1 绪 论

* 1. MATLAB软件发展过程简介

MATLAB软件是由美国Mathworks公司推出的用于数值计算和图形处理的科学计算系统环境。MATLAB是英文MATrix LABoratory（矩阵实验室）的缩写。它的第一版发行于1984年，目前已发展到7.0及其以上版本，在新的版本中集中了日常数学处理中的各种功能，包括数值计算，矩阵运算，信号处理和图形生成等。MATLAB提供了一个人机交互的数学系统环境，该系统的基本数据结构是矩阵，在生成矩阵对象时，不要求说明明确的文书，与利用C语言或FORTRAN语言进行数值计算相比，利用MATLAB可以节省大量的编程时间。目前在数学领域，MATLAB已经应用于解决数值线性代数问题和一些其他高等应用数学问题；在工程技术领域，MATLAB也用于解决一些实际课题和数学建模问题。典型的MATLAB应用问题包括数值计算，算法预设计与验证，以及一些特殊的矩阵计算等。因而，MATLAB是集数值计算，图形管理和程序开发为一体的操作环境，并且与主系统一起开发了各种各样的工具箱，以完成一些特定的任务，在很大程度上简化了操作，提升了软件的实用性和便捷性。

* 1. 优势特点

1. 高效的数值计算及符号计算功能，能使用户从繁杂的数学运算分析中解脱出来
2. 具有完备的图形处理功能，实现计算结果和编程的可视化
3. 友好的用户界面及接近数学表达式的[自然化](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=455313&ss_c=ssc.citiao.link" \t "https://baike.sogou.com/_blank)语言，使学者易于学习和掌握
4. 功能丰富的应用工具箱(如信号处理工具箱、通信工具箱等)，为用户提供了大量方便实用的处理工具

1.3 交通标志识别介绍

1.3.1 选题背景

随着交通技术的日益完善和交通工具的多样化发展，交通安全成为当前最热门的问题，道路交通关系着人们的生命以及财产安全。

交通标志的识别是保证交通安全的一个重要环节。交通标识包含丰富的道路交通信息，为驾驶员提供警示、指示等辅助信息，对减轻驾驶员的驾驶压力、降低道路的交通压力、减少交通事故的发生起着重要的辅助作用。

如果完全依靠驾驶员注意和发现交通标识并做出正确的反应，难免会增加驾驶员的负担，加速疲劳，严重地可能导致交通事故。道路交通标识识别是通过用安置于车辆前方的摄像头（这里常借助车道保持辅助系统中的摄像头），检测道路标识（限速、禁止超车），通对交通标志的采集、处理、识别，将所获取的信息以最快的速度第一时间传送给驾驶员或直接对汽车采取相应的操作．从而降低了交通安全隐患并节省了人力资源。

1.3.2 识别的交通标志要求

①图片中交通标志位置不固定，可以居中、倾斜  
 ②交通标志类型以及颜色不唯一，做到多类型多颜色的交通标志识别  
 ③识别出交通标志的图形，并显示出对应的标志含义

1. 设计思想

2.1交通标志识别的主要步骤

①寻找交通标志所在位置

②二值化处理

③边缘检测提取边缘

④腐蚀、膨胀

⑤切割交通标志

⑥模板匹配

交通标志识别的流程图如图1。

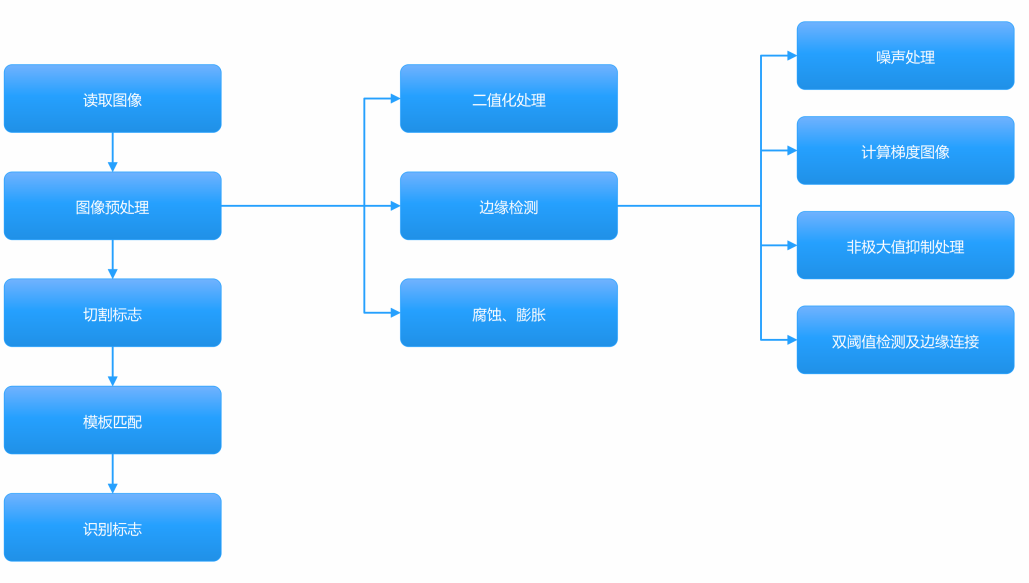


图1 交通标志识别流程图

2.2 各模块算法设计

2.2.1寻找交通标志所在位置

本文的交通标志识别算法主要是针对大陆交通标志来识别。

在汽车行驶过程中，利用安装于汽车上的数字摄像机对远处的交通标志进行动态拍摄，将拍摄到的连续图片传送给计算机，计算器通过逐行扫描法完成对预处理图像采集。在提取交通标志位置的时候，我们可以利用交通标志的颜色特征来确定其位置，即特征色彩提取法。主要的实现是用MATLAB工具箱中的颜色阈值器Color Thresholder确定标志位置。

在GB5768．2—2009《道路交通标志和标线：第 2部分道路交通标志》中，对交通标志的分类、特征颜色、轮廓、尺寸大小、内部包含的字符、数字、几何图样和数量等作了明确的规定．每种交通标志具有鲜明的颜色特征和明确的含义．每种交通标志具有明显的背景和内景颜色特征．禁令标志、指示标志、警告标志的背景特征颜色分别为红色、蓝色、黑色，禁令标志、指示标志、警告标志的内景特征颜色 分别为白色、白色、黄色．因此，标准交通标志的背景和内景特征颜色作为提取和识别交通标志的依据．

设计核心思想是对图像首先进行分块处理，提取色彩特征点，计算出每一块颜色所占比例；人为设置一个颜色阈值，使得交通标志有效区域可以保留；经过对图像的颜色筛选，可以计算出图像中颜色区域的大致位置。

2.2.2 对图像进行二值化处理

得到交通标志图像后，进行二值化处理，得到二值图像。二值图像又称黑白图像，图像的每个像素只能是黑或者是白，也就是说，其每个图像的像素值只有两个状态．即：0 和1，二值图像的表示是用0和1组成的二维矩阵来表示的。

2.2.3对图像进行边缘检测

数字图像处理中的边缘检测是图像处理过程中关键的一步，边缘是目标与背景的分割线，只有进行边缘检测提取边缘，才能将目标物与背景分开，因此是图像分割、图像识别分析领域的基石。

边缘检测的方法有很多种，如：Roberts算子、Prewitt算子、Canny算子、Sobel算子、Log算子等。Roberts算子定位比较精确，但由于不包括平滑，所以对于噪声比较敏感。Prewitt算子和Sobel算子分别是平均滤波、加权平均滤波，且检测的图像边缘可能大于2个像素，两者对灰度渐变低噪声的图像又较好的检测效果，但对于混合多复杂噪声图像处理效果并不理想。

本识别系统将采取Canny算子进行边缘检测，Canny算子是边缘检测中最具有代表的一种局部极值边缘检测，具有滤波、增强效果，无论从视觉效果还是客观评价来看，提取的边缘线性连接程度较好，对线类的边缘提取的比较完整，边缘线很细腻。

2.2.4腐蚀、膨胀

膨胀与腐蚀处理是形态学操作的一部分，膨胀就是求局部最大值的操作，即用一个结构元素 B，对原始图像A的每一个像素进行扫描，再用结构元素B与其覆盖的图像A做“或”操作， 其中结构元素B可以是方形结构也可以是圆盘结构。经过膨胀处理后的图像会“变胖”，因为 其在运行的过程中，用结构元素B中心点与图像A中的像素一个一个的对比，一旦结构元素B中有一点属于A中，则结构元素B被A所吸收， 所以原始图像会变胖膨胀一圈，其膨胀的大小由结构元素的大小所决定。

而与膨胀具有相反意义的腐蚀就是求局部最小值的操作。同样是用结构元素B对原始图像A中的每个像素进行扫描，但此时结构元素B与原始图像A做的是“与”运算，即结构元素 B中心点与原始图像A一个一个的对比，只有当结构元素B中的所有点都包含在A中时，结构元素B才会被保留，所以原始图像经过腐蚀运算后，图像会“变瘦”部分边界像素会被删除。

本文运用的识别系统是先进行腐蚀后进行膨胀的运算，也叫开运算。开运算能够消除一些孤立的小点、毛刺等细小的点，而总的位置和形状不发生改变。所以，开运算也能够达到滤波降噪，改善图像视觉质量的效果。

2.2.5切割交通标志

对图像分割我们首先找到它的轮廓图，根据交通标志5类图形轮廓进行有效部分的切割，由于交通标志的形状有5种，分别是正三角形、圆形、倒三角形、八边形和矩形。通过对轮廓图的研究分析提取出符合这五种形状的有效区域，为下面图像的有效区域的裁剪识别提供基础。

在获取了边缘图和轮廓图后，根据边缘图和轮廓图上的圆形、正三角形、倒三角形、八边形和矩形进行有效部分的切割，提取出图片上的交通标志。

2.2.6 模板匹配

模板匹配作为图像分析和处理的一项重要手段，在计算机视觉和模式识别的众多领域获得广泛的应用。模板匹配的基本思想是利用模板叠放在待搜索图上平移，模板覆盖下的那块待搜索图成为子图，记为，其中，为子图的左上角像点在中的坐标，通过比较模板与搜索子图的匹配度，进行目标的搜索。

模板匹配技术中最简单的匹配方法是采用图像像素的统计特性。根据计算匹配度进行图像的匹配判断．采用图像像素数的统计特性进行图像匹配，具有较好的准确性。在交通标志识别中，以交通标志所在区域图像作为待搜索图而不是整个交通场景图像，减少在待搜索图像中的匹配区域，节省时间开销。

1. 核心算法
   1. 特定色彩提取法

在MATLAB中，彩色图像大多数是基于RGB三色彩模型。交通标志的特征颜色有三种。我们需要提前选定交通标志的颜色，利用颜色作为特征点，提取出整个图像中该颜色最大区域。以红色为例，对于一个RGB图像，我们可以分三通道R、G、B分别提取颜色，在该算法下，我们采取分别提取三通道色彩信息，然后红色信息加倍，与蓝色、绿色区域相减，这样可以得到一副只含有红色的图像。然后在利用区域生长法，找出最大红色区域，即为交通标志所在区域。

主要的实现是使用MATLAB工具箱中的颜色阈值器Color Thresholder中的RGB颜色空间，在点云上绘制ROI选择图像颜色并且分割图像，从而确定标志位置。

* 1. 交通标志分类处理

为了将交通标志切割分类，我们可以采取以下算法：首先，对车牌图像进行二值化；其次对图像进行边缘检测，采取Canny算子进行边缘检测，分为四个步骤：噪声处理、计算梯度图像、对图像进行非极大值抑制处理和双阈值检测及边缘连接；随后对图像不断进行腐蚀、膨胀操作，使得交通标志中噪声、杂质点消除，同时使图像更加突出。

* 1. 模板匹配识别法

为判断识别交通标志，采样基础的模板匹配法。它是通过将所有像素值看成一个坐标（无论它是多少维的），通过最小化这两个坐标的欧式距离来确定最终在原始图像上匹配的位置。所谓欧式距离算法，既是对两幅图像中的每一块进行相减再平方，得出其欧式距离作为相似性的判断依据，从而利用计算两幅图像的欧式距离得出相似性，取最优解，最终识别出交通标志的含义。

1. 核心代码
2. 识别标志

Color = 0; % 按照颜色来分 颜色标记 1是红色 2 是蓝色 3是黄色

if get(handles.radiobutton3,'Value')

Color = 1;

elseif get(handles.radiobutton4,'Value')

Color = 2;

elseif get(handles.radiobutton5,'Value')

Color = 3;

end

handles.Color = Color;

Image = handles.I;

switch Color

case 1

DI = Image(:,:,1);

GI = (Image(:,:,1)>110 & Image(:,:,2)<150 & Image(:,:,3)<150 ...

&abs(double(Image(:,:,2))-double(Image(:,:,3)))<50& abs(double(Image(:,:,1))-double(Image(:,:,2)))>30);

case 2

DI = Image(:,:,3);

GI = (Image(:,:,1)<100 & Image(:,:,2)<150 & Image(:,:,3)>120 ...

& abs(double(Image(:,:,2))-double(Image(:,:,3)))>30);

case 3

DI = rgb2gray(Image);

GI = (Image(:,:,1)>160 & Image(:,:,2)>90 & Image(:,:,3)<90 ...

&abs(double(Image(:,:,1))-double(Image(:,:,2)))<100& abs(double(Image(:,:,1))-double(Image(:,:,3)))>90);

end

axes(handles.axes2);

imshow(DI);

axes(handles.axes3);

imshow(GI);

handles.GI = GI;

guidata(hObject, handles);

1. 边缘检测

I = imread('待检测图片.bmp');

%I = double(I);

%I=rgb2gray(I);

I=imnoise(I,'gaussian',0,0.0005);

figure;

imshow(I)

title('noise image')

[height,width] = size(I);

J = I;

conv = zeros(5,5);%高斯卷积核

sigma = 1;%方差

sigma\_2 = sigma \* sigma;%临时变量

sum = 0;

for i = 1:5

for j = 1:5

conv(i,j) = exp((-(i - 3) \* (i - 3) - (j - 3) \* (j - 3)) / (2 \* sigma\_2)) / (2 \* 3.14 \* sigma\_2);%高斯公式

sum = sum + conv(i,j);

end

end

conv = conv./sum;%标准化

%对图像实施高斯滤波

for i = 1:height

for j = 1:width

sum = 0;%临时变量

for k = 1:5

for m = 1:5

if (i - 3 + k) > 0 && (i - 3 + k) <= height && (j - 3 + m) > 0 && (j - 3 + m) < width

sum = sum + conv(k,m) \* I(i - 3 + k,j - 3 + m);

end

end

end

J(i,j) = sum;

end

end

figure,imshow(J,[])

title('高斯滤波后的结果')

%求梯度

dx = zeros(height,width);%x方向梯度

dy = zeros(height,width);%y方向梯度

d = zeros(height,width);

for i = 1:height - 1

for j = 1:width - 1

dx(i,j) = J(i,j + 1) - J(i,j);

dy(i,j) = J(i + 1,j) - J(i,j);

d(i,j) = sqrt(dx(i,j) \* dx(i,j) + dy(i,j) \* dy(i,j));

end

end

figure,imshow(d,[])

title('求梯度后的结果')

%局部非极大值抑制

K = d;%记录进行非极大值抑制后的梯度

%设置图像边缘为不可能的边缘点

for j = 1:width

K(1,j) = 0;

end

for j = 1:width

K(height,j) = 0;

end

for i = 2:width - 1

K(i,1) = 0;

end

for i = 2:width - 1

K(i,width) = 0;

end

for i = 2:height - 1

for j = 2:width - 1

%当前像素点的梯度值为0，则一定不是边缘点

if d(i,j) == 0

K(i,j) = 0;

else

gradX = dx(i,j);%当前点x方向导数

gradY = dy(i,j);%当前点y方向导数

gradTemp = d(i,j);%当前点梯度

%如果Y方向幅度值较大

if abs(gradY) > abs(gradX)

weight = abs(gradX) / abs(gradY);%权重

grad2 = d(i - 1,j);

grad4 = d(i + 1,j);

%如果x、y方向导数符号相同

%像素点位置关系

%g1 g2

% C

% g4 g3

if gradX \* gradY > 0

grad1 = d(i - 1,j - 1);

grad3 = d(i + 1,j + 1);

else

%如果x、y方向导数符号反

%像素点位置关系

% g2 g1

% C

%g3 g4

grad1 = d(i - 1,j + 1);

grad3 = d(i + 1,j - 1);

end

%如果X方向幅度值较大

else

weight = abs(gradY) / abs(gradX);%权重

grad2 = d(i,j - 1);

grad4 = d(i,j + 1);

%如果x、y方向导数符号相同

%像素点位置关系

% g3

% g4 C g2

% g1

if gradX \* gradY > 0

grad1 = d(i + 1,j + 1);

grad3 = d(i - 1,j - 1);

else

%如果x、y方向导数符号反

%像素点位置关系

% g1

% g4 C g2

% g3

grad1 = d(i - 1,j + 1);

grad3 = d(i + 1,j - 1);

end

end

%利用grad1-grad4对梯度进行插值

gradTemp1 = weight \* grad1 + (1 - weight) \* grad2;

gradTemp2 = weight \* grad3 + (1 - weight) \* grad4;

%当前像素的梯度是局部的最大值，可能是边缘点

if gradTemp >= gradTemp1 && gradTemp >= gradTemp2

K(i,j) = gradTemp;

else

%不可能是边缘点

K(i,j) = 0;

end

end

end

end

figure,imshow(K,[])

title('非极大值抑制后的结果')

%定义双阈值：EP\_MIN、EP\_MAX，且EP\_MAX = 2 \* EP\_MIN

EP\_MIN = 3;

EP\_MAX = EP\_MIN \* 2;

EdgeLarge = zeros(height,width);%记录真边缘

EdgeBetween = zeros(height,width);%记录可能的边缘点

for i = 1:height

for j = 1:width

if K(i,j) >= EP\_MAX%小于小阈值，不可能为边缘点

EdgeLarge(i,j) = K(i,j);

else if K(i,j) >= EP\_MIN

EdgeBetween(i,j) = K(i,j);

end

end

end

end

%把EdgeLarge的边缘连成连续的轮廓

MAXSIZE = 999999;

Queue = zeros(MAXSIZE,2);%用数组模拟队列

front = 1;%队头

rear = 1;%队尾

edge = zeros(height,width);

for i = 1:height

for j = 1:width

if EdgeLarge(i,j) > 0

%强点入队

Queue(rear,1) = i;

Queue(rear,2) = j;

rear = rear + 1;

edge(i,j) = 255;%EdgeLarge(i,j);

EdgeLarge(i,j) = 0;%避免重复计算

end

while front ~= rear%队不空

%队头出队

temp\_i = Queue(front,1);

temp\_j = Queue(front,2);

front = front + 1;

%8-连通域寻找可能的边缘点

%左上方

if EdgeBetween(temp\_i - 1,temp\_j - 1) > 0%把在强点周围的弱点变为强点

edge(temp\_i - 1,temp\_j - 1) = K(temp\_i - 1,temp\_j - 1);

EdgeBetween(temp\_i - 1,temp\_j - 1) = 0;%避免重复计算

%入队

Queue(rear,1) = temp\_i - 1;

Queue(rear,2) = temp\_j - 1;

rear = rear + 1;

end

%正上方

if EdgeBetween(temp\_i - 1,temp\_j) > 0%把在强点周围的弱点变为强点

edge(temp\_i - 1,temp\_j) = K(temp\_i - 1,temp\_j);

EdgeBetween(temp\_i - 1,temp\_j) = 0;

%入队

Queue(rear,1) = temp\_i - 1;

Queue(rear,2) = temp\_j;

rear = rear + 1;

end

%右上方

if EdgeBetween(temp\_i - 1,temp\_j + 1) > 0%把在强点周围的弱点变为强点

edge(temp\_i - 1,temp\_j + 1) = K(temp\_i - 1,temp\_j + 1);

EdgeBetween(temp\_i - 1,temp\_j + 1) = 0;

%入队

Queue(rear,1) = temp\_i - 1;

Queue(rear,2) = temp\_j + 1;

rear = rear + 1;

end

%正左方

if EdgeBetween(temp\_i,temp\_j - 1) > 0%把在强点周围的弱点变为强点

edge(temp\_i,temp\_j - 1) = K(temp\_i,temp\_j - 1);

EdgeBetween(temp\_i,temp\_j - 1) = 0;

%入队

Queue(rear,1) = temp\_i;

Queue(rear,2) = temp\_j - 1;

rear = rear + 1;

end

%正右方

if EdgeBetween(temp\_i,temp\_j + 1) > 0%把在强点周围的弱点变为强点

edge(temp\_i,temp\_j + 1) = K(temp\_i,temp\_j + 1);

EdgeBetween(temp\_i,temp\_j + 1) = 0;

%入队

Queue(rear,1) = temp\_i;

Queue(rear,2) = temp\_j + 1;

rear = rear + 1;

end

%左下方

if EdgeBetween(temp\_i + 1,temp\_j - 1) > 0%把在强点周围的弱点变为强点

edge(temp\_i + 1,temp\_j - 1) = K(temp\_i + 1,temp\_j - 1);

EdgeBetween(temp\_i + 1,temp\_j - 1) = 0;

%入队

Queue(rear,1) = temp\_i + 1;

Queue(rear,2) = temp\_j - 1;

rear = rear + 1;

end

%正下方

if EdgeBetween(temp\_i + 1,temp\_j) > 0%把在强点周围的弱点变为强点

edge(temp\_i + 1,temp\_j) = K(temp\_i + 1,temp\_j);

EdgeBetween(temp\_i + 1,temp\_j) = 0;

%入队

Queue(rear,1) = temp\_i + 1;

Queue(rear,2) = temp\_j;

rear = rear + 1;

end

%右下方

if EdgeBetween(temp\_i + 1,temp\_j + 1) > 0%把在强点周围的弱点变为强点

edge(temp\_i + 1,temp\_j + 1) = K(temp\_i + 1,temp\_j + 1);

EdgeBetween(temp\_i + 1,temp\_j + 1) = 0;

%入队

Queue(rear,1) = temp\_i + 1;

Queue(rear,2) = temp\_j + 1;

rear = rear + 1;

end

end

%下面2行用于观察程序运行的状况

i

j

end

end

figure,imshow(edge,[])

title('双阈值后的结果')

1. 分割标志（包括腐蚀、膨胀处理）

d = handles.GI;

se = ones(3); % 腐蚀膨胀模版

d = imdilate(d,se);% 再做膨胀运算

d = imerode(d,se); % 先做腐蚀运算

% 先膨胀后腐蚀的过程称为开运算。用来填充物体内细小空洞、连接邻近物体、平滑其边界的同时并不明显改变其面积。

d = bwareaopen(d,100); % 移除小对象 小区域肯定是噪声

L = bwlabel(d,8); %标记连通

Num = max(max(L)); % 有几个区域

S = zeros(1,Num);

for i = 1:Num

S(i) = sum(sum(L == i));

end

[S,id] = sort(S,'descend');

Ran = zeros(Num,4); % 范围

Fig = zeros(1,Num); % 长宽比

for i = 1:Num

[ix,iy] = find(L == id(i));

Ran(i,:) = [min(ix),max(ix),min(iy),max(iy)]; % 每一个区域的范围

end

for i = 1:Num

Fig(i)=max(Ran(i,2)-Ran(i,1),Ran(i,4)-Ran(i,3))/min(Ran(i,2)-Ran(i,1),Ran(i,4)-Ran(i,3));

End

in = 0;

for i = 1:Num

if Fig(i) < 1.5 % 长宽比是接近1的 所以不会太离谱

in = i;

break;

end

end

if in == 0

in = 1; % 如果都不满足就认定面积最大的一个了 这时候肯定是有干扰的

end

axes(handles.axes3);

imshow(L == id(in));

if handles.Color ~= 3

xmin = Ran(in,1);

xmax = Ran(in,2);

ymin = Ran(in,3);

ymax = Ran(in,4);

II = handles.I(xmin:xmax,ymin:ymax,:);

elseif handles.Color == 3 % 黄色的多切一点 有黑色边框

xmin = Ran(in,1)-round((Ran(in,2)-Ran(in,1))\*0.15);

xmax = Ran(in,2)+round((Ran(in,2)-Ran(in,1))\*0.15);

ymin = Ran(in,3)-round((Ran(in,4)-Ran(in,3))\*0.15);

ymax = Ran(in,4)+round((Ran(in,4)-Ran(in,3))\*0.15);

II = handles.I(xmin:xmax,ymin:ymax,:);

end

axes(handles.axes4);

imshow(II);

% imwrite(II,'5.png','png')

handles.Divice = II;

guidata(hObject, handles);

1. 识别标志（包括模板匹配）

I = handles.Divice;

load Name Name;

load Trained\_BP Trained\_BP;

load Data Data;

if size(I,3) ~= 1

I = rgb2gray(I);

end

I = imadjust(I); % 调整对比度

I = imresize(I,[40,40]); % 都变成40\*40的图像

I = im2bw(I); % 二值化

% axes(handles.axes4);

% imshow(I);

I = I(:);

distance = zeros(1,size(Data,2));

for i = 1:size(Data,2)

distance(i) = norm(I-Data(:,i),2);

end

%模板匹配:距离最小的就可以被认为是模版

id = find(distance == min(distance));

str = Name{id(1)};

set(handles.text2,'string',str);

1. 测试数据或截图
   1. 交通标志识别实验用图

为了更好地呈现程序对多交通标志的处理，测试时选取了一幅有三个标志两种不同颜色的图像，且需要识别的是黄色警告类标志。



图2 交通标志实验用图

* 1. 边缘检测

边缘检测的目的是找到图像中亮度变化剧烈的像素点构成的集合，表现出来往往是轮廓。如果图像中边缘能够精确的测量和定位，那么就意味着实际的标志能够被定位和测量，包括标志的面积、直径、形状等就能被测量。边缘检测分为四个步骤，分别如下。

5.2.1噪声处理

噪声，简而言之就是图像中随机出现的灰度变化大的离散的像素点。噪声可能来自于图像采集，量化等过程，也可能产生于图像传送过程中，具有离散性和随机性等特点。而高斯滤波实质上是一种信号的滤波器，其用途是信号的平滑处理。

首先进行噪声处理。用imnoise函数添加高斯噪声并用高斯滤波去除噪声。结果如下图3。



图3 噪声处理的结果

5.2.2 对图像求梯度

当用高斯滤波降低图像噪声的时候，会带来图像模糊的副作用。从逻辑上考虑，图像模糊是因为图像中物体的轮廓不明显，轮廓边缘灰度变化不强烈，层次感不强造成的。

我们知道微分就是求函数的变化率，即导数。由于梯度的定义只适用于连续的函数，而图像是二维的离散函数，因此对于图像来说我们需要使用有限差分来近似计算梯度。

梯度提供了两种信息——大小和方向。梯度的方向告诉我们函数在这个点沿着这个方向上升最快(负梯度就代表函数在这个点沿着负梯度方向下降最快)。把函数比作一座“山”，我们站在半山腰上，如果我们要上山则沿着梯度方向可以最快到达山顶，如果是下山则沿着梯度的反方向（负梯度方向）则可以最快到达山脚。梯度的大小代表了沿着这个方向的变化率，即导数。所以我们用微分来表示图像灰度的变化率。

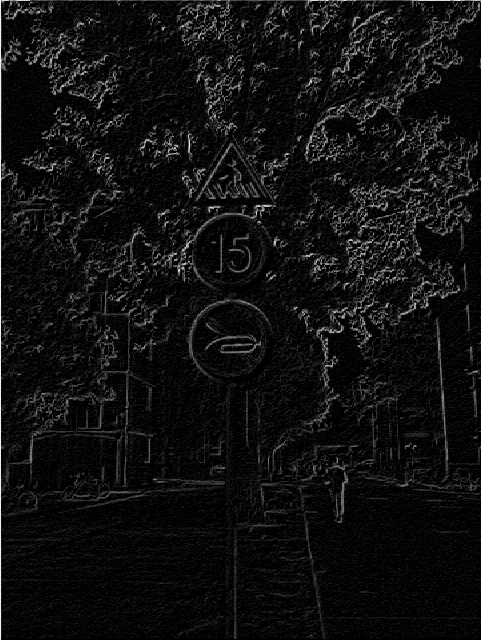


图4 求梯度后的结果

5.2.3 非极大值抑制

非极大值抑制是抑制不是极大值的元素,可以理解为局部最大搜索，这个局部代表的是一个邻域,邻域有两个参数可变,一是邻域的维数,二是邻域的大小。它可以帮助抑制除局部最大值之外的所有梯度值,使其指示具有最强烈的强度值变化的位置。

在交通标志检测过程中在同一目标的位置上会产生大量的候选框,这些候选框相互之间可能会有重叠,此时我们需要利用非极大值抑制找到最佳的目标边界框,消除冗余的边界框。



图5 非极大值抑制后的结果

5.2.4 双阈值检测和边缘连接

它利用图像中要提取的目标与背景在灰度上的差异，通过设置阈值来把像素级分成若干类，从而实现目标与背景的分离。

先设置高、低两个阈值（一般高阈值是低阈值的2至3倍），遍历整个灰度矩阵，若某点的梯度高于高阈值，则在结果中置1，若该点的梯度值低于低阈值，则在结果中置0，若该点的梯度值介于高低阈值之间，则需要进行如下判断：检查该点（将其视为中心点）的8邻域点，看是否存在梯度值高于高阈值的点，若存在，则说明该中心点和确定的边缘点相连接，故在结果中置1，否则置0。结果如下图6 。



图6 边缘检测结果

* 1. 腐蚀膨胀

图像的腐蚀操作类似于中值平滑，首先要取每个位置的一个邻域内的最小值(中值平滑是取中间值)，将其作为该位置的输出像素值。这里的邻域不局限于矩形结构，还包括椭圆形结构和十字交叉形结构。它的具体定义为结构元，作用类似于平滑操作中的卷积核。

膨胀和腐蚀操作原理相似，膨胀是选取每个位置邻域内的最大值作为输出灰度值。膨胀后的图像的整体亮度会有提高，图形中较亮物体的尺寸变大，而较暗物体的尺寸会减小甚至消失。结果如图7。

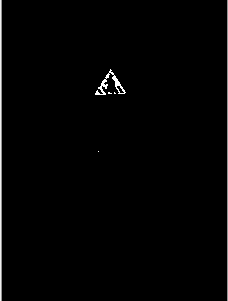


图7 腐蚀膨胀结果

* 1. 切割标志

根据MATLAB工具箱中的颜色阈值器Color Thresholder的RGB颜色空间，在点云上绘制ROI选择黄色并确定标志位置。

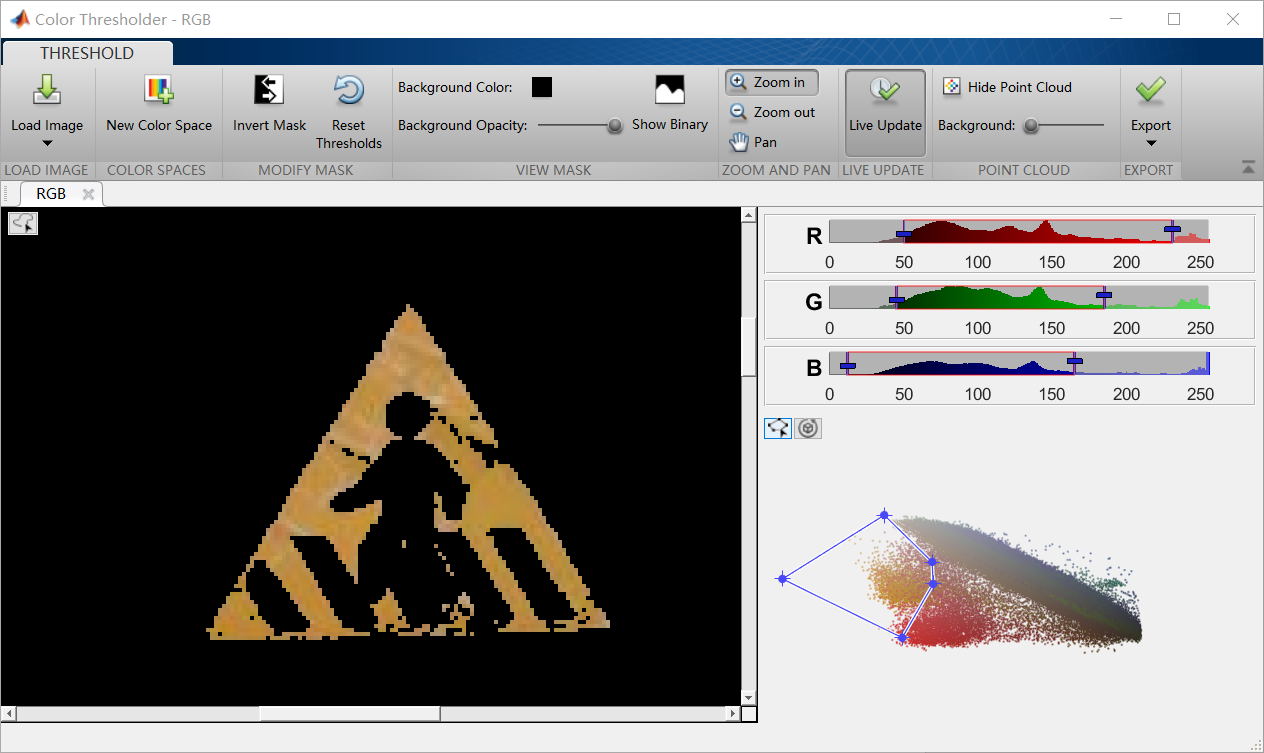


图8 Color Thresholder确定的结果

根据图像处理后的结果及颜色阈值器Color Thresholder确定的位置对原图像进行切割，得到交通标志，如图9。



图9 切割标志结果

* 1. 识别结果

模板匹配是数字图像处理的重要组成部分之一。把不同传感器或同一传感器在不同时间、不同成像条件下对同一景物获取的两幅或多幅图像在空间上对准,或根据已知模式到另一幅图中寻找相应模式的处理方法就叫做模板匹配。简单而言，模板就是一幅已知的小图像。模板匹配就是在一幅大图像中搜寻目标，已知该图中有要找的目标，且该目标同模板有相同的尺寸、方向和图像，通过一定的算法可以在图中找到目标，确定其坐标位置。

采用模板匹配法对切割后的标志进行识别，系统成功识别出结果——注意行人。

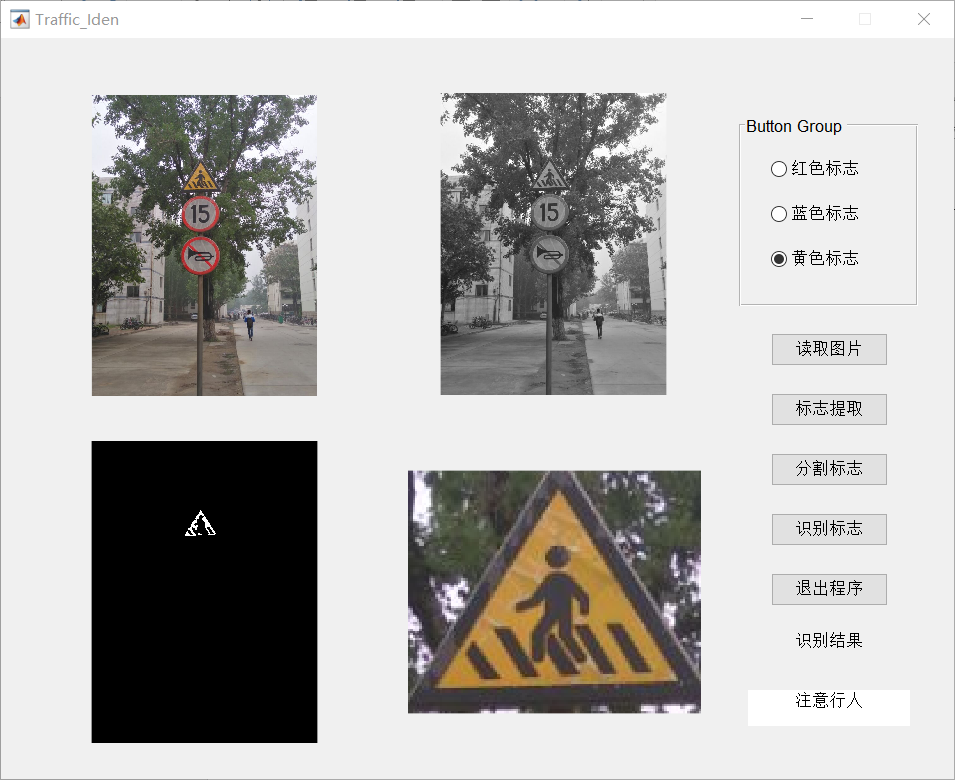


图10 识别界面及结果图

结论

正如绪论中所说的一样，MATLAB软件在数值计算，矩阵运算，信号处理和图形生成等方面有着强大的实现功能和广泛的应用。本次MATLAB软件实习，我主要选取了MATLAB在软件应用方面进行探究，分别从图像预处理、交通标志定位、腐蚀膨胀以及标志识别几个方面对系统进行分析。

主要工作内容如下：

1. 收集，整理和总结了国内外在交通标志定位、分割、识别方面的研究成果和发展方向，系统介绍了我国交通标志的固有特征，以及标志识别等特点。
2. 在Color Thresholder中采用点云选取ROI对标志定位后，我们对先图像进行噪声处理、求梯度、非极大值抑制、双阈值检测和边缘连接，再进行腐蚀、膨胀处理，得到标志区域的结果图。实验表明本方法不仅保留了标志区域的信息，又减少了噪声的干扰，从而简化了处理过程，提高了后面分割的准确性。
3. 采用彩色分量的定位方法，通过对黄色像素点的统计是对车牌是黄色的警告类标志进行定位，实验表明，通过该方法得到的标志定位准确率较为理想。
4. 该设计采用的图像预处理、边缘检测、交通标志颜色特征识别等对标志的定位都比较成功。而且对我国的三种颜色红色禁止类、黄色警告类以及蓝色指示类都适用，且结果较好。因此，此课设提出的交通标志的分割算法实验证明在实际运用中是准确、有效、可行的。

在整个实习过程中，我学习了Color Thresholder工具箱用法、特征色彩提取法、Canny算子边缘检测、腐蚀膨胀处理及模式匹配识别法，进一步地提高了自己发现问题，分析问题和解决问题的能力；与此同时还看到MATLAB软件在作图及GUI呈现方面的功能强大，使用起来也较为方便。因此，在今后的学习中我将进一步地学习图像处理的理论知识和MATLAB在此方面的应用。

参考文献

1. 张建波. 大学数学实验：MATLAB版[M]. 北京：人民邮电出版社，2020.8.
2. 崔盼,张荣辉. 基于MATLAB图像处理的道路交通标志处理技术的研究[J]. 黑龙江交通科技,2017,40(10):182-183.
3. 房泽平,段建民,郑榜贵. 基于特征颜色和SNCC的交通标志识别与跟踪[J]. 交通运输系统工程与信息,2014,14(1):47-52.
4. 关鑫. 自然环境下交通标志牌的检测与识别[J]. 电脑知识与技术,2016,12(15):190-192.
5. 娄月新. 基于Matlab的交通标志识别系统设计与实现[J]. 电脑编程技巧与维护,2014(6):83-84.
6. 孙巍,孙国荣,张瑞龙. 基于MATLAB的道路交通标志识别[J]. 教育教学论坛,2016(12):55-57.
7. 王国栋,姚力. 基于MATLAB的图像处理技术在足迹图像中的应用[J]. 云南警官学院学报,2019(5):107-113.