

1、图像引导滤波算法介绍：

图像引导滤波是一个线性平移变的滤波过程。当引导图为输入图像时，引导滤波就成为一个保持边缘的滤波操作，可以用于图像重建的滤波。

其包括引导图像 I ，输入图像 p 和输出图像 q 。其中引导图像 I 是需要根据具体应用事先设定的，也可以直接取为输入图像 p 。对于输出图像中第 i 个像素而言，起计算方法可表达为

$$q_i = \sum_j W_{ij}(I)p_j, \quad (1)$$

式中， i 和 j 为像素索引， W_{ij} 为滤波核函数，其定义如下

$$W_{ij}(I) = \frac{1}{|\omega|^2} \sum_{k:(i,j) \in \omega_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right). \quad (9)$$

式中， ω_k 为第 k 个核函数窗口， $|\omega|$ 为窗口内的像素个数， μ_k 和 σ_k^2 是引导图像 I 在窗口内的均值和方差， ϵ 为平滑因子。

式(1)、式(9)为论文构造引导滤波器化简后的式子。

在这里介绍下引导滤波器具体的处理过程：

假设输入图像为 p ，输出图像为 q ，引导图为 I ， q 与 I 在以像素 k 为中心的窗口中存在局部线性关系：

$$q_i = a_k I_i + b_k, \forall i \in \omega_k, \quad (3)$$

窗口半径为 r ， a ， b 为线性系数，且在局部窗口 k 中为常数。这个模型保证了只有在 I 存在边缘的情况下， q 才会存在边缘。这是因为： $\nabla q = a \nabla I$ 。这与在去雾、超分辨率、抠图等研究中使用的模型是一致的。

q 即 p 去除噪声或者纹理之后的图像：

$q_i = p_i - n_i$

为确定以上公式中的线性系数，并满足使得 q 与 p 的差别最小，转化为最优化问题：

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2). \quad (4)$$

以上公式的求解可以利用线性回归：

$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon} \quad (5)$$

$$b_k = \bar{p}_k - a_k \mu_k. \quad (6)$$

在这里， u_k 和 σ_k^2 表示 I 在局部窗口 w_k 中的均值和方差。 $|\omega|$ 是窗口内的像素数， p_k 表示 p 在窗口 w_k 中的均值。当求的 a_k 和 b_k 后：

$$q_i = \frac{1}{|\omega|} \sum_{k:i \in \omega_k} (a_k I_i + b_k) \quad (7)$$

$$= \bar{a}_i I_i + \bar{b}_i \quad (8)$$

其中 $\bar{a}_i = \frac{1}{|\omega|} \sum_{k \in \omega_i} a_k$ 和 $\bar{b}_i = \frac{1}{|\omega|} \sum_{k \in \omega_i} b_k$.

算法过程

Algorithm 1 Guided Filter.

- 1: $\text{mean}_I = f_{\text{mean}}(I, r)$
 $\text{mean}_p = f_{\text{mean}}(p, r)$
 $\text{corr}_I = f_{\text{mean}}(I \cdot I, r)$
 $\text{corr}_{Ip} = f_{\text{mean}}(I \cdot p, r)$
 - 2: $\text{var}_I = \text{corr}_I - \text{mean}_I \cdot \text{mean}_I$
 $\text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I \cdot \text{mean}_p$
 - 3: $a = \text{cov}_{Ip} ./ (\text{var}_I + \epsilon)$
 $b = \text{mean}_p - a \cdot \text{mean}_I$
 - 4: $\text{mean}_a = f_{\text{mean}}(a, r)$
 $\text{mean}_b = f_{\text{mean}}(b, r)$
 - 5: $q = \text{mean}_a \cdot * I + \text{mean}_b$
-

2、实现细节

a、由于使用的是 matlab 编程其主算法确实按照 algorithm1 的实现，修改不多。

b、改进均值滤波器

但发现实现该算法效率过慢在进行彩色增强的时候，于是想一下算法耗时大概会在哪个地方，由上图可知均值滤波器是该算法用的最多的滤波器，所以可能是均值滤波时间复杂度为($w \cdot h \cdot r \cdot r$)过慢导致，于是查询网上有无改进方法，发现有时间复杂度为($w \cdot h$)。

该算法主要先计算行累积和 rowcum 然后根据滤波器行宽度 w , $\text{imgdes}[i] = \text{rowcum}[i+w] - \text{rowcum}[i]$ 计算得出的结果再跟据列累计和同样的方法得出 imgdes 。最后再除以($w \cdot h$)，就是图像均值滤波后的结果。

这算法极大的加快运行时间。

c、图像像素归一化

在实现过程中，还需注意一点，在进行算法前图像的像素值都需进行归一化，如 $\text{img}[i] = \text{img}[i]/255$ ，否则由于 ε 的取值很小导致获得的图像没法进行准确标定。
d、快速引导滤波算法

Algorithm 2 Fast Guided Filter.

- 1: $I' = f_{\text{subsample}}(I, s)$
 $p' = f_{\text{subsample}}(p, s)$
 $r' = r/s$
 - 2: $\text{mean}_I = f_{\text{mean}}(I', r')$
 $\text{mean}_p = f_{\text{mean}}(p', r')$
 $\text{corr}_I = f_{\text{mean}}(I' \cdot * I', r')$
 $\text{corr}_{Ip} = f_{\text{mean}}(I' \cdot * p', r')$
 - 3: $\text{var}_I = \text{corr}_I - \text{mean}_I \cdot * \text{mean}_I$
 $\text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I \cdot * \text{mean}_p$
 - 4: $a = \text{cov}_{Ip} ./ (\text{var}_I + \epsilon)$
 $b = \text{mean}_p - a \cdot * \text{mean}_I$
 - 5: $\text{mean}_a = f_{\text{mean}}(a, r')$
 $\text{mean}_b = f_{\text{mean}}(b, r')$
 - 6: $\text{mean}_a = f_{\text{upsample}}(\text{mean}_a, s)$
 $\text{mean}_b = f_{\text{upsample}}(\text{mean}_b, s)$
 - 7: $q = \text{mean}_a \cdot * I + \text{mean}_b$
-

利用之前作业上的双线性插值法进行下采样和上采样。

这里还需注意由于图像宽度和高度不能整除 s ，所以我这里在上采样时再传入 (originw, originh) 两个参数(原图像大小)进行扩充。

3、实验结果

普通的引导滤波：

其参数从左到右为 $\varepsilon = 0.1 * 0.1, \varepsilon = 0.2 * 0.2, \varepsilon = 0.4 * 0.4$

从上到下为 $r=2, r=4, r=8$

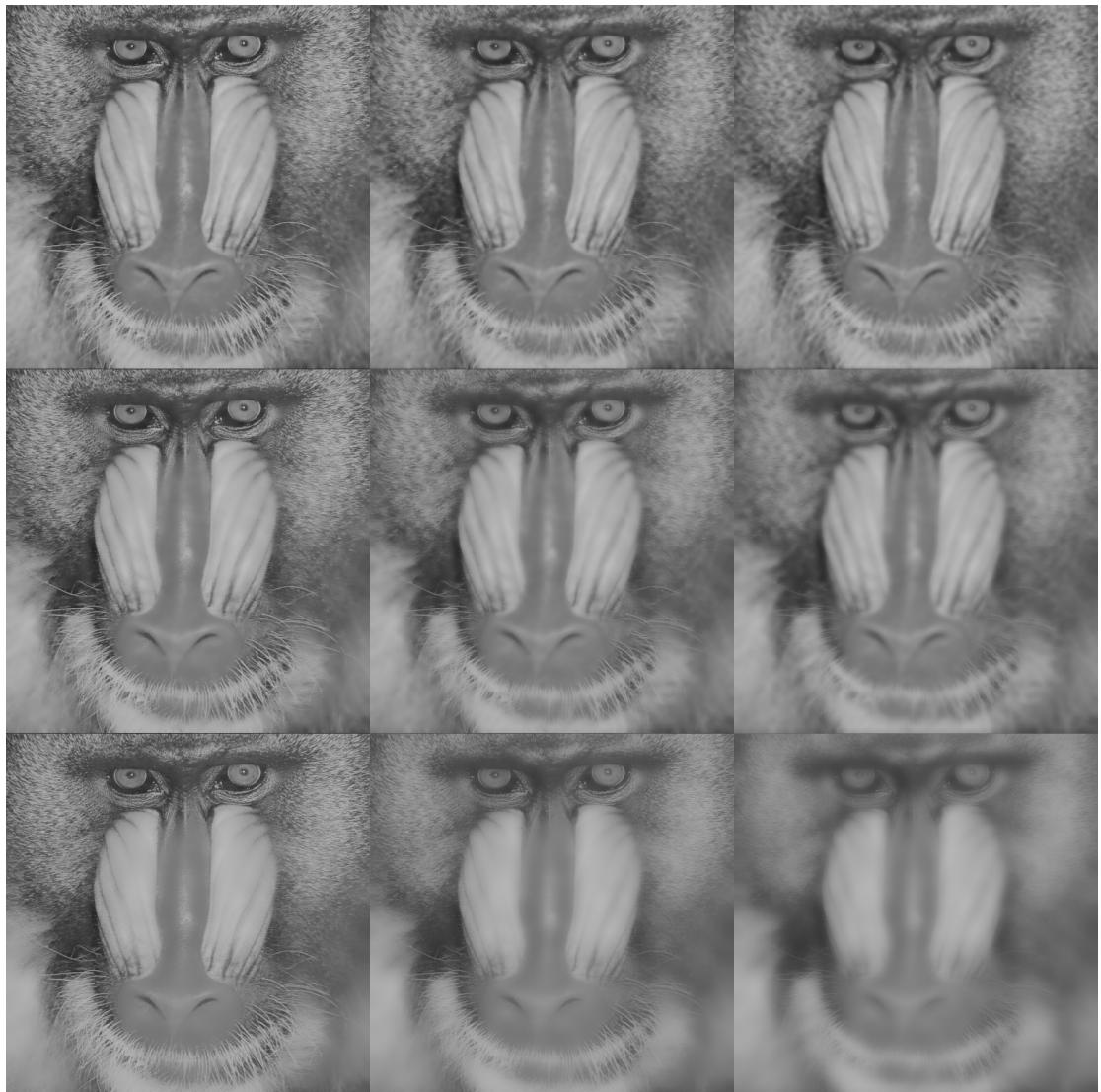


如图所示， ε 的取值比 r 的取值影响大一点，当 $r=2$, $\varepsilon = 0.1 * 0.1$ 变成 $r=4$, $\varepsilon = 0.2 * 0.2$ 比当 $r=2$, $\varepsilon = 0.1 * 0.1$ 变成 $r=2$, $\varepsilon = 0.2 * 0.2$ 更清晰点。

该算法的应用使边缘平滑上取得较好的效果

以下为其它图像实验结果。







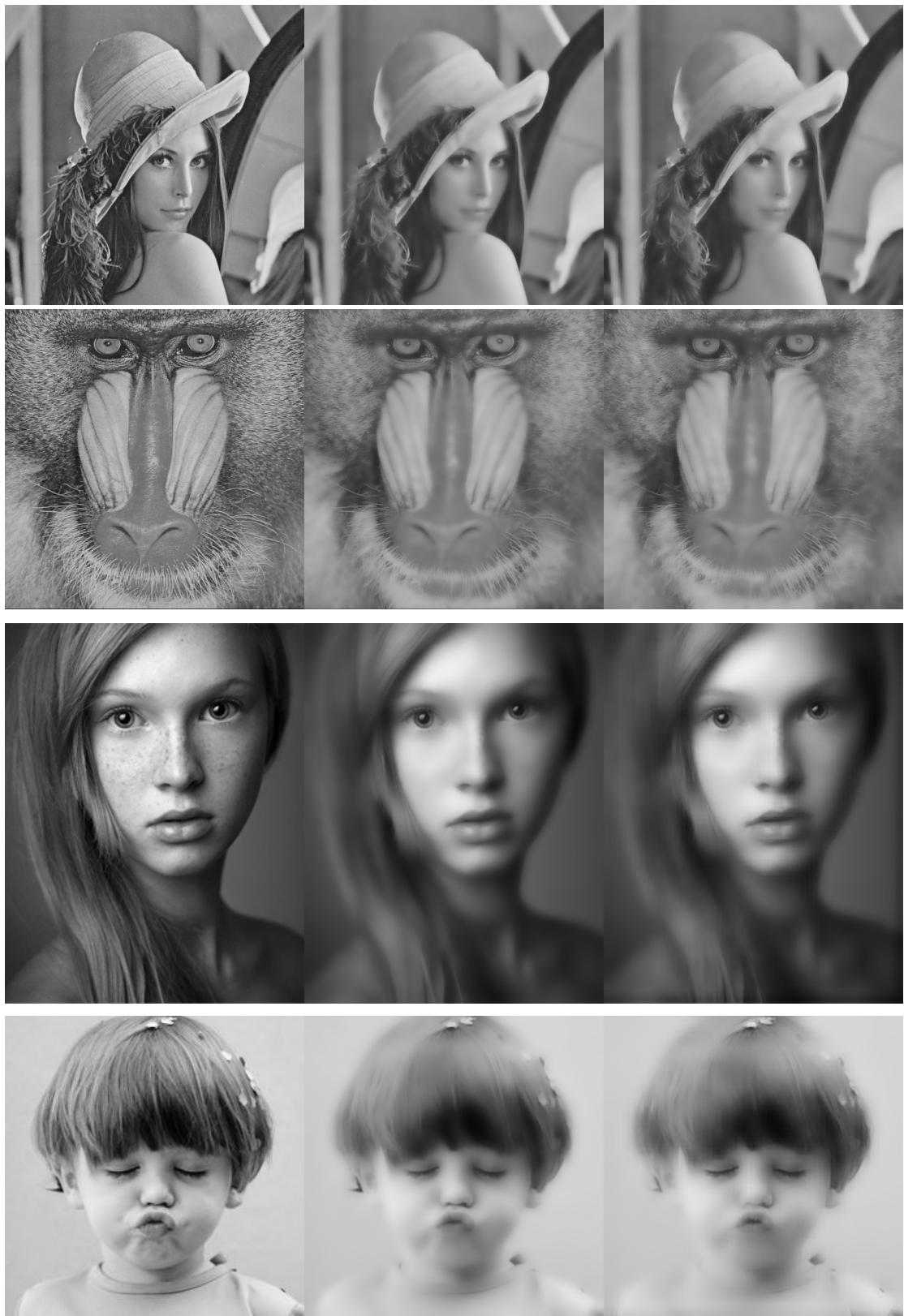


对于快速引导滤波的实验结果

$$r = 4, \epsilon = 0.2^2, s=4$$

第一张为原图，第二张为普通引导滤波，第三张为快速引导滤波





快速滤波有一些图片产生块效应且处理的图像更为模糊，这可能是取样方法导致的问题，但时间确实加快了，

```
>> examples_fastguided_smooth  
0.0980
```

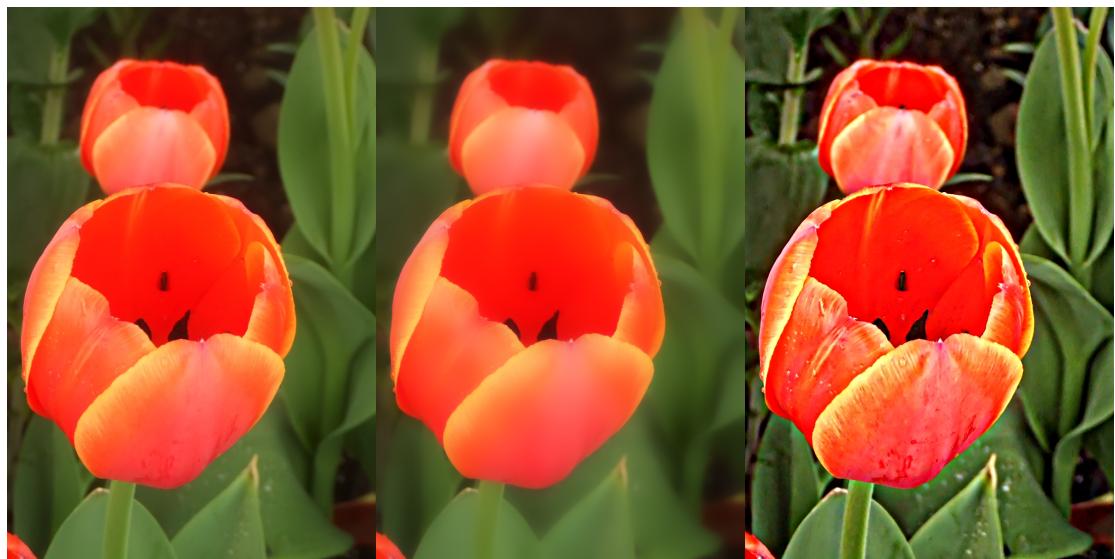
```
>> examples_guided_smooth  
0.1506
```

时间提升了一半差不多。

图像增强对比

普通引导滤波结果

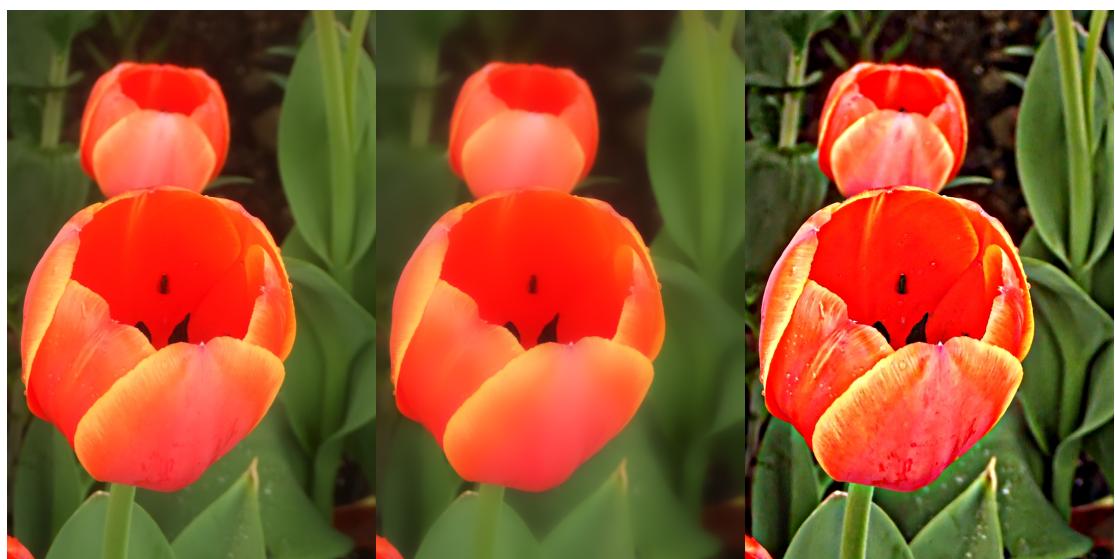
$r=16, \epsilon = 0.1^2, \text{boosted}^*5$



图像有所增强，且在边缘上有很好的渐变过程，这是算法的结论

$\nabla q \approx \bar{a} \nabla I$, 产生的很好的效果。

快速引导滤波的结果：





快速滤波:



快速引导滤波:





快速引导滤波:



快速引导滤波:



其时间对比:

```
>> examples_guided_enhancement  
0.9758
```

```
>> examples_fastguided_enhancement  
0.5587
```

整整快了一倍多，也不会出现块效应，但快速滤波后的图比普通滤波的图更模糊些，但其中有几幅图，经过快速滤波后亮度增加了。

4、该算法的缺点在于：

- 1、由于该算法利用了均值滤波器，所以也同样有固有的缺陷，在去噪的过程中破坏图像细节，使图像变得模糊。
- 2、在边缘平滑上导致边缘更亮，有光晕效果。
- 3、无法直接应用在较稀少输入的 **strokes** 上。

自己想出的改进办法：

针对第 1 点，把窗口半径 r 和 ϵ 控制在较小的范围内，再使用拉普拉斯算子，将局部变化不大的图像提取出来，进行图像增强，这样又保证了边缘平滑，而且图像细节有所增强。

针对第 2 点、如果为彩色图，尝试在 HSI 模型的 I 度图上进行图像引导滤波，我发现灰度图的光晕效果要小一点。

针对第 3 点，通过一些深度学习的方法进行图像特征提取之后，进行图像上采样，间接的应用在较稀少输入的 **strokes** 上。