

# 信号建模与 算法实践



图像处理-01

Xu Weiye

2022-09-22

## Content

1	Introduction	2
2	process	2
2.1	Setup	2
2.2	Basic Operation	3
2.3	Smooth Filter	4
2.4	Edge Detection Filter	6

## 1 Introduction

本次实验进行图像处理的基本操作和底层处理算法，了解数字图像的基本形式和基本滤波操作，具体包括：

- 图像读取、写入、平移、旋转、缩放等操作；
- 图像滤波：平滑（均值滤波、高斯滤波、中值滤波、双边滤波）、边缘提取（Sobel、Canny、DOG、LOG）；
- 图像特征：灰度直方图、颜色直方图（RGB 空间、HSV 空间）、方向梯度直方图（Histogram Of Gradient）。

## 2 process

### 2.1 Setup

读取图像：使用的是 OpenCV 默认的读取库，得到的读取图片是

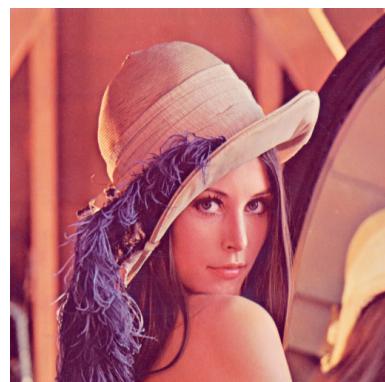


Figure 1: 原图

然后通过不同的算法添加噪声

---

**Algorithm 1** Add Guass Noise

**Require:**  $Image, \mu, \sigma$

$Noise \leftarrow random(\mu, \sigma)$

$Image \leftarrow Noise + Image$

---

---

**Algorithm 2** Add Possion Noise

**Require:**  $Image$

$Noise \leftarrow poisson(Image)$

$Image \leftarrow Noise + Image$

---

**Algorithm 3** Add Salt-and-pepper noise

---

**Require:**  $Image, p$

```

for all  $pixel \in Image$  do
     $model \leftarrow random(0, 1)$ 
    if  $model \leq p$  then
         $pixel \leftarrow 0$ 
    else if  $model \geq 1 - p$  then
         $pixel \leftarrow 255$ 
    else
         $pixel \leftarrow pixel$ 

```

---



(a) 添加高斯噪声

(b) 添加柏松噪声

(c) 添加椒盐噪声

**Figure 2:** 添加噪声

## 2.2 Basic Operation

对图像进行旋转操作分为三步

第一步将图像中心平移到原点,  $x_m, y_m$  分别表示中心点坐标

$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_m & -y_m & 1 \end{bmatrix}$$

再将变换后的图片按照原点逆时针旋转  $\theta$

$$\begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

之前由于需要旋转操作将图片的中心平移到了原点, 现在要反变换回去

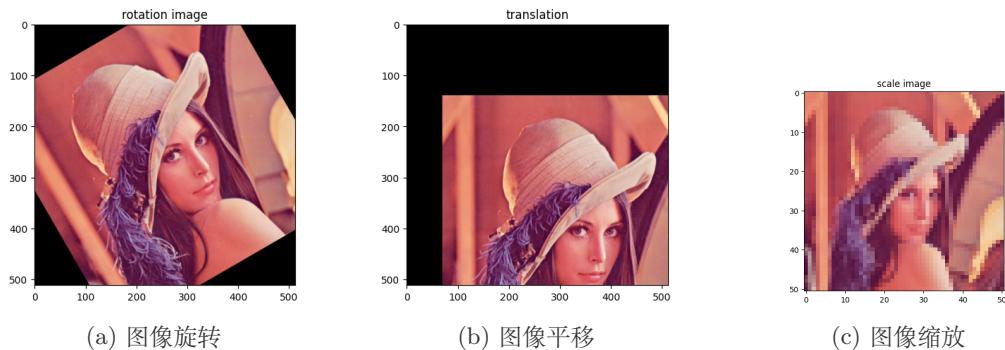
$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_m & y_m & 1 \end{bmatrix}$$

对图像平移相当于指进行旋转操作的第一步

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x^* & -y^* & 1 \end{bmatrix}$$

对图像的缩放就是

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

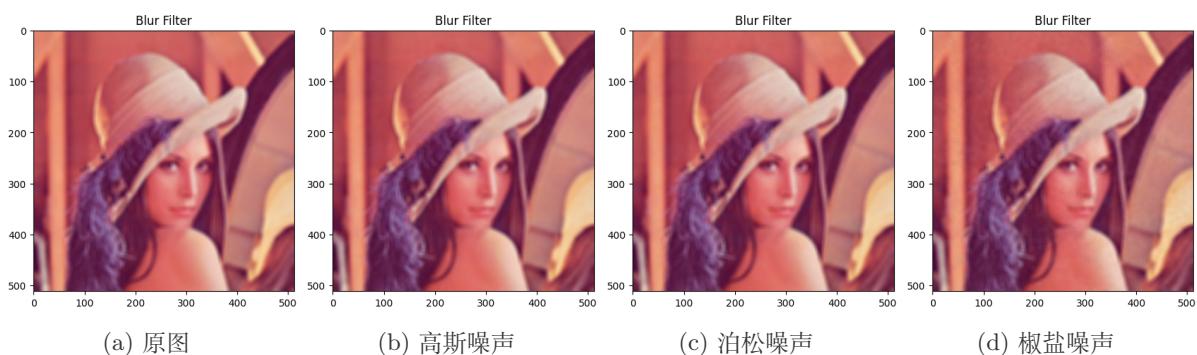


**Figure 3:** 图像基本操作

### 2.3 Smooth Filter

对图像进行均值 Blur 滤波, 需要使用滑动卷积

$$I = I_o * \frac{1}{K_{width} \cdot K_{height}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ \vdots & \ddots & \ddots & \dots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$



**Figure 4:** 图像均值滤波

值得注意的细节是当使用滑动平均的时候都会出现边界问题，OpenCV 提供了不同的 padding 方案，常见的有如下，默认选取第一种镜像填充

NAME	EXAMPLE
cv2.BORDER_DEFAULT	gfedcb abcdefgh gfedcba
cv2.BORDER_CONSTANT	iiiiii abcdefgh iiiiii

对图像进行高斯滤波，需要高斯滤波核函数，坐标按照如下排列

(-1,1)	(0,1)	(1,1)
(-1,0)	(0,0)	(1,0)
(-1,-1)	(0,-1)	(1,-1)

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

再将得到的高斯滤波核函数整体乘上归一化系数使得所有框内的数值和为 1，使整个图片的亮度不发生变化

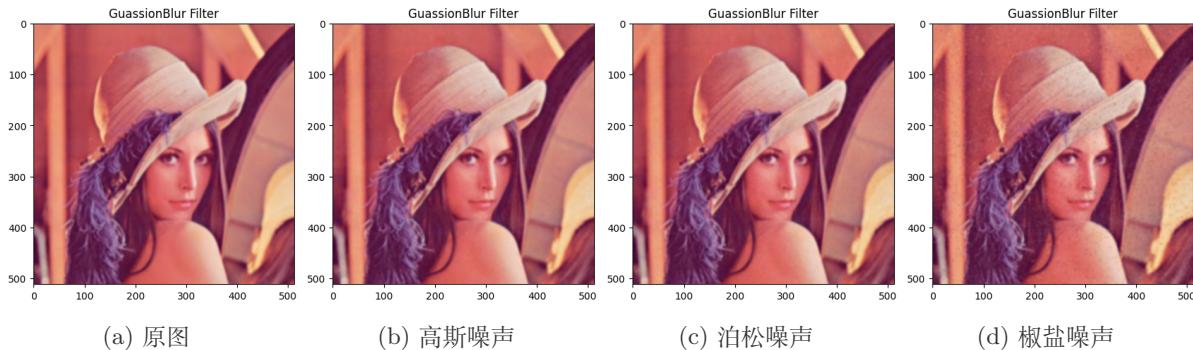


Figure 5: 图像高斯滤波

对图像进行中值滤波也非常好理解，就是在核函数大小的区域内选取中位数来代替中间的像素值

双边滤波的设计初衷是为了改变高斯滤波在某些情况下会是的图像变得模糊的情况，由于高斯滤波核函数的系数只取决于位置关系，在某些情况下不能适用，所以需要再加入像素值之间的差异：

$$G_s = \frac{1}{2\pi\sigma_s^2} e^{-\frac{x^2+y^2}{2\sigma_s^2}}$$

$$G_r = \frac{1}{2\pi\sigma_r^2} e^{-\frac{||I_p - I_0||}{2\sigma_r^2}}$$

$$h(x, y) = w_{x,y} G_s G_r$$

其中  $w_{x,y}$  是归一化系数

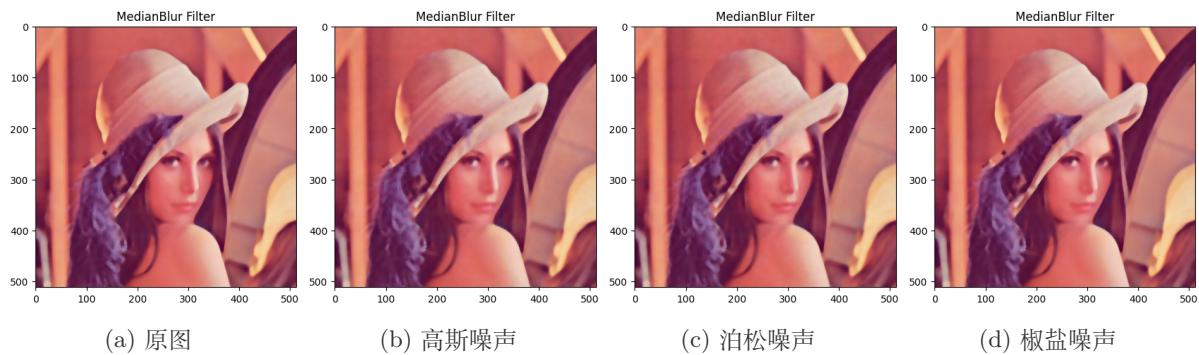


Figure 6: 图像中值滤波

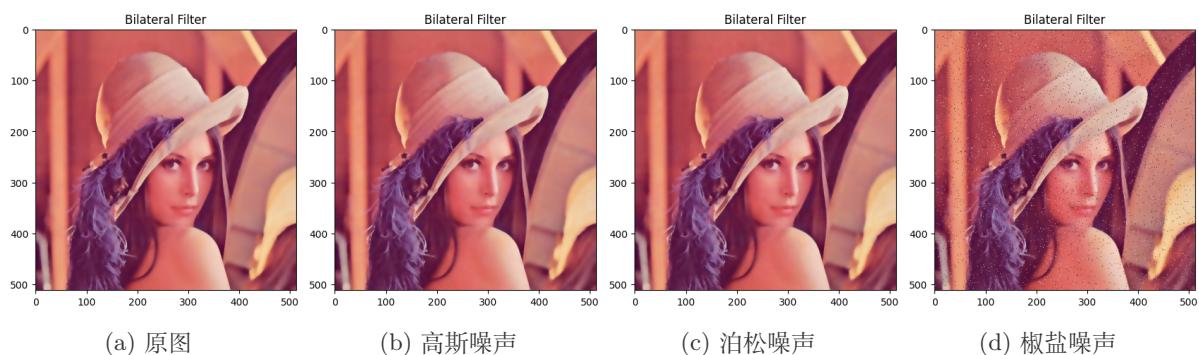


Figure 7: 图像双边滤波

## 2.4 Edge Detection Filter

Sobel 边缘提取是通过对图像求两个方向的梯度 (这里举例为 3x3 的核函数)

其中在 X 方向上的梯度是

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

在 Y 方向上的梯度为

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

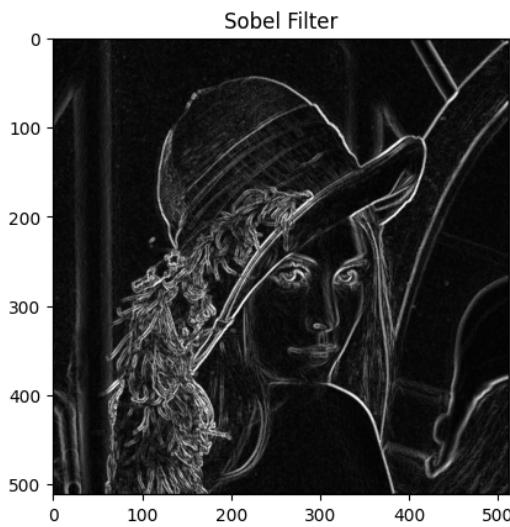
最后得到的两个方向的梯度求平方和

$$I^* = \sqrt{G_x^2 + G_y^2}$$

Canny 边缘提取需要如下五个步骤:

首先通过高斯滤波将图像去噪,

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

**Figure 8:** Sobel

$$I_1 = h * I_0$$

再将图像进行和 sobel 类似的寻找梯度

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

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

得到两个特征

$$S = \sqrt{G_x^2 + G_y^2} \quad \Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

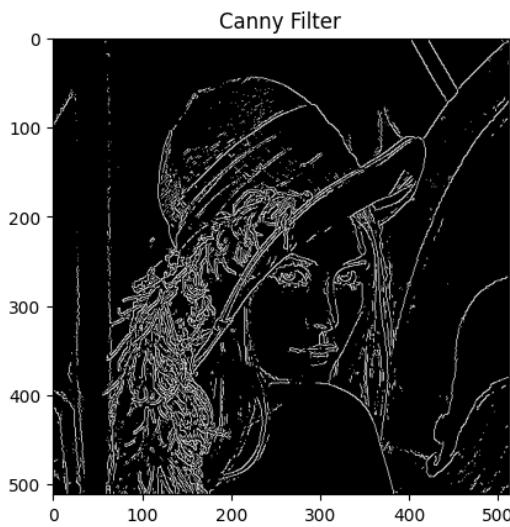
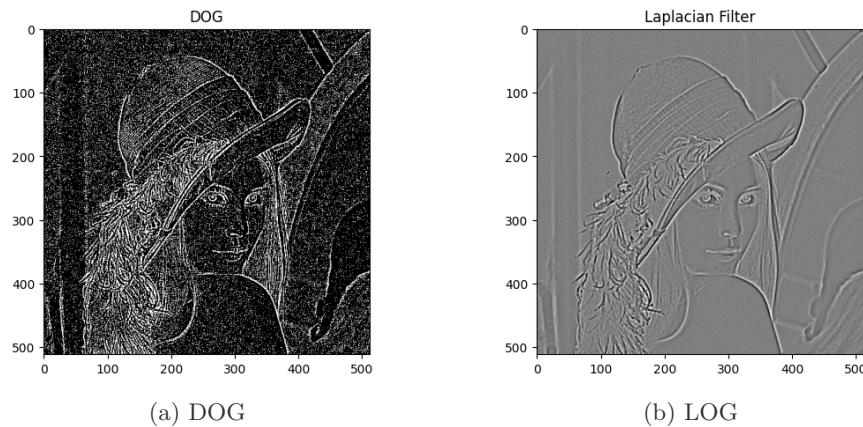
将得到的特征  $\Theta$  近似到东南西北以及四个 45 度总共 8 个方向，如果对于每个像素都有  $\Theta, S$  两个特征，对于每个像素首先按照  $\Theta$  得到其方向上的相邻两个像素，如果在  $S$  中这个像素比相邻两个像素都大那么就在  $S$  中保留其像素值，否则就置为 0.

设定两个阈值，分别为上阈值和下阈值，如果  $S$  中大于上阈值的保留并称这些为强边界，小于下阈值的置为 0，如果在两者之间的像素和强边界相连就保留否则也置为 0

通过高斯差分检测边缘是将两个不同高斯滤波器得到的图片相减

$$G(x, y\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$I^* = G(\sigma_1) * I - G(\sigma_0) * I$$

**Figure 9:** Canny

(a) DOG

(b) LOG

**Figure 10:** DOG & LOG

通过拉普拉斯检测器是二阶微分算子

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

一般使用的核函数为

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

最后通过卷积得到结果