# 《计算机视觉（1）》实验报告

## 实验四 局部处理的边缘连接

| 实验小组成员<br>（学号+班级+姓名） | 分工及主要完成任务 | 成绩 |
| --- | --- | --- |
| 201800830004<br>18 数科 叶家辉 | 全部 | |
| | | |
| | | |

山东大学

2021 年 3 月

完成《数字图像处理》P468页例10.10的编程实验，编程语言可以选择Matlab，C，C++，OpenCV，Python等。设计方案可参照教科书中的分析，也可以自行设计新的方案。

图(a)显示了一辆汽车尾部的图像。该例的目的是说明用前述算法来寻找大小适合车牌的矩形的应用。该矩形可以通过检测强的水平和垂直边缘构成。图(b)显示了梯度幅度图像M(x, y)，图(c)和(d)显示了该算法步骤3和步骤4的结果，其中，令TM等于最大梯度值的30%，A = 90°，TA = 45°，并填充了全部25个或更少像素的缝隙(约为图像宽度的5%)。

为检测车牌壳的全部拐角和汽车的后窗，要求使用一个较大范围的容许角度方向。图(e)是前两幅图像逻辑“或”(OR)操作的结果，图(f)是使用9.5.5节讨论的细化过程细化图(e)得到的。如图(f)所示，在图像中清楚地检测到了对应于车牌的矩形。如果汽车牌照的宽高比有与众不同的2∶1 的比例，所以利用这一事实从图像的所有矩形中简单地分离出牌照是一件简单的事情。

图(a)一幅大小为534×566的汽车后部图像；(b)梯度幅度图像；(c)水平连接的边缘像素；(d)垂直连接的边缘像素；(e)前两幅图像的逻辑“或”(OR)；(f)用形态学细化得到的最终结果

原始图像的电子版图像在 Images 文件夹中。实验报告写在如下空白处，页数不限。

# 一、实验目的与原理

## 1.1 梯度阵列

在一幅图像的$(x,y)$处寻找边缘的强度和方向，就需要借助梯度。梯度由下式来进行定义：

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

梯度向量指出了图像在$(x,y)$处灰度的最大变化率的幅度和方向。具体地，幅度$M(x,y)$和方向$\alpha(x,y)$通过以下式子得到：

$$M(x,y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \qquad (1)$$

$$\alpha(x,y) = arctan\left[\frac{g_y}{g_x}\right] \qquad (2)$$

为了计算在每个像素点处的偏导数$\frac{\partial f}{\partial x}$和$\frac{\partial f}{\partial y}$，我们选择用于计算梯度偏导数的滤波器模版，通常称为梯度算子。本实验中选用的是 Sobel 算子，其 x 和 y 方向的算子分别为以下形式：

| | | | | | |
|---|---|---|---|---|---|
| −1 | −2 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −2 | 0 | 2 |
| 1 | 2 | 1 | −1 | 0 | 1 |

其近似 x 和 y 方向的一阶偏微分分别为：

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

我们使用检测水平方向的 Sobel 算子和检测垂直方向的 Sobel 算子就可以分别得到 x 和 y 方向上的梯度分量$|g_x|$和$|g_y|$。计算梯度可用上面式(1)，或用$M(x,y) = |g_x| + |g_y|$近似代替。计算梯度的方向角度，需要用上面式(2)。

## 1.2 阈值处理

为了消除弱小边缘，我们可以对梯度图像进行阈值处理。经过阈值处理后的图像中边缘更少，并且图像中的边缘会锐利很多。我们对上述生成的幅度阵列和角度阵列引入一定的约束，形成一幅二值图像：

$$g(x,y)=\begin{cases}1, & M(x,y)>T_M \ and \ \alpha(x,y)=A\pm T_A \\ 0, & others\end{cases} \quad (3)$$

其中$T_M$是一个阈值，$A$是一个指定的角度方向，$A\pm T_A$定义了一个关于$A$的方向的“带宽”。

### 1.3 边缘连接

在一般的局部处理的边缘连接问题中，判断边缘点的所有邻点是否属于边缘的相似点需要按照一定的准则（包括梯度向量幅度准则和梯度向量方向准则）将相似点连接起来，以形成根据指定准则满足相同特性像素的一条边缘。当某个像素点同时满足幅度准则和方向准则，

$$|M(s,t)-M(x,y)\le E| \quad and \quad |\alpha(s,t)-\alpha(x,y)\le A|$$

它就被连接到图像的边缘中，将已连接的点记录下来，然后在下一个像素点处重复上述步骤。可以看出，这种计算方式复杂性较高，因为必须对每个点的所有邻点做检验。

在实际应用中，我们往往采用一种简化的但仍然行之有效的方法。即我们对图像$g(x,y)$的每一行，对长度不超过指定值 K 的某个空隙进行填充，也就是说，如果一行 0 的集合的两端是 1 或多个 1，就对这些 0 进行置 1。然后对各行分别地处理，它们之间无记忆性。

如果要在其他方向 $\theta$ 上检测缝隙，则先将$g(x,y)$旋转 $\theta$，然后进行上述扫描过程，最后将其旋转- $\theta$ 即可。在本实验中， $\theta$ 取 90 度。

## 二、实验内容与步骤

### 2.1 界面总体设计与图像导入

本实验的 GUI 界面延续之前一贯的简洁明快的设计风格。同时为了集成计算机视觉（1）的所有实验项目，在 GUI 的顶部特别设计了导航栏，可以在同一窗口定位到不同的实验项目。点击“4-局部处理的边缘连接”后即可得到本实验的整体操作界面。

和前几个实验一样，左上角有“打开文件”和“保存图像”两个图标。点击“打开文件”图标选择相应图片，即可在坐标图区 axes1（第一行第一列图像显示区）导入这张图片。“打开文件”按钮的回调函数如下图所示。前四行获取了所选定图片的文件名和路径名，再根据得到的路径名和文件名读取文件。后两行则把读取到的文件显示到坐标图区，并设定顶部的文本框显示图片路径名/文件名。

```
% --------------------------------------------------------------------
function uipushtool1_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename,pathname]=uigetfile({'*.bmp;*.jpg;*.png;*.jpeg;*.tif'},'请选择图片');
str=[pathname filename];
OriginalPic = imread(str);
axes(handles.axes1);

imshow(OriginalPic);
set(handles.edit1,'String',str);
```

实验过程中，点击“显示原图”即可在第一行第一列的图片区显示文本框内文件名对应的图片。这按钮的设置主要是为了一种形式上的和谐，仔细想来其实并没有实际效用。该按钮的回调函数如下。

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = get(handles.edit1,'string');
img = imread(file);
axes(handles.axes1);
imshow(img);
```

### 2.2 计算与显示梯度幅度

要计算一幅图像的梯度大小，首先需要计算在每一个像素点上分别沿 x 轴和 y 轴方向上的一阶导数，然后利用式(1)即可得到梯度的幅度。梯度的方向则与边缘的方向相垂直。一幅图像的边缘处的梯度幅度比其它处更大，所以梯度幅度的图像反映了边缘的情况。下图展示的是“梯度幅度”按钮的回调函数，首先得到图像每一点处的灰度值，然后进行归一化，再通过 imgradientxy( )函数计算图像$f$在$(x,y)$坐标处沿 x 轴和 y 轴方向的导数（这里选择 Sobel 算子）。然后根据式子(1)计算结果并展示在坐标图区 axes2（第二行第一列图像显示区）。

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = get(handles.edit1,'string');
f = double(imread(file))/255;
[gx gy] = imgradientxy(f, 'sobel');
M = (gx.^2+gy.^2).^0.5;
axes(handles.axes2);
imshow(M);
```


## 2.3 水平边缘连接

首先对图像进行阈值处理。根据式(3)生成一幅二值图像$g(x,y)$，在本实验中给出了参考值$T_M = 30\% \times max(M)$，$T_A = 45°$。当然，实验中用户也可以对参数进行自定义，以探索不同的效果。

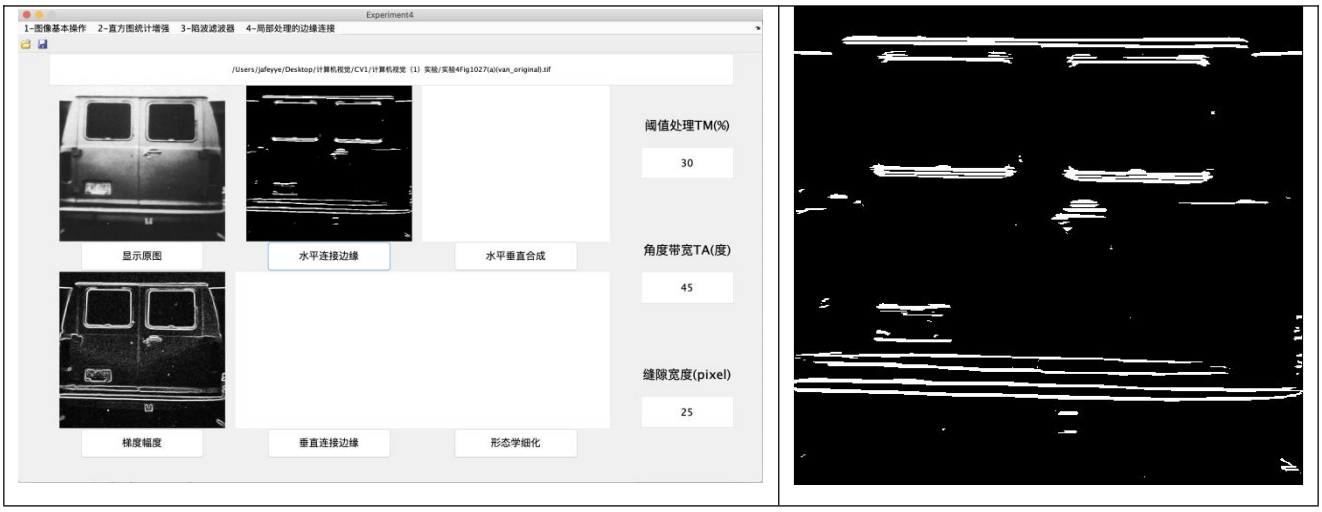
```
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
TM = str2num(get(handles.edit2,'string'))/100;
A = 90;
TA = str2num(get(handles.edit3,'string'));
file = get(handles.edit1,'string');
f = double(imread(file))/255;
[gx gy] = imgradientxy(f,'sobel');
M = sqrt(gx.^2+gy.^2);
max_M = max(max(M));
T_M = max_M*TM;
alpha = atan2(gy,gx)*180/pi;
g = zeros(size(f));
[H,W] = size(f);
for i = 1:H
    for j = 1:W
        if M(i,j)>T_M
            if (alpha(i,j)<=A+TA && alpha(i,j)>=A-TA) || (alpha(i,j)<=A-180+TA && alpha(i,j)>=A-180-TA)
                g(i,j) = 1;
            end
        end
    end
end
```

阈值处理的回调函数如上图所示。首先从 GUI 界面的文本框中分别读入参数$TM$和$T_A$的值，由于梯度方向与边缘方向是垂直的，所以取$A = 90°$。再按照 2.2 中的步骤计算梯度幅度矩阵$M(x,y)$和梯度方向矩阵$\alpha(x,y)$，并得到幅度矩阵中的最大值，以便计算参数$T_M = TM \times max(M)$（$TM$表示阈值相对于幅度最大值的百分数）。然后创建与原图象同维的零矩阵，根据式(3)来对每一个元素进行赋值。有两点值得注意。第一是分成两句 if 语句有利于提高运行速度，即代码先进行幅度的判定，如满足再进行接下来角度的判定，如不满足则直接进入下一个像素。第二是第二步 if 判定，要注意有$A - T_A \le \alpha(i,j) \le A + T_A$和$(A - \pi) - T_A \le \alpha(i,j) \le (A - \pi) + T_A$两种情况，因为 atan2( )得到的范围除了 $0 \sim \pi$，还有$-\pi \sim 0$。

接下来对二值图像进行水平方向的边缘连接，从 GUI 文本框中读入参数$L$的值，先对二值图像进行填充，使其每行的最后添上$L-1$个 0，再创建与原图象同维的零矩阵。进行判定，如果$g_pad$中某个 1 元素后 L 个以内的元素中有 1，就将其置为 1，然后修改$g_g$中的元素也为 1。最后得到的$g_g$即为填充后的二值图像。

```
L = str2num(get(handles.edit4,'string'));
%水平二值图像沿x方向进行扩展
g_pad=padarray(g,[0,L-1],'post');
g_g=zeros(H,W);
for i=1:H
    for j=1:W
        if g_pad(i,j)==1 && g_pad(i,j+1)==0
            Block=g_pad(i,j+2:j+L-1);
            ind=find(Block==1);
            if ~isempty(ind)
                ind_Last=j+2+ind(1,length(ind))-1;
                g_pad(i,j:ind_Last)=1;
                g_g(i,j:ind_Last)=1;
            end
        else
            g_g(i,j)=g_pad(i,j);
        end
    end
end

axes(handles.axes3);
imshow(g_g);
```



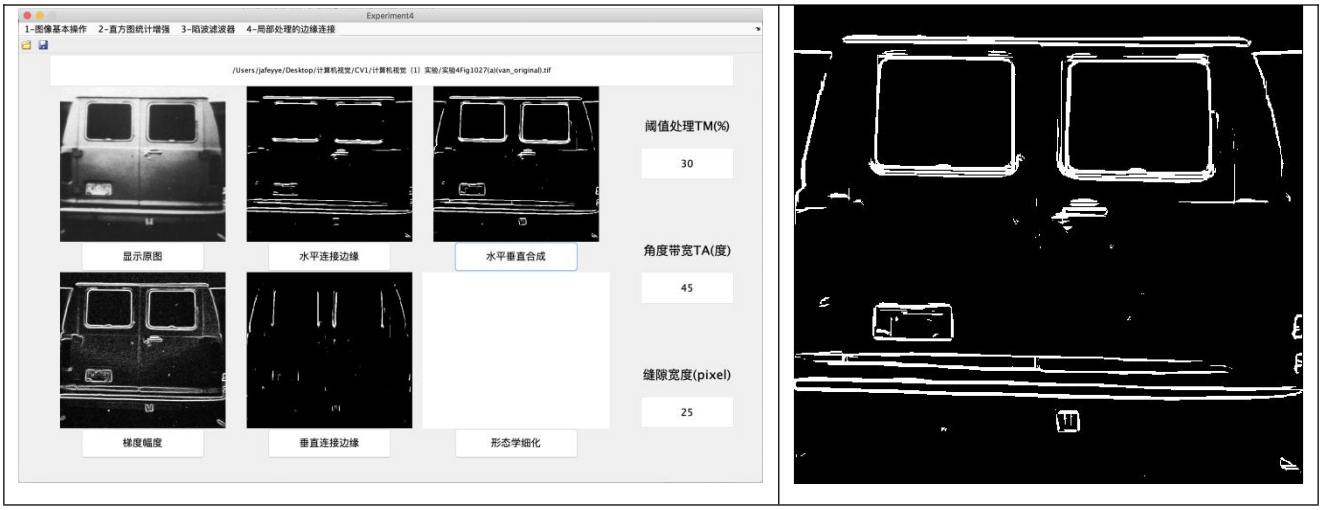
## 2.4 垂直边缘连接

垂直边缘连接和水平边缘连接基本类似。由于边缘连接的步骤是按行进行操作的，所以在生成二值图像前需要先将图像旋转 90 度。（当然，也可以在生成二值图像后，进行边缘连接前旋转，只是这样的话就需要修改生成二值图像时的方向阈值。为保证参数设置上的便利，本文选取前种方式。）然后在完成边缘连接操作后旋转-90 度即可。具体的代码就不再赘述。


## 2.5 水平垂直合成

水平与垂直合成的操作相当于对水平连接边缘和垂直连接边缘的两张图像进行“or”操作。也就是说，合成图像中的像素灰度为 0 当且仅当该像素在水平连接边缘和垂直连接边缘两张图像中均为 0。体现在代码上就是如下的样式。读取中间一列两张图像分别存储为矩阵 X 和 Y，然后对两个矩阵进行“或”操作（|）。

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
X = getimage(handles.axes3);
Y = getimage(handles.axes4);
img = X | Y;
axes(handles.axes5);
imshow(img);
```



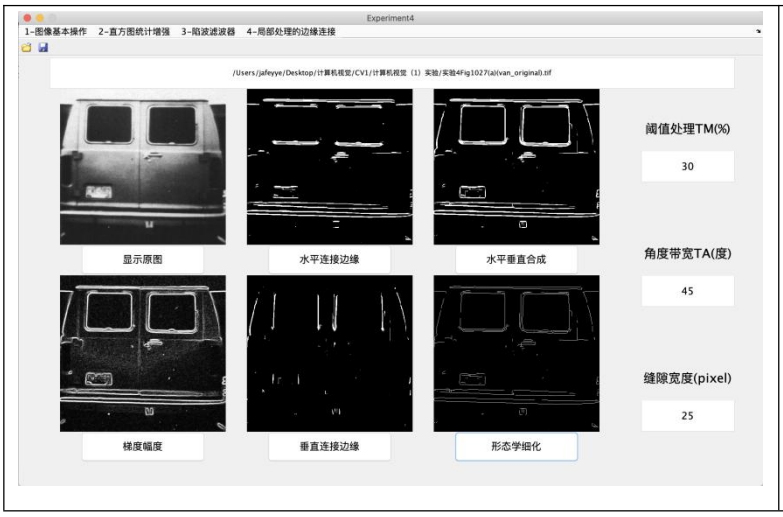
## 2.6 形态学细化

形态学细化可以根据击中击不中变换来定义：

$$A \otimes B = A - (A \circledast B) = A \cap (A \circledast B)^C$$

在 Matlab 中有专门的函数进行细化操作，如下图所示。

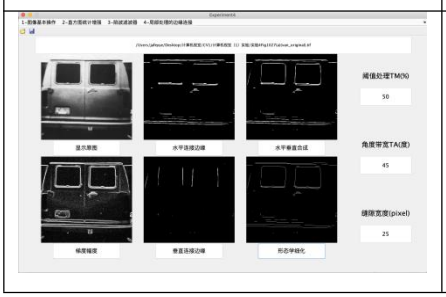
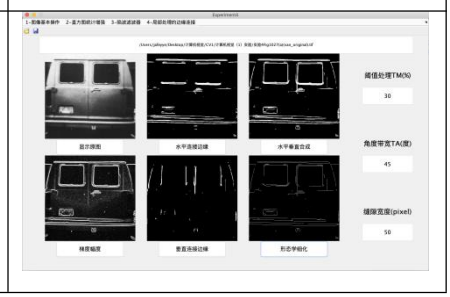
```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
img = getimage(handles.axes5);
thin_f = bwmorph(img,'thin',Inf);
axes(handles.axes6);
imshow(thin_f);
```




# 三、实验结果与总结

各步得到的结果（包括 GUI 显示和图像处理结果）已经展示在 2.2～2.6 节每一节的最后。对于所有结果，参数使用的是参考值。这些参考值是经实践检验具有明显优势的值，不过通过设定不同的参数可以直观地认识不同参数所起到的作用。

比如，通过增大$TM$会使得图像中更少的边缘被保留，通过增大$T_A$会使得图像中更倾斜方向上的边缘得到保留，通过增大缝隙宽度可以导致更宽的缝隙也被允许填充。

| 增大$TM$ | 增大$T_A$ | 增大缝隙宽度 |
| --- | --- | --- |
|  |  |  |