基于KNN的车牌识别算法

实验目的：从汽车照片中提取车牌，并借助**MNIST**数据库完成对车牌号码的识别

实验人：李东嵘

实验平台：**MATLAB R2016a**

1.任务综述

在本次实验中，我们需要对从下图中汽车的车牌号码进行识别：

这不是一个简单的任务，其涉及到图像分割，语义识别与数字匹配等问题。为了使任务脉络更加清晰，我们将图像识别任务分为以下几个部分：

1.定位车牌位置

2.分割出数字

3.对手写数字进行识别

在编程实现中，我们需要分别实现上述的三个功能模块，最后将其组合至main函数中。

但是，我们知道，上述需求中除了手写数字识别外其他难度均较高，因此我们做出如下假设：

1.车牌总是出现在图片的下半部分

2. 需要我们识别的车牌往往是白底黑字

3.需要我们识别的车牌只有四个数字

这些不算太强的假设。在这些假设下，我们设计提取车牌号码的程序。

2.定位车牌区域

首先，我们要定位出车牌所在的矩形区域，并将它单独抠出来。为了完成这一目的，一个朴素的想法是，我们将图像二值化，然后对图像进行去噪，最后在图片的下半部分用灰度匹配的方法求出图像区域。

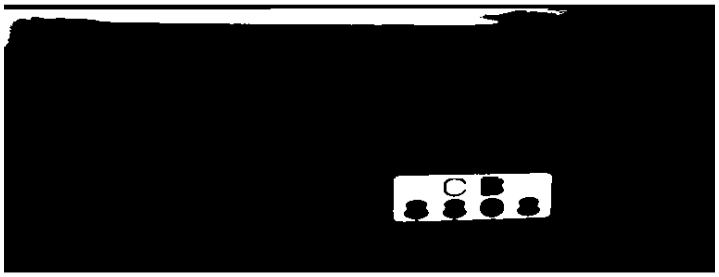
具体而言，其算法流程如下：

1. 将图像G二值化。
2. 将图像中较大快的白色噪声去掉，这样在图像下半部分基本上只剩下车牌照处为白色。
3. 找出下半部分白色点的白色点坐标集合X,Y,
   令$x_{min} = min(X)$，$x_{max} = max(X)$，$y_{min} = min(Y)$，$y_{max} = max(Y)$
4. 令G=[$x_{min}:x_{max}$,$y_{min}:y_{max}$]，所得区域G即为车牌部分

我们用Cutboard函数实现提取车牌区域的功能（见附录1）。

为了验证算法有效性，我们先对图像进行二值化去噪：

```
car=im2bw(car);
%图像去噪，去除像素数量在2000以上的点集
car=bwareaopen(car,2000);
imshow(car)
```



我们可以看到，去噪后的二值图像下半部分只有车牌号。于是，我们就可以通过Cutboard函数提取车牌部分：

```
board=Cutboard(car)%注意，这里的car是原图像
```

我们也就得到了车牌区域：

车牌号区域得到了较好的定位。接下来，我们需要从车牌号码图片中提取数字。

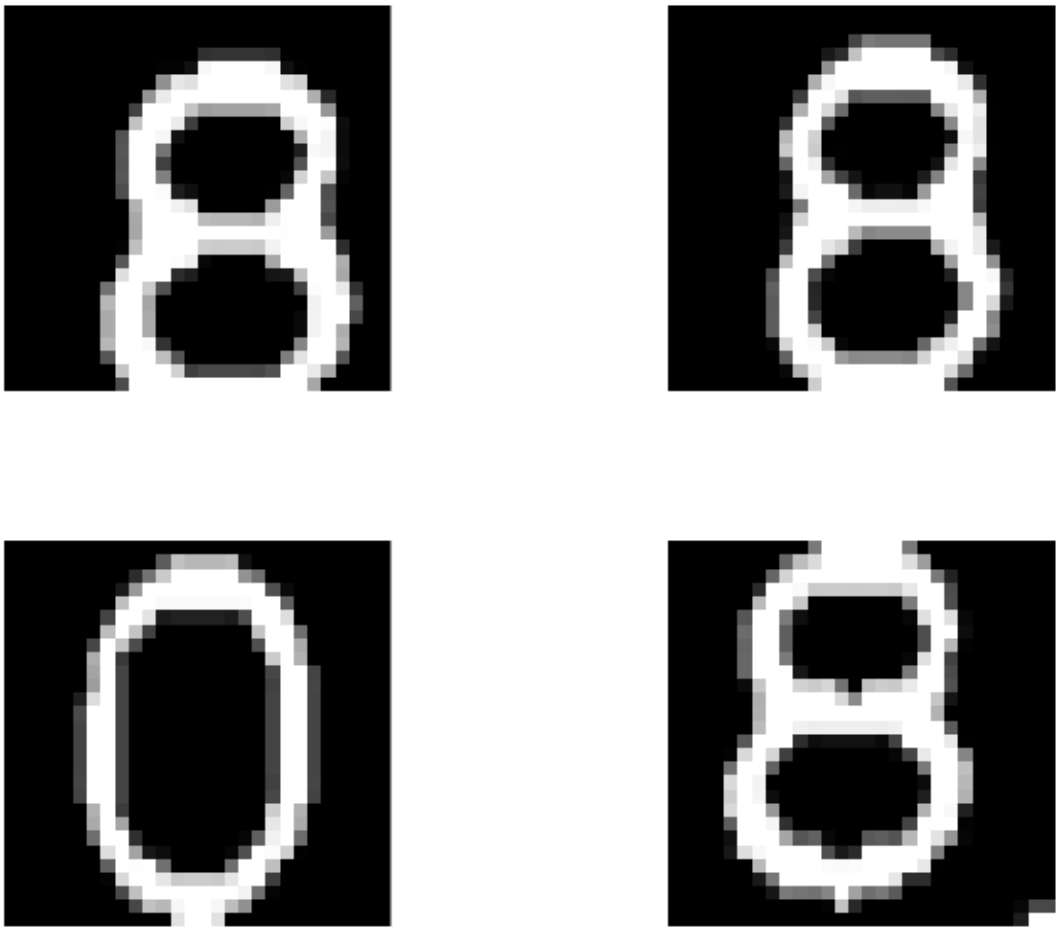
3.数字提取

在本节中，我们实现AcquireNumber函数，用于提取车牌号码图片中的数字。

由于我们已经假设车牌号码会遵循一定的格式，因此可以基于车牌号码图片开发出通用的分割车牌数字算法。在图片中，我们知道，车牌数字往往位于车牌号的下半部分，而且在水平上四等分整张图片。对图片的进一步分析可以帮助我们确定分割的具体参数。然后，不光对于这张车牌本身，事实上对于所有的符合格式的车牌我们均可以对其按照这种方法分割。分割方法的具体实现可以参照附录2中的AcquireNumber函数.

下述代码实现了数字分割与显示:

```
numbers=AcquireNumber(board);
figure()
subplot(2,2,1)
imshow(numbers(:,:,1))
subplot(2,2,2)
imshow(numbers(:,:,2))
subplot(2,2,3)
imshow(numbers(:,:,3))
subplot(2,2,4)
imshow(numbers(:,:,4))
```



这是对车牌数字进行识别后的结果，显然，车牌号是8808. 至此，我们已经成功实现了车牌号码的分割。接下来，我们进行数字匹配。

4.数字匹配

在数字匹配中，我们使用knn算法对手写数字进行匹配，训练集为MNIST。

KNN的思想是简单的，我们将每个数字看作是一个m*n维的高维向量，对于输入u，我们在训练集中定位与其距离前k近的k个向量，并考察这k个向量的标签，将占比最大的标签作为该向量的标签。这是因为，我们可以合理地假设输入向量与同标签向量在几何上往往是相近的。附录3实现了knn具体算法。

我们用mnist数据集的前60000个作为训练集，第60001-60100个作为测试集，测试knn效果：

```
%采用50nn,c为标签向量
output=knn(100,mnist(:,:,1:60000),c(1:60000),mnist(:,:,60001:60100));
rate=output==c(60001:60100);
sum(rate)/100
```

```
ans =

    0.9100
```

可以发现，最简单的50nn都能轻易达到91%的准确率，这说明knn，尽管算法简单，但是识别准确率还是比较高的。

在此基础上，我们用knn技术对手写数字进行识别。

为了更好地反应数字特征，我们用Sobelev算子对mnist数据集与车牌号码进行滤波以提取图像的边界。在此基础上，我们再分别将这些图像二值化，以达到更好的特征提取和模式匹配效果。对mnist数据集的Sobel滤波由于耗时极长，因此再次按下不表。下面的代码实现的是对车牌号码进行滤波与二值化后，用300nn算法对其进行匹配，并最后汇报识别结果。

```
for i=1:4
    tmp=numbers(:,:,i);
    tmpx=fil(tmp,sobelx);
    tmpy=fil(tmp,sobely);
    numbers(:,:,i)=sqrt(tmpx.^2+tmpy.^2);
    numbers(:,:,i)=im2bw(numbers(:,:,i));
end

figure()
subplot(2,2,1)
imshow(numbers(:,:,1))
subplot(2,2,2)
imshow(numbers(:,:,2))
subplot(2,2,3)
imshow(numbers(:,:,3))
subplot(2,2,4)
imshow(numbers(:,:,4))


knn(3000,mnist,c,numbers)
```

由于MNIST数据集有70000幅图片，因此300nn并不会造成过拟合。

车牌号码在sobel滤波并二值化后得到的结果如下图所示：

识别结果如下所示：

```
ans =

     8
     8
     0
     3
```

可以看到，前三个数字都得到了较好的识别，但最后一个数字识别失败，算法将8错误识别成了3.

5.APPENDIX

1.Cutboard函数

```
%输入：汽车图片
```

```
%输出：车牌区域
%function board=Cutboard(car)

carbw=im2bw(car);
%去噪，删去多于2000个像素点的白色区域
carbw=bwareaopen(carbw,2000);

%获取车牌坐标，并舍去x小于100（即在图像上半部分）的点
[x,y]=find(carbw==1);
index=find(x>100);
coordinates=[x,y];
coordinates=coordinates(index,:);

%用x1，x2,y1,y2定位车牌坐标
x1=min(coordinates(:,1));
x2=max(coordinates(:,1));
y1=min(coordinates(:,2));
y2=max(coordinates(:,2));

board=car(x1:x2,y1:y2);

%end
```

2.AcquireNumber函数

```
%输入：车牌图片
%输出：numbers为一28*28*4图片，每一层为一个数字
%function numbers=AcquireNumber(board)

[m,n]=size(board);

numbers=board(round(m/2):m,1:n);
numbers=im2bw(numbers);
numbers=abs(numbers-1);

[m,n]=size(numbers);

%提取数字区域
number1=numbers(1:round(0.8*m),2:round(n/4));
number1=imresize(number1,[28,28]);
number2=numbers(1:round(0.8*m),round(n/4):round(n/2));
number2=imresize(number2,[28,28]);
number3=numbers(1:round(0.8*m),round(n/2):round(3*n/4));
number3=imresize(number3,[28,28]);
number4=numbers(1:round(0.8*m),round(3*n/4):n-4);
number4=imresize(number4,[28,28]);
numbers=zeros(28,28,4);

%将数字图片嵌入numbers三维矩阵
numbers(:,:,1)=number1;
numbers(:,:,2)=number2;
numbers(:,:,3)=number3;
numbers(:,:,4)=number4;

%end
```

3.knn函数：

```
%输出：测试集上每个图片的标签
%输入：t：k的具体数字，因为函数体内有同名变量，因此用t代替
%train：训练集
%train_label：训练集标签
%test：测试集
%function output=knn(t,train,train_label,test)

[tm,tn]=size(train_label);
[m,n,k]=size(test);
[a,b,c]=size(train);
output=zeros(k,1);

%vec存储训练集中所有图片的拉直向量
vec=zeros(a*b,c);
%距离矩阵
dis=zeros(c,1);

%初始vec，每一列为训练集中图片的拉直向量
for i=1:c
    tmp=train(:,:,i);
    vec(:,i)=tmp(:);
end

%对每个测试集，获得其knn匹配
for i=1:k
    %将测试集中第i张图片拉长
    temp=vec;
    testing=test(:,:,i);
    testing=testing(:);

    %计算测试集第i张图片与其他图片距离
    i;
    temp=temp-repmat(testing,1,c);
    dis=sum(temp.^2).^(1/2);

    %对距离进行排序，获取前k近向量的标签
    [num,val]=sort(dis);
    val=val(1:t);
    labels=train_label(val);

    %找出占比最多的标签，作为测试集标签
    table=tabulate(labels);
    max_percent=max(table(:,3));
    [row,col]=find(table==max_percent);
    maxlabel=table(row(1),1);

    output(i)=maxlabel;

end


%end
```