作业二

15331191 廖颖泓

1. 给定图像car.png和模版图像wheel.png，利用相关检测实现对car图像中的wheel检测，具有最大相关值的位置可以解释为所检测到的wheel位置。程序的输入是图像和模版，要求：

（i）显示图像的相关值结果；

（ii）列出在图像中检测到的所有目标的（x，y）坐标。

a. 算法描述：

(1) 用 Matlab 的 imread()函数分别扫描给定图像 car.png 和模版图像 wheel.png，获取 car 图像矩阵 scene 和模板矩阵 template；

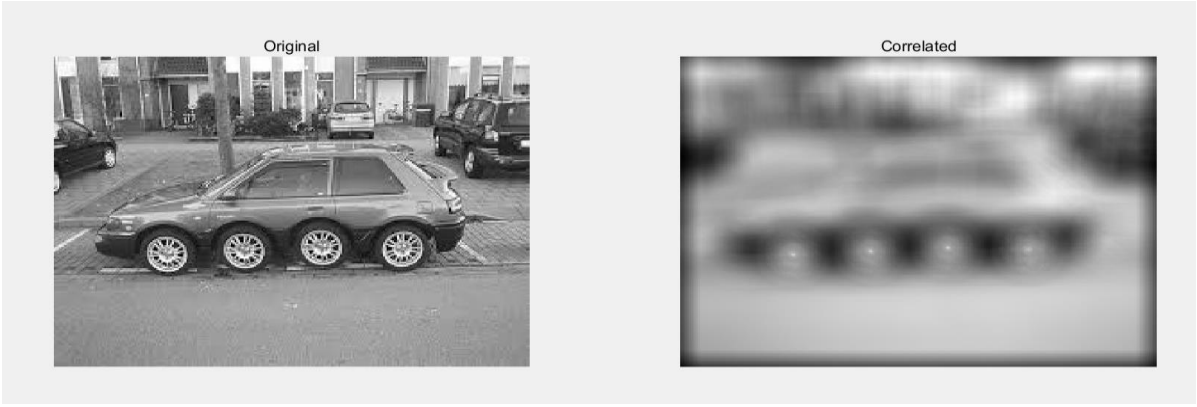
(2) 用 Matlab 的 size()函数获取 car 图像矩阵 scene 的行数 m 和列数 n 和模板矩阵 template 的行数 x 和列数 y；

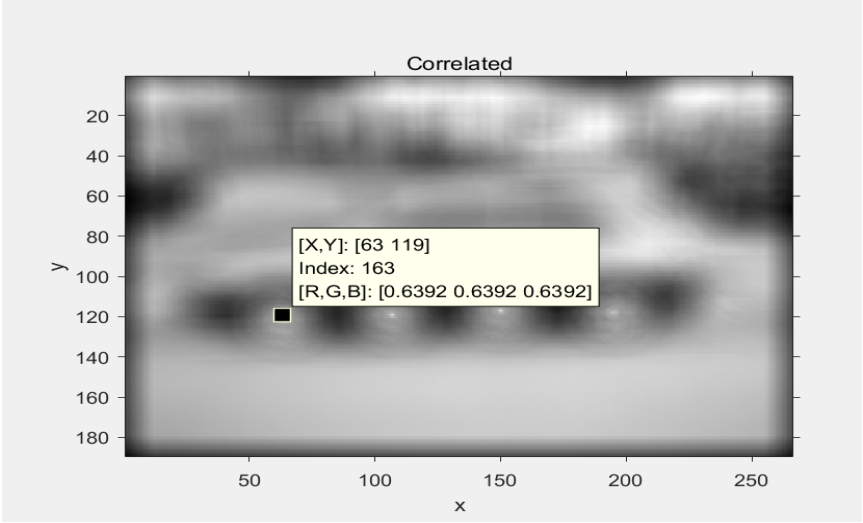
(3) 运用循环对图像的每一个像素用模板矩阵 template 套用以下公式进行相关运算,得到相关值矩阵 correlated_img，矩阵中较大值说明匹配程度越高，而较大值集中的区域即为车轮的位置;

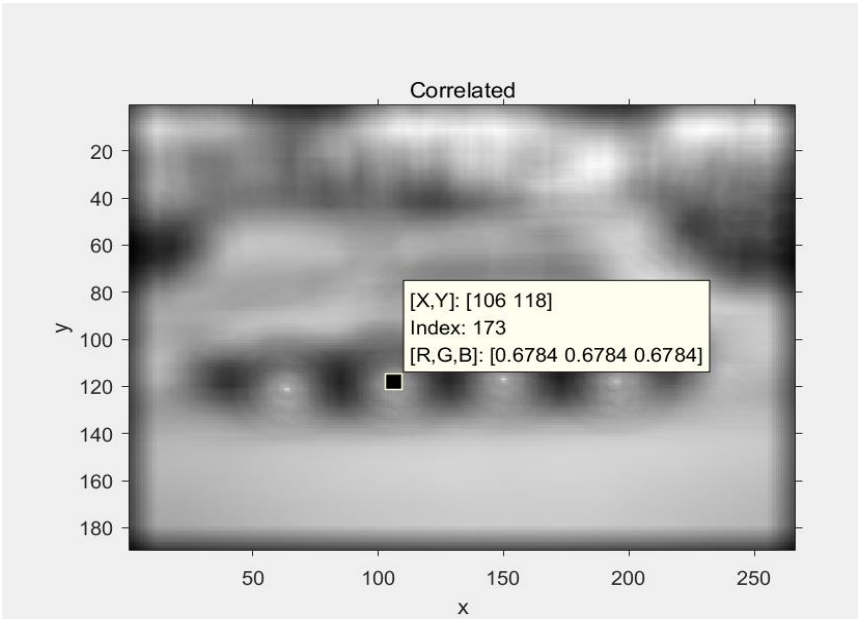
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

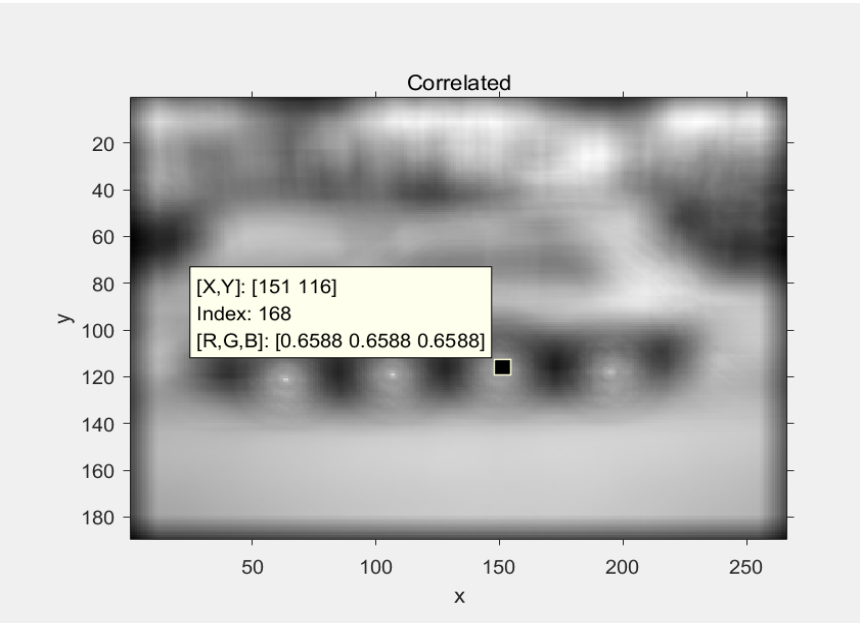
(4) 显示该矩阵代表的图像，我们可以观察到匹配到的四个车轮出有四个白点，即所检测到得 wheel 的位置，用三维坐标系表示，我们可以观察到有四个峰值，利用 Matlab 图像的工具，我们可以得到四个坐标分别为(63,119),(106, 118),(151,116),(195,117)(结果可能有一定误差)。

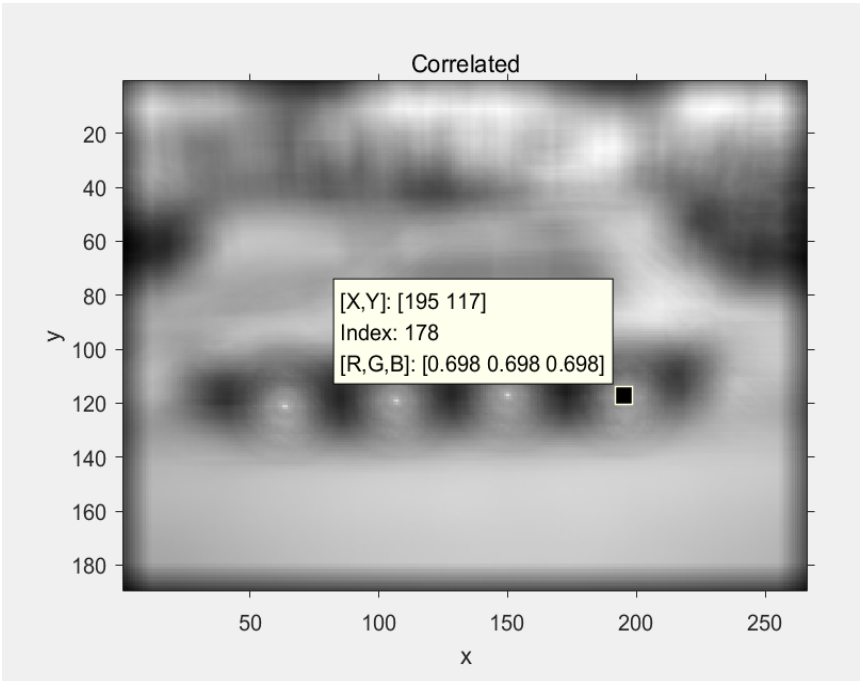
b. 程序效果示意图：

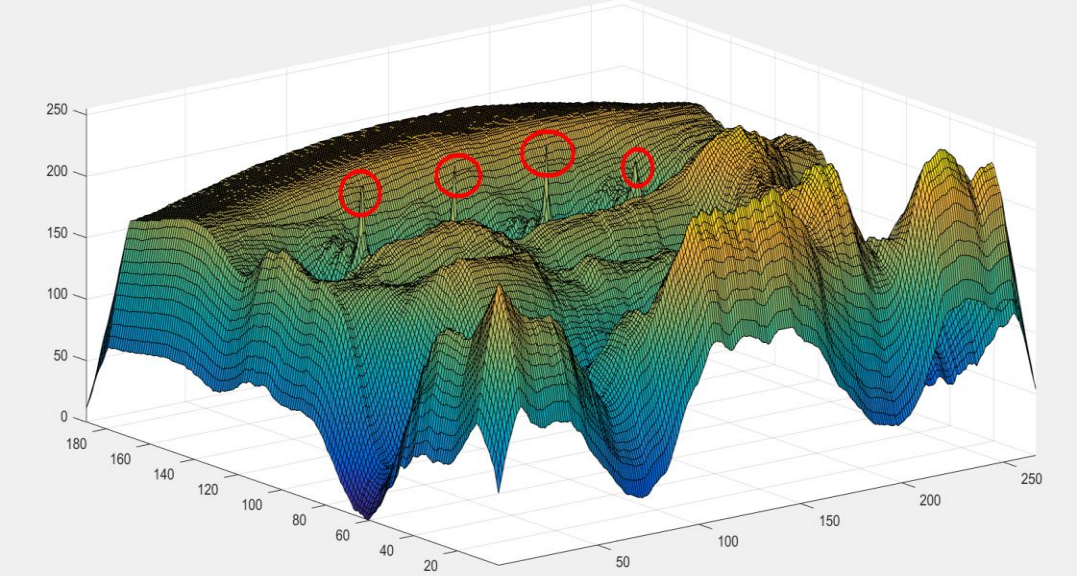





Correlated
[X,Y]: [106 118]
Index: 173
[R,G,B]: [0.6784 0.6784 0.6784]
y
x
20
40
60
80
100
120
140
160
180
50
100
150
200
250


Correlated
[X,Y]: [151 116]
Index: 168
[R,G,B]: [0.6588 0.6588 0.6588]
y
x
20
40
60
80
100
120
140
160
180
50
100
150
200
250


Correlated
[X,Y]: [195 117]
Index: 178
[R,G,B]: [0.698 0.698 0.698]
y
x
20
40
60
80
100
120
140
160
180
50
100
150
200
250



c. Mtalab 代码：

```
function [correlated_img] = correlation_filtering(img1, img2)
    % 读取给定图像和模板图像
    scene = imread(img1);
    template = imread(img2);
    m = size(scene, 1);
    n = size(scene, 2);
    x = size(template, 1);
    y = size(template, 2);
    % 将图像矩阵转换成双精度数
    scene = im2double(scene);
    template = im2double(template);
    % 套用公式进行相关运算
    correlated_img = scene;
    for i = 1:m
        for j = 1:n
            correlated_img(i,j) = 0;
            for p = 1:x
                for q = 1:y
                    a = round(i + p - (x - 1)/2 - 1);
                    b = round(j + q - (y - 1)/2 - 1);
                    if  a >= 1 && b >= 1 && a <= m && b <= n
                        correlated_img(i, j) = correlated_img(i, j) +
template(p, q) * scene(a, b);
                    end
                end
            end
        end
    end
    % 将相关值矩阵转换成uint8
    correlated_img = im2uint8(mat2gray(correlated_img));
    % 打印图像
```

```
    figure;
    subplot(2,2,1),imshow(scene),title('Original');
    subplot(2,2,2),imshow(correlated_img),title('Correlated'),  axis
on,xlabel x, ylabel y;
    subplot(2,2,3),surf(correlated_img),axis tight
```

2. 产生椒盐噪声图像，实现采用中值滤波：

(i) 分别产生2个独立、在区间[0,255]内均匀分布的随机矩阵 $t_1(x,y)$和 $t_2(x,y)$，这里$t_1(x,y) \neq t_2(x,y)$。（提示：采用Matlab命令'rand'）

a. 算法描述：

(1) 用Matlab的imread()函数分别扫描给定图像sport car.pgm得到图像矩阵I，再用用Matlab的size()函数获取图像矩阵I的行数m和列数n；

(2) 构建两个m×n的矩阵t1和t2,乘以10000后对256进行取模得到2个独立、在区间[0,255]内均匀分布的随机矩阵。

b. Matlab代码：

```
    I = imread(img1);
    [m,n] = size(I);
    t1 = uint8(mod(rand(m, n)* 10000,256));
    t2 = uint8(mod(rand(m, n)* 10000,256));
```

(ii) 设输入图像sport car.pgm为$f_0(x,y)$，采用下式产生椒盐噪声图像：

$$f(x,y)=\begin{cases}255 & \text{if } f_0(x,y) > t_1(x,y)\\ 0 & \text{if } f_0(x,y) < t_2(x,y)\\ f_0(x,y) & \text{otherwise}\end{cases}$$

a. 算法描述：

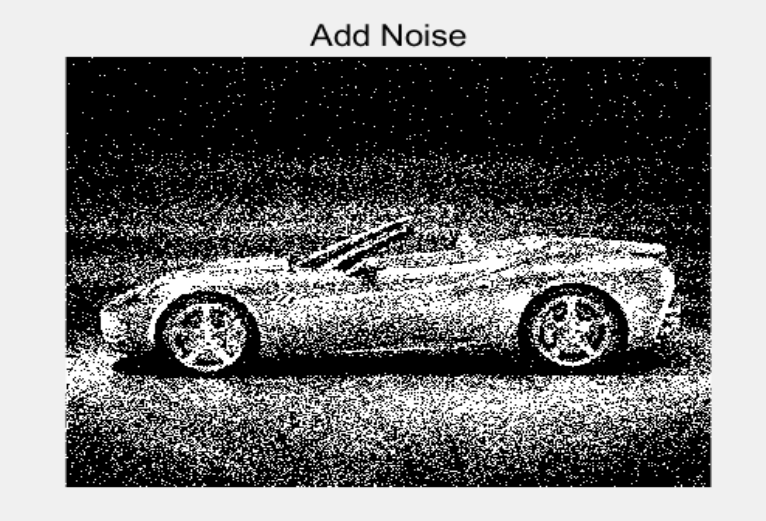
根据以上式子产生的噪声的方法不够严谨，没有考虑$t_1(x,y)$和$t_2(x,y)$的大小关系。这里我比较了$t_1(x,y)$和$t_2(x,y)$的大小，让$f_0(x,y)$与两者的最小值和最大值进行比较，这样产生的噪声方法会在数学上会更加严谨一些。

b. Matlab代码：

```
% 生成噪声
psnoise_img = I;
for i = 1:m
    for j = 1:n
        if t2(i, j) >= t1(i, j)
            if psnoise_img(i, j) > t1(i , j)
                psnoise_img(i, j) = 255;
            elseif psnoise_img(i, j) < t2(i , j)
                psnoise_img(i, j) = 0;
            end
        else
            if psnoise_img(i, j) > t2(i , j)
                psnoise_img(i, j) = 255;
            elseif psnoise_img(i, j) < t1(i , j)
                psnoise_img(i, j) = 0;
```

```
            end
        end
    end
end
```

c.图像效果：



(iii) 采用3×3窗口实现中值滤波，注：不能使用Matlab中的‘medfilt2’。

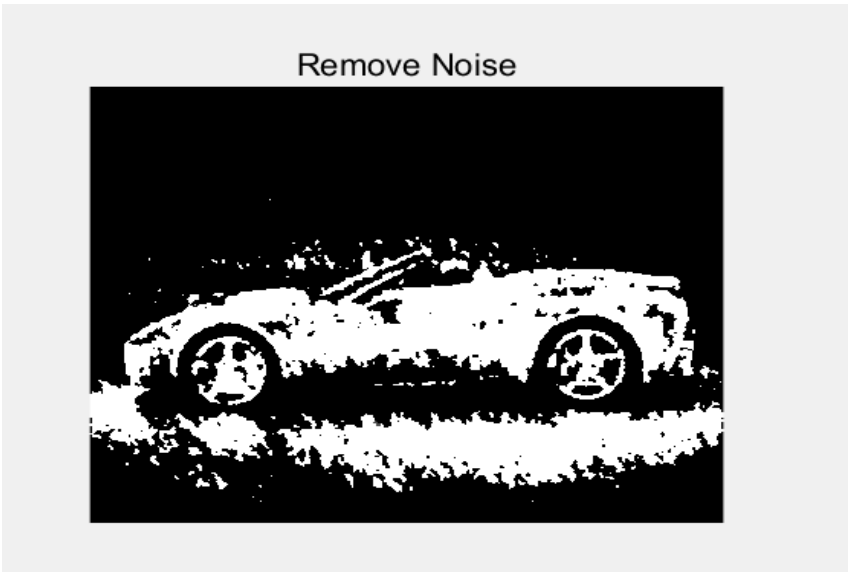
a. 算法描述：

构造3×3矩阵neighbors，用循环遍历整个噪声图像psnoise_img，对每个像素，其周围的像素值都存入矩阵neighbors中，不在图像中的像素点记为0，然后将neighbors向量化并调用median()求出这些值的中位数，然后将当前像素值替换成中值。

b. Matlab代码：

```
% 中值滤波
nonoise_img = psnoise_img;
for i = 1:m
    for j = 1:n
        neighbors = zeros(3,3);
        for p = -1:1
            for q = -1:1
                if i + p >= 1 && j + q >= 1 && i + p <= m && j + q <= n
                    neighbors(p + 2,q + 2) = nonoise_img(i + p, j + q);
                end
            end
        end
        % 矩阵向量化
        neighbors = neighbors(:);
        % 计算中位数并赋值
        nonoise_img(i, j) = median(neighbors);
    end
end
```

c. 图像效果：

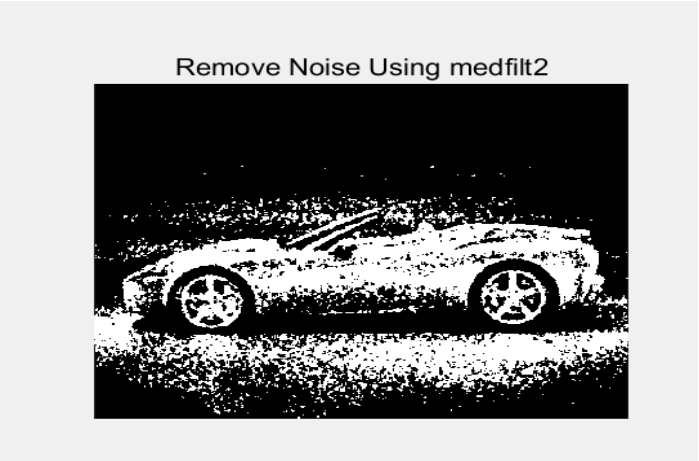


(iv) 显示原图像、椒盐噪声图像、中值滤波图像，并和采用Matlab‘medfilt2’的结果做比较。

a. Matlab代码：

```
% 用medfilt2进行中值滤波
nonoise_img_medfilt2 = medfilt2(psnoise_img, [3,3]);
```

b. 图像效果：



c. 对比分析：
用Matlab函数medfilt2()得到的图像效果比我自己写的效果更好，在过滤掉噪音的同时保留了更多的细节。我的自己实现的中值过滤在基本算法上没有问题，但是被白化的区域过多，需要进行一定的优化。