

《计算机视觉（1）》实验报告

实验二 使用直方图统计的局部增强

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	

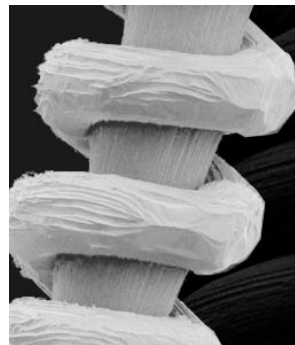
山东大学

2021 年 3 月

完成《数字图像处理》P87 页例 3.12 的编程实验，编程语言可以选择 Matlab, C, C++, OpenCV, Python 等。如图显示了一根绕在支架上的钨丝的 SEM(扫描电子显微镜)图像。图像中央的钨丝及其支架很清楚并很容易分析。在该图像的右侧即图像的暗侧，有另一根钨丝的结构，但几乎不能察觉到，其大小和特征当然也不容易辨认。通过对比度操作进行局部增强是解决这种图像中包含部分隐含特征问题的理想方法。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。

实验完成主要任务：

1) 参照教材给出的基于局部直方图统计矩（均值及方差）的亮度及对比度增强方案，编写实现图像全局直方图、局部直方图统计程序、基于直方图的全局均值及方差计算程序、局部方差及方差计算程序、选择性局部图像增强程序；



2) 通过实验，探讨增强关键参数选择方法，包括参数 E , K_0 , K_1 , K_2 以及局部区域 S_{xy} 大小, 分析改变这些参数对增强图像质量的影响规律。

原始图像

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 全局（或局部）直方图计算与绘制

假设一幅灰度图像在 $[0, G]$ 内共有 L 个灰度级，则其灰度图定义为

$$h(r_k) = n_k$$

其中 r_k 表示第 k 级灰度， n_k 表示 r_k 的像素个数。对于一般的图像（unit8 类），取 $G = 255$ 。一般采取归一化得到频率直方图，即

$$p(r_k) = \frac{h(r_k)}{n} = \frac{n_k}{n}$$

对于局部的情况，只需将整幅图像替换为以某点为中心在给定邻域大小下的子图即可。对于边界上的点，需要用 0 进行填充（padding），其他道理相同。

1.2 全局（或局部）均值和方差计算

一方面，均值和方差作为某种矩，由于一旦得到了直方图就可以很简单地得到所有矩，所以我们可以通过直方图来计算均值和方差。另一方面，我们也可以直接从取样值计算取样均值和取样方差：

$$m = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$
$$\sigma^2 = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f(x, y) - m]^2$$

同样地，对于局部均值和方差，只需要对边界上的点用 0 进行填充，其他均同此理。

1.3 图像局部增强

若 $f(x, y)$ 表示在图像 (x, y) 处的像素值， $g(x, y)$ 表示在图像 (x, y) 处增强后的像素值，则令

$$g(x, y) = \begin{cases} E \cdot f(x, y), & m_{S_{xy}} \leq k_0 m_G \text{ and } k_1 \sigma_G \leq \sigma_{S_{xy}} \leq k_2 \sigma_G \\ f(x, y), & \text{others} \end{cases}$$

其中 E, k_0, k_1, k_2 是人为设定的参数， m_G 是全局均值， σ_G 是全局标准差， $m_{S_{xy}}$ 是局部均值， $\sigma_{S_{xy}}$ 是局部标准差。另外，为了保留细节，通常选取的邻域大小要尽可能小。虽然在 GUI 中这个值是可以自定义的，但是一般建议取边长为 3 的正方形。

在判定条件中， $m_{S_{xy}} \leq k_0 m_G$ 是为了保证增强的区域是暗区，避免亮区受影响， $k_1 \sigma_G \leq \sigma_{S_{xy}} \leq k_2 \sigma_G$ 是为了保证增强的区域要有一定的灰度变化，避免把全黑的背景也增强了。在这个实验中，有如下参考值： $E = 4.0$ ， $k_0 = 0.4$ ， $k_1 = 0.02$ ， $k_2 = 0.4$ 。

二、实验内容与步骤

2.1 界面总图设计与图像导入

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“2-直方图统计增强”后即可得到本实验的整体操作界面。界面按功能大致可划分为六个区块，如下图所示。蓝色区域是导航栏与功能区，用以进行实验的选择和读取保存图像的操作。红色部分是图像展示区，其中上部的两个文本框可以显示图像尺寸大小和路径名/文件名。绿色区域是全局统计信息计算与展示区，点击“计算”按钮后全局均值和全局标准差就会显示在相应文本框内，同时右侧的显示区显示全局直方图。黄色区域是局部统计信息计算与展示区，需要预先设定邻域大小并通过鼠标选定邻域中心位置，同时位置坐标返回到相应文本框内，也可以直接修改“x”和“y”文本框内的数值来选定位置。点击“计算”按钮后局部均值和局部标准差就会显示在相应文本框内，同时右侧的显示区显示局部直方图。紫色区域是局部增强参数设置区，可以自定义输入的主要参数，也可以通过“参考值”按钮给定内置的参考值（点击“参考值”时会同时将黄色区的“邻域大小”设为 3）。灰色区域是功能实现区，点击“原图”按钮可以恢复原图，点击“全局均衡”按钮可以显示全局直方图均衡化后的图像，点击“局部增强”按钮则可以根据设定的参数计算暗处局部增强后的图像。这样可以形成对照，帮助我们观察全局均衡与局部增强它们效果上的差异。



左上角有与实验一相同的“打开文件”和“保存图像”两个图标（预计以后的实验也都会有这两个图标）。点击“打开文件”图标选择相应图片，即可在坐标图区 axes1 导入这张图片，可以对其进行接下来的操作。顶部的可编辑文本框 edit1 和 edit5 分别显示了图片的路径名/文件名和尺寸大小。



“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。要注意在这里 uigetfile() 返回的文件名和路径名与作为 imread() 参数的路径名和文件名的顺序是相反的。后三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分别显示图片尺寸和图片路径名/文件名。

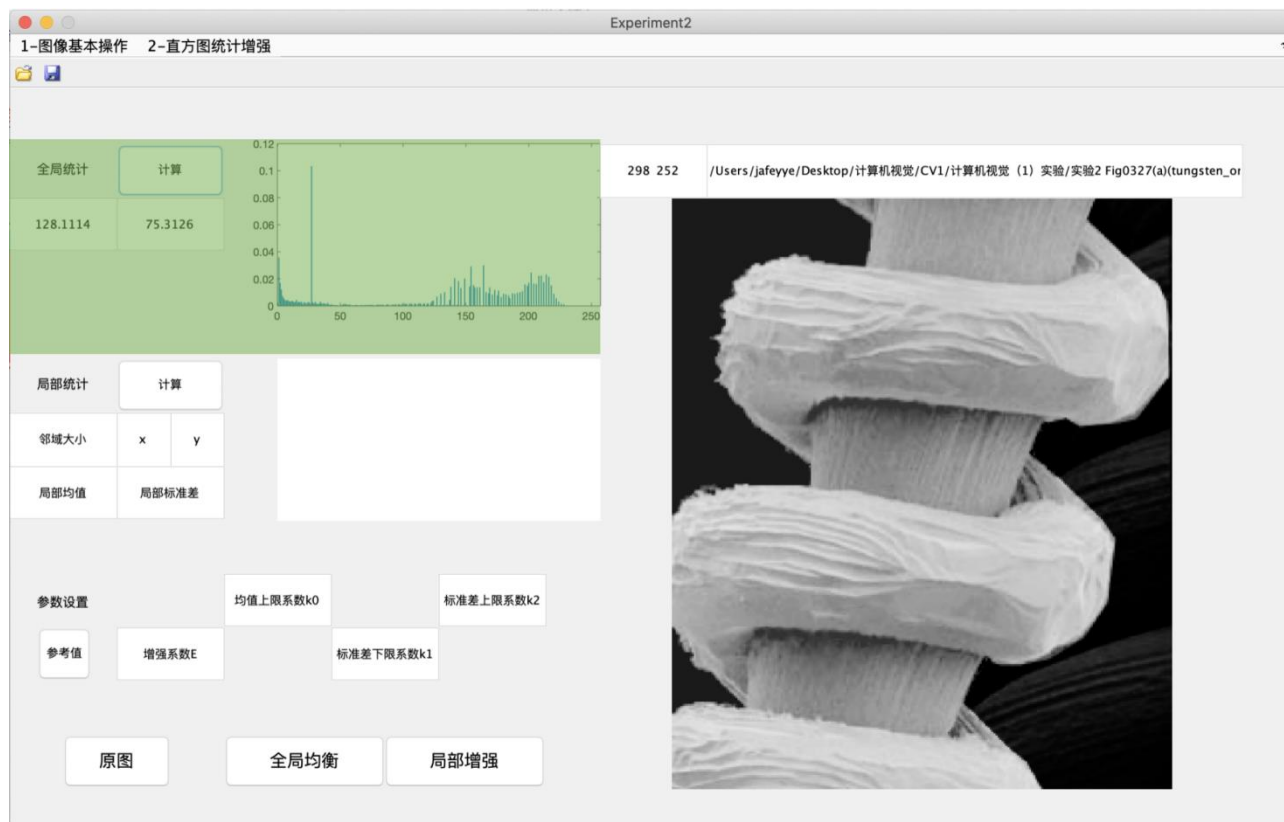
```
% -----
function uipushtool1_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename,pathname]=uigetfile({'*.bmp;*.jpg;*.png;*.jpeg;*.tif'},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

imshow(OriginalPic);
set(handles.edit1, 'String', str);
set(handles.edit5, 'String', num2str(size(OriginalPic)));
```

接下来我们就可以对导入的图像进行处理了。

2.2 全局统计（直方图、均值、方差）计算与呈现

点击绿色区域“全局统计”中的“计算”按钮，就可以计算当前坐标图区所显示的图像的全局统计信息。左边文本框显示全局均值，右边文本框显示全局标准差，右侧图像显示全局直方图。



这部分的回调函数如下图所示。首先从坐标图区获取当前图片信息（这样后期也可以计算全局均衡化或局部增强后的直方图），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 中。然后通过图像的采样值计算灰度值的均值和标准差，分别显示在文本框 edit10 和 edit12 中。（要注意的是 std() 函数的输入值首先要转化成 double 类型）

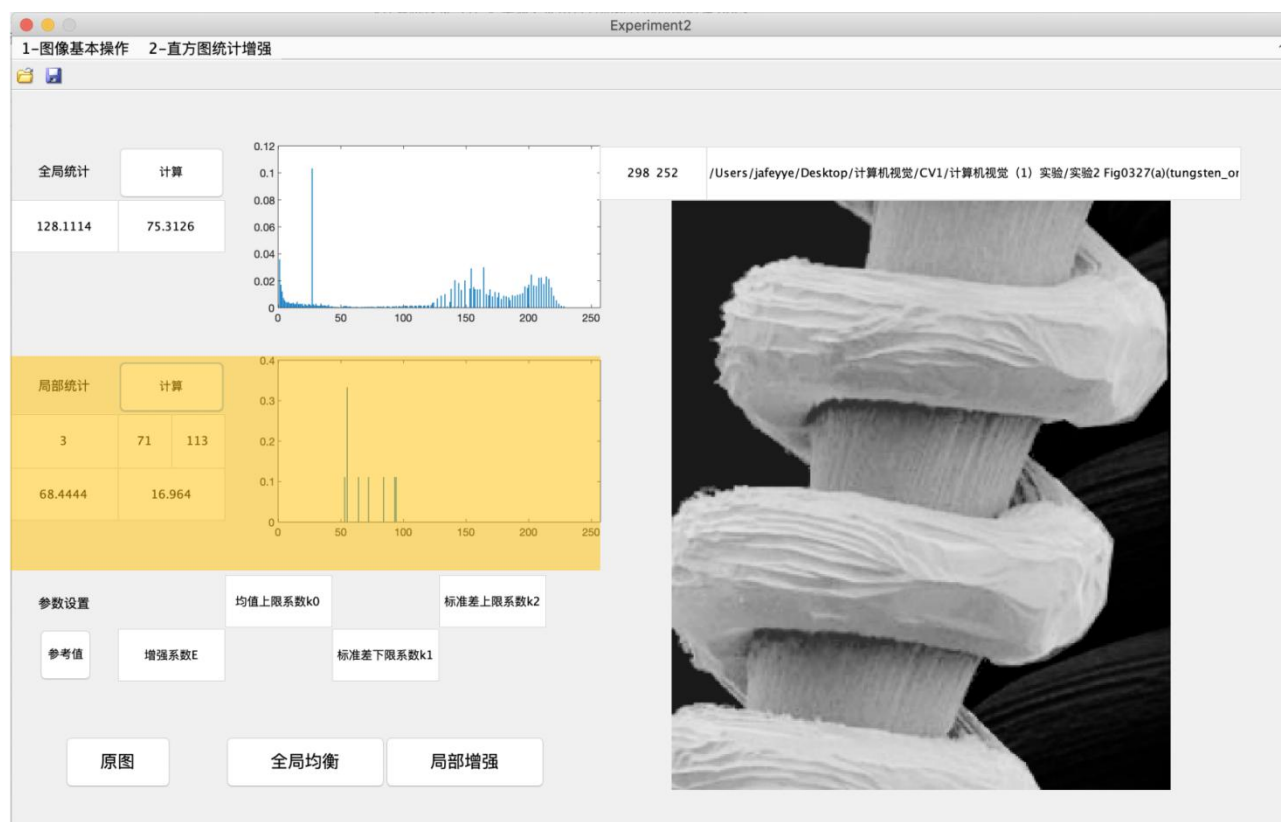
```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
img = getimage(handles.axes1);
h = imhist(img)/numel(img);
h1 = h(1:1:256);
horz=1:1:256;
axes(handles.axes2);
bar(horz,h);
set(handles.edit10,'string',num2str(mean(mean(img))));
set(handles.edit12,'string',num2str(std(double(img(:)))));
```


2.3 局部统计（直方图、均值、方差）计算与呈现

局部统计首先要确定一个小的子图范围，主要是通过设置邻域大小和选定中心位置。所以我们需要添加一个鼠标响应，在鼠标点击图中某点后返回该点坐标。该鼠标响应的回调函数如下图所示。首先以 (x,y) 记录横纵坐标，再将其分别显示在两个文本框中。

```
% --- Executes on mouse press over figure background, over a disabled or
% --- inactive control, or over an axes background.
function figure1_WindowButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[x,y]=ginput(1);
x=int64(x(1,1));
y=int64(y(1,1));
set(handles.edit3,'string',num2str(x));
set(handles.edit4,'string',num2str(y));
```

一般设定邻域大小为 3，然后点击“计算”按钮就可以计算某个小邻域内图像的局部统计信息。左边文本框显示局部均值，右边文本框显示局部标准差，右侧图像显示局部直方图。



计算局部统计信息的回调函数如下图所示。首先从文本框中获得邻域大小和邻域中心坐标的信息，然后从坐标图区 axes1 中获得全局的图像，再构建邻域

$$\left[y - \frac{\text{size} - 1}{2} : y + \frac{\text{size} - 1}{2}, x - \frac{\text{size} - 1}{2} : x + \frac{\text{size} - 1}{2} \right]$$

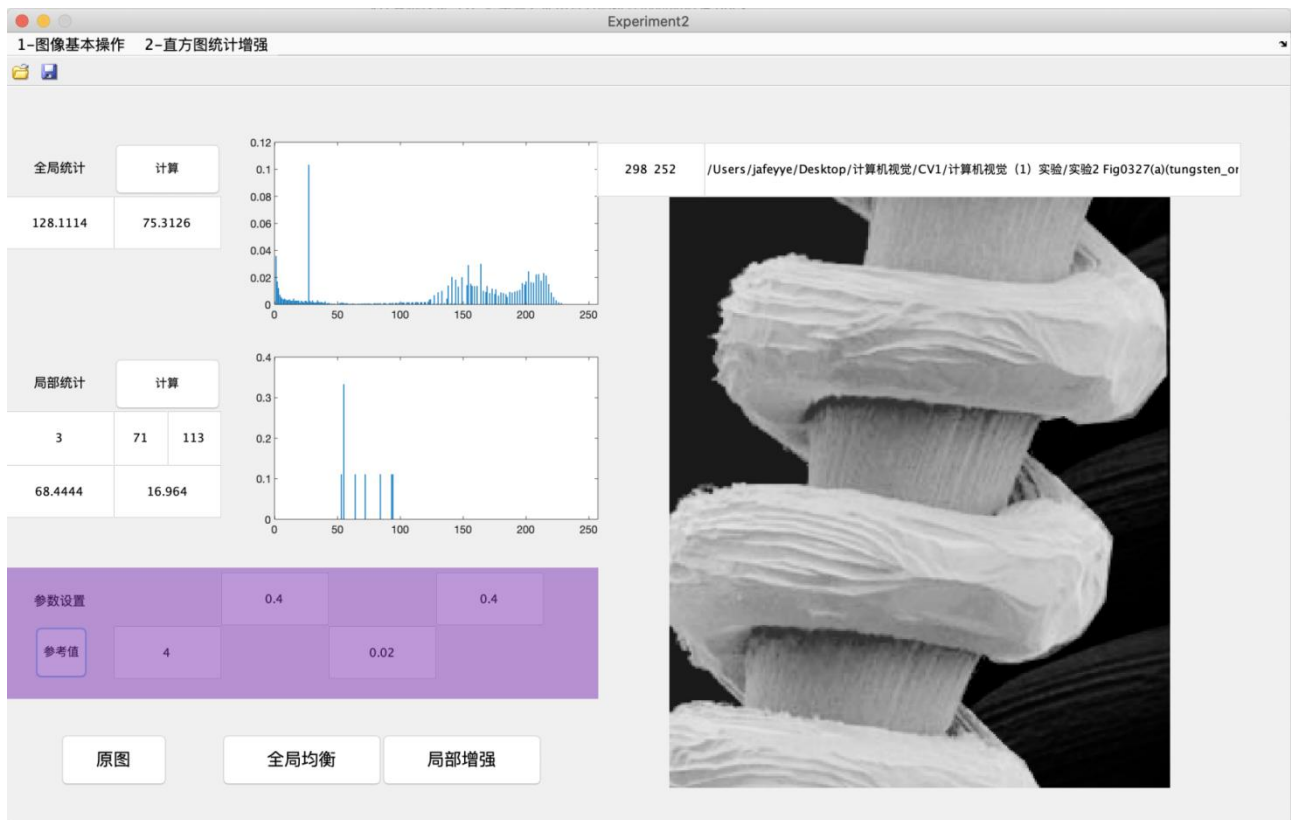
(注意鼠标相应的 (x,y) 和图像中的 (y,x) 顺序是相反的) 内的子图 f ，参照全局直方图的方法

做 f 的直方图显示在坐标图区 axes3，参照全局均值和标准差的计算方法计算 f 的均值和标准差并分别输出在文本框 edit11 和 edit13。

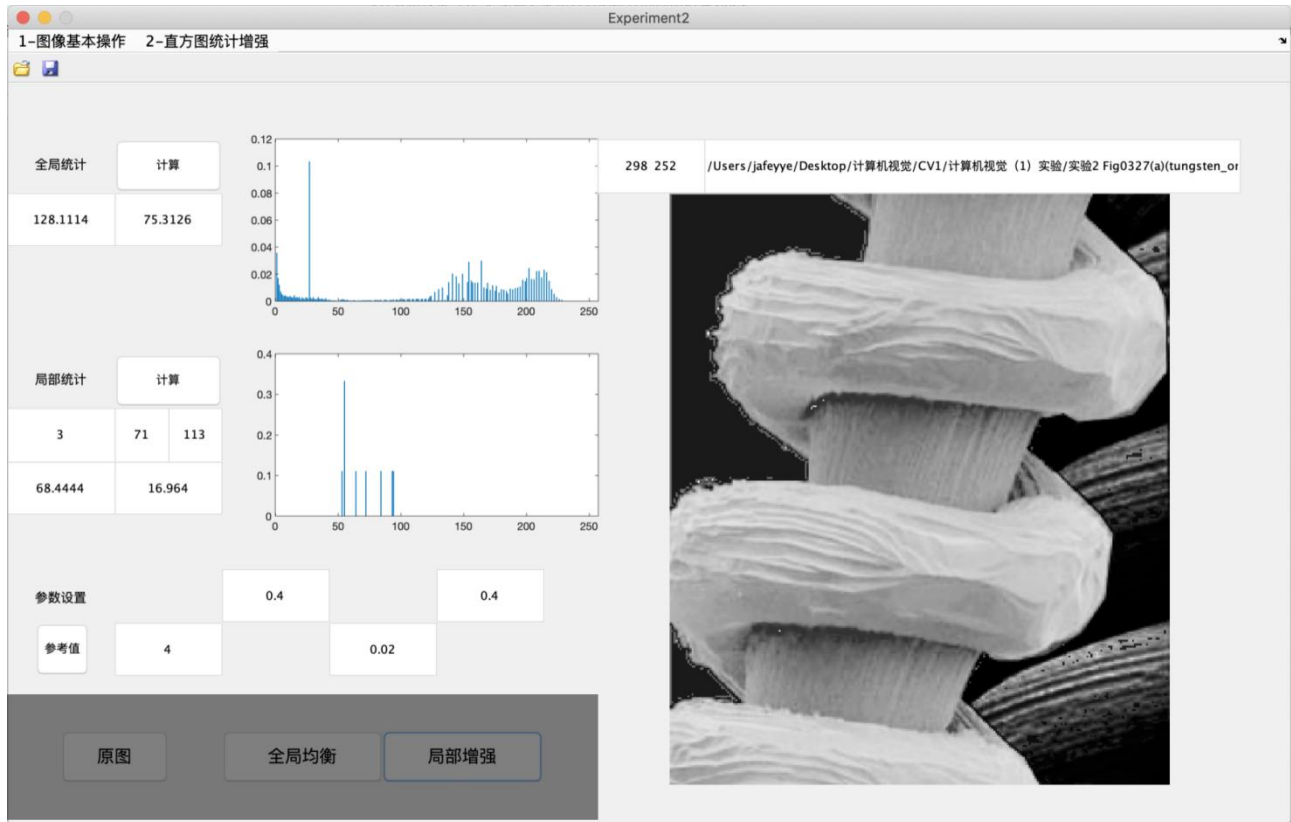
```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
size = str2num(get(handles.edit2,'string'));
x = str2num(get(handles.edit3,'string'));
y = str2num(get(handles.edit4,'string'));
img = getimage(handles.axes1);
f = img(y-(size-1)/2:y+(size-1)/2,x-(size-1)/2:x+(size-1)/2);
h = imhist(f)/numel(f);
h1 = h(1:1:256);
horz=1:1:256;
axes(handles.axes3);
bar(horz,h);
set(handles.edit11,'string',num2str(mean(mean(f))));
set(handles.edit13,'string',num2str(std(double(f(:)))));
```

2.4 暗处局部增强

首先是局部增强的参数设置部分，可键入紫色区域内的四个文本框来修改参数的大小。当然，实验中也提供了参考值 $E = 4.0$ ， $k_0 = 0.4$ ， $k_1 = 0.02$ ， $k_2 = 0.4$ ，可以通过点击“参考值”按钮进行设置（会同时将邻域大小设为 3）。



后面的程序会对紫色区域内的参数进行读取，然后进行局部增强操作。点击“局部增强”后，原图就变成了按照参数设置（下图呈现的是按照参考值局部增强后的效果图，更多参数的探索将在 3.2 中进行讨论）进行局部增强后的图像。



这里的“局部增强”回调函数如下图所示。程序的前九行，首先将设置好的参数读取进来，并且读入原图，再计算原图的全局均值和全局标准差，以备后用。第十行返回原图的尺寸。后面两行对原图进行填充，在宽度和高度的两端各填充 $\frac{s-1}{2}$ 列（行）的 0 得到 G_1 ，并创建一个 G_1 的副本 G_2 。for 循环对 G_1 中每一个非填充像素执行操作。对于每一个 G_1 中的非填充像素，首先计算它的邻域内的局部均值和局部标准差，然后进行判定，当局部均值和局部标准差与全局均值和全局标准差之间的关系满足条件 $m_{S_{xy}} \leq k_0 m_G$ and $k_1 \sigma_G \leq \sigma_{S_{xy}} \leq k_2 \sigma_G$ 时，将该像素的灰度值变为原来的 E 倍，否则不变。这一改变的操作在 G_2 中进行，避免对 G_1 直接进行修改而影响后续相邻像素的计算。执行完 for 循环后，得到的 G_2 就是增强后的图加上四周的填充元素。接下来我们只取 G_2 中非填充像素的部分，即位于

$$\left[\frac{s+1}{2} : height + \frac{s-1}{2}, \frac{s+1}{2} : width + \frac{s-1}{2} \right]$$

部分的像素点，即可得到最终的图像。最后我们将该图像显示在坐标图区 axes1 处即可。

```

% --- 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)
E = str2num(get(handles.edit6,'string'));
k0 = str2num(get(handles.edit8,'string'));
k1 = str2num(get(handles.edit7,'string'));
k2 = str2num(get(handles.edit9,'string'));
s = str2num(get(handles.edit2,'string'));
path = get(handles.edit1,'string');
img = imread(path);
mean_img = mean(mean(img));
std_img = std(double(img(:)));

[height,width] = size(img);

img_padding = padarray(img,[(s-1)/2,(s-1)/2]);
new_img = img_padding;

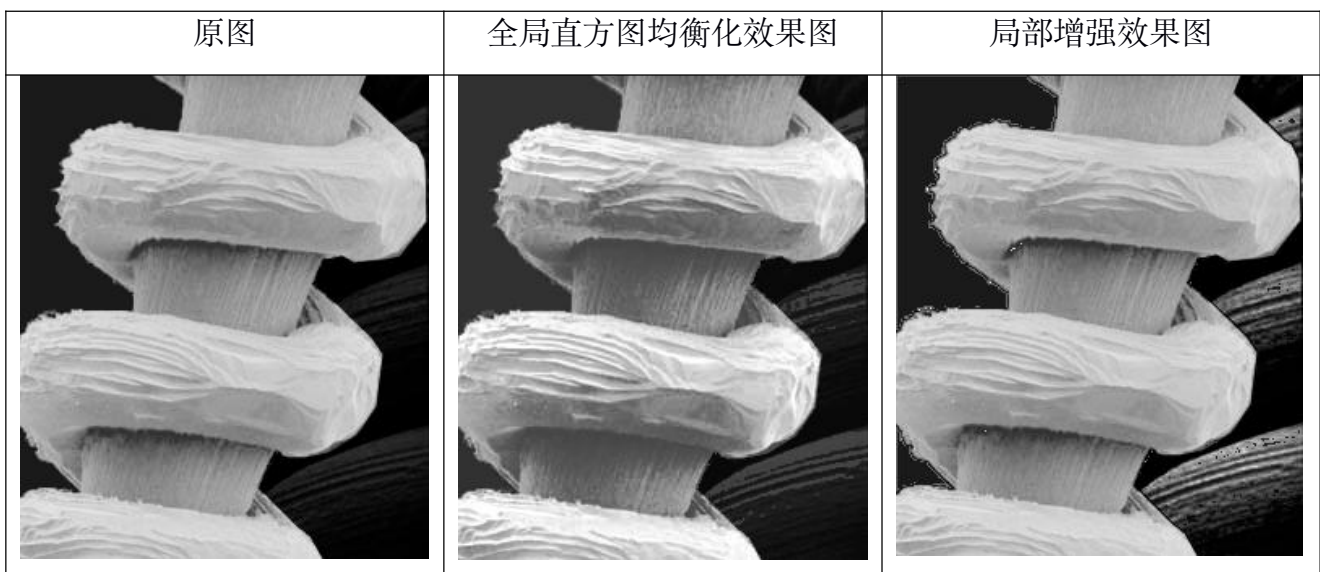
for i = (s+1)/2:width+(s-1)/2
    for j = (s+1)/2:height+(s-1)/2
        % 邻域子图
        f = img_padding(j-(s-1)/2:j+(s-1)/2,i-(s-1)/2:i+(s-1)/2);
        mean_f = mean(mean(f));
        std_f = std(double(f(:)));
        if mean_f <= k0*mean_img && std_f >= k1*std_img && std_f <= k2*std_img
            new_img(j,i) = E*img_padding(j,i);
        end
    end
end

new_img = new_img((s+1)/2:height+(s-1)/2,(s+1)/2:width+(s-1)/2);
axes(handles.axes1);
imshow(new_img);

```

三、实验结果与讨论

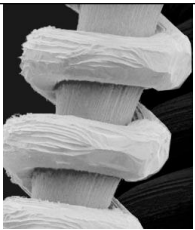
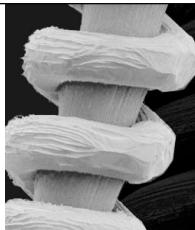



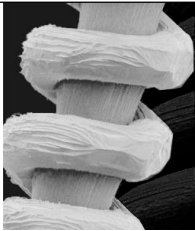
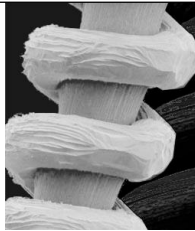
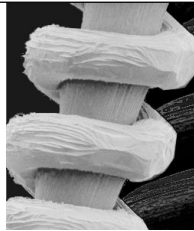
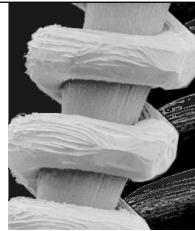

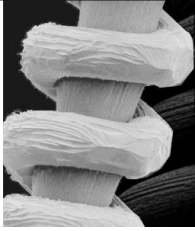




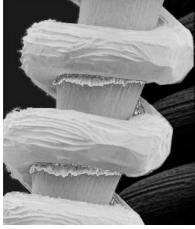
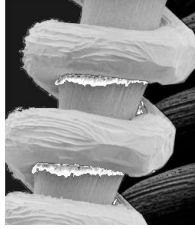
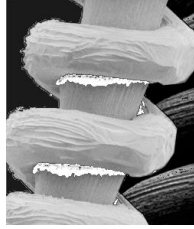
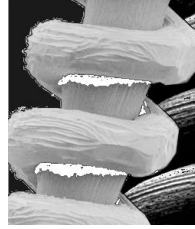
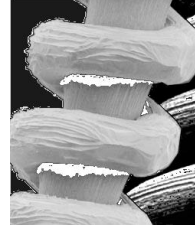
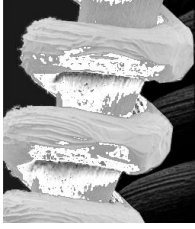
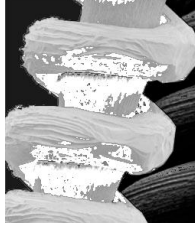
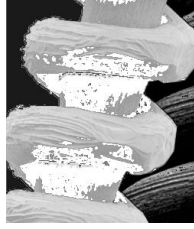
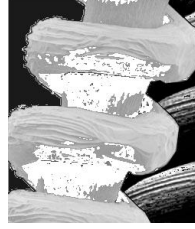
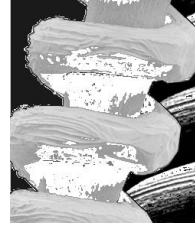
3.1 全局均衡与局部增强的效果对比



从上图不难看出，原图虽然亮处的灯丝显示清晰，但是暗处的灯丝几乎看不见。经过全局均衡后，虽然暗处的灯丝有一些显现，但是亮处的灯丝变暗了，这是我们所不希望发生的。我们希望只使得暗处变得明显，同时尽量保持亮的区域。我们发现局部增强就做得很好。一方面，它像原图一样保持明亮灯丝的亮度，另一方面，它又像全局增强一样（甚至更优）使得暗的灯丝更加明显。所以对于这一类特殊的图像处理任务，局部增强具有优异的表现。

3.2 探索参数改变对增强图片质量的影响

3.2.1 E 和 k_0 的改变对图片质量的影响




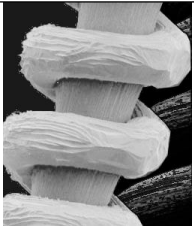






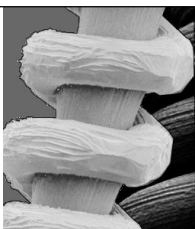










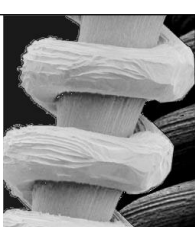



$k_1 = 0.02, k_2 = 0.4, s = 3$					
	$E = 2.0$	$E = 3.0$	$E = 4.0$	$E = 6.0$	$E = 8.0$
$k_0 = 0.1$					
$k_0 = 0.2$					
$k_0 = 0.4$					
$k_0 = 0.8$					
$k_0 = 1.2$					

通过横向的对比我们发现， E 的改变并不影响被增强的像素的集合，而只影响那些被增强的像素具体增强多少，所以我们发现随着 E 的增大，图像中一部分区域（被增强的区域）变得

越来越亮，另一部分则保持不变。所以 E 需要取一个大于 1 的适当值，使得那些需要被增强的区域被增强得恰到好处。

通过纵向的对比我们发现，当 k_0 取的比较小时，被增强的像素就越少，有可能增强不到需要的区域，而 k_0 取的比较大时，随着被增强的像素变多，可能导致不需要被增强的区域（原图中较明亮的区域）也被增强了。所以 k_0 需要取一个 0 到 1 之间适中的值，保证能够增强到较暗（但不至于是黑色）的区域同时避免增强较亮的区域。但是从书上了解到，一般 $k_0 \leq 0.5$ ，因为经验上需要增强的区域确实比全局均值的一半还要暗。

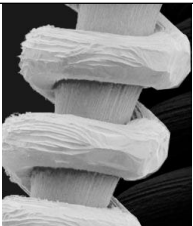


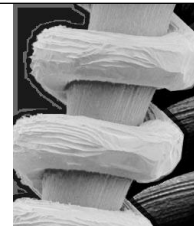
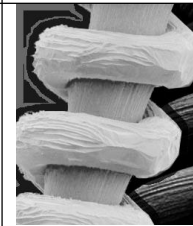
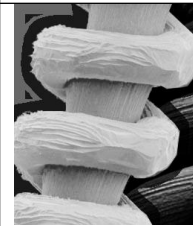
3.2.2 k_1 和 k_2 的改变对图片质量的影响

$E = 4.0, k_0 = 0.4, s = 3$					
	$k_1 = 0$	$k_1 = 0.01$	$k_1 = 0.02$	$k_1 = 0.03$	$k_1 = 0.04$
$k_2 = 0.1$					
$k_2 = 0.2$					
$k_2 = 0.4$					
$k_2 = 0.8$					
$k_2 = 1.2$					

通过横向的对比我们发现，当 $k_1 = 0$ 的时候，图像左上角当黑色区域也被增强了，这说明这部分区域应该是纯色区域，同时灰度值比较低但不等于 0（Matlab 中是从 1 开始的）。但是当 $k_1 = 0.01$ 时情况就有了明显好转。这说明 k_1 很大程度上不能等于 0。同时随着 k_1 的增大，对被增强像素的要求也越来越高，即要求该像素附近要有越高的对比度才能够被增强，而低灰度值区较高的对比度意味着该区域存在目标物体的可能性较高，就可以避免增强很多无意义的区域。但毕竟处于低灰度值区，所以对比度也不会太大，所以如果 k_1 取得太大可能就会跳过需要增强的区域，使得目标物体的部分特征被忽略。

通过纵向的对比我们发现，当其它参数不变时， k_2 取不同值带来的改变是相对较小的。 k_2 取得较小时，可能导致目标物体纹理中对比度较大的区域没有办法得到增强，但由于处于暗区，所以一般目标图像本身的对比度还是落在 $\leq k_2$ 的范围内。 k_2 取得较大时，这个条件对所有像素就不再构成限制，但是像素们同时还要满足其它条件。由此可以看出， k_2 是一个比较“松弛”的约束。

3.2.3 s 的改变对图片质量的影响

$E = 4.0, k_0 = 0.4, k_1 = 0.02, k_2 = 0.4$					
$s = 1$	$s = 3$	$s = 7$	$s = 13$	$s = 21$	$s = 31$
					

通过横向对比不难发现，当 $s = 1$ 时，整张图没有得到增强，因为邻域内的标准差为 0。当 $s = 3$ 时，增强效果最“精细”，随着 s 的增大，增强的过程越来越“粗糙”，保留的图像细节越来越少，特别是物体边缘的区域。而整幅图像的四周也出现了一圈增强的带，其原因在于 s 增大导致填充的 0 比较多，就使得整体的标准差变大。可能在其它索引从 0 开始的编程语言（如 Python）中不会出现这种情况。