

图像去噪 Filtering Denoise

学 院	控制科学与工程学院
任课教师	姜伟
学生姓名	申炳琦
学号	22132125

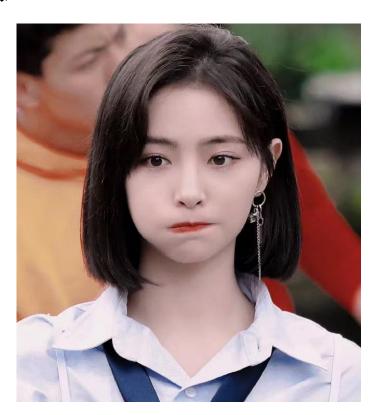
2022 年 12 月 2 日

一、 实验目的与要求

- (1) 任选一张图像(彩色、灰度任意),模糊降质(或下采样降质),加入噪声(此步骤可选)。
- (2) 用所学方法,或自选方法对步骤(1)结果图像去模糊(或 SR 处理)
- (3) 书写简单的实验报告,包括所选用方法原理简述、原图像、结果图像、以及方法效果评价。(建议选用不同方法,并比较不同方法的处理结果)

二、 实验过程与结果

1. 原始图像



2. 随机添加均值为 20 的高斯白噪声和相应滤波处理

• 随机添加均值为 20 的高斯白噪声后的图像



·均值滤波(Mean Filter)后的图像



• 中值滤波(Median Filter)后的图像



· 高斯滤波(Gaussian Filter)后的图像



· 双边滤波(Bilateral Filter)后的图像



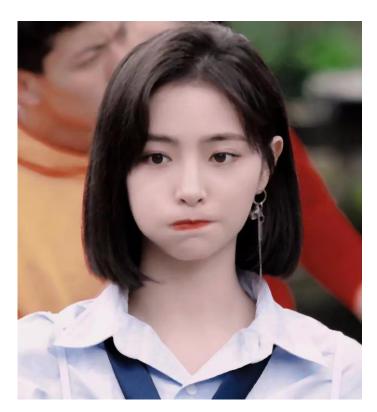
- 3. 添加比例为 0.01 的椒盐噪声和相应滤波处理
 - •添加比例为 0.01 的椒盐噪声后的图像



• 均值滤波(Mean Filter)后的图像



•中值滤波(Median Filter)后的图像



• 高斯滤波(Gaussian Filter)后的图像



·双边滤波(Bilateral Filter)后的图像



本实验在对原图像加入高斯白噪声后,共采用四种滤波方法进行图像去噪:分别是均值滤波,中值滤波,高斯滤波和双边滤波。根据上述结果,可对比分析这四种滤波方法的优缺点:

- (1) 均值滤波是一种典型的线性滤波方法,它由一个归一化卷积框完成的——用卷 积框覆盖区域所有像素的平均值来代替中心元素,卷积框尺寸越大,效率越高, 但精度会随之降低。该方法思路简单,效率高。但它不能很好地保护图像细节, 在图像去噪的同时也破坏了图像的细节部分,从而使图像变得模糊,不能很好 地去除噪声点,特别是椒盐噪声。
- (2) 与均值滤波相比,中值滤波是一种非线性平滑技术,它将图像上每个像素点的 灰度值都设置为该点邻域窗口内所有像素点灰度值的中值。由上图可看出,中 值滤波在消除椒盐噪声的方面展现出优越的性能。
- (3) 高斯滤波的作用原理与均值滤波类似,只是高斯滤波是按照加权平均的,输出 图像的每个像素点是原图像上对应像素点与周围像素点的加权和,距离越近的 点权重越大,距离越远的点权重越小,能够有效地抑制噪声,平滑图像,尤其 能有效地抑制高斯噪声。
- (4) 双边滤波是一种非线性的滤波方法,它是基于空间分布的高斯滤波函数,结合 图像的空间邻近度和像素值相似度的一种折中处理,同时考虑空域信息和灰度 相似性,对边缘等细节信息保留的较好。但是由于保存了过多的高频信息,对 于彩色图像里的高频噪声,双边滤波器不能够干净的滤掉,只能够对于低频信 息进行较好的滤波。

三、 实验代码

本实验的代码如下所示:

```
# -*-coding:utf-8-*-
"""
File Name: image_deeplearning.py
Date: 2022/12/5
Create File By Author: Bingqi Shen
"""
import cv2
import numpy as np
import random

def clamp(pv):
   if pv > 255:
     return 255
```

```
if pv < 0:
       return pv
# 加高斯噪声
def gaussian noise(image):
   image ori = image.copy()
   h, w, c = image.shape
   for row in range(h):
       for col in range(w):
           s = np.random.normal(0, 20, 3)
           b = image[row, col, 0] # blue
           g = image[row, col, 1] # green
           r = image[row, col, 2] # red
           image[row, col, 0] = clamp(b + s[0])
           image[row, col, 1] = clamp(g + s[1])
           image[row, col, 2] = clamp(r + s[2])
   cv2.imwrite('noise kiki.png', image)
   return image
# 加椒盐噪声
def sp noise(image):
   prob = 0.01
   output = np.zeros(image.shape,np.uint8)
   thres = 1 - prob
   for i in range(image.shape[0]):
       for j in range(image.shape[1]):
           rdn = random.random()
           if rdn < prob:</pre>
              output[i][j] = 0
           elif rdn > thres:
              output[i][j] = 255
              output[i][j] = image[i][j]
```

```
return output
# 均值滤波
def mean filter(noise img):
   result = cv2.blur(noise_img, (5, 5))
   cv2.imwrite('mean filter kiki.png', result)
# 中值滤波
def median filter(noise img):
   result = cv2.medianBlur(noise img, 5)
def gaussian filter(noise img):
   gaussian blurred = cv2.GaussianBlur(noise img, (5, 5), 0)
   cv2.imwrite('gaussian filter kiki.png', gaussian blurred)
def bilateral filter(noise img):
   bilateral blurred = cv2.bilateralFilter(noise img, d=20,
sigmaColor=50, sigmaSpace=15)
   cv2.imwrite('bilateral filter kiki.png', bilateral blurred)
   path = 'kiki.png'
   img = cv2.imread(path, cv2.IMREAD COLOR)
   noise img = gaussian noise(img)
   mean filter(noise img)
   median filter(noise img)
   gaussian filter(noise img)
   bilateral filter(noise img)
```