DIP 大作业

选题一

在对研究现状进行调研的基础上，力争提出创新思路，通过设计合适的滤波器，编程实现彩色图像去噪，恢复彩色图像，评价指标采用 RMSE 及主观效果。

小组分工

姓 名	学 号
蒙亚愿	17341125

摘要（200 字以内）

现有的保边滤波器在降噪的同时，也破坏了边缘和梯度。何凯明等人提出一种新型的保边滤波器——引导滤波器。引导滤波器通过一张引导图，对初始图像进行滤波，使得最后的输出图像大致与初始图像相似，但纹理部分和引导图的相似。根据引导图的不同，该滤波器还有去雾、边缘感知平滑、HDR 压缩等功能。引导滤波在保留双边滤波有效保留边缘的优点的同时，时间复杂度降低，且不会随着窗口大小变化而变化很多，在主要边缘附近没有梯度的变化。

引言

一幅图是由过度平滑、梯度较小的区域和过度尖锐、梯度较大的组成的。而噪声是影响图像质量的不利因素。噪声的特点是一起为中心的各个方向上的梯度都比较大而且差不多。边缘也是梯度大，但是边缘只有在其法向方向上才会有很大的梯度，而在切向方向上梯度则较小。

在原来的平滑滤波（如高斯滤波）中，图像中的每个像素的值是由以这个像素为中心的一个窗口中所有像素加权平均而计算出的，这都属于各向同性滤波，它们对噪声和边缘信息都是一样的处理，噪声被抹平的时候，图像中重要的边缘、纹理和细节也都被抹平。针对这个问题，现有双边滤波、维纳滤波可以解决，但是也有一种不错的方法——引导滤波。

引导滤波是属于保边滤波，复杂度为 O(N)，相对于双边滤波有更好的边缘保持性，且不会出现梯度反转的现象。在不同的引导图像的引导下，可广泛应用于降噪、去雾和高动态压缩等范围。联合双边滤波器的事件复杂度与窗口大小有关，窗口越大就越久；而引导滤波的时间复杂度与窗口大小无关。

改进的快速引导滤波算法相对于原始的引导滤波得到的效果相差无几，但是减少了运算量。主要是通过采样初始图像和引导图像，得到较小的图像，用它们来计算参数，然后通过线性插值的方法恢复到原始大小，得到近似的原参数，用来计算输出图像。可以看出参数的计算量已经不是在原始的全尺寸的图像上了，从而减少了计算量，但结果和原来的引导滤波的结果在视觉上并没有很大区别。

方法介绍

设引导图像为G，输入图像为P，输出图像为Q。导向滤波的目标是使得输入图像P和输出图像Q尽可能相同，同时纹理部分和引导图像G相似。

为了使输入图像P和输出图像Q尽可能相同，有

$$\min(Q-P)^2$$

为了使得输出图像Q的纹理部分尽可能与引导图像G相似，有

$$\nabla \mathrm{Q} = \mathrm{a}\,\nabla \mathrm{G}$$

积分得到

$$\mathrm{Q} = \mathrm{aG} + \mathrm{b}$$

考虑一个小窗口W_k，在该窗口内a和b不变，设为a_k和b_k，则根据上式可知W_k内的像素i有

$$i \in W_k \quad q_i = a_k\, g_i + b_k \tag{1}$$

把(1)代入第一个目标，则窗口内的像素就同时满足了以上两个条件：

$$E(a_k, b_k) = \sum_{i \in W_k} ((a_k\, g_i + b_k - p_i)^2 + \varepsilon a_k^2) \tag{2}$$

其中，ε是惩罚大的a_k的正则化参数。

要使(2)最小，有

$$\begin{cases} \dfrac{\partial E}{\partial a_k} = \sum\limits_{i \in W_k} (2g_i(g_i a_k + b_k - p_i) + 2\varepsilon a_k) = 0 \\ \dfrac{\partial E}{\partial b_k} = \sum\limits_{i \in W_k} \big(2(g_i a_k + b_k - p_i)\big) = 0 \end{cases}$$

$$\begin{cases} (\sum\limits_{i \in W_k} g_i^2 + \varepsilon |W|)a_k + \sum\limits_{i \in W_k} g_i b_k - \sum\limits_{i \in W_k} p_i g_i = 0 \\ \sum\limits_{i \in W_k} g_i a_k + |W| b_k - \sum\limits_{i \in W_k} p_i = 0 \end{cases}$$

其中，|W|是窗口W_k的像素总数。

解得

$$a_k = \frac{|W| \sum_{i \in W_k} p_i g_i - \sum_{i \in W_k} p_i \sum_{i \in W_k} g_i}{|W| \sum_{i \in W_k} g_i^2 + \varepsilon |W|^2 - \left(\sum_{i \in W_k} g_i\right)^2} \tag{3}$$

$$\mathrm{b_k} = \frac{\sum_{i \in W_k} p_i - \sum_{i \in W_k} g_i a_k}{|W|} \tag{4}$$

设$\overline{p_k}$是输入图P在窗口W_k的平均值，μ_k和δ_k是引导图G在窗口W_k的平均值和方差，则有

$$\mathrm{a_k} = \frac{\frac{1}{|W|} \sum_{i \in W_k} p_i g_i - \overline{p_k} \mu_k}{\delta_k^2 + \varepsilon} = \frac{cov(p_k, g_k)}{\delta_k^2 + \varepsilon} \tag{5}$$

$$\mathrm{b_k} = \overline{p_k} - \mu_k a_k \tag{6}$$

其中，$\mathrm{cov}(\mathrm{p_k}, g_k) = \frac{1}{|W|} \sum_{i \in W_k} p_i g_i - \overline{p_k} \mu_k$是引导图G和输入图P在$W_k$的协方差。

计算出a_k和b_k后，就可以根据(1)来计算窗口W_k的输出像素。对于一个像素i，输出值q_i和所有有关覆盖像素i的窗口W_k有关。所以当W_k不同，q_i的值也不同。一个简单的策略是平均所有可能的q_i值。计算了所有可能覆盖i的窗口的W_k的个数为|W|，那么

$$\mathrm{q_i} = \frac{1}{|W|} \sum_{k | i \in W_k} (a_k g_i + b_k) \tag{7}$$

$$\mathrm{q_i} = \overline{a_\iota} g_i + \overline{b_\iota} \tag{8}$$

其中，$\bar{a_i} = \frac{1}{|W|}\sum_{k|i\epsilon W_k} a_k$，$\bar{b_i} = \frac{1}{|W|}\sum_{k|i\epsilon W_k} b_k$

假定在窗口W_k内，a_k和b_k是常数，只与W_k的位置有关。取W_k为半径是r的方形窗口，使在窗口内的中心像素位置遍历整个图像，那么W_k就取到了不同的所有位置，在每个位置计算出相应的a_k和b_k。所有的a_k和b_k组成了和输入图像P相同大小的数据集A和B。对于任意的像素i，$\bar{a_i}$和$\bar{b_i}$分别为以i为中心、半径为r的窗口W_k内A和B的数据均值，从而就可以从(8)计算出输出图像。

流程如下：

Algorithm 1. Guided Filter.

Input: filtering input image p, guidance image I, radius r, regularization ϵ

Output: filtering output q.

1: $\text{mean}_I = f_{\text{mean}}(I)$
$\text{mean}_p = f_{\text{mean}}(p)$
$\text{corr}_I = f_{\text{mean}}(I. * I)$
$\text{corr}_{Ip} = f_{\text{mean}}(I. * p)$

2: $\text{var}_I = \text{corr}_I - \text{mean}_I. * \text{mean}_I$
$\text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I. * \text{mean}_p$

3: $a = \text{cov}_{Ip}./(\text{var}_I + \epsilon)$
$b = \text{mean}_p - a. * \text{mean}_I$

4: $\text{mean}_a = f_{\text{mean}}(a)$
$\text{mean}_b = f_{\text{mean}}(b)$

5: $q = \text{mean}_a. * I + \text{mean}_b$

在实现里，对图像进行滤波时是对每一个 RGB 单独最为一个二维的图像处理，并在还需要将图像归一化，用浮点数进行计算。f_{mean}是方框滤波后的图像点除方框滤波全是 1 的矩阵得到的结果。

引导滤波相对于双边滤波最大的优势在于算法得复杂度与窗口大小无关，对于处理较为大型的图像时效率有明显的提升。引导滤波中每个像素虽然由多个窗口包含，但是求某一像素点值时，只需要将包含该点的所有线性函数求平均就可以。而双边滤波考虑了每个点的几何差距与强度差距，当处理较大的图像时，运算量会明显增大。

同时假设存在线性关系，可以保证梯度的一致性，很好的克服了双边滤波中出现的梯度反转的现象。对于双边滤波而言，对于梯度变化大的地方，由于周围没有相似的像素，高斯函数的权重不稳定，导致梯度出现反转现象。

实验结果

源图像

目标图像

引导滤波

快速引导滤波

高斯滤波

双边滤波

Table 1 客观效果

	RMSE	PSNR
guassion Img	8.109795058285064	26.75226372189195
bilateral Img	9.01151743470926	22.60481128907579
guideImg	80.97139264168213	9.964171431678029
fastGuideImg	80.97082938415477	9.964231853137735

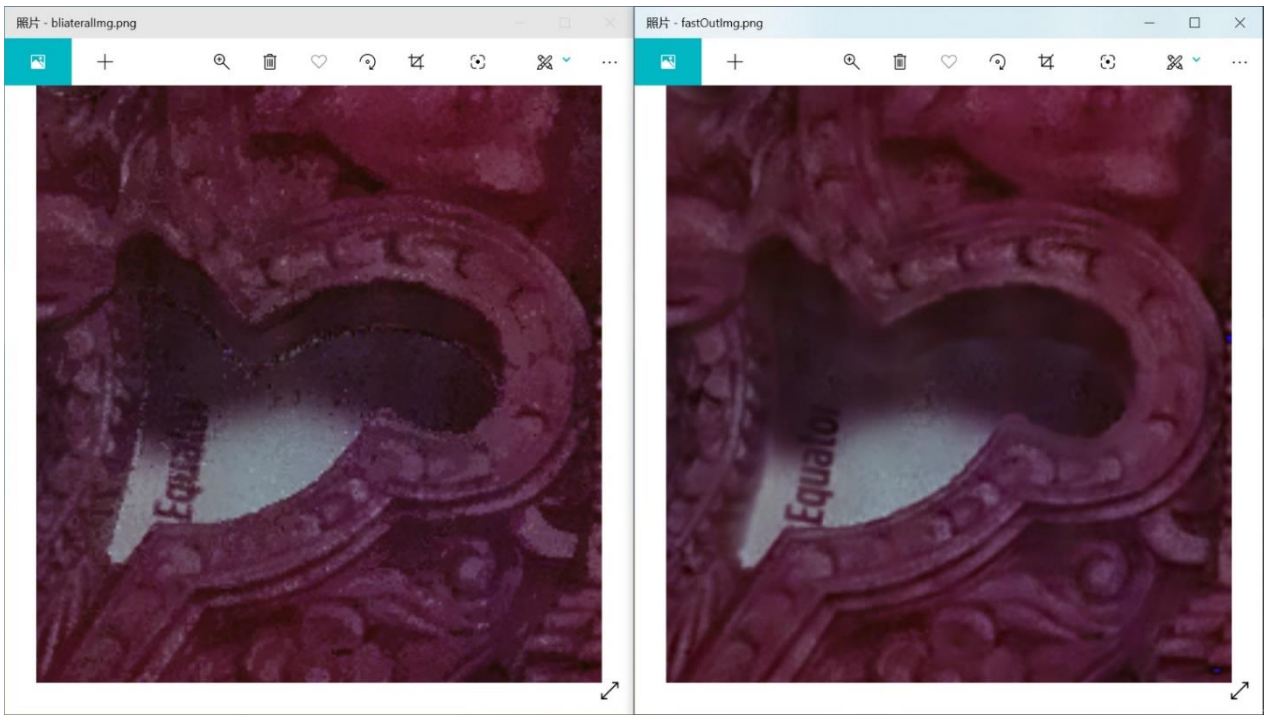
Table 2 计算时间

	time(seconds)
guideFilter	2.5491762161254883
fastGuideFilter	0.7460122108459473

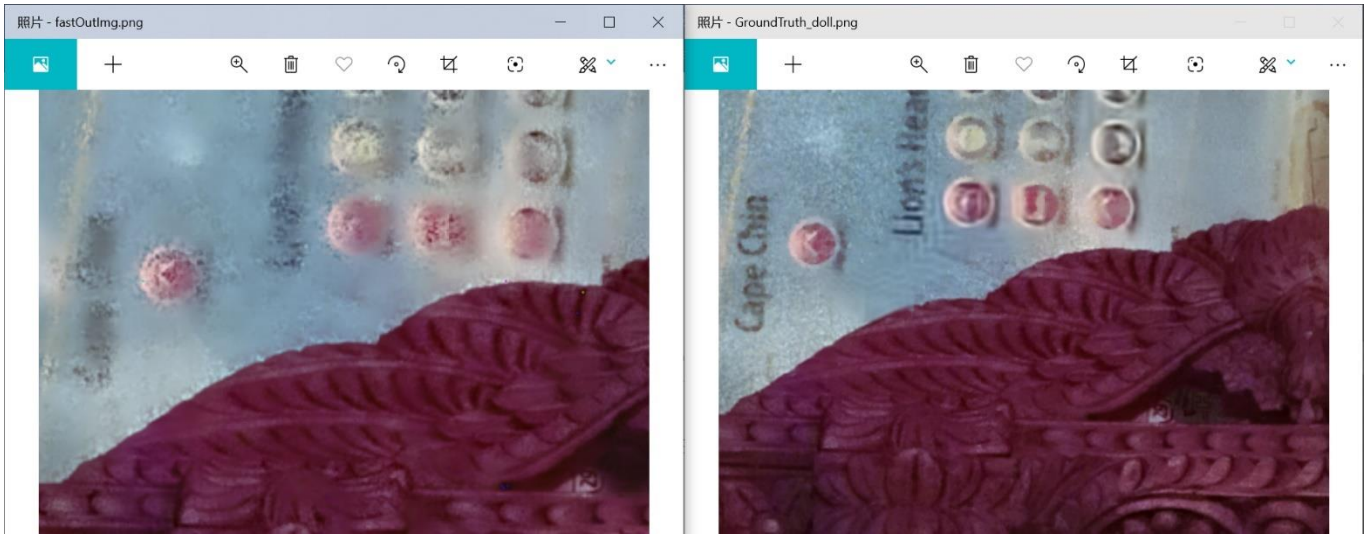
以对 rgb_doll_noise.png 图像进行高斯滤波、联合双边滤波、引导滤波以及快速引导滤波为例，计算得到的图像和 GroundTruth.png 的RMSE和PSNR。

引导滤波的窗口大小设置为 5，惩罚误差为 0.000001。从表 1 可知，引导滤波和快速引导滤波的效果比其余两个的处理相比，效果特别好，这一点从视觉效果上也能看出的来，得到的图像明显平滑了很多，和 GroundTruth.png 很相似，同时信噪比也明显很小，失真也很小。但是快速引导滤波效果和引导滤波的几乎一样，但是消耗的时间就大大减了。

而高斯滤波明显从主观视觉效果、客观效果上都差尽人意。但联合双边滤波得到的效果虽从视觉上成功地降噪，但是在客观结果上看，却也是差之千里。按照论文所述，放大图像来看，在某些边缘确实有梯度反转伪影现象。



虽然经过引导滤波后，图像能够成功的平滑降噪，但是不可避免地在一些很小很细节的地方会有点模糊，有一些平滑过渡的感觉。



其余图像的效果图可在相应文件夹中查看。

程序说明

数据集：

名称	修改日期	类型	大小
book	2020/7/6 22:24	文件夹	
bowl	2020/7/6 22:21	文件夹	
cave	2020/7/6 22:14	文件夹	
data	2020/6/29 16:49	文件夹	
peacock_window	2020/7/6 22:29	文件夹	
teapot	2020/7/6 22:17	文件夹	

代码：

名称	修改日期	类型	大小
.idea	2020/7/6 22:49	文件夹	
__pycache__	2020/7/6 17:15	文件夹	
boxFilter.py 方框滤波	2020/7/6 17:14	Python File	2 KB
fastGuideFilter.py 快速引导滤波	2020/7/3 14:46	Python File	1 KB
guideFilter.py 引导滤波	2020/7/6 15:32	Python File	3 KB
main.py main	2020/7/6 22:36	Python File	4 KB

- **main.py**

1) 读取源图像、引导图像和目标图像

2) 进行双边滤波

3) 图像归一化

4) 设置引导滤波参数、并进行引导滤波

5) 设置快速引导滤波参数、并进行快速引导滤波

6) 把双边滤波、引导滤波、快速引导滤波都得到的图像和目标图像两两计算 RMSE 和 PSNR

- **guideFilter.py**

参数：引导图的某一 RGB 平面guideImg，源图像的某一 RGB 平面srcImg，窗口大小radius，惩罚eps

返回：引导滤波后的某一平面dstImg

1) 对全是 1 的矩阵进行方框滤波得到均值矩阵N

2) 对guideImg进行方框滤波进行方框滤波，然后点除矩阵N，得到$\mathrm{mean_I}$

3) 对srcImg进行方框滤波进行方框滤波，然后点除矩阵N，得到mean_p

4) 对guideImg $*$ guideImg进行方框滤波进行方框滤波，然后点除矩阵N，得到$corr_I$

5) 对guideImg $*$ srcImg进行方框滤波进行方框滤波，然后点除矩阵N，得到$\mathrm{coor_{Ip}}$

6) $\mathrm{var_I} = \mathrm{corr_I} - \mathrm{mean_I} * \mathrm{mean_I}$

7) $\mathrm{cov_{Ip}} = \mathrm{corr_{Ip}} - \mathrm{mean_I} * \mathrm{mean_p}$

8) $\mathrm{a} = \frac{\mathrm{cov_{Ip}}}{(var_\mathrm{I} + \mathrm{eps})}$

9) $\mathrm{b} = \mathrm{mean_p} - \mathrm{a} * \mathrm{mean_I}$

10) 对a进行方框滤波，然后点除矩阵N得到$\mathrm{mean_a}$

11) 对b进行方框滤波，然后点除矩阵N得到$\mathrm{mean_b}$

12) 返回引导滤波后的图像$\mathrm{mean_a} * guideImg + mean_b$

- **fastGuideFilter.py**

参数：参数：引导图的某一 RGB 平面guideImg，源图像的某一 RGB 平面srcImg，窗口大小radius，惩罚eps，变化大小zise

返回：引导滤波后的某一平面dstImg

fastGuideFilter.py 和 guideFilter.py 主要的区别就是在进行所有的计算之前把guideImg和srcImg通过双线性插值，把图像变为原来的$\frac{1}{size}$，得到较小的图像sub_I和sub_p，然后之后的计算就是对sub_I和sub_p滤波、计算。

而在最后得到a和b之后，再对它们进行双线性插值变回原来的大小，得到$mean_a$和$mean_b$，再去计算输出图像。

1) 对 guideImg和srcImg进行双线性插值，缩小size倍，得到sub_I和sub_p
2) 对全是 1 的矩阵进行方框滤波得到均值矩阵N
3) 对sub_I进行方框滤波进行方框滤波，然后点除矩阵N，得到$mean_I$
4) 对sub_p进行方框滤波进行方框滤波，然后点除矩阵N，得到$mean_p$
5) 对guideImg ∗ guideImg进行方框滤波进行方框滤波，然后点除矩阵N，得到$corr_I$
6) 对guideImg ∗ srcImg进行方框滤波进行方框滤波，然后点除矩阵N，得到$coor_{Ip}$
7) $var_I = corr_I - mean_I * mean_I$
8) $cov_{Ip} = corr_{Ip} - mean_I * mean_p$
9) $a = \frac{cov_{Ip}}{(var_I + eps)}$
10) $b = mean_p - a * mean_I$
11) 把a恢复到原来的大小，即放大size倍，然后进行方框滤波，点除矩阵N得到$mean_a$
12) 把b放大size倍，然后进行方框滤波，点除矩阵N得到$mean_b$
13) 返回引导滤波后的图像$mean_a * guideImg + mean_b$

- **boxFilter.py**

参数：源图像的某一个平面srcImg，窗口大小radius

返回：方框滤波后的某一平面dstImg

1) 创建一个零矩阵dstImg
2) 列累加
    a) srcImg每一列累加到下一列得到sumY

b) dstImg的$[0,\ radius]$列等于sumY的$[radius,\ 2radius]$列

c) dstImg的$[radius+1,\ H-radius]$列等于sumY的$[2radius+1,\ H-1]$列减去sumY的$[0,\ H-2radius-2]$ 列

d) dstImg的$[H-radius,\ H-1]$列等于sumY的最后一行扩张radius列减去sumY的$[H-2radius-1,\ H-radius-2]$ 列

3) 行累加

e) dstImg每一行累加到下一行得到sumX

f) dstImg的$[0,\ radius]$等于sumY的$[radius,\ 2radius]$行

g) dstImg的$[radius+1,\ H-radius]$行等于sumY的$[2radius+1,\ H-1]$行减去sumY的$[0,\ H-2radius-2]$行

h) dstImg的$[H-radius,\ H-1]$行等于sumY的最后一行扩张radius行减去sumY的$[H-2radius-1,\ H-radius-2]$行

具体分工说明

蒙亚愿（17341125）：由于个人原因，这次的大作业全部由我一个人完成。一开始看论文的时候原理部分有点看不懂，后来看到论文中有给了具体的算法，然后就很好算了。首先算法中的f_{mean}是均值滤波的意思，于是我用方框滤波，并且用加法，这样事件复杂度比较低。

最后出来的图像不管怎么改radius和eps出来的图像都有朦胧感，还以为是方框滤波写错了，但改成了 opencv 里的函数还是一样的结果，最后慢慢调试，才发现在计算$mean_I$的时候减法写成了加法，一改然后图像就清晰了，果然有 bug 这种事要从自己的角度反思，代码是不会骗人的。在实现快速引导滤波时，本来缩放的话可以用之前写过的双线性插值，但为了避免不必要的时间开销，就直接用了库函数来改。

这次大作业其实很有趣，学到了东西。最后感谢老师和 TA 姐姐这一学期的教导和帮助，希望以后还能再见 ahhh