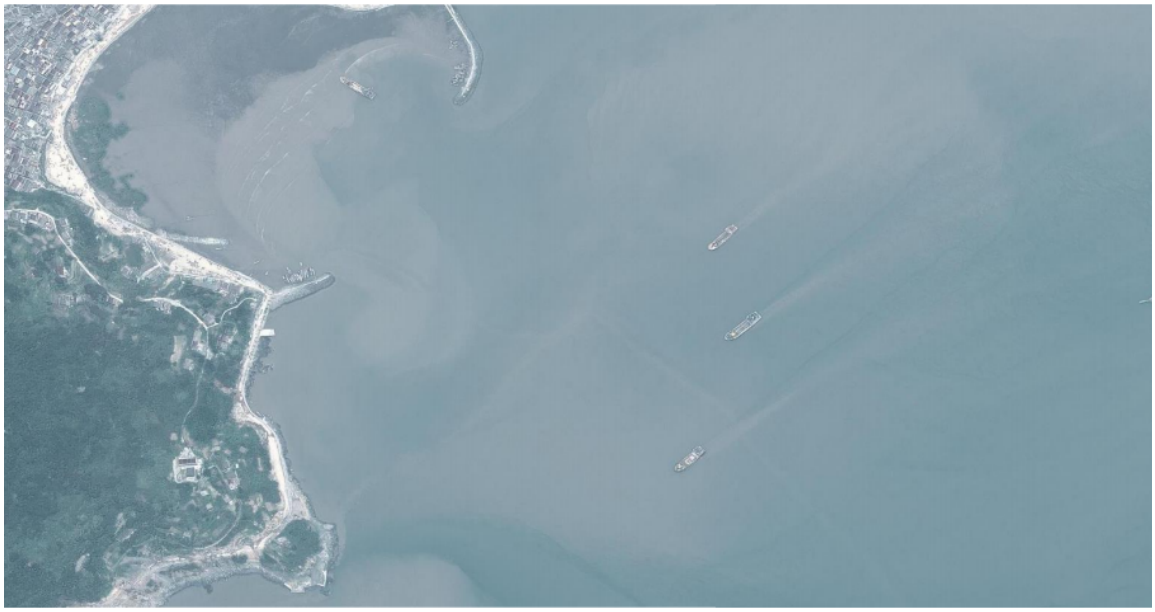
统计信号处理大作业

无88 刘子源 2018010895

一、问题描述

现有某光学传感器拍摄的一幅港口地区的遥感图像（见附件及下图）。请利用所学知识尝试检测出本图像中的陆地区域以及海面上的目标（本图像中为舰船）。



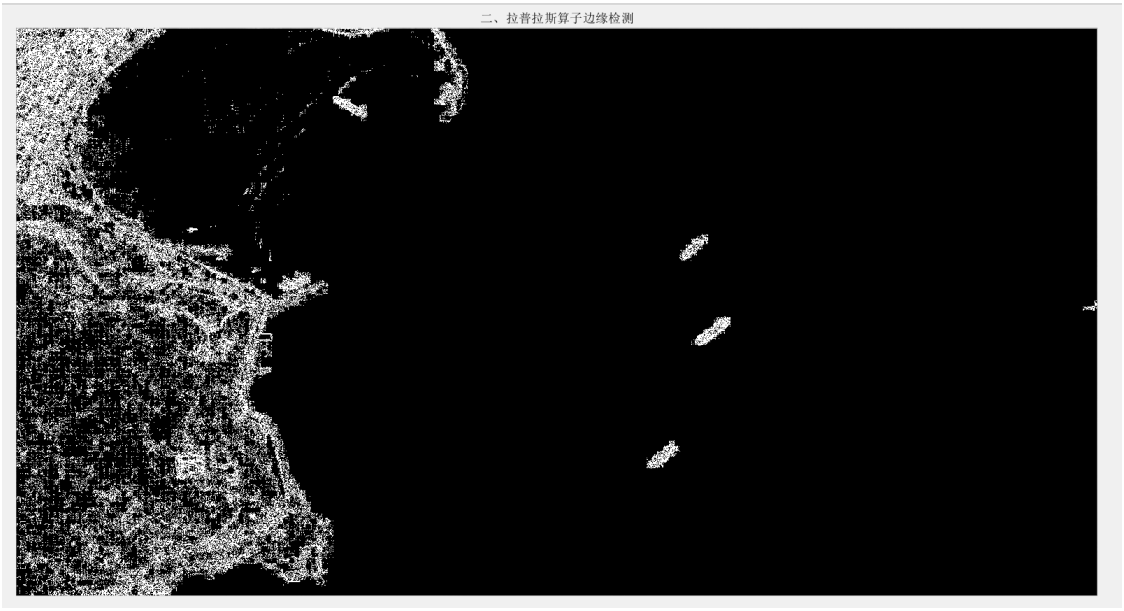
二、具体实现

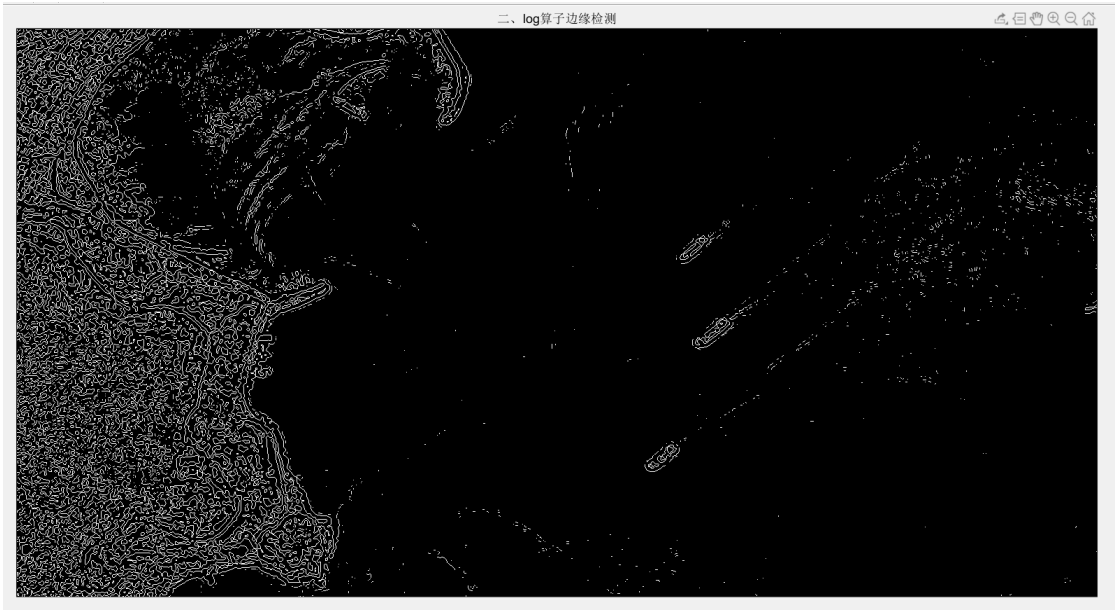
2.1图像预处理

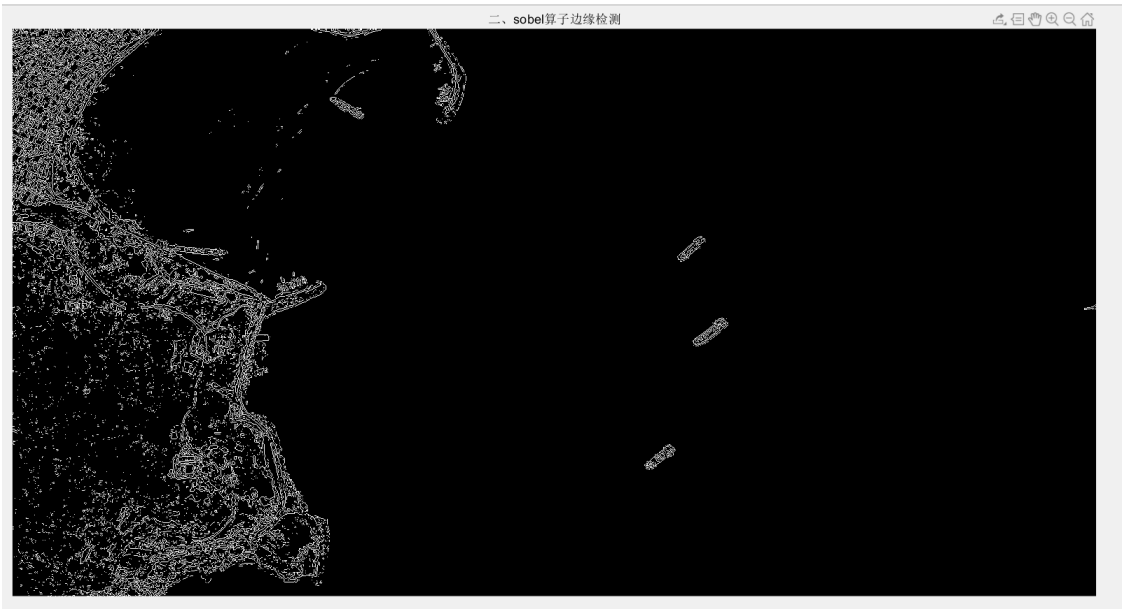
将图像矩阵归一化并转化为灰度图，以便于后续图像处理。

2.2边缘检测

进行图像的边缘检测，常用的检测方法有Laplace算子、Log算子、Sobel算子检测等，其各自效果如下：







其中Log算子对舰船的检测效果较差，Sobel算子和Laplace算子检测效果较好，这里选用Laplace算子。

Laplace算子的检测原理是，当图像灰度剧烈变化时，在一阶微分上会出现一个局部的极值，则在二阶微分上会形成一个过零点，并在零点两边产生一个波峰和波谷，所以我们只要通过设置一个阈值，检测过零点来进行图像的边缘检测。

以p点为中心构建一个3×3的像素块，有如下公式：

$$\Delta f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

如果Δf的绝对值超过了设定的阈值，则p点就是一个过零点像素。

将过零点像素的灰度设置为255（白色），其余像素的灰度设置为0（黑色），这样在Laplace算子边缘检测后即可将灰度图转化为二值图。经过提取后，已可以看出陆地和舰船的轮廓，这是因为它们与水面的交界处灰度值有剧烈的变化，从而被Laplace算子检测出来。

2.3降噪、粘合

经过Laplace算子检测后的水面上有较多的白色噪点我用以下代码进行初步的降噪处理：

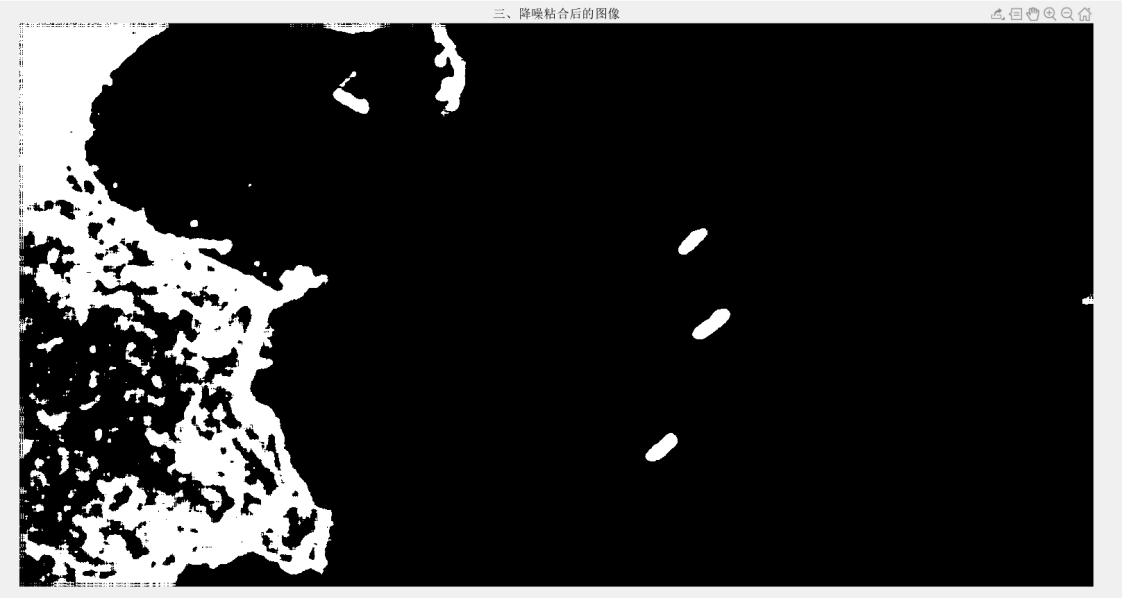
```
if sum(sum(pic(i - blocksize / 2:i + blocksize / 2, j - blocksize / 2 : j + blocksize / 2) > 0)) < blocksize ^ 2 * denoiserate
    newpic(i, j) = 0;
else
    newpic(i, j) = pic(i, j);
```

其意思是，检测某个像素点周围10×10区域的所有像素点，若该区域内灰度值为0的像素点少于某一阈值，则认为该点时孤立的白色噪点，将p点去除。

此外我们还可以看到，陆地和舰船的内部分布着许多的白色噪点，可以用上面相似的代码将其粘合：

```
if sum(sum(newpic(i - blocksize / 2:i + blocksize / 2, j - blocksize / 2 : j + blocksize / 2) > 0)) > blocksize ^ 2 * connectrate
    new2pic(i, j) = 1;
else
    new2pic(i, j) = 0;
```

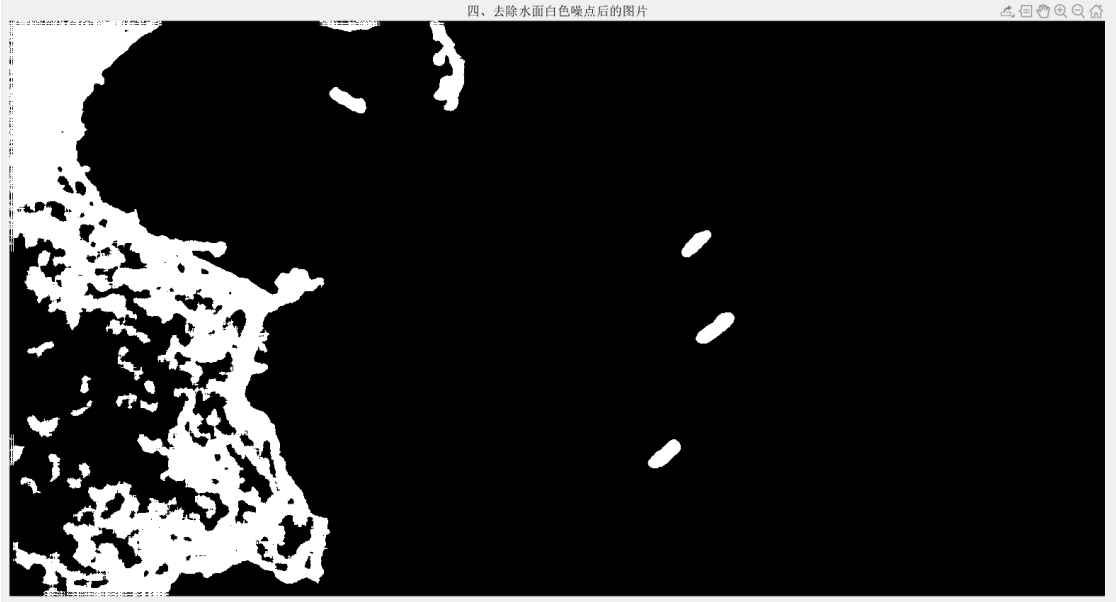
其意思是，检测某个像素点p周围10×10区域的所有像素点，若该区域内灰度值为0的像素点高于某一阈值，则认为该点在连通域之内，将该点设置为1。经过降噪粘合处理后，得到如下图像：



经过该步处理，水面上的白色噪点基本去除，但还有一些较大的噪点未能除掉，需要后续进一步处理；陆地边缘基本连通，其内部仍有黑色噪点，也需要后续进一步处理。

2.4除掉水面噪点

水面上靠近陆地边缘处有一些白色噪点，需要将其除去。观察发现，噪点面积与船只面积有明显差别。所以可以检测所有的连通区域，设置阈值下限200，将小于阈值的连通域统统视为噪点除去，得到如下图像：



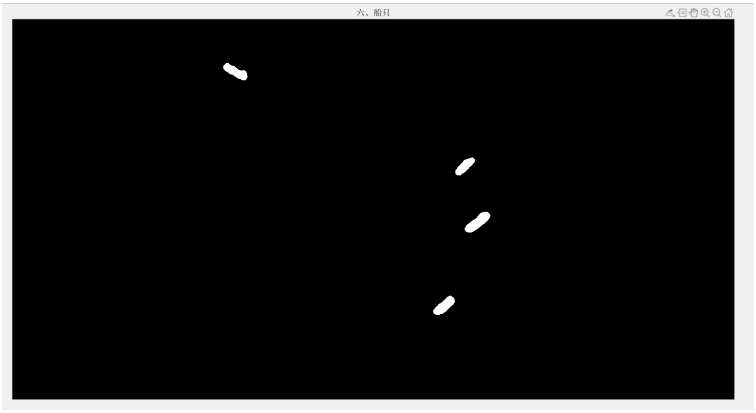
2.5大陆内部连通

由于陆地内部没有较强的边缘信息，Laplace算子并没有将其提取出来。故可通过两次0-1翻转，第一次翻转得到最大的连通域——水面后，再次翻转即可得到陆地和船只的连通域。得到如下图像：

2.6鉴定大陆、船只

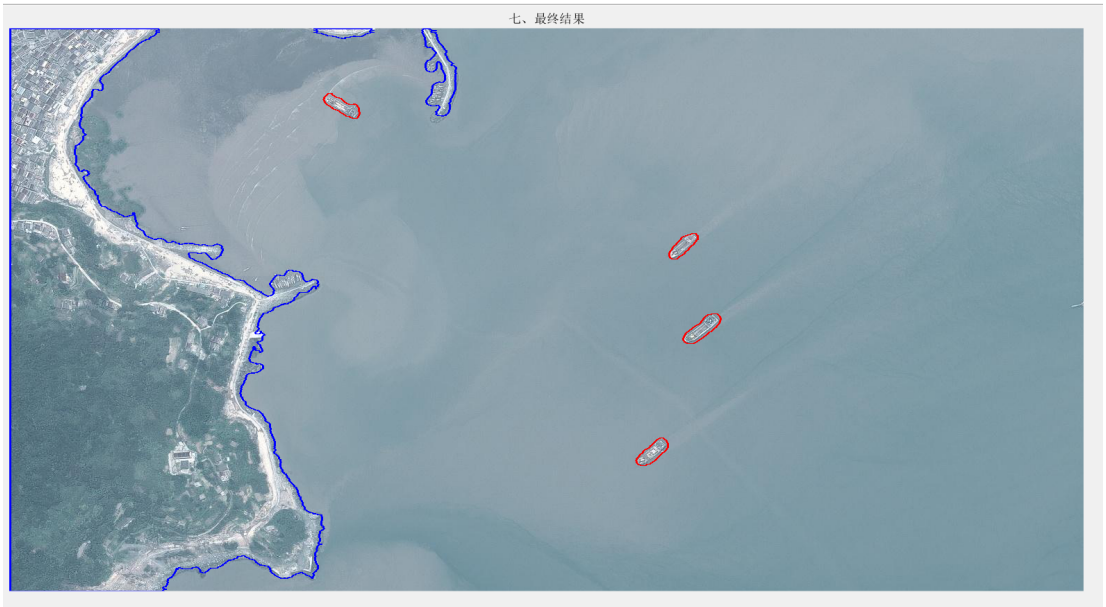
最初我想的比较简单，面积最大的连通域肯定是陆地，又因为噪点已被我全部除掉，剩下的在合理面积范围内的连通域就是船只，但是出现了下面的情况：

这时我发现，图中一共有3块陆地，它们彼此是不连通的。其中一块较小的陆地面积与船只相似，而被误判成了船只，所以单单靠面积来分类是行不通的。在二值图中，除了连通域的面积，剩下的信息就是形状了。观察发现，船只的形状近似矩形，比较工整，所以可以考虑在连通域外围画一个最小的外接矩形，用连通域面积与外接矩形面积之比作为判别标准，当比值较大时，说明该图形近似矩形，判定为船只，反之判定为陆地。判定效果如下：



2.7可视化

根据上面判定结果，在原图片绘出陆地和船只的边界，陆地边界用蓝线画出，船只边界用红线标出：



三、性能分析

检测方面，陆地边缘和海上船只检测效果较好，检测到了三块陆地和海面上所有的船只，但是由于岸边的船只与陆地太过接近而被当做了陆地。

程序运行方面，由于我不太熟悉matlab众多强大的函数，很多地方使用了for循环，导致程序耗时较长，比如在粘合连通域步骤中，由于每个点都要进行检测，复杂度为$O(m \times n \times area)$，m、n是图片的长和宽，area是像素块的面积（设定为100），该步用掉了很多的时间。

普适性方面，程序中有一些参数是手动调整的，比如说拉普拉斯算子的阈值，像素块的面积等等，都是根据反复观测运行结果找到的最优参数，在该图片中效果很好，可能在另一张图片中就没有这么好的效果了，主要原因还是作业中只给了一张图片，样本太少，由此想到若有大量的遥感图片的话，我可以用机器学习方面的知识自动训练出合适的参数，增强程序的普适性。

总体来看，该程序很好地完成了任务，但也有很多优化的空间。

四、总结

本次作业我完成了遥感图像的陆地和船只的检测，并给出了可视化的结果，完成了作业要求。

在本次作业中我有如下收获：

1、这次作业很大地提高了我的matlab编程能力。这是我第一次用matlab写这么多代码完成一次大作业级别的项目，由于 还没上过小学期matlab编程课和数字信号处理，我一开始对matlab的诸多内置函数不太了解，经过这次锻炼，我掌握了matlab中很多强大的数字图像处理函数如bwlabel，bwboundaries，regionprops等。这些函数很大程度上减轻了工作量。

2、我对统计信号处理有了更深刻的认知。程序中虽然没有直接用到课上学到的知识，却处处体现着统计信号处理的思想，比如阈值的选取，阈值过低则无法滤掉噪点，阈值过高又可能把有用的信息除掉了，这正是检测中虚警概率和漏检概率的制衡与折中。

3、这次作业提升了我解决问题的能力。现实问题永远不会有那么理想，需要随机应变，比如说检测陆地时其内部有黑色噪点，就换个思路去检测区域更大的海面，翻转过来就得到了平滑连通的陆地。又比如说小块的陆地被误检成船只，这时就要考虑，在一个只有0和1的二值图中，除了像素面积，还有什么有用的信息？也只剩下形状和位置了，船只可能出现在海面上各个位置，所以就剩下形状因素可以下功夫，然后我又发现网上以有人完成了内接矩形实现的工作，我直接拿来用就好了。

本次作业中也发现了自身的不足，尤其是对matlab编程的不熟练，我会在日后多加练习改进。

这是一次比较综合的作业，提升了我图像处理的能力，也让我对统计信号处理有了更加深刻的认识，感谢老师和助教的耐心指导！

五、参考文献

1、拉普拉斯算子边缘检测https://blog.csdn.net/u014485485/article/details/78364573?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1

2、matlab最小外接矩形检测https://blog.csdn.net/yeyang911/article/details/51384179?locationNum=13

3、matlab遥感图像处理https://wenku.baidu.com/view/b738c3f2d5bbfd0a7856732a.html

六、文件清单

文件名	说明
统计信号处理大作业.pdf	作业报告
image.bmp	原遥感图片
main.m	入口函数
preprocess.m	预处理函数
laplace_edge_test.m	拉普拉斯算子边缘检测函数
adhesion_and_denoise.m	降噪粘合函数
surface_denoise.m	海面去噪函数
land_denoise.m	陆地去噪函数
classify.m	判定函数
finish.m	可视化函数

七、源代码

1、主函数main.m

```
%读取图像
pic0 = imread('image.bmp');           %读入图像

%step 1：图像预处理，将图像转为灰度图后易于处理
pic1 = preprocess(pic0);

%step 2：Laplace算子边缘检测
pic2 = laplace_edge_test(pic1, 0.16);
%pic2 = log_edge_test(pic1);
%pic2 = sobel_edge_test(pic1);


%step 3：降噪、粘合，但此时较大的白色噪点仍未去除
pic3 = adhesion_and_denoise(pic2);

%step 4：去除水面白色噪点
pic4 = surface_denoise(pic3);

%step5：去除陆地黑色噪点
pic5 = land_denoise(pic4);

%step6：判定
[pic6_1, pic6_2] = classify(pic5);

%step7：在原图片上标定
pic7 = finish(pic0, pic6_1, pic6_2);
```

2、预处理preprocess.m

主要进行读取图像，图像矩阵归一化，得到原图的灰度图。

```
function [graypic] = preprocess(pic)
%PREPROCESS 图像预处理
%读取图像，图像矩阵归一化，得到灰度图及图像尺寸

graypic = mat2gray(pic);            %图像矩阵归一化
graypic = rgb2gray(graypic);        %得到灰度图

figure();
imshow(graypic);
title('一、原图的灰度图');

end
```

3、拉普帕斯算子检测laplace_edge_test.m

使用拉普拉斯算子检测边缘，输入参数中的laplace_threshhold为设定的阈值，可在入口函数main.m中调参。

```
function[laplace_graypic] = laplace_edge_test(graypic, laplace_threshhold)

[m, n] = size(graypic);
laplace_graypic = graypic;

for j = 2:m - 1
for k = 2 : n - 1
tmp = abs(4 * graypic(j, k) - graypic(j - 1, k) - graypic(j + 1, k) - graypic(j, k + 1) -
graypic(j, k - 1));
if (tmp > laplace_threshhold)
laplace_graypic(j, k) = 255;  %白
else
laplace_graypic(j, k) = 0;    %黑
end
end
end

figure();
imshow(laplace_graypic);
title('二、拉普拉斯算子边缘检测');

end
```

4、降噪粘合adhesion_and_denoise.m

设置像素块边长$\text{blocksize} = 10$，降噪阈值$\text{denoiserate} = 0.1$，和粘合阈值$\text{connectrate} = 0.3$，其意为当某一像素点p周围的像素块中，值为“1”的像素比例小于0.1的话，认为p点为孤立的白色噪点，将其设置为“0”，否则保持不变；当某一像素点p周围的像素块中，值为“1”的像素比例大于0.3的话，认为p点在连通域内，将其设置为“1”，否则设置为0。

```
function[new2pic] = adhesion_and_denoise(pic)

[m, n] = size(pic);
newpic = pic;
blocksize = 10;          % 10 * 10像素块
edgesize = 4;            %处理边缘，去掉边缘噪点，否则在最后会出问题
denoiserate = 0.1;       %去除像素块白色噪点
connectrate = 0.3;       %连接白色像素块

%降噪
%可取消白色噪点
for i = m - 1 - blocksize / 2 : -1 : 1 + blocksize / 2
for j = n - 1 - blocksize / 2 : -1 : 1 + blocksize / 2
if sum(sum(pic(i - blocksize / 2:i + blocksize / 2, j - blocksize / 2 : j + blocksize / 2) > 0)) <
blocksize ^ 2 * denoiserate
    newpic(i, j) = 0;
else
newpic(i, j) = pic(i, j);
end
end
end

%newpic = medfilt2(newpic);

%粘合
%将成片的白色像素块粘合
new2pic = newpic;
for i = 1 + blocksize / 2 : m - 1 - blocksize / 2
for j = 1 + blocksize / 2 : n - 1 - blocksize / 2
if sum(sum(newpic(i - blocksize / 2:i + blocksize / 2, j - blocksize / 2 : j + blocksize / 2) >
0)) > blocksize ^ 2 * connectrate
```

```
new2pic(i, j) = 1;
else
new2pic(i, j) = 0;
end
end
end

%处理边缘
new2pic(1:edgesize, : ) = new2pic(edgesize:edgesize * 2 - 1, : );                    %处理上边
new2pic(m - edgesize + 1:m, : ) = new2pic(m - 2 * edgesize + 1:m - edgesize, : );    %处理下边
new2pic(:, 1 : edgesize) = new2pic(:, edgesize : edgesize * 2 - 1);                  %处理左边
new2pic(:, n - edgesize + 1 : n) = new2pic(:, n - 2 * edgesize + 1 : n - edgesize); %处理右边

figure()
imshow(new2pic)
title('三、降噪粘合后的图像')

end
```

5、去除水面白色噪点surface_denoise.m

将面积小于200的像素块认为是噪点并将其除去，这样海面上所有的白色噪点将都被除掉。

```
function[newpic] = surface_denoise(pic)
% 去除水面较大的白色噪点

newpic = pic;
bw = bwlabel(newpic);                              %标记连通区域
stats = regionprops(bw, 'Area');                   %得到各连通区面积
[b, index] = sort([stats.Area], 'descend');        %从大到小排序
numend = find(b > 200, 1, 'last');                 %找到面积大于200的像素块
newpic = ismember(bw, index(1:numend));            %面积小于200的像素块变成黑色
figure();
imshow(newpic)
title('四、去除水面白色噪点后的图片');

end
```

6、去除陆地黑色噪点land_denoise.m

核心在于两次反转，第一次反转后最大的连通域变成了海面，由于在上一步中已经消除了海面上的所有噪声，所以这是一个非常平滑的连通域，然后在反转回来，这样陆地内部所有黑色的“洞”就都被除掉了。

```
function[newpic] = land_denoise(pic) % 去除陆地黑色噪点

newpic = -pic + 1;                             %黑白像素反转
bw = bwlabel(newpic);                          %标记连通区域
stats = regionprops(bw, 'Area');               %得到各连通区面积
[~, index] = sort([stats.Area], 'descend');    %从大到小排序
newpic = ismember(bw, index(1));               %最大的那个是海面
newpic = -newpic + 1;                          %黑白像素反转回来
figure();
imshow(newpic);
title('五、去除陆地黑色噪点后的图片');

end
```

7、判定陆地和船只classify.m

此时连通区域只剩下陆地和船只了，再根据连通区域的面积及其形状就能判定陆地和船只。

```
function[newpic1, newpic2] = classify(pic) % 判定大陆和船只

[m, n] = size(pic);
newpic = pic;
newpic1 = zeros(m, n);
newpic2 = zeros(m, n);
[bw, index] = bwlabel(newpic);                 %标记连通区域

siz = [];
for i = 1:index
siz = [siz, size(find(bw == i), 1)];           %得到所有连通区域的面积
end

for i = 1:index
[r, c] = find(bw == i);
area = minboundrect(c, r);                     %得到连通区域内接矩形的面积
```

```
if (siz(i) / area > 0.8)                    %将连通区域面积与内接矩形面积作比
newpic2(find(bw == i)) = 1;
else
newpic1(find(bw == i)) = 1;
end
end

figure();
imshow(newpic1);
title('六、陆地');
figure();
imshow(newpic2);
title('六、船只');

end

```

8、可视化

```
function[newpic] = finish(pic0, pic1, pic2)

newpic = pic0;
landboudary = bwboundaries(pic1);      %得到陆地边界
shipboundary = bwboundaries(pic2);     %得到船只边界
linewidth = 3;                         %检测框宽度

for i = 1 : size(landboudary, 1)
for j = 1 : size(landboudary{ i }, 1)
for k = 0 : linewidth - 1
h = landboudary{ i }(j, 1);
w = landboudary{ i }(j, 2);
newpic(h, w + k, :) = [0 0 255];       %蓝线划定陆地边界
end
end
end

for i = 1 : size(shipboundary, 1)
for j = 1 : size(shipboundary{ i }, 1)
for k = 0 : linewidth - 1
h = shipboundary{ i }(j, 1);
w = shipboundary{ i }(j, 2);
newpic(h, w + k, :) = [255 0 0];       %红线划定船只边界
end
end
end

figure();
imshow(newpic);
title('七、最终结果');

end
```