

双边滤波说明

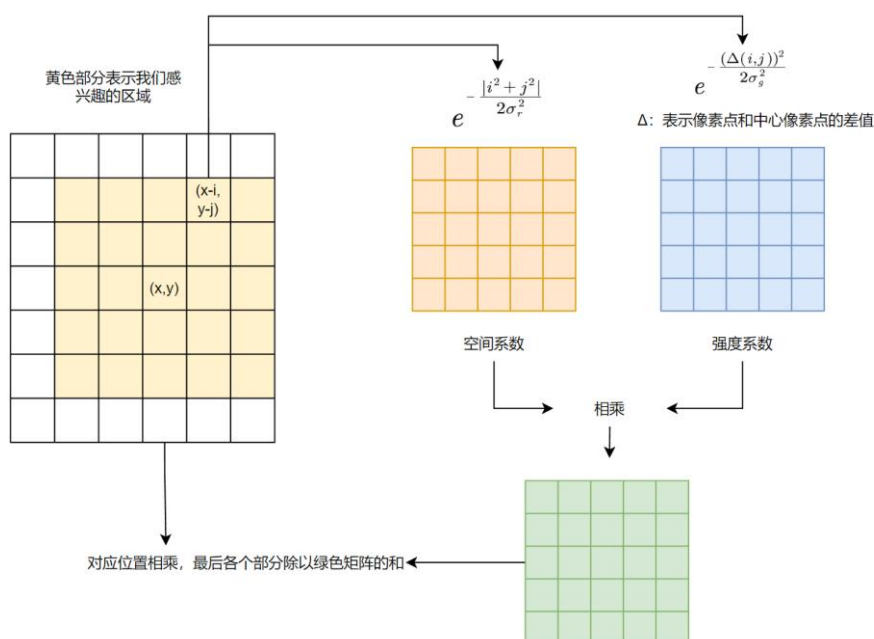
对于一个像素，双边滤波计算公式如下：

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}, \quad (3)$$

where normalization factor $W_{\mathbf{p}}$ ensures pixel weights sum to 1.0:

$$W_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|). \quad (4)$$

图例解释：



实现流程，主要是两个部分：空间系数和强度系数

1. 空间系数

用户设置有一个参数为窗口大小，要求只能为 3*3、5*5、7*7。

公式中有另一个参数：空间标准差，即上图的 σ_r 。这个参数不需要用户输入，因为当标准差乘以 6 和窗口直径接近的时候效果较好，所以可以直接通过用户输入的窗口大小计算得到。

根据用户输入的窗口，空间系数可以直接选择为下面的对应数据：

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

2. 强度系数

用户设置有一个参数为强度标准差，即示意图的 σ_g 。

以像素为 12bit 为例，两个像素值之差的绝对值 Δ 最大为 4095，所以运算 Δ 为 0-4095 时的结果，即上图中的 $e^{-\Delta(i,j)^2/2\sigma_g^2}$ ，计算后存储入 LUT。该步骤不应该让 FPGA 实现，应该事先存储形成一个文件，或者 MCU 来完成。真正图像处理时，根据 Δ （像素点和中心像素点的差值）进行查表，得到对应系数后进行后续计算。

此外，如果为了节省资源，像素值之差可以进行简单的量化，比如每次算完像素差之后除以 16，这样用 256 个值代替 4096。

3. FPGA 处理

- 黑白图片：正常按照图例流程即可
- 彩色图片：每个颜色都要执行一次操作，注意每个颜色计算时用到的像素差是红绿蓝的平均值（或者使用 YUV 中的 Y，其他模块有过计算，如果可以复用则使用 Y 最好），这也意味着红绿蓝的像素差都是一样的。切记不要使用每个颜色各自的像素值 s 。

第 2 步的 MCU 计算表的举例，称这张强度表叫 G 表，假设他是 64 个 6bit 数目：

先认为 G 表是浮点数，这样 $G[i] = e^{-(64 \times i)^2 / (2 \times (16 \times \sigma)^2)}$ ，解释一下分子和分母：分子是因为 G 表压缩成只有 64 个，不压缩即最真实的时候是 4096 个， $4096/64=64$ ，所以对于 G 表的第 x 项，放在实际中应该是 $(64 \times x)$ 。分母是因为 σ 压缩成 0-255，不压缩最真实的时候也是 4096（即需要保证 σ 可取范围能达到 Δ 最大值），所以对于用户输入的 σ ，放在实际中应该是 $(16 \times \sigma)$ 。

化简得到 $G[i] = e^{-8(i/\sigma)^2}$ ，而由于我们存储的是 6bit，因为 G 表正常范围是 0-1，所以这里就是乘以 63 即可。

提供了 FPGA 模拟双边滤波的 python 实现，有快速版和慢速易懂版。

3*3 大小的双边滤波实现流程

不需要看这里的文档，该文档已作废。

用户输入一个参数：强度系数的标准差，以下用 sigma 来代替。

有三个参数控制 LUT 大小：位宽 x、G 表缩放值 Gs、W 表缩放值 Ws。其中 S 表示位宽 x 最大数加一，如 x 为 3 时，S 为 8

1. MCU 计算强度系数 LUT 表，设其为 G，大小为【4096/Gs】，值的位宽为 x bit。计算方式：
从 0 到 4096 按照如下公式计算，假设当前索引值是 i：

$$G[i] = (S - 1) \times e^{-\frac{1}{2} \left(\frac{Gs \times i}{sigma} \right)^2}$$

2. MCU 计算权重系数 LUT 表，设其为 W，W 的大小为【12*S/Ws】，值的位宽为 x-2 bit。

$$W[i] = \frac{S \times S}{Ws \times i + 4 \times S} - 1$$

3. 数据传入 FPGA，开始计算：

```
# 空间固定 3x3, 注意中间变成了 0
F = np.array([[1, 2, 1], [2, 0, 2], [1, 2, 1]])
new_img = np.zeros(img.shape)
for i in range(1, img.shape[0]-1):
    for j in range(1, img.shape[1]-1):
        # 和中心的差值
        D = np.abs(img[i-1:i+2, j-1:j+2] - img[i, j])
        # 高斯和颜色相乘
        now_W = F * G[D//Gs]
        b = np.sum(now_W)

        if b == 0:
            new_img[i, j] = img[i, j]
        else:
            # 分母
            a = np.sum(now_W * img[i-1:i+2, j-1:j+2]) + img[i, j]*4* S
            # 分子, +1 是因为事先计算 W 的时候减了 1 (否则最大值为S/4, 溢出)
            b = W[b//Ws] + 1
            new_img[i, j] = a * b / (S * S)
return new_img
```

4. LUT 需要 $\frac{4096}{G_s} \times (x) + \frac{2^x}{W_s} \times (x - 2)$ bit
5. 代码在 Denoise 目录中, BF_FPGA.py