双边滤波说明

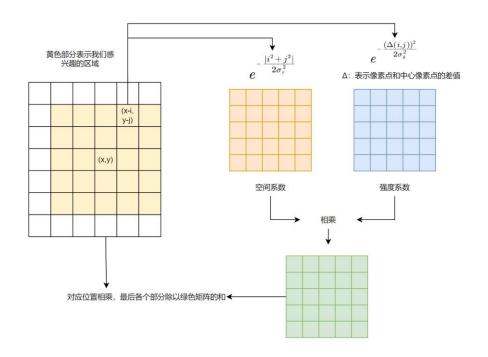
对于一个像素, 双边滤波计算公式如下:

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

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

$$W_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_{\mathbf{s}}}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_{\mathbf{r}}}(|I_{\mathbf{p}} - I_{\mathbf{q}}|). \tag{4}$$

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

用户输入为窗口大小,要求只能为 3*3、5*5、7*7; 公式中的空间标准差忽略 (因为当标准 差乘以 6 和窗口直径接近的时候效果较好,所以直接使用这样的标准差)。

将下面的数据直接存储,根据用户输入调用不同的窗口。

					1	4	7	4
1/16	1	2	1	1/273	4	16	26	16
	2	4	2		7	26	41	26
	1	2	1		4	16	26	16
					1	4	7	4

3	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

用户输入为强度标准差,事先进行运算 Δ 为 0-4095 时的结果,共需要 4096 次除法和 409 6 次指数运算,计算后存储入 LUT。

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。

3*3 大小的双边滤波实现流程

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 S 时, S 为 S

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:

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

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 $\frac{4096}{Gs} \times (x) + \frac{2^x}{Ws} \times (x-2)$ bit
- 5. 代码在 Denoise 目录中, BF_FPGA.py