数字图像处理实验报告

代码详见 https://github.com/CSY-tvgo/Digital-Image-Processing-Tutorial

1. 图像的归一化直方图

原理

离散图像用直方图表示该图像中不同灰度级像素的出现概率。归一化灰度直方图的横坐标是图像的各灰度级，纵坐标是各个灰度出现的频率。

算法

对于 8 位的灰度图而言：

- 1 创建一个大小为 256 的数组$hist[256]$，其 0~255 的索引值表示灰度级
- 2 遍历每个像素，若当前遍历至的像素灰度为i，则$hist[i]$加一
- 3 将所有次数除以总的像素数量，即可获得每个灰度级出现的频率

处理结果

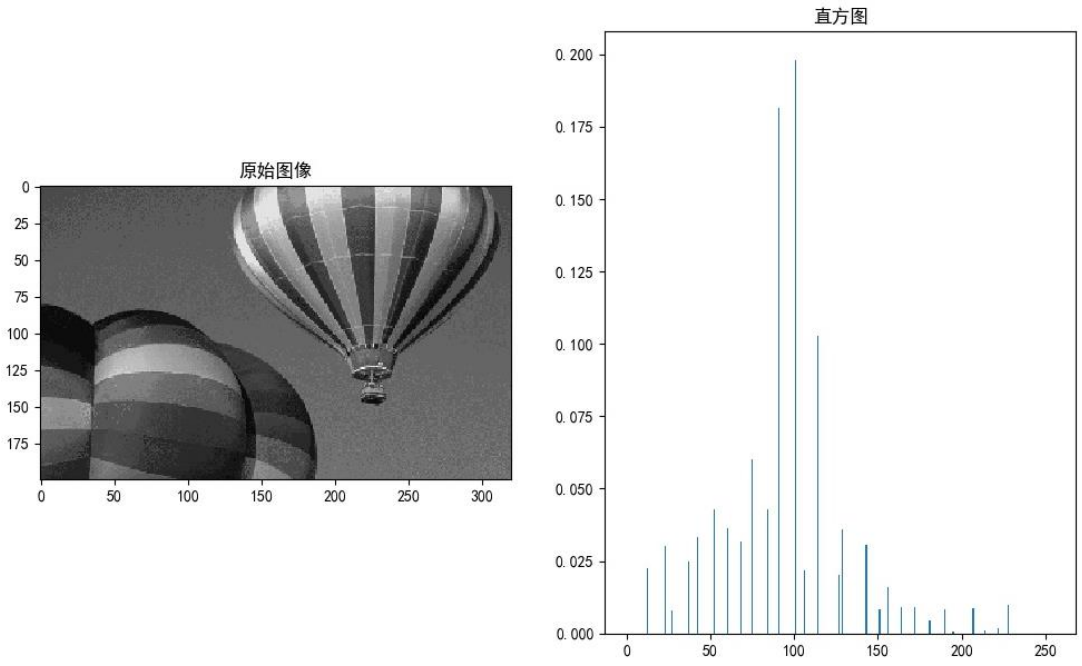

Figure 1 求归一化直方图

结果分析

显示的直方图能够反映原图灰度级的分布情况。

2. 图像的线性变换、图像的直方图均衡化

原理

（1） 线性变换

设原始图像像素灰度f的范围为[a, b]，线性变换后图像像素灰度g的范围为[a′, b′]，则灰度g与灰度f之间的关系为

$$g = a' + \frac{b' - a'}{b - a}(f - a)$$

（2） 直方图均衡化

直方图均衡化是使图像的直方图变为均匀分布的直方图，从而达到增强的效果。对于灰度级为离散值的数字图像，用频率代表概率，则r_k至s_k的变换函数$T(r_k)$的离散形式可表示为

$$s_k = T(r_k) = \sum_{i=0}^{k} p_r(r_i) = \sum_{i=0}^{k} \frac{n_i}{n}$$

式中$p_r(r_i)$指在原图中灰度级为r_i的像素出现的概率，$0 \le r_k \le 1$, $k = 0,1,2,\dots,L-1$，可见，均衡后各像素的灰度值s_k可直接由原图像的直方图算出。

算法

（1） 线性变换

- 1 输入变换后的灰度范围a'和b'
- 2 求原图的最小灰度值a和最大灰度值b
- 3 遍历每个像素，直接计算新图每个像素的灰度值，

例如，第r行第c列的像素变换后的灰度值为

$$g[r][c] = round\left(a' + \frac{b' - a'}{b - a}(f[r][c] - a)\right)$$

其中$round$函数表示四舍五入

（2） 直方图均衡化

- 1 求原图各个灰度级出现的频率$hist[i]$
- 2 计算累积概率

$$prob\_sum[k] = \sum_{i=0}^{k} hist[i]$$

- 3 计算变换前后各个灰度级的映射关系

$$T[k] = round(prob\_sum[k] * 255)$$

- 4 对每个像素进行变换

处理结果

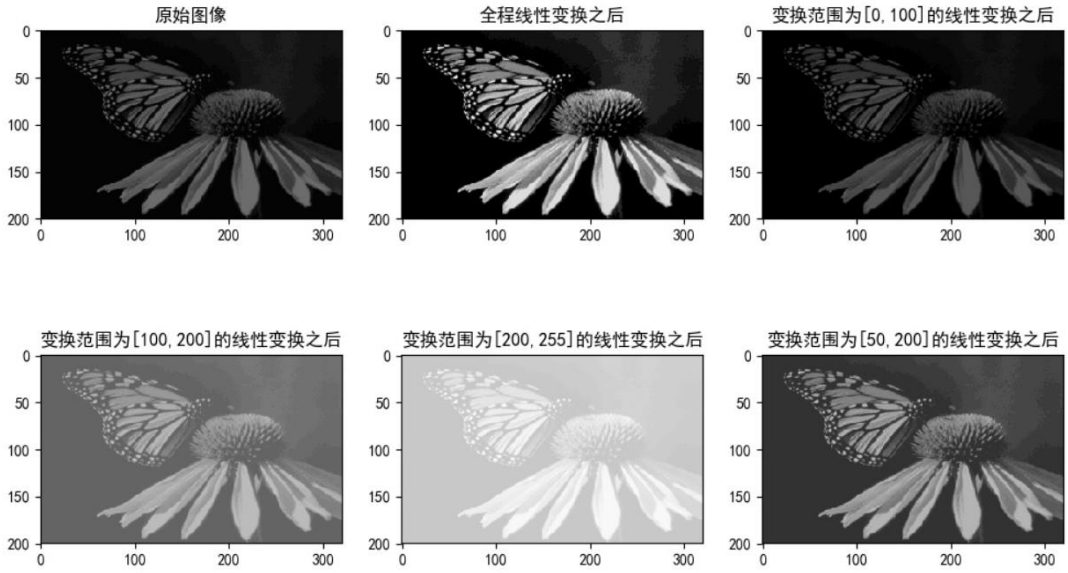

Figure 2 **线性变换**

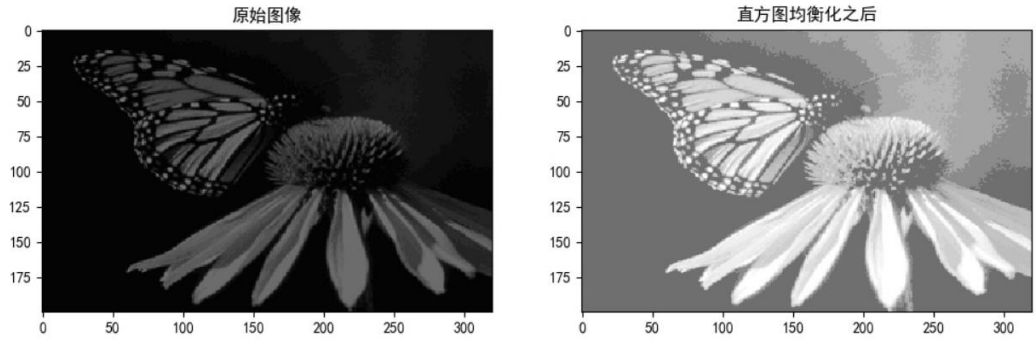

Figure 3 **直方图均衡化（灰度图）**

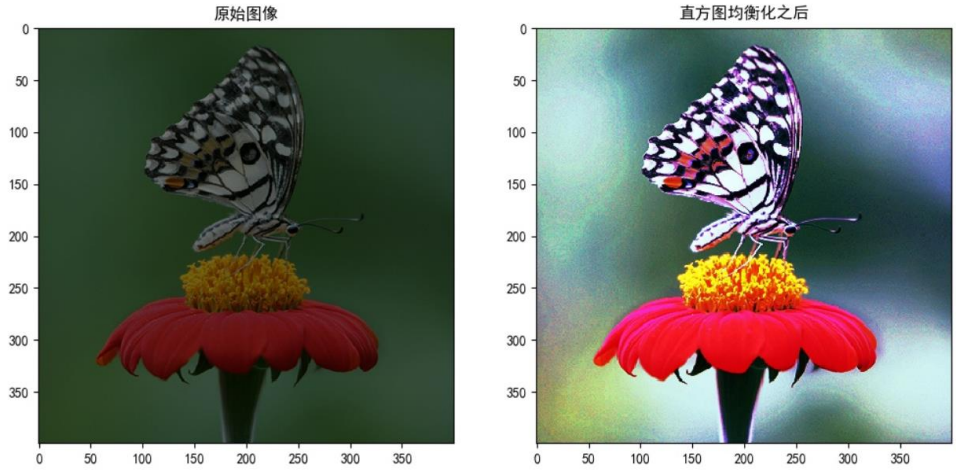

Figure 4 **直方图均衡化（彩色图）**

结果分析

实验效果与预想相同。

3. 图像平滑消噪处理（邻域平均法）

原理

通过邻域平均法，可以一定程度上消除某些噪声。原图像$f(x,y)$邻域S的均值$g(x,y)$可以表示为

$$g(x,y)=\frac{1}{M}\sum_{(x,y)\in S} f(x,y)$$

式中，M为邻域S中像素的个数。

由此式可知，邻域平均法是以图像模糊为代价来换取噪声的降低的。

算法

以 3*3 均值模板为例

• 1 定义一个处理模板(mask, or template)

$$H_0=\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

• 2 根据模板的长宽，将原图在四周拓展一圈

• 3 遍历每个像素，根据下式计算新的灰度值

$$g(x,y)=\frac{1}{M}\sum_{(x,y)\in S} f(x,y)$$

处理结果

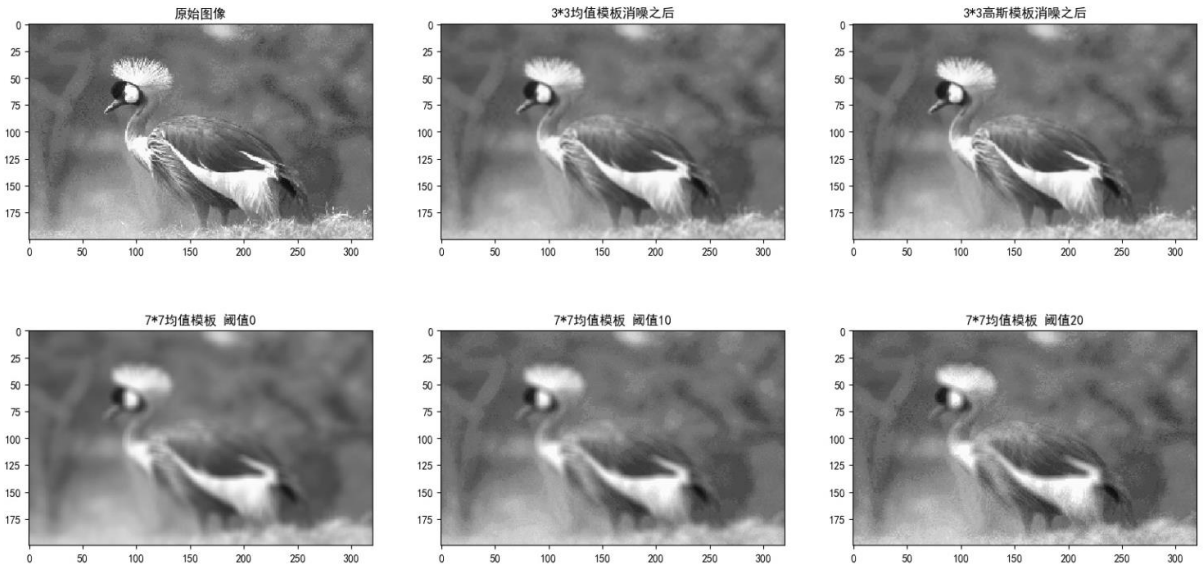

Figure 5 邻域平均法平滑

结果分析

实验效果与预想相同。

4. 图像锐化处理（以拉普拉斯算子为例）

原理

图像锐化处理（sharpening）的目的是加强图像中景物的边缘和轮廓，使模糊图像变得更清晰。这里以通过拉普拉斯算子锐化图像为例。

拉普拉斯算子（Laplacian）是具有各向同性的二阶微分算子。一个连续的二元函数$f(x,y)$，其拉普拉斯运算定义为

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

通过拉普拉斯算子锐化图像可以由下式来表示

$$\mathrm{g} = f + k\tau \cdot \nabla^2 f$$

式中，f和g分别是锐化前后的图像，$k\tau$为常系数，可以表示锐化强度。

对于数字图像，通过拉普拉斯算子进行锐化的过程可以表示为

$$\mathrm{g}(i,j) = f(i,j) + k\tau[4f(i,j) - f(i+1,j) - f(i-1,j) - f(i,j+1) - f(i,j-1)]$$

也可以表示为卷积的形式，即

$$\mathrm{g} = f + k\tau \cdot f * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

算法

• 1 定义一个处理模板

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

• 2 根据模板的长宽，将原图在四周拓展一圈

• 3 遍历每个像素，根据下式计算新的灰度值

$$\mathrm{g}(i,j) = f(i,j) + k\tau[4f(i,j) - f(i+1,j) - f(i-1,j) - f(i,j+1) - f(i,j-1)]$$

处理结果

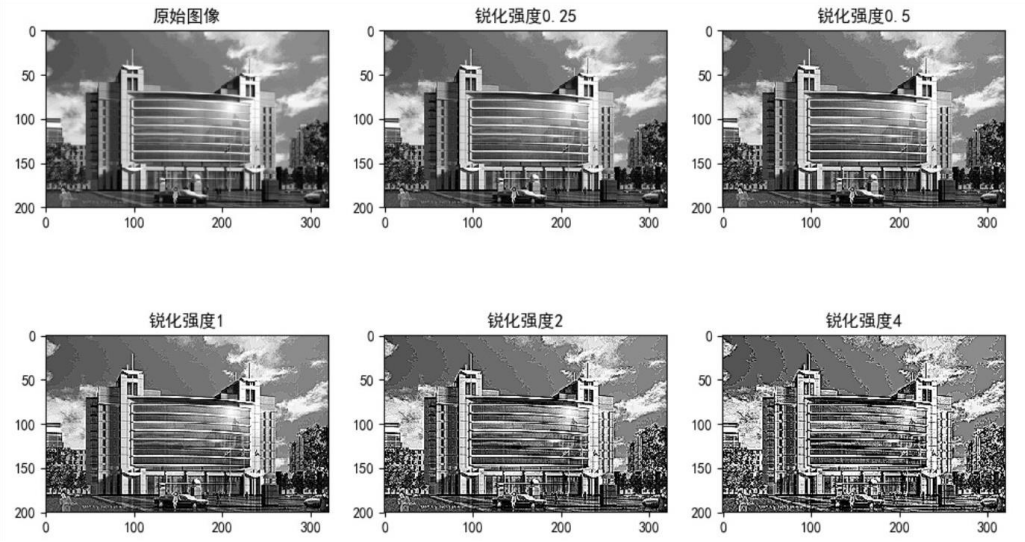

Figure 6 通过拉普拉斯算子锐化灰度图

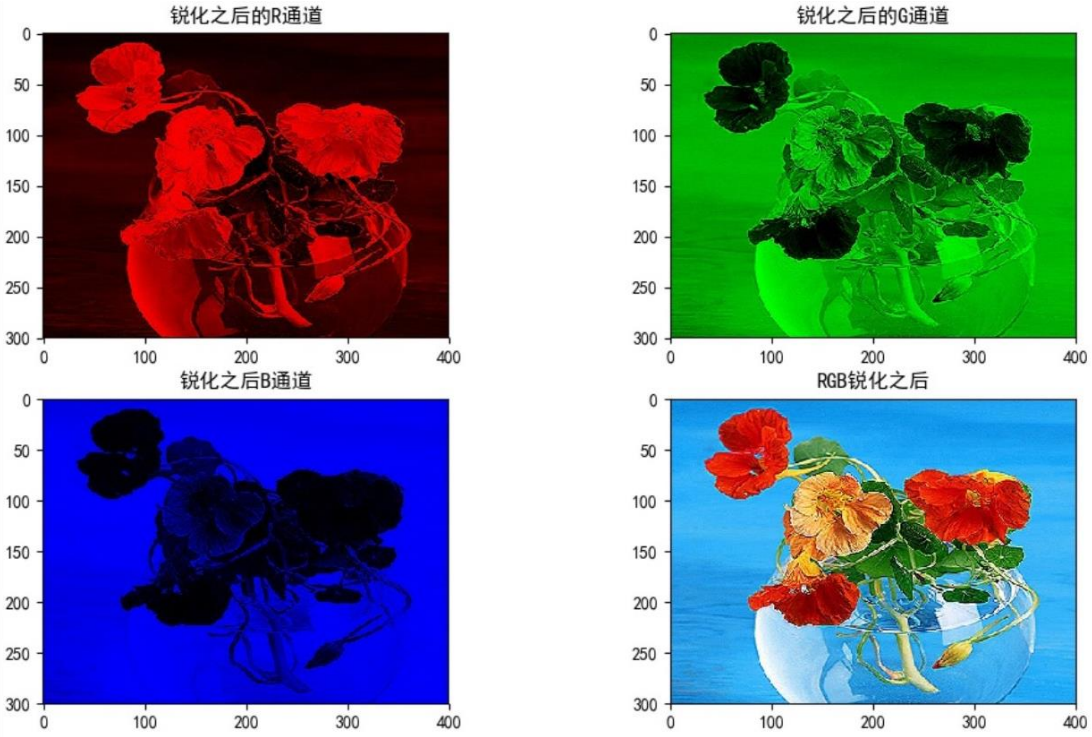

Figure 7 **通过拉普拉斯算子锐化彩色图**

结果分析

锐化强度越大，锐化效果越明显，但是同时也会发现出现了更多噪点。

5. 中值滤波处理

原理

使用邻域平均法消噪的同时，图像上很多边界也会变得模糊，中值滤波是一种消噪的同时还能尽可能保留边缘信息的方法。选一个长宽均为奇数的滑动窗口W，将这个窗口在图像上扫描，把该窗口中所含的像素点按灰度级的升序（或降序）排列，取位于中间的灰度值来取代该点的灰度值。即

$$\mathrm{g}(m,n) = Med\{f(m-k,n-l),(k,l) \in W\}$$

算法

以 3*3 十字中值滤波为例：

- 1 定义一个处理模板$H = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
- 2 根据模板的长宽，将原图在四周拓展一圈
- 3 遍历每个像素，例如，在处理第r行第c列的像素$f(r,c)$时，通过滑动窗口能获得$f(r,c), f(r-1,c), f(r+1,c), f(r,c-1), f(r,c+1)$ 这 5 个值
- 4 将滑动窗口取得的值进行快速排序
- 5 取中值，该值即为变换后的$\mathrm{g(r,c)}$

处理结果

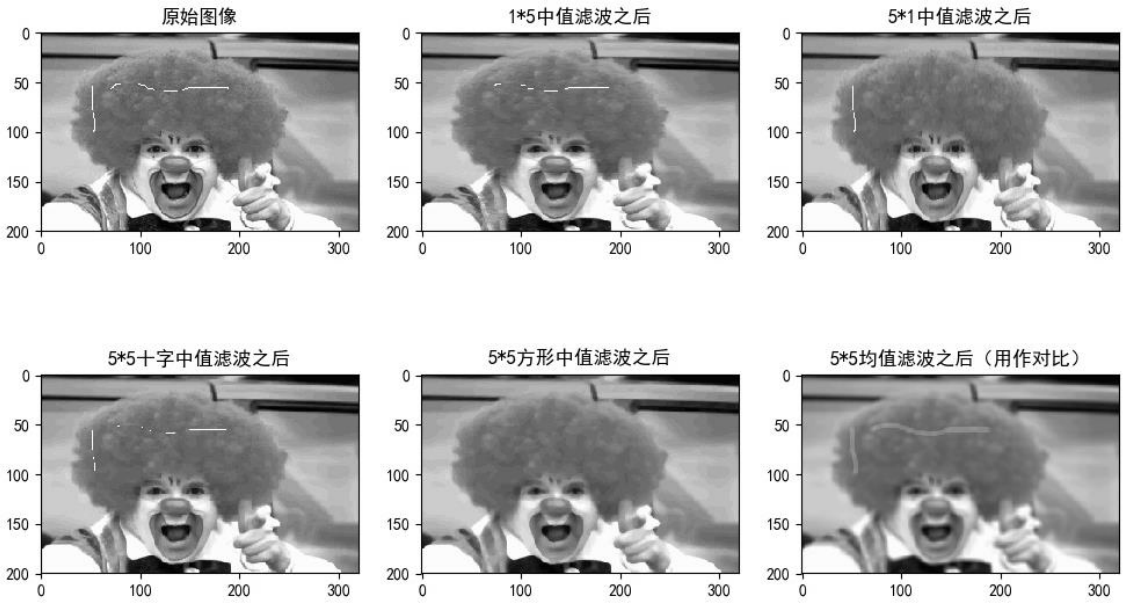

Figure 8 **中值滤波**

结果分析

由处理效果可见，使用 1*5 模板能够有效去除竖向裂纹，使用 5*1 模板能够有效去除横向裂纹，使用 5*5 方形模板能够去除横向和竖向的裂纹，而使用均值滤波无法去除裂纹。

6. 伪彩色增强（简单映射）

原理

伪彩色增强（Pseudo Color）是将灰度图变换为彩色图像，从而把人眼不能区分的微小的灰度差别显示为明显的色彩差异。伪彩色增强有简单映射、频率域伪彩色增强等多种方法。

例如，简单映射即为设置一个表达式，直接根据原灰度值算出 R、G、B 三个通道的值。

算法

以映射关系如下图的简单映射为例：

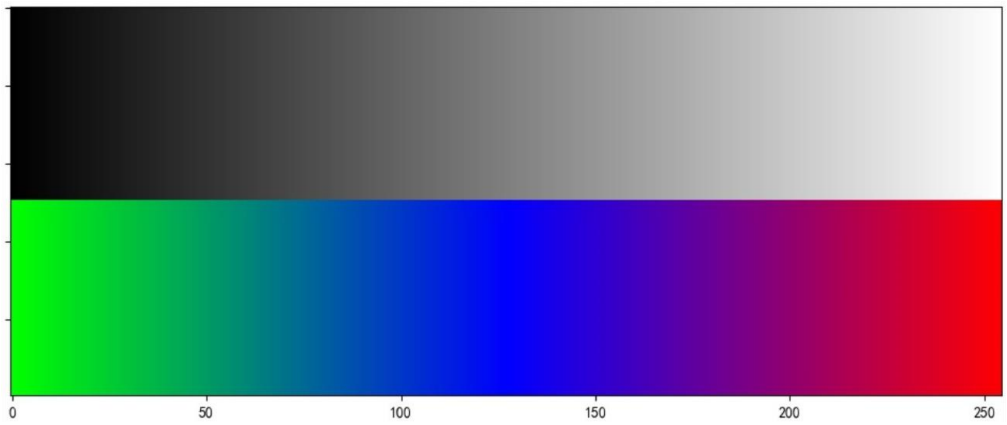

Figure 9 简单映射的映射关系

• 1 假设原灰度图f的高度为h，宽度为w，新建一个大小为$h*w*3$的全零矩阵用以存储伪彩色增强后的图g

• 2 遍历每个像素，例如，在处理第r行第c列的像素$f(r,c)$时，直接计算该像素映射至的 RGB 值

R 通道：

$$\mathrm{g(r,c,0)} = suppress(f(r,c) \times 2 - 255)$$

G 通道：

$$\mathrm{g(r,c,1)} = suppress(255 - f(r,c) \times 2)$$

B 通道：

$$\mathrm{g(r,c,2)} = suppress(255 - |f(r,c) - 128| \times 2)$$

其中$suppress$函数表示把值抑制在 0~255 的范围内。

处理结果

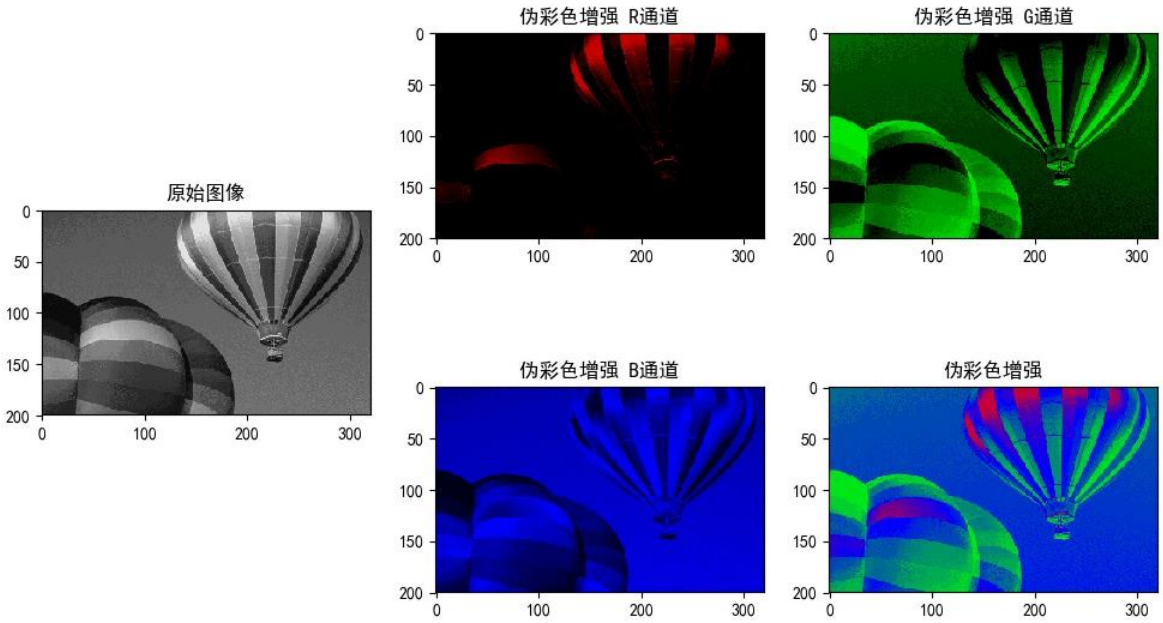

Figure 10 **通过简单映射法进行伪彩色增强**

结果分析

实验效果与预想相同。

7. 边缘检测（以拉普拉斯算子为例）

原理

目标的边缘一般表现为灰度的特变。对于人类的视觉感知，图像边缘对理解内容起到关键作用。边缘检测可以通过一阶微分算子（也称梯度算子，如：Roberts 算子、Prewitt 算子、Sobel 算子等）、也可以通过二阶微分算子（如：Laplacian 算子、Laplacian of Gaussian 算子、Canny 算子等）来实现。

这里以 Laplacian 算子为例，拉普拉斯算子（Laplacian）是具有各向同性的二阶微分算子。一个连续的二元函数$f(x,y)$，其拉普拉斯运算定义为

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

对于数字图像，可以通过拉普拉斯算子遍历每个像素，对每个像素计算一个临时值

$$temp(i,j) = 4f(i,j) - f(i+1,j) - f(i-1,j) - f(i,j+1) - f(i,j-1)$$

然后设定一个阈值th，如果某个像素对应的临时值大于该阈值，即表示该像素为边缘。

算法

• 1 定义一个处理模板

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

• 2 根据模板的长宽，将原图在四周拓展一圈

• 3 遍历每个像素，根据下式计算每个像素对应的临时值

$$temp(i,j) = 4f(i,j) - f(i+1,j) - f(i-1,j) - f(i,j+1) - f(i,j-1)$$

• 4 设定一个阈值th，如果$temp(i,j) \geq th$，则在新图中将对应的像素灰度值置为 0，如果$temp(i,j) < th$，则在新图中将对应的像素灰度值置为 255

处理结果

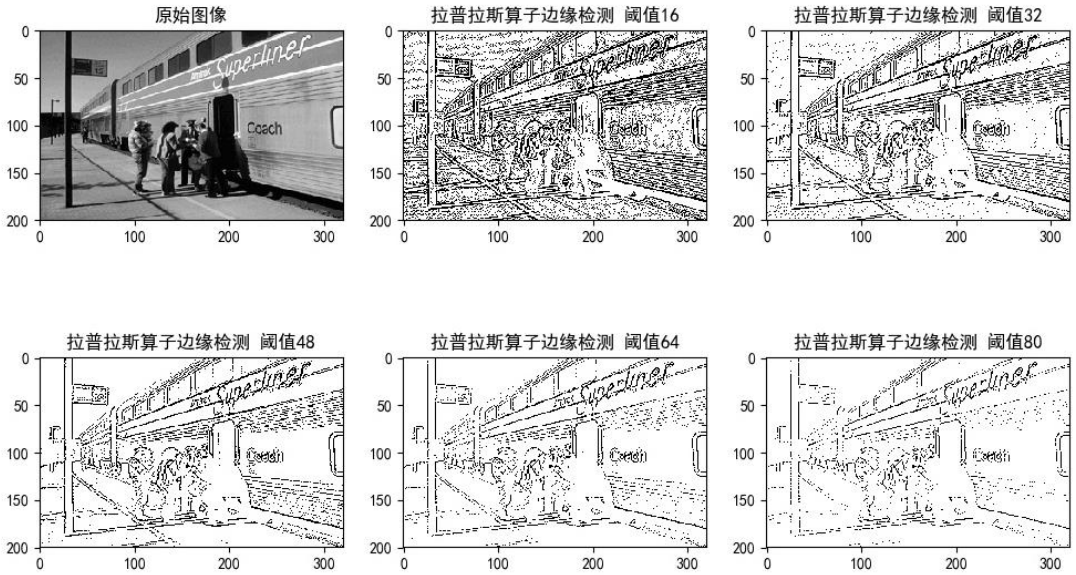

Figure 11 **通过拉普拉斯算子进行边缘检测**

结果分析

实验效果与预想相同。