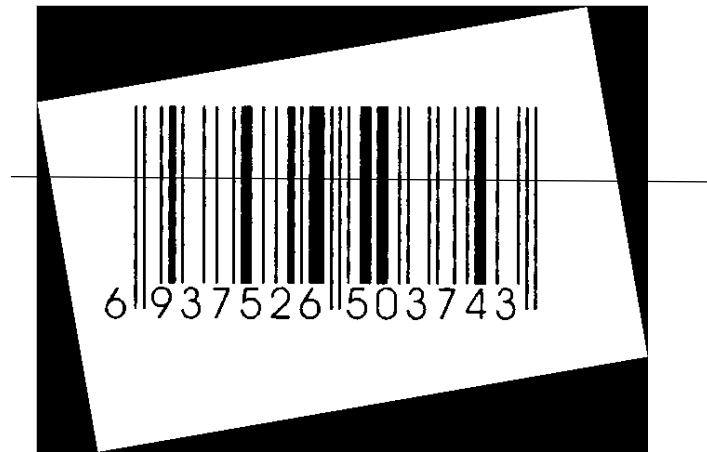
这里就是利用 `hough()` 函数返回的参数值选取线段；参数 50 是一个正的标量，指定了与相同的 hough 变换相关的两条线段的距离，小于该距离则将线段合并；参数 10 是一个正的标量，指定合并的线是丢弃还是保留。`lines` 里的成员是一个结构体，包含了线段端点的坐标等信息。

```
[L1,Index1]=max(Len(:));
x1=[lines(Index1).point1(1) lines(Index1).point2(1)];
y1=[lines(Index1).point1(2) lines(Index1).point2(2)];
K1=-(lines(Index1).point1(2)-...
lines(Index1).point2(2))/(lines(Index1).point1(1)-
lines(Index1).point2(1))
angle=atan(K1)*180/pi
A = imrotate(I,90-angle,'bilinear');
```

先找到最长线段 `L1` 以及索引 `Index1`，根据端点求出斜率 `K1`，然后用反正切函数 `atan()` 找到偏离角 `angle`。`imrotate()` 默认逆时针旋转，所以最后的结果是将原二值图像逆时针转 $90-\text{angle}$ 。图一是线段标识图，图二是校正后的图：



图一



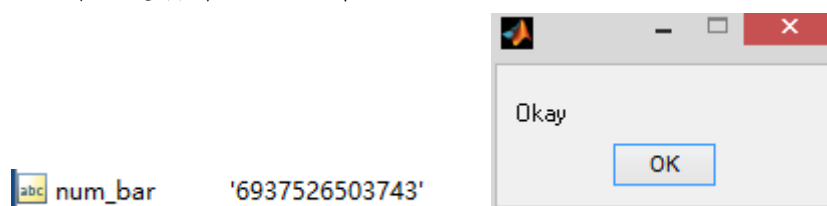
图二

这里还有一个很蛋疼的地方，可以发现经过旋转后的图比原图更大，而且四周出现了四个角，关键这四个角还是黑色的，这就会引起一个很严重的问题：在遍历图像某一行时，条/空的数量会比原来多（单单考虑图二的话，确切的说是所有有效行多了两条），画条线看的更清楚。所以在程序中这部分加了一个特判：对于旋转过的图像，计算宽度值的方法要和未旋转的图像区分开来，相关代码如下：

```
%以下是针对校正的图像的
if angle~=90
    if width_id==62
        number=number+1;
        for k=1:59
            total_len(number,k)=width(i,k+1);
        end
    end
%以下是针对未校正的图像的
else
    if width_id==60
        number=number+1;
        for k=1:59
            total_len(number,k)=width(i,k);
        end
    end
end
```

当然，上述代码只是针对特定的图而言的，更一般的做法是：如果某一行的条/空总数大于 59，如 61，则要去掉第一个值和第 61 个值，保留中间的 59 个值作为有效值；如果总数等于 59 则满足要求；小于 59 则直接忽略改行。代码很容易，在这里不再赘述。

下面是该图的验证结果：



3.2 判断一维码图是否符合 EAN-13 码制的标准：以群里给出的图 1. jpg 为例：



这张图乍一看工工整整的，但是在 0/1 标注后，得到的 0/1 数组的大小为 100；除了检验起始符（101）、中间位（01010）、终止符（101）之外，还有一种检验是否符合 EAN-13 码的方法：

```
%在得到条空比例之后
test=round(proposition);
test_sum=sum(test);
temp=0;
isValid=true;
ii=0;
for k=4:28
    if mod(ii,4)==0
        if(temp==0 || temp==7)
            isValid=true;
        else
            isValid=false;
            break;
        end
        temp=0;
    end
    temp=test(k)+temp;
    ii=ii+1;
end
```

这里充分利用了 EAN-13 码的性质之一：每个字符有两个条和两个空组成，一共有 7 个模块。

3.3 处理含有椒盐噪声的图像：对于一张含有椒盐噪声的图像，我们做识别处理肯定是会增大误差的。下面是一张例图：



我们滤波的对象是二值图，所以先需要用前文中提及的方法来将这张 RGB 图转化成二值图，再做滤波处理。相关代码如下：

```
A=imread('5.jpg');  
figure(1),imshow(A);  
A=rgb2gray(A);  
A=im2bw(A,graythresh(A));  
A=double(A);  
K = medfilt2(A,[2,2]);  
figure(2),imshow(K);
```

这是直接使用中值滤波函数 `medfilt2()` 的例子，滤波后的图像如下：



这里值得注意的一点是 `medfilt2()` 函数的第二个参数，是一个 $[N, M]$ 大小的滑动窗口，对于某一个像素点 (x, y) ，仅处理它邻域的响应。窗口越大，就有越多的像素点对中心像素点有影响。一般而言当图像比较小时，选取的滑动窗口也应该相应的小。对于上面那段代码，如果将滑动窗口改成 3×3 的话，就会牺牲更多的清晰度，效果很差。图像如下所示：



4. 总结和体会：

体会了一遍一维码译码的流程，虽然说比较简单，使用的算法都是 `matlab` 封装好的，但是还是有很多细节让我印象深刻，学会了不少东西。物联网技术导论虽然感觉课堂讲的一些内容好想很水，但是这种有点工程性质的作业还是很不错的，包括这次的一维码验证，还有实验课的 `MFC` 程序。希望戒骄戒躁，继续努力！