

## Demosaic 系列二：自适应插值

HA 算法用到了梯度，它的思想是根据梯度判断方向，按照方向来插值。但是方向可能会判错，以及其他的方向也可能有贡献。之后有了一些方法，根据梯度来自适应调整周围像素的权重，来进行加权平均。

### 一、DLMMSE

*Color Demosaicking via Directional Linear Minimum Mean Square-Error Estimation, TIP 2005, Lei Zhang, McMaster University (加拿大一所公立大学)*

引用次数为 429 (截至 2023/11)。作者是香港城市大学的张老师，在去马赛克领域比较有影响力，后续也有一些其他比较好的文章。这篇文章比以往的方法提升了 5 个 dB。

主要贡献在于融合了两个不同方向的预测，而不是以往的二选一（后续会提到）。

### 二、GBTF

*GRADIENT BASED THRESHOLD FREE COLOR FILTER ARRAY INTERPOLATION, ICIP 2010, Ibrahim, Georgia IT (佐治亚理工)*

引用次数为 110 (2023.11)，个人认为是相当不错的文章，效果拔群，运算量也还可以，不知道为啥引用这么低。

回顾 DLMMSE，两个部分：第一部分按照横和竖方向分别计算；第二部分根据方差结合。当然还有第零个部分，就是先拿双线性差值预测，然后计算 R-G 这个差值，之后对差值进行操作。

对于第一部分，DLMMSE 的计算很复杂，如下图公式， $y$  是高斯滤波后的结果，高斯滤波后还要进行花里胡哨的操作。

$$\hat{x} = \mu_x + \frac{\sigma_x^2}{(\sigma_x^2 + \sigma_v^2)}(y - \mu_x)$$

而 GBTF 其实就是拿高斯滤波后的结果作为上图公式中的  $x$  了，也就是没必要还要计算  $x$  平均值、 $x$  方差这些（DLMMSE 是计算高斯滤波后的  $y$  平均值和方差来等价这两个指标）。其实我当时看 DLMMSE 时就疑惑，既然你都假定  $y$  平均值和方差与  $x$  一样，那为什么还要拿  $y$  去计算新的  $x$ 。

下图就是 GBTF 第一部分的公式。第一，忽略几个  $w$ ，就当 1 来看待；第二，只看  $V$  方向先

不看 H 方向；第三，GBTf 中的 f 是均值滤波，不是高斯滤波。因此下面公式其实就是做一个大小为 9 的均值滤波。

$$\begin{aligned}\tilde{\Delta}_{g,r}(i,j) = & [w_N * f * \tilde{\Delta}_{g,r}^V(i-4:i,j) + \\ & w_S * f * \tilde{\Delta}_{g,r}^V(i:i+4,j) + \\ & w_E * \tilde{\Delta}_{g,r}^H(i,j-4:j) * f' + \\ & w_W * \tilde{\Delta}_{g,r}^H(i,j:j+4) * f'] / w_T\end{aligned}$$

对于第二部分，DLMMSE 根据方差进行融合，如下图所示。

$$w_h(n) = \frac{\sigma_{\tilde{x}_v}^2(n)}{\sigma_{\tilde{x}_h}^2(n) + \sigma_{\tilde{x}_v}^2(n)}, \quad w_v(n) = \frac{\sigma_{\tilde{x}_h}^2(n)}{\sigma_{\tilde{x}_h}^2(n) + \sigma_{\tilde{x}_v}^2(n)}$$

上面这个公式有个麻烦地方是：明明是 H 的 w 系数，但是分子确实 V 方向的方差。其实只要转换为下面的式子就好了，这样其实我可以直接抛弃分母，最后除以各个 w 系数之和即可：

$$w_h = \frac{1/\sigma_h^2}{1/\sigma_h^2 + 1/\sigma_v^2}$$

$$\text{let } w_h = \frac{1}{\sigma_h^2}, w_v = \frac{1}{\sigma_v^2} \text{ then } \frac{w_h \times H + w_v \times V}{w_h + w_v}$$

知道这个转换后，理解 GBTF 的论文就轻松了。GBTf 无非就是利用梯度信息来替换 DLMMSE 的方差计算，如下图所示。

$$\begin{aligned}w_N &= 1 / \left( \sum_{a=i-4}^i \sum_{b=j-2}^{j+2} D_{a,b}^V \right)^2 \\ w_S &= 1 / \left( \sum_{a=i}^{i+4} \sum_{b=j-2}^{j+2} D_{a,b}^V \right)^2\end{aligned}$$

### 三、LED

*Low Cost Edge Sensing for High Quality Demosaicking, TIP 2019, Yan Niu, Jilin University*

引用次数 26 (2023.11), 非常简单, 这也能发文章, TIP 是真水了。

本质上就是把 HA 算法的 1 和 0 换成差异值。下图是 HA 算法的实际情况, 要么横要么竖:

The green channel demosaicking process of the HA algorithm, as shown in Eq.4, can be rewritten as

$$\hat{g}(i, j) = \omega_h(\bar{g}_h - \partial_h^2 \mathbf{M}(i, j)) + (1 - \omega_h)(\bar{g}_v - \partial_v^2 \mathbf{M}(i, j)), \quad (8)$$

where

$$\omega_h = \begin{cases} 0 & \text{if } v_h > v_v \\ 1 & \text{if } v_h < v_v \\ \frac{1}{2} & \text{if } v_h = v_v. \end{cases} \quad (9)$$

把 wh 替换成常见的这个 logistic 函数, 然后没了:

It can be shown that the **logistic function**

$$f_k(x) = \frac{1}{1 + e^{kx}}, \quad (12)$$

where  $k$  is a positive real number adjusting the convergence of  $f_k(x)$ , fulfills all requirements on  $\omega_h$ . Thus we define

$$\omega_h = \frac{1}{1 + e^{k(v_h - v_v)}}. \quad (13)$$

It can be verified that

$$1 - \omega_h = \frac{1}{1 + e^{k(v_v - v_h)}}. \quad (14)$$