图像图形编程实践

姓名：马照桅　　学号：1825103025

目录

1. 编程实现图像的二值化，分析不同的阈值对二值化图像的影响

1.1 计算方法

图像二值化需要先定义一个阈值(thresh)。在处理图象时，对比图像的像素点和设定的阈值的大小，当大于该阈值时，将像素点的灰度值设定为最大值(255)；当小于阈值时，像素点的灰度值设定为。具体计算方法如下：

$$dst[x][y]=\begin{cases}255 & if\ \ img[x][y]>thresh\\ 0 & if\ \ img[x][y]<=thresh\end{cases}$$

1.2 实验结果分析

图 1.1 原图的灰度图

图 1.2 阈值为 50 时的二值化图像

图 1.3 阈值为 150 时的二值化图像

对比图 1.2 和图 1.3，不难发现阈值设置的越大，可以保留的整体信息，但是会丢失一些细节，如头发丝的分叉。阈值较低时，虽然会保留一些细节，但是会丢失大部分的信息，如面部的鼻子和嘴巴等。这一特性在下一个例子中更加明显。

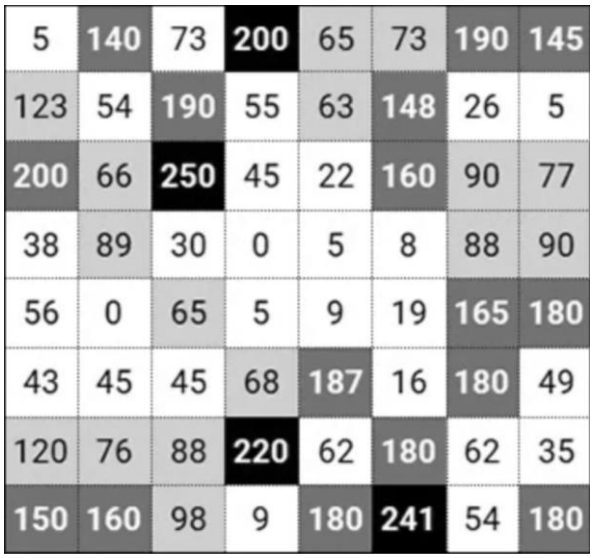

图 1.4 原图

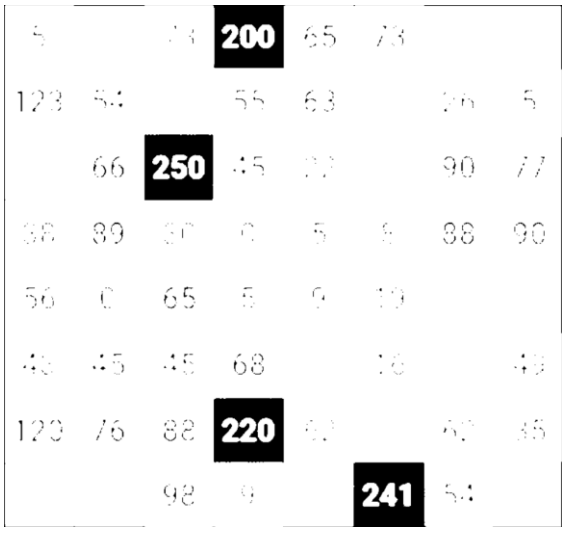

图 1.5 阈值设置为 50 时的二值化图形

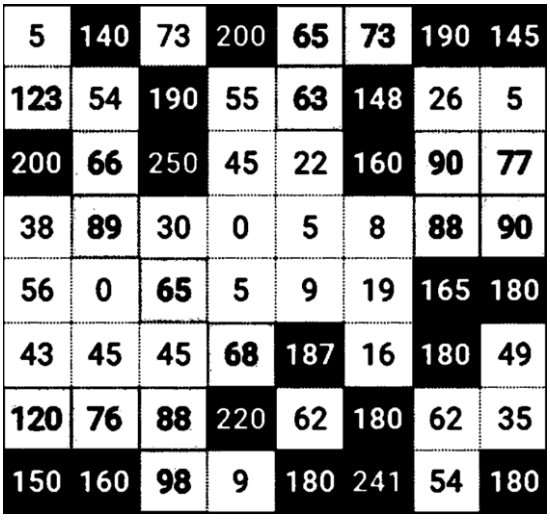

图 1.6 阈值设置为 150 时的二值化图像

虽然低阈值的图保留了一些黑色的细节，但是丢失了大部分的数字细节，而高阈值的图很好的保留了数字的信息，同时保留了深色的细节。

2. 编程实现图像的基本运算：两幅图像相加和相减，并分析这两种运算的作用

2.1 计算方法

对图像的运算都是基于像素点的运算，加法运算就是将对应的像素点的灰度值或者 RGB 的值进行加法运算。减法同理。具体的计算方法如下：

$$dst = img1 + img2$$

$$dst = img1 - img2$$

2.2 实验结果分析

因为需要对图像的像素点进行运算，所以在读入图像后，需要做的第一步是将两幅图的大小统一。

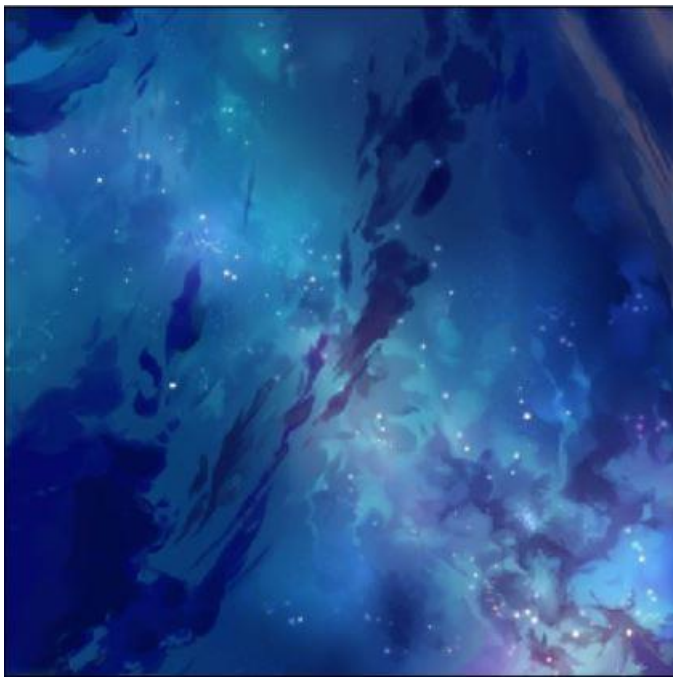

图 2.1 原图 1

图 2.2 原图 2

图 2.3 灰度图的加法运算

图 2.4 灰度图的减法

图 2.5 彩色图的加法

图 2.6 彩色图像的减法

图像的加法有两种用法，一种就是减少图像采集时的噪声，对同一张图重复采样多次，进行相加，然后取平均值，可以实现噪声的消除。另一种用法就是将两张图像叠加起来，做特效处理。

图像的减法通常用于查找图像的差异，如医学上血管的影像和造影后的血管影像比对就可以看出血液流动的情况。同样，减法也可以用在图像的特殊处理上。

3. 编程实现图像拉普拉斯锐化

3.1 计算方法

拉普拉斯锐化图像是根据图像某个像素的周围像素到此像素的突变程度有关，也就是说它的依据是图像像素的变化程度。我们知道，一个函数的一阶微分描述了函数图像是朝哪里变化的，即增长或者降低；而二阶微分描述的则是图像变化的速度，急剧增长下降还是平缓的增长下降。那么据此我们可以猜测出依据二阶微分能够找到图像的色素的过渡程度，例如白色到黑色的过渡就是比较急剧的。

当邻域中心像素灰度低于它所在的领域内其它像素的平均灰度时，此中心像素的灰度应被进一步降低，当邻域中心像素灰度高于它所在的邻域内其它像素的平均灰度时，此中心像素的灰度应被进一步提高，以此实现图像的锐化处理。

我们首先使用一阶微分确定图像边缘是否存在，具体计算过程如下：

$$\frac{\partial f}{\partial x} = f(x,y) - f(x-1,y)$$

$$\frac{\partial f}{\partial y} = f(x,y) - f(x,y-1)$$

$$\nabla f = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} = 2f(x,y) - f(x-1,y) - f(x,y-1)$$

接着使用二阶微分，即拉普拉斯算子确定边缘的位置。即：

$$\nabla^2 f = 4f(x,y) - f(x-1,y) - f(x,y-1) - f(x+1,y) - f(x,y+1)$$

就可以得到如下的一个模板矩阵，也被称为四领域矩阵：

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$$

3.2 实验结果分析

图 3.1 原图

图 3.2 四领域滤波

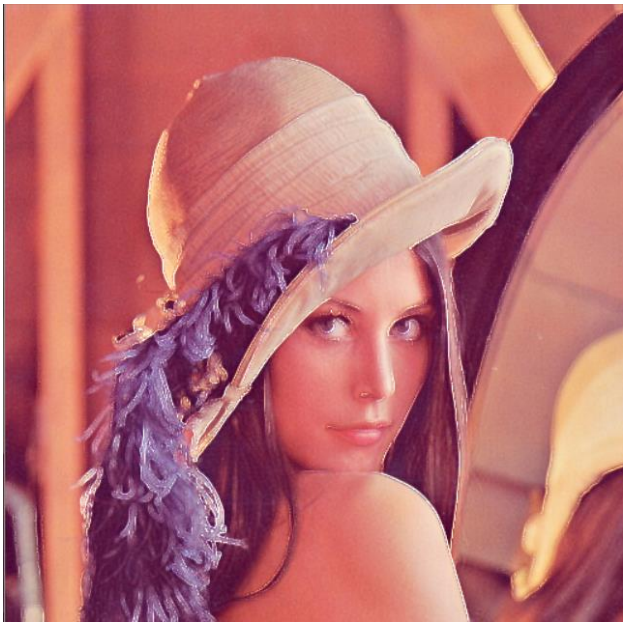

图 3.3 八领域滤波

使用卷积运算，将拉普拉斯算子和原图进行卷积运算，实现边缘的检测。再将处理后的边缘图像和原图进行相加运算，得到锐化后的图像。

4. 编程实现 Canny 图像边缘检测方法，对不同参数进行验证

4.1 计算方法

首先使用高斯滤波，将图片的噪声去除。然后计算图像的梯度幅值和方向。

运用 Sobel 滤波器的步骤来操作。使用如下的运算公式计算梯度幅值和方向：

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * img$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * img$$

$$G = \sqrt{G_x^{\,2} + G_y^{\,2}}$$

$$\theta = \arctan(\frac{G_y}{G_x})$$

计算得出图像的边缘像素，然后使用滞后阈值。当某一像素的值超过高阈值时，该像素点就被保留为边缘像素。当某一像素的值低于低阈值，该像素被排除。若某一像素的值在两个阈值之间，该像素仅仅在连接到一个高于高阈值的像素时被保留。

4.2 实验结果分析

图 4.1 canny 图像边缘检测

高阈值是将要提取轮廓的物体与背景区分开来，就像阈值分割的参数一样，是决定目标与背景对比度的；低阈值是用来平滑边缘的轮廓，有时高阈值设置太大了，可能边缘轮廓不连续或者不够平滑，通过低阈值来平滑轮廓线，或者使不

连续的部分连接起来。

5. 编程实现彩色图像的切割。设计一个通用的方法，基于 RGB 空间，从下列的图像中，分割出指定的目标

5.1 计算方法

截取一个草莓区域，将这个区域的 RGB 向量的均值作为中心点。计算这个区域的 RGB 向量的方差作为阈值。

分别使用欧氏距离和绝对值距离计算图像的像素和中心点的距离，判断这个距离和阈值的关系，若大于阈值，则舍弃，若小于阈值，则保留。即:

$$(Z_R - a_R)^2 + (Z_G - a_G)^2 + (Z_B - a_B)^2 < thresh^2$$

5.2 实验结果分析

图 5.1 原图

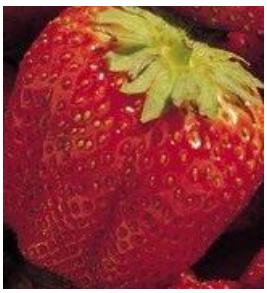

图 5.2 截取的一个草莓区域

图 5.3 欧氏距离的图像切割

图 5.4 绝对值距离的图像切割

由于欧氏距离需要进行平方等较复杂的运算，所以使用绝对值距离在计算上会比欧氏距离方便许多。对比图 5.3 和图 5.4，可以发现，欧氏距离可以将草莓

的主题切割出来，而不会保留叶子，但是绝对值距离则会保留一些草莓叶子。欧氏距离可以更好的切割图像。

附录：

本次实践的所有代码和所用的图片都存在以下的 Github 中

https://github.com/Damage0413/shu_practice