引导滤波

Dezeming Family

2023 年 2 月 6 日

DezemingFamily 系列文章和电子书**全部都有免费公开的电子版**，可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列电子书，可以从我们的网站 [https://dezeming.top/] 找到最新的版本。对文章的内容建议和出现的错误也欢迎在网站留言。

目录

一　基本介绍

引导滤波 (Guided image filtering) 是 Kaiming He 博士的经典之作，也成为了 OpenCV3 和 Matlab2014 的内置算法。该算法不但可以进行保边平滑去噪，还能够用于去雾，抠图，HDR 压缩，细节增强，联合上采样 (noise reduction, detail smoothing/enhancement, HDR compression, image matting/feathering, haze removal, and joint upsampling) 等多种功能。网上讲解引导滤波的通俗简化的文章有很多了，本文会根据论文 [1] 来介绍它的基本原理，会损失一些通俗性。我们还会在其他文章中介绍各种引导滤波器的变体。

有许多线性平移不变滤波器 (linear translation-invariant (LTI) filters)，比如 Gaussian filter，Laplacian filter 和 Sobel filter，用于图像模糊/平滑、边缘检测以及特征提取 (feature extraction)。LTI 滤波器的核是空间不变的，与图像内容无关，但有些时候我们可能希望将另一幅图像（引导图）的内容集成到当前图像，有两大类方式可以实现：(1) 通过约束求解方法可以实现，并广泛用于黑白着色、图像泊松融合等领域；(2) 通过双边滤波方法。

双边滤波器会出现梯度反转伪影 (gradient reversal artifacts)。注意直接对图像做双边滤波是不会出现图像梯度反转的，但是当使用另一张图作为引导图时，双边滤波会在图像边缘区域出现一条更明显的边，具体可以参考论文原文的 Fig.4、Fig.6 和 Fig.8。

最优化方法也是一种图像滤波技术，通过使用一个二次损失函数 (quadratic cost function) 求解线性系统，等同于通过一个逆矩阵来滤波一幅图像。但是最优化滤波器求解需要不少时间，有些人发现最优化滤波可以用其他显式滤波器来近似，更容易实现，速度也更快。

本文 [1] 中提出了一种新的方法叫做引导滤波器，滤波器输出是引导图的线性变换。它不但可以有效保边平滑，还可以避免双边滤波器中的梯度反转伪影。注意它也与 matting Laplacian matrix（可以参考 Kaiming He 博士的暗通道去雾算法）有关，因此也可以用在其他领域，而不仅仅只是用来平滑图像噪声。

二　算法流程

先把论文公式贴一下：

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

$$W_{ij}^{\mathrm{bf}}(I) = \frac{1}{K_i} \exp(-\frac{|\mathbf{x}_i - \mathbf{x}_j|^2}{\sigma_{\mathrm{s}}^2}) \exp(-\frac{|I_i - I_j|^2}{\sigma_{\mathrm{r}}^2}). \quad (2)$$

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

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

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

$$b_k = \bar{p}_k - a_k \mu_k. \quad (6) \quad \bar{p}_k = \frac{1}{|\omega|}\sum_{i\in\omega_k} p_i$$

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

$$= \bar{a}_i I_i + \bar{b}_i \quad (8) \quad \boxed{\begin{aligned} \bar{b}_i &= \tfrac{1}{|\omega|}\textstyle\sum_{k\in\omega_i} b_k \\ \bar{a}_i &= \tfrac{1}{|\omega|}\textstyle\sum_{k\in\omega_i} a_k \end{aligned}}$$

$$\boxed{a_k = \sum_j A_{kj}(I) p_j \qquad b_k = \sum_j B_{kj}(I) p_j}$$

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

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

对于一个引导图 I 和输入图像 p 以及输出图像 q，设像素 i 周围像素的滤波结果为公式 (1)。

$W_{i,j}$ 是引导图 I 的函数，与 p 无关。对于联合双边滤波，$W_{i,j}$ 可以写为公式 (2) 的形式。当引导图和原图相同时，则公式 (2) 从联合双边滤波退化为原始的双边滤波方法（权重同时考虑临近像素的距离和临近像素颜色）。

对于引导滤波，假设引导滤波器是引导图 I 和滤波器输出结果 q 之间的局部线性模型。设中心在像素 k 的窗 ω_k，假设 q 是 I 在窗 ω_k 内的线性变换，见公式 (3)。(a_k, b_k) 是在 ω_k 内的常量线性系数，我们使用半径为 r 的方窗。该局部线性模型由于 $\nabla q = a\nabla I$，所以当 I 有一个边时，q 也会有一个边。

我们求解公式 (3) 的系数，以最小化 q 和 p 的差别，即公式 (4)。这里的 ϵ 是一个正则化参数，避免 a_k 太大（后面会描述其重要性）。我们可以看到，公式 (4) 的损失值的计算包含了一个窗内的全部像素。仅思考一个窗口，则输出结果与引导图是成正比的，损失值的意义更像是一种能量约束，比如如果引导图的像素值都是好几百，输入图像值 p_i 都是小于 1 的数，使用约束项就可以使得输出图像的能量更接近于输入图像。

其解可以用线性回归得到，见公式 (5) 和 (6)。这里的 μ_k 和 σ_k^2 表示 ω_k 内的均值和方差，$\bar{p}_k$ 是 p 在窗 ω_k 内的均值。

当把线性模型扩展到整个图像上时，注意一个像素 i 会出现在多个窗内，在这多个窗中计算的结果很可能是不同的，一个方案是取平均，即公式 (7) 和 (8)。

由于线性系数 $(\bar{a}_i, \bar{b}_i)$ 在空间上是变化的，此时 ∇q 就不再是 $nablaI$ 的放缩。不过由于 $(\bar{a}_i, \bar{b}_i)$ 是平均滤波器的输出，在很强的边缘区域，$(\bar{a}_i, \bar{b}_i)$ 的梯度会比 I 的梯度小得多（这里的梯度表示“变化率”，意味着各个窗的 $(\bar{a}_i, \bar{b}_i)$ 值几乎相同，下面有蓝字解释），因此也能得到 $\nabla q \approx \bar{a}\nabla I$，即 I 中的数据值突变（梯度较大的位置）可以大部分被保留在 q 中。

解释一下上面的内容，比如只考虑两个窗 ω_j 和 ω_l，它们叠加的区域中的某个像素 i，$q_i = \frac{1}{2}(a_j + a_l)I_i + \frac{1}{2}(b_j + b_l)$。前面说过，每个窗口计算的系数 (a_k, b_k) 更像是一种能量约束，注意 a_k 的计算，如果 I 在两个窗叠加的区域中有一条强边，那么这条边会使得参数 $(\bar{a}_i, \bar{b}_i)$ 在多个窗口中都会有类似的值。

经过改写成权重和的形式，可以得到公式 (9) 的形式。经过进一步计算可以发现权重和满足相加为 1，因此不用再归一化。

三　当 $I = p$ 时的保边平滑效果

假设 $I = p$，$\epsilon = 0$，则公式 (4) 的解就是 $a_k = 1, b_k = 0$：

$$q_i = p_i \tag{三.1}$$

如果 $I = p$，$\epsilon > 0$，则考虑两种情况：(1) 当 I 在窗口内是常量，则 $a_k = 0, b_k = \bar{p}_k$；(2) 当 I 方差比较高，那么 a_k 趋近于 1，而 b_k 趋近于 0：

$$case(1):\ \ q_i \approx \bar{p}_k \tag{三.2}$$

$$case(2):\ \ q_i \approx p_i \tag{三.3}$$

当结合公式 (8) 的平均以后，如果像素在高方差区域的中心，那么该值不变（把一堆值接近为 1 的 a_k 取平均，值仍然接近于 1），如果在较为平坦的区域，该值就会变为临近像素值的平均。ϵ 用来评判一个区域是平滑的还是高方差的，当一个区域的方差小于 ϵ 就会被平滑掉，一个区域的方差大于 ϵ 就会被保留。注意这里的 ϵ 和双边滤波中的方差起到了同样的效果。

边缘区域以及高细节区域的方差都会很大，因此引导滤波是保边平滑的。

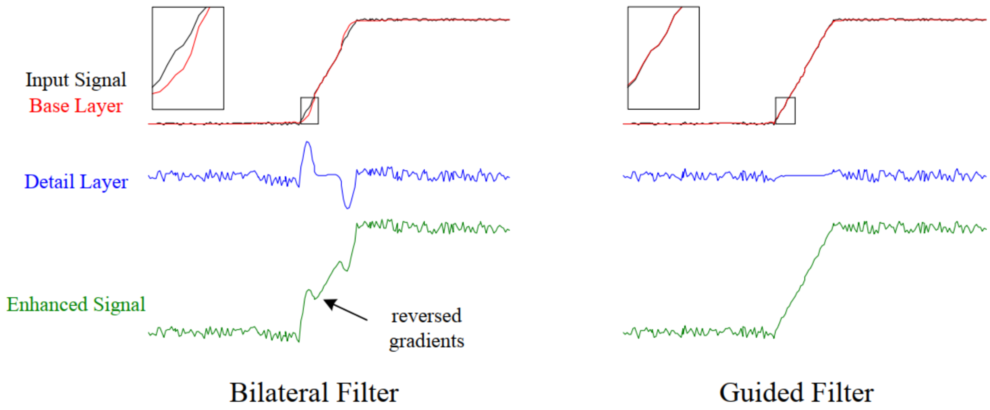
四　滤波核的保边性

假设在一个窗口中，引导图的中心像素值为 I_i，邻域有 I_j, I_l。假设该窗口的均值是 μ_k，且 $I_j < \mu_k < I_i \approx I_l$ 那么，可知公式 (9) 中的 $1 + \frac{(I_i-\mu_k)(I_j-\mu_k)}{\sigma_k^2+\epsilon}$ 会非常小，意味着 i 点处的内容不会被 j 点处的内容平均掉。且 $1 + \frac{(I_i-\mu_k)(I_l-\mu_k)}{\sigma_k^2+\epsilon}$ 值比较大，也就是相近的强边信息会被保留。

当 σ_k^2 趋近于 0 时，会导致公式 (9) 中的所有 $1 + \frac{(I_i-\mu_k)(I_j-\mu_k)}{\sigma_k^2+\epsilon}$ 都接近于 1，那么就相当于低通滤波。

五　引导滤波的保梯度性

引导滤波可以避免梯度反转现象，比如对于下面的信号增强例子，黑色是输入信号，红色是保边平滑输出，用于作为基信号，基信号和输入信号的差作为细节层信号（蓝色）。蓝色信号被放大以增强细节。增强信号（绿色）是增强的细节层和基信号的组合。



由图可知，增强后的信号出现了梯度反转的现象，这种现象会在边缘处比较常见。因为在靠近边缘时，原本应该低的值会被高斯滤波器拉高，比较高的值会被高斯滤波器拉低（本质在于双边滤波的保边性没有那么好）。

而对于引导滤波，由于在边缘处 $\nabla q \approx \bar{a}\nabla I$

六　与 Matting Laplacian Matrix 之间的关系

引导滤波不但可以用于平滑算子，还与 matting Laplacian matrix（可以参考去雾算法）有关，这就带来了新的应用（比如 matting 和去雾算法）。

$$E(\alpha) = (\alpha - \beta)^{\mathrm{T}}\Lambda(\alpha - \beta) + \alpha^{\mathrm{T}}\mathrm{L}\alpha, \quad (10)$$

$$\mathrm{L}_{ij} = |\omega|(\delta_{ij} - \mathrm{W}_{ij}), \quad (12)$$

$$\mathrm{L}_{ij} = \sum_{k:(i,j)\in\omega_k}(\delta_{ij} - \frac{1}{|\omega|}(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon})). \quad (11)$$

在 matting 中，公式 (10) 中的 L 叫做 matting Laplacian 矩阵，L 大小是 $N \times N$ 的（N 是图像像素数），每个元素的定义如公式 (11)；β 是 trimap（在去雾中，就是细化前的暗通道图）。比较一下公式 (11) 和公式 (9)，可以看到 matting Laplacian 矩阵可以写成公式 (12) 的形式，附加材料中证明，引导滤波器是优化过程的一次 Jacobi 迭代，如果 β 是一个 matte 的比较好的猜测（比如在去雾中的暗通道图就是比较好的猜测，而 matting 中的涂鸦就是不太好的猜测），那么我们可以运行一步 Jacobi 过程（在去雾中，就是用原图对暗通道图进行滤波）：

$$\alpha_i \approx \sum_j W_{i,j}(I)\beta_j$$

七　扩展到 RGB 空间

引导滤波算法是 $O(N)$ 量级的（至于为什么它是 $O(N)$ 量级的，有些人可能不理解，觉得它既然有个窗口，那么窗口越大不就意味着每个像素要求平均的区域就得越大吗？实际上它的实现是借助了 box 滤波器，这是一种用空间换取时间效率的线性时间滤波器），意味着它的滤波跟窗口半径 r 没有关系，可以用任意大的窗口进行滤波。

为了去泛化彩色引导图，重新局部线性模型为：

$$q_i = \mathbf{a}_k^{\mathrm{T}}\mathbf{I}_i + b_k, \forall i \in \omega_k. \quad (13)$$

$$\mathbf{a}_k = (\Sigma_k + \epsilon \mathrm{U})^{-1}(\frac{1}{|\omega|}\sum_{i\in\omega_k}\mathbf{I}_i p_i - \mu_k\bar{p}_k) \quad (14)$$

$$b_k = \bar{p}_k - \mathbf{a}_k^{\mathrm{T}}\mu_k \quad (15)$$

$$q_i = \bar{\mathbf{a}}_i^{\mathrm{T}}\mathbf{I}_i + \bar{b}_i. \quad (16)$$

引导滤波器就写为公式 (14)(15)(16)。$\mathbf{I}_i$ 就变为了 3×1 的颜色向量，$\mathbf{a}_k$ 是一个 3×1 的系数向量；Σ_k 是引导图 $\mathbf{I}$ 的窗口 ω_k 内的协方差矩阵，U 是与 ω_k 同样大小的单位矩阵。

很多人在实现的时候，使用自身当引导图时，就会三个通道分别计算（使用公式 (5)(6)），把三通道分别自引导来去噪，而不是整体使用彩图。但是如果用 $\mathbf{I}$ 和 $\mathbf{p}$ 不同，则需要使用 (14)(15)(16)。

八　算法流程与代码实现

算法分为两个，一是原始的引导滤波算法，二是基于下采样滤波再上采样恢复的快速算法。本文只介绍基础算法的实现，并会在《快速引导滤波》文章中介绍快速算法的实现。

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: $q = \text{mean}_a.*I + \text{mean}_b$
/* f_{mean} is a mean filter with a wide variety of O(N) time methods. */

Algorithm 2 Fast Guided Filter.

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

滤波算法的实现是根据公式 (5) 和 (6)（注意公式 (9) 只是写成权重和的形式，为了做进一步分析使用）。

代码实现如下，cv2.blur() 就是平局滤波：

```python
def guided_filter(I,p,win_size,eps):

    mean_I = cv2.blur(I,(win_size,win_size))
    mean_p = cv2.blur(p,(win_size,win_size))

    corr_I = cv2.blur(I*I,(win_size,win_size))
    corr_Ip = cv2.blur(I*p,(win_size,win_size))

    var_I = corr_I-mean_I*mean_I
    cov_Ip = corr_Ip - mean_I*mean_p

    a = cov_Ip/(var_I+eps)
    b = mean_p-a*mean_I

    mean_a = cv2.blur(a,(win_size,win_size))
    mean_b = cv2.blur(b,(win_size,win_size))

    q = mean_a*I + mean_b
    return q
```

其中，公式 (5) 可以写为：

$$\frac{\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i p_i - \frac{1}{|\omega|}\sum_{i\in\omega_k}\mu_k \overline{p}_k}{\sigma_k^2+\epsilon} \tag{八.1}$$

代码中的 corr_Ip 就对应了 $\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i p_i$，mean_I*mean_p 对应于 $\frac{1}{|\omega|}\sum_{i\in\omega_k}\mu_k \overline{p}_k$。

参考文献

[1] K. He, J. Sun, and X. Tang. Guided image filtering. In ECCV, pages 1-14. 2010.

[2] He, Kaiming, Jian Sun, and Xiaoou Tang. "Guided image filtering." IEEE transactions on pattern analysis and machine intelligence 35.6 (2012): 1397-1409.