《计算机视觉（1）》实验报告

实验六 使用区域生长的分割

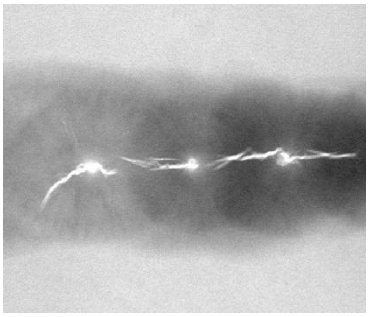
实验小组成员 （学号+班级+姓名）	分工及主要完成任务	成绩
201800830004 18 数科 叶家辉	全部	

山东大学

2021 年 4 月

完成《数字图像处理》P494页例10.23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。

参照例10.23中“使用区域生长的分割”算法，对如下的有缺陷焊缝的X射线图像得到图10.51中(b)~(i)中的实验结果。



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 区域生长

区域生长是根据预先定义的生长准则将像素区域组合为更大区域的过程。基本方法是从一组“种子”点开始，将与种子预先定义的性质相似的那些邻域内的像素添加到每个种子上形成这些生长区域。区域生长还需要用一定的方法表示出终止规则，当不再有像素点满足加入某个区域的准则时，区域生长就会停止。令$f(x,y)$表示一个输入图像阵列；$S(x,y)$表示一个种子阵列二值图像，其中种子点处为 1，其它位置为 0；Q表示在每个位置(x,y)处所用的属性准则。一种基于八连接的基本区域生长算法包含以下步骤：

1、在种子阵列图像$S(x,y)$中寻找所有连通分量，并把每个连通分量腐蚀为一个像素，把找到的所有这种像素标记为 1，把$S(x,y)$中的所有其他像素标记为 0；

2、在坐标对(x,y)处形成图像f_Q：如果输入图像$f(x,y)$在该坐标处满足给定属性 Q，则令$f_Q(x,y)=1$，否则令$f_Q(x,y)=0$；

3、令$g(x,y)$是这样形成的图像：即把$f_Q(x,y)$中为 8 连通种子点的所有 1 值点添加到$S(x,y)$中的每个种子点；

4、用不用的区域标记（如 1，2，3，……）标出$g(x,y)$中的每个连通分量，这就是由区域生长得到的分割图像。

在本实验中，采用分割有缺陷的焊接区域来说明区域生长的应用。

工作的第一步是确定种子点。从问题的物理性质来看，裂缝和孔隙与实的焊缝相比，对 X 射线的衰减要小，因此包含这种类型缺陷的区域要比 X 射线图像中其他部分明显亮一些。由此看来，可以用在高百分比处设置阈值对原始图像进行阈值处理来提取种子点。在前面的实验中也已经涉及过阈值处理，这里的思路和之前类似。这里的阈值T通常由百分数n表示，指的是大于给定集合内所有数字的$n\%$的最小数字。然后将每个$f(x,y)>T$的点设为 1，反之则设为 0。即由下式生成一幅二值图像：

$$g(x,y)=\begin{cases}1, f(x,y)>T\\ 0, f(x,y)\le T\end{cases}$$

第二步是确定一个属性准则。在这个例子中，要求添加到每个种子中的像素点满足且仅满足以下两个条件：（1）对于某个种子是 8 邻接的；（2）与种子满足某种相似性度量的判定。如果使用灰度差的绝对值作为一种相似性度量，则应用于每一个位置(x,y)处的属性是：

$$Q=\begin{cases}TRUE, & \textit{种子和点}(x,y)\textit{处的像素间灰度差的绝对值}<T\\ FALSE, & \textit{其它}\end{cases}$$

1.2 多阈值处理

前面的实验中我们关注了单一全局阈值对图像进行分割。我们可以将实验 5 中的阈值处理方法扩展到任意数量的阈值。在有 K 个类C_1，C_2，C_3，……，C_K的情况下，类间方差可以由下式定义：

$$\sigma_B^2 = \sum_{k=1}^{K} P_k (m_k - m_G)^2$$

其中，

$$P_k = \sum P_i \, , m_k = \frac{1}{P_k} \sum i P_i \, , m_G = \sum_{i=1}^{L} i p_i$$

对于由三个灰度间隔组成的三个类（就像本实验中的情况，这时需要两个阈值来分割），类间方差由下式给出：

$$\sigma_B^2 = P_1 (m_1 - m_G)^2 + P_2 (m_2 - m_G)^2 + P_3 (m_3 - m_G)^2 \quad (1)$$

其中，

$$P_1 = \sum_{i=1}^{K_1} P_i \, , P_2 = \sum_{i=K_1+1}^{K_2} P_i \, , P_3 = \sum_{i=K_2+1}^{L} P_i \quad (2)$$

$$m_1 = \frac{1}{P_1} \sum_{i=1}^{K_1} i P_i \, , m_2 = \frac{1}{P_2} \sum_{i=K_1+1}^{K_2} i P_i \, , m_3 = \frac{1}{P_3} \sum_{i=K_2+1}^{L} i P_i \quad (3)$$

并且有以下两个式子成立：

$$P_1 m_1 + P_2 m_2 + P_3 m_3 = m_G, \; P_1 + P_2 + P_3 = 1 \quad (4)$$

则最佳阈值k_1^*和k_2^*满足：

$$[k_1^*, k_2^*] = argmax(\sigma_B^2 (k_1, k_2)) \quad (5)$$

然后，阈值处理的图像由下式给出：

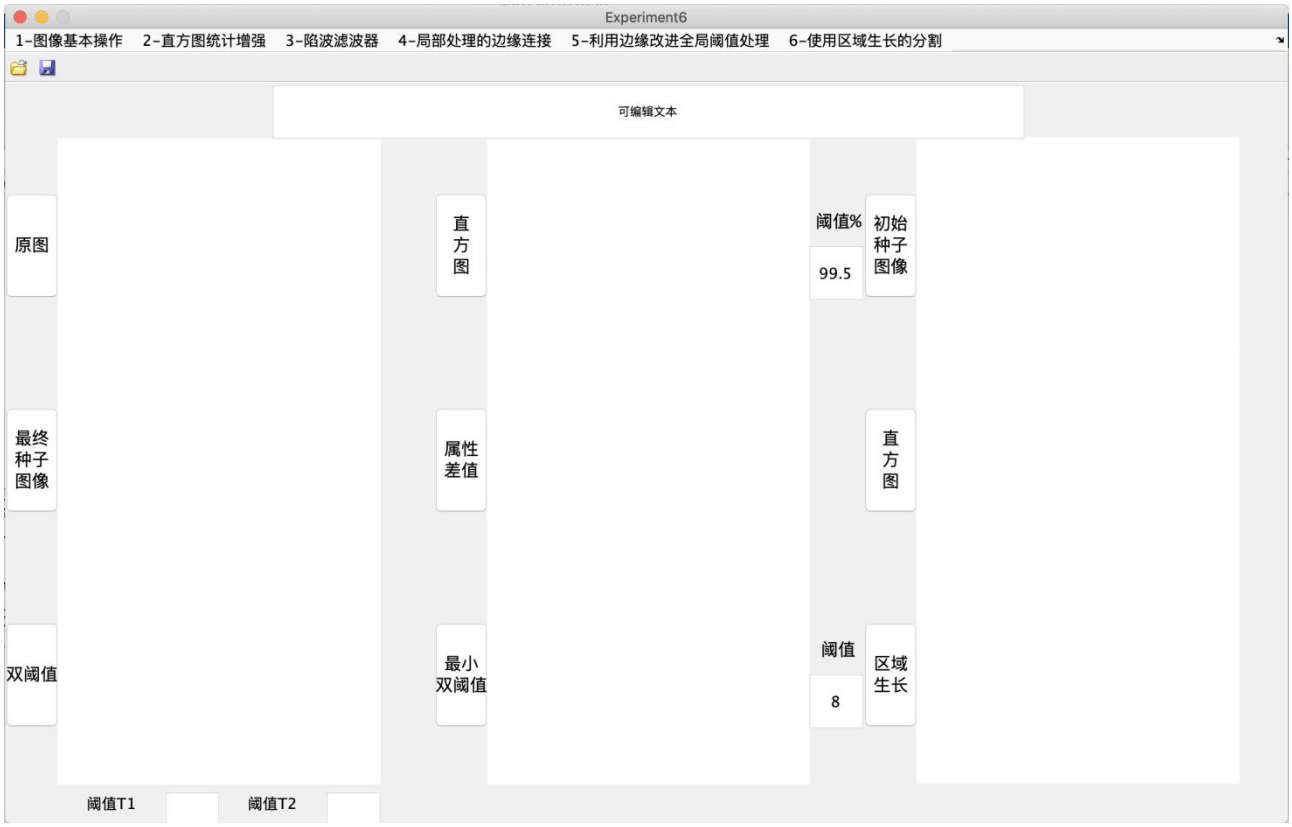
$$g(x,y) = \begin{cases} a, & f(x,y) \le k_1^* \\ b, & k_1^* < f(x,y) \le k_2^* \\ c, & f(x,y) > k_2^* \end{cases} \quad (6)$$

其中，a,b,c 是任意三个有效的灰度值。

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“6-使用区域生长的分割”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。

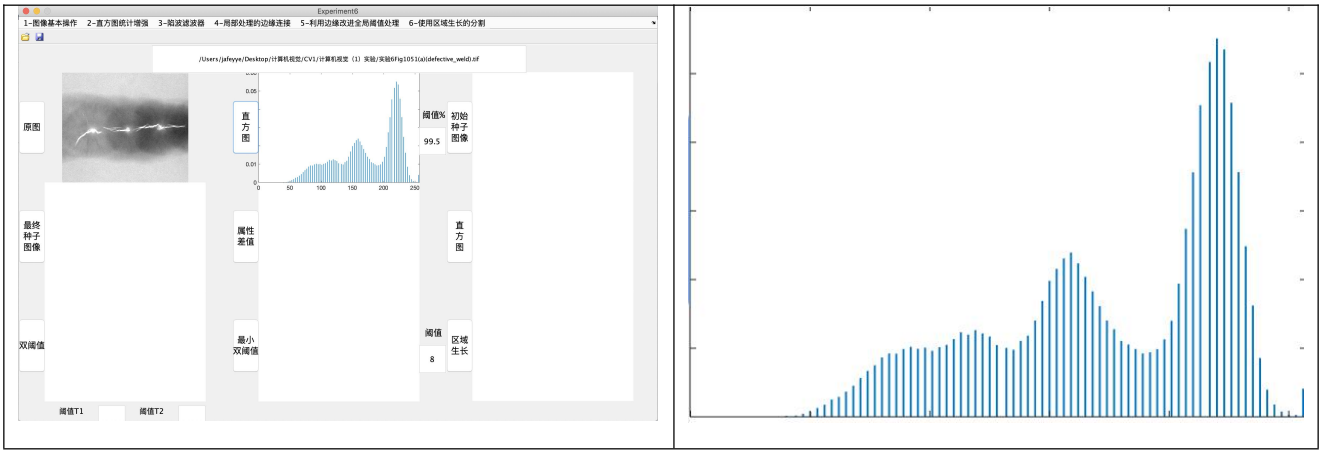
```
% --------------------------------------------------------------------
function uipushtool1_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (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);
```

2.2 做原图的直方图

在前面的实验中已多次涉及过直方图的绘制，在此结合本实验做一个简单的回顾。通过点击“直方图”按键实现直方图的绘制，并将结果返回至第一行第二列的显示区域中。首先

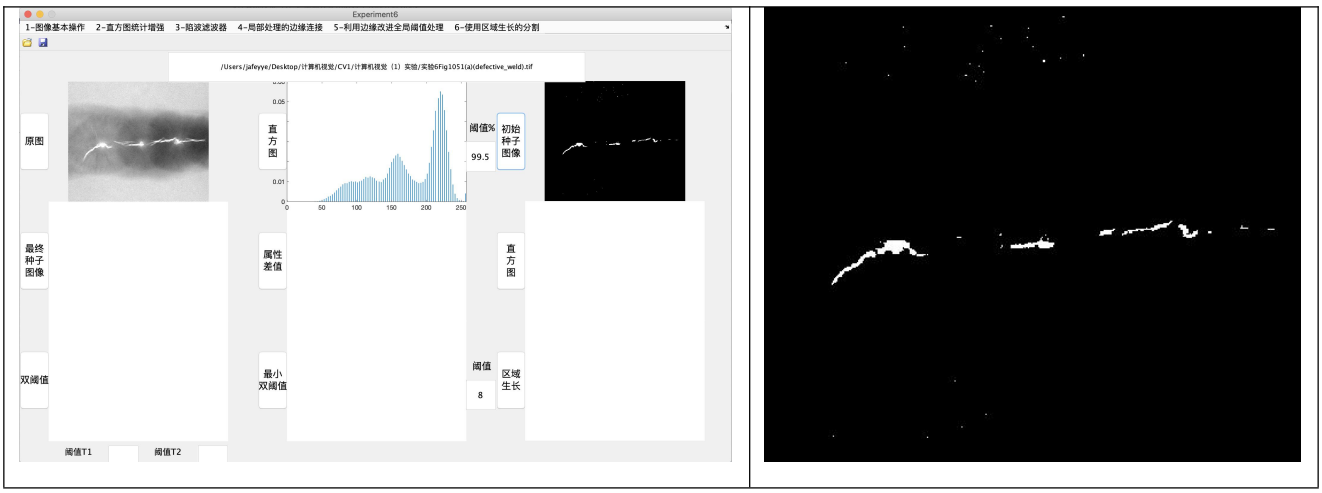
从顶部文本框处获取图像路径名和文件名，并由此读入图像信息，然后用每个灰度像素出现的个数除以整张图的像素个数来计算这张图的直方图（注意 Matlab 中的范围是 1 到 256），并绘制在坐标图区 axes2（第一行第二列的显示区域）中。



2.3 初始种子图像

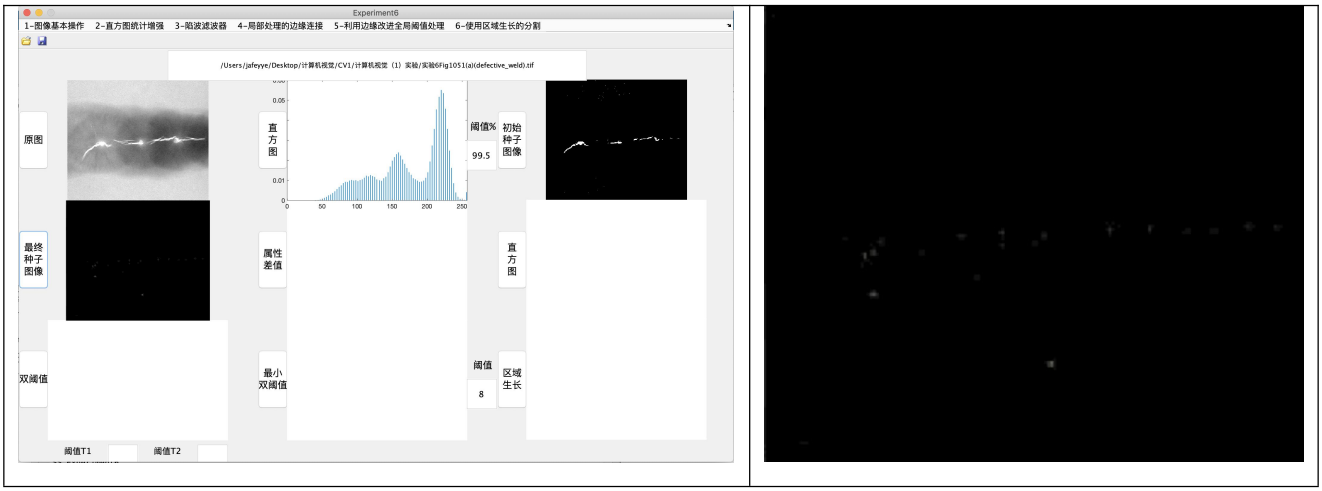
阈值处理前面的实验中也已经涉及过，其基本思想是当且仅当像素(x, y)处的某些特征（此处是灰度值）满足一定条件时，将该像素置为 1，否则置为 0。落实到操作上，下图中代码的其余部分展示了这一过程。首先从文本框中读取设定的阈值。该值通常由百分数n表示，指的是大于给定集合内所有数字的n%的最小数字。所以接下来通过这个百分数计算出该阈值在整个集合中所排的位置，并且找到这个位置所代表的具体值。然后创建一个与原图象同维的零矩阵，并对原图中的每个像素做判断：如果该元素大于阈值，就将零矩阵对应位置的元素置 1，否则保持为 0。最后把如是生成的二值图像显示在坐标图区 axes3（第一行第三列的显示区域）中。

```
% --- Executes on button press in pushbutton3.
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)
T = str2num(get(handles.edit2,'string'))/100;
file = get(handles.edit1,'string');
f = imread(file);
num = int64(numel(f) * T);
sort_f = sort(f(:));
threshold = sort_f(num);
C = zeros(size(f));
[H,W] = size(f);
for i = 1:H
    for j = 1:W
        if f(i,j)>threshold
            C(i,j)=1;
        end
    end
end
axes(handles.axes3);
imshow(C);
```



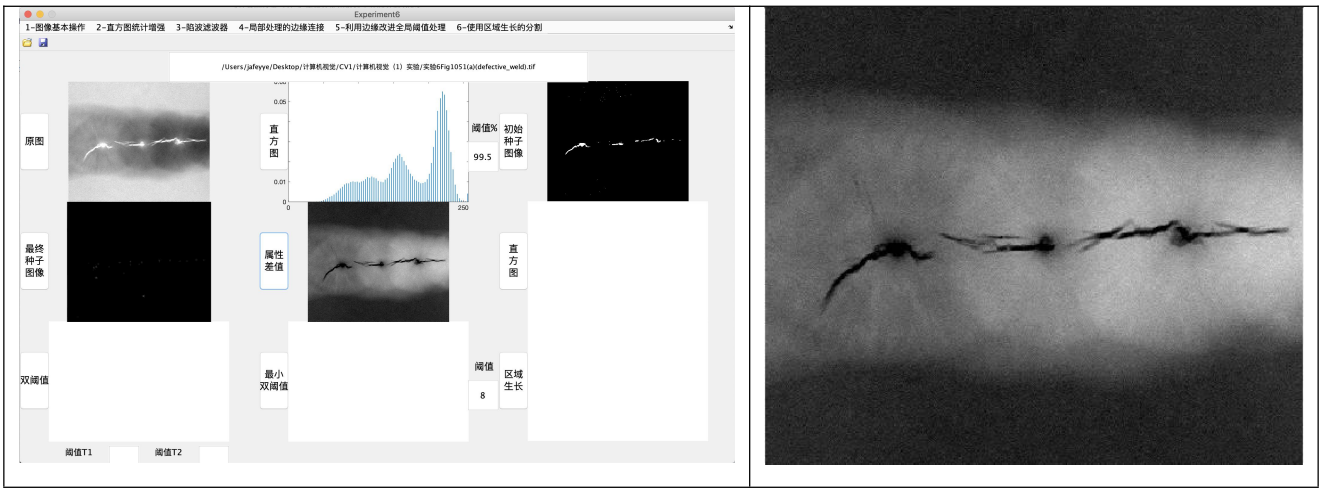
2.4 最终种子图像

初始种子虽然已经达到了百分比的 99.5%，但是点仍然比较多，所以我们需要将初始种子中的每个连通分量用形态学腐蚀为一个单点，得到最终种子图像。



2.5 原图与种子差值的绝对值图像

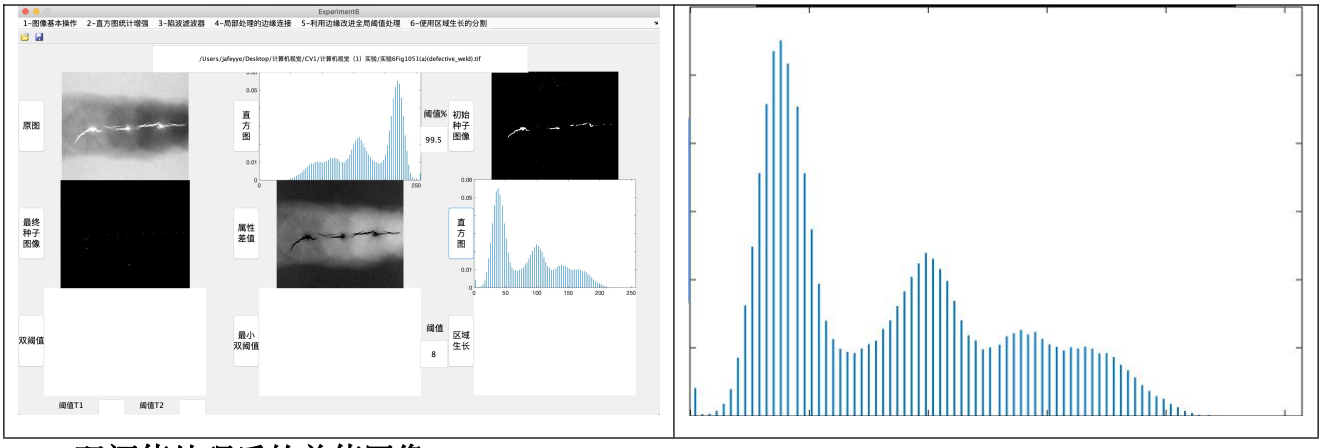
下面我们计算用最大灰度（256，Matlab 中索引从 1 开始，所以灰度范围为 1～256）减去原图的灰度再取绝对值，得到的图像主要是用于接下来计算每个位置(x, y)处的属性所需要的差值。



2.6 原图与种子差值的绝对值图像的直方图

接下来我们绘制 2.5 中的图像的直方图。该步骤和 2.2 基本一致，只是处理的目标图像不一样。将 2.5 的直方图绘制在坐标图区 axes6（第二行第三列的显示区域）中。

可以很明显的看出，该直方图有三个波峰两个波谷，有三个主要模式，所以可以使用双阈值来处理差值图像，这就是 2.7 所展示的。



2.7 双阈值处理后的差值图像

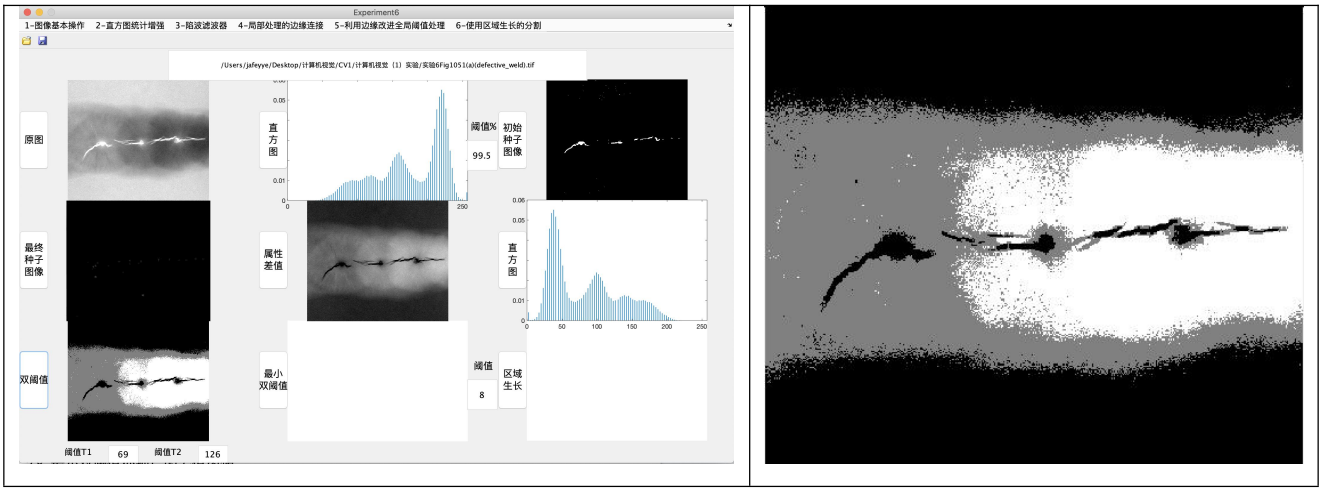
通过观察 2.6 的直方图可以看出，2.5 中的图像包含三种主要的模式，所以可以选择双阈值处理的技术。

如下图所示。首先得到图像的直方图。再用两个 for 循环取遍k_1，k_2的值，根据式(1)~(4)得出σ_B^2的表达式，然后通过比较来得到最大的σ_B^2，并记录下此时k_1，k_2的值为k_1^*，k_2^*

```
404 -   img = getimage(handles.axes5);
405 -   h = imhist(img)/numel(img);
406 -   sigmaB2 = 0;
407 -   for k1 = 1:254
408 -       for k2 = k1:255
409 -           P1 = 0;
410 -           m1 = 0;
411 -           P2 = 0;
412 -           m2 = 0;
413 -           P3 = 0;
414 -           m3 = 0;
415 -           for i = 1:k1
416 -               P1 = P1+h(i);
417 -               m1 = m1+i*h(i);
418 -           end
419 -           m1 = m1/P1;
420 -           for i = k1+1:k2
421 -               P2 = P2+h(i);
422 -               m2 = m2+i*h(i);
423 -           end
424 -           m2 = m2/P2;
425 -           for i = k2+1:256
426 -               P3 = P3+h(i);
427 -               m3 = m3+i*h(i);
428 -           end
429 -           m3 = m3/P3;
430 -           mG = P1*m1+P2*m2+P3*m3;
431 -           if P1*(m1-mG)^2+P2*(m2-mG)^2+P3*(m3-mG)^2 > sigmaB2
432 -               sigmaB2 = P1*(m1-mG)^2+P2*(m2-mG)^2+P3*(m3-mG)^2;
433 -               k1_s = k1;
434 -               k2_s = k2;
435 -           end
436 -       end
437 -   end
```

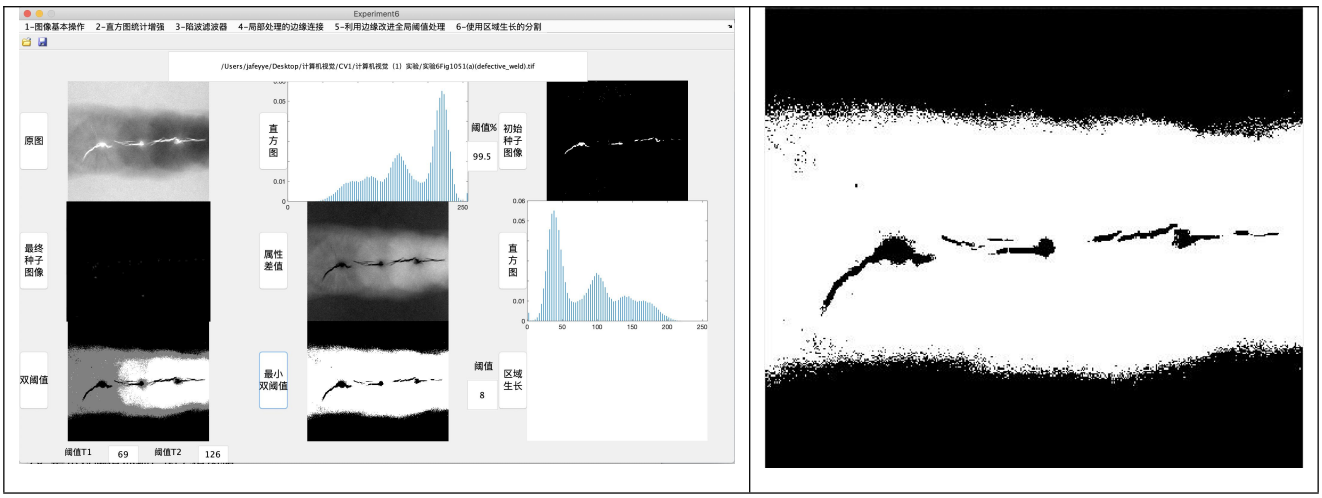
然后根据得到的两个阈值，由式(6)得到三分图，其中$a=0$(对应灰度值为 1)，$b=0.5$(对应灰度值为 128)，$c=1$(对应灰度值为 256)。最后将得到的图像绘制在坐标图区 axes7(第三行第一列的显示区域)中，将两个阈值分别显示在两个文本框中。我们发现双阈值分割不能解决缺陷的分割问题，尽管阈值处在主要波谷中。

```
438 -    [H,W] = size(img);
439 -    for i = 1:H
440 -        for j = 1:W
441 -            if img(i,j)<=k1_s
442 -                g(i,j) = 0;
443 -            elseif img(i,j)>k1_s & img(i,j)<=k2_s
444 -                g(i,j) = 0.5;
445 -            elseif img(i,j)>k2_s
446 -                g(i,j) = 1;
447 -            end
448 -        end
449 -    end
450 -    set(handles.edit3,'string',num2str(k1_s));
451 -    set(handles.edit4,'string',num2str(k2_s));
452 -    axes(handles.axes7);
453 -    imshow(g);
```



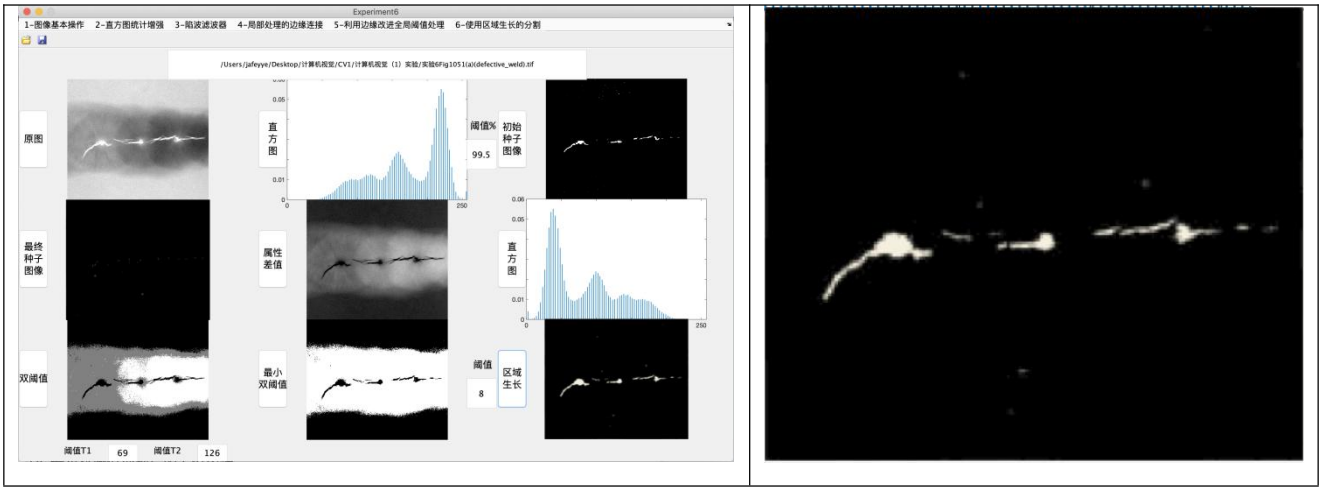
2.8 最小双阈值处理后的差值图像

首先按照 2.7 的步骤求出两个阈值，然后根据小的阈值k_1^*使用 im2bw()函数对差值图像做二值分割。



2.9 区域生长得到的分割图像

按照 1.1 中描述的步骤(2)~(4), 首先判定差值图像中的像素是否满足灰度小于设定值(本实验取 8），对于满足该条件并且还是种子点的 8 邻域点的点，将其添加到种子图像中，然后重复该步骤，直到没有点再满足加入条件为止。可以发现，尽管我们选用了单一阈值，但它的表现依旧很好。



三、实验结果与总结

原图	双阈值分割	最小双阈值分割	区域生长