Amortized supersampling

Dezeming Family

2022 年 12 月 18 日

正常字体：表示论文的基本内容解释。
粗体：表示需要特别注意的内容。
红色字体：表示容易理解错误或者混淆的内容。
蓝色字体：表示额外增加的一些注释。
绿色字体：表示额外举的一些例子。

目录

abstract

我们提出了一种实时渲染方案，该方案重用早期时序帧中的着色样本，以实现程序着色器的抗锯齿效果。使用重投影策略，我们以亚像素精度维护多组着色估计，并逐步更新这些估计，以便对于大多数像素每帧仅评估一个新的着色样本。

关键的困难是防止在连续的重投影过程中累积模糊。我们对重投影方法引入的模糊进行了理论分析。基于这一分析，我们介绍了一种非均匀空间滤波器、一种自适应递归时间滤波器和一种用于局部估计空间模糊的原理方案。

我们的方案适用于随时间缓慢变化的抗锯齿着色属性（意味着光照不能剧烈变化，同时，高光随着视角移动时也不能变化太快）。它只需在商业图形硬件上进行一次渲染，就可以提供质量超过 4×4 分层超采样的结果，这只需一小部分计算成本。

一 Introduction 和相关工作

常见的抗锯齿方法有两种，这两种都是可以硬件支持的。一种是基于预滤波的 mipmapped 纹理，另一种是 framebuffer 的超采样。

对于一些实时应用中，可能包含程序化材质或者复杂的着色函数，这跟预滤波的纹理不同，这些信号通常不是带宽受限 (bandlimited) 的（[Ebert et al. 2003]）。因此，对这种情况产生有限带宽版本的程序着色器比较困难。程序化材质有两种含义：(1) 是对创作过程的描述，不是保存最终纹理结果，而是记录创建过程；(2) 材质是随着时间或者交互变化的。这里是第二种意思。程序着色器 (Procedural shaders)（或简单的着色器）是通过数学和算法方法创建的纹理。它是在 GPU 或图形卡上运行的一段代码。它们可以提供一系列效果，例如使对象看起来像卡通或模拟蜡烛火焰。

抗锯齿技术一般是增加空间采样率，然而对于复杂的着色计算多次会很消耗时间。幸运的是，给定任意表面点，表面着色的昂贵的计算（比如 albedo）变化比较缓慢，甚至在时序中是常量。许多技术可以自动将程序着色器分解为静态和动态层。我们的想法是以较低的时时序速率对静态和弱动态层进行采样，以在相同的计算预算下实现较高的空间采样率（用时序样本弥补空间样本的不足）；强动态层可以以原来的分辨率或者超分辨率来采样。

我们的技术，仅仅对着色计算中的静态或者弱动态组件估计一次，并重用先前帧的样本来实现好的空间抗锯齿。我们也是借助了时序重投影，即把当前帧的信息重投影到先前帧来获得样本。先前有些工作，比如 [Nehab et al. 2007] 重用两帧之间的着色信息；[Scherzer et al. 2007] 重用时序之间缓慢变化的 shadow map。

由于场景是运动的，先前帧的样本集可能由于遮挡等原因没法再使用。因此，时-空样本对于重建来说，它的结构性远不如空间分层样本的结构性好。

我们的方法是基于先前的抖动采样 (jittered sampling) 和递归指数平滑 (recursive exponential smoothing)。但我们有一个重要的贡献，即对时空样本造成的模糊进行了理论分析。通过自适应地调整平滑因子，基本的重投影算法可以在静态场景的平稳视觉中收敛到完美的重建（无限超采样）（见论文第 4.1 节）。

贡献主要是：

- 使用多个更高分辨率的子像素 buffer 来保留重投影估计值。
- 对这些子像素缓冲器进行不规则的循环更新，以提高重建质量，同时仍然只需要每帧每个像素一次采样估计。
- 一个原理性的方法来估计和限制重投影和指数平滑中的模糊。
- 对未遮挡像素的自适应的额外样本估计来减少伪影。
- 一个对时序变化慢的着色的估计的策略。

摊余超采样适合于实现在消费级显卡的光栅管线上，它是轻量级的，不需要预处理。而且它可以与现有的硬件超采样技术结合。它实现的效果质量上可与 4×4 的分层超采样相媲美，且计算消耗低很多（见图 (1)d）。

相关工作主要是三个方面：数据缓存和重用；程序着色器抗锯齿；渲染视频的后处理。

对于数据缓存，以前的方法主要目的是重用着色结果（比如光线追踪昂贵的光照计算），我们主要是为了做时序抗锯齿，我们也分析了重投影带来的模糊。

对于抗锯齿，mipmaps 是经常被使用的，频域处理也会使用；然而最常采用的还是超采样技术。

对于视频渲染，像素追踪滤波与我们的技术比较相关。对于渲染好的视频序列，追踪不同帧的点在屏幕空间的位置关系，然后实现空间抗锯齿。

二　重投影回顾

论文中的全部公式：

$$f_t[p] \leftarrow (\alpha) s_t[p] + (1-\alpha) f_{t\text{-}1}(\pi_{t\text{-}1}(p)). \tag{1}$$

$$f_N[p] \leftarrow \frac{1}{N}\sum_{i=1}^{N} S(p + g_i[p]). \tag{2}$$

$$\mathrm{Var}(f_N[p]) = \frac{1}{N}\mathrm{Var}(f_1[p]), \tag{3}$$

$$f_t[p] \leftarrow (\alpha_t[p])\, S(p + g_t[p]) + (1-\alpha_t[p])\, f_{t\text{-}1}[\pi_{t\text{-}1}(p)]. \tag{4}$$

$$\alpha_t[p] = \frac{1}{t}, \tag{5}$$

$$\alpha_t[p] \leftarrow \frac{1}{N_{t\text{-}1}[p]+1} \tag{6}$$

$$N_t[p] \leftarrow N_{t\text{-}1}[p] + 1. \tag{7}$$

$$N_t[p] \leftarrow \left(\alpha_t[p]^2 + \frac{(1-\alpha_t[p])^2}{N_{t\text{-}1}[\pi_{t\text{-}1}(p)]}\right)^{-1}. \tag{8}$$

$$f_t[p] = \sum_{i=1}^{n(t,p)} \omega_{t,i}\, S(p + \delta_{t,i}[p]) \quad \text{with} \quad \sum_{i=1}^{n(t,p)} \omega_{t,i} = 1. \tag{9}$$

$$\sigma_t^2[p] = \tfrac{1}{2}\mathrm{Var}_x(\{\delta_{t,i}[p]\}_{i=1}^{n(t,p)}) + \tfrac{1}{2}\mathrm{Var}_y(\{\delta_{t,i}[p]\}_{i=1}^{n(t,p)}). \tag{10}$$

$$\sigma_v^2 = \sigma_G^2 + \frac{1-\alpha}{\alpha}\,\frac{v_x(1-v_x)+v_y(1-v_y)}{2}. \tag{11}$$

$$b_{i(t)}[p] \leftarrow (\alpha)\, s_t[p] + (1-\alpha)\, \tilde{b}(p + \phi_{i(t)}). \tag{12}$$

$$\Lambda_r(v) = \mathrm{clamp}(1 - \tfrac{|v_x|}{r}, 0, 1)\ \mathrm{clamp}(1 - \tfrac{|v_y|}{r}, 0, 1). \tag{13}$$

$$p_k = \pi_{t\text{-}k\text{-}1}(p + \phi_{i(t)}) - \phi_{i(t\text{-}k\text{-}1)}, \quad k \in \{0,1,2,3\}. \tag{14}$$

$$r_k = \left\| J_{\pi_{t\text{-}k\text{-}1}}[p] \begin{pmatrix}0.5\\0.5\end{pmatrix} \right\|_\infty, \tag{15}$$

$$\tilde{b}(p + \phi_{i(t)}) = \frac{\sum_{k,\Delta} w_{k,\Delta}\, b_{i(t\text{-}k\text{-}1)}[\lfloor p_k \rfloor + \Delta]}{\sum_{k,\Delta} w_{k,\Delta}}, \tag{16}$$

$$w_{k,\Delta} = \Lambda_{r_k}(p_k - (\lfloor p_k \rfloor + \Delta)). \tag{17}$$

$$\tilde{b}(p + \phi_{i(t)}) = \frac{\sum_k w_k\, b_{i(t\text{-}k\text{-}1)}(\lfloor p_k \rfloor + o_k)}{\sum_k w_k} \tag{18}$$

$$o_k = \begin{pmatrix} \frac{w_{k,\binom{0}{1}} + w_{k,\binom{1}{1}}}{w_k} & \frac{w_{k,\binom{1}{0}} + w_{k,\binom{1}{1}}}{w_k} \end{pmatrix}^{\mathrm{T}}. \tag{19}$$

$$f_t[p] \leftarrow \left(\frac{\alpha}{K}\right) s_t[p] + \left(1 - \frac{\alpha}{K}\right) \tilde{f}(p). \tag{20}$$

$$\alpha_t[p] \leftarrow K \max\left(\frac{1}{N_{t\text{-}1}+1},\ \alpha_{\tau_b}(v)\right). \tag{21}$$

$$\epsilon_t[p] = (S * G)_t(p) - f_t[p]. \tag{22}$$

$$e_k[p] = s_{t\text{-}k}[p] - b_{i(t\text{-}k)}[p],\ k \in \{0,1,2,3\}. \tag{23}$$

$$e_{\mathrm{smin}}[p] = e_j[p] \quad \text{where} \quad j = \arg\min_k |e_k[p]|, \tag{24}$$

$$\hat{\epsilon}_t[p] = (B_3 * e_{\mathrm{smin}})[p] \tag{25}$$

$$\alpha_t[p] \leftarrow K \max\left(\frac{1}{N_{t\text{-}1}+1},\ \alpha_{\tau_b}(v),\ \alpha_{\tau_\epsilon}\right), \tag{26}$$

$$\alpha_{\tau_\epsilon} = 1 - \frac{\tau_\epsilon}{|\hat{\epsilon}_t[p]|}. \tag{27}$$

我们把当前帧的表面点重投影到先前帧，测试它们之间的深度是否是匹配的。

令 f_t 表示时间 t 中缓存的像素属性，d_t 表示像素深度，$p = (x, y)$ 表示像素。定义重投影运算符 π，将 t 时刻的像素 p 重投影到 $t-1$ 时刻的位置：$(x', y', z') = \pi_{t-1}(p)$。如果 z' 和上一帧的深度 $d_{t-1}(x', y')$ 之间差距很小，则认为它们之间是相关的。

递归指数平滑是常用的混合方法，定义一个 α 值，计算 $f_t[p]$ 可见公式 (1)，其中 $s_t[p]$ 是当前像素。

三　Amortized supersampling 的理论

空间抗锯齿可以用低通滤波器完成。我们使用 jitter 方法，见公式 (2)。$g_i[p]$ 表示抖动偏移，这是根据高斯分布来采样的随机值。此时估计器的方差见公式 (3)，其实就是根据多个抖动样本的平均值的方差会比一个样本的方差低。根据公式 (4)，可以将计算 (2) 的和的计算成本平摊到多帧中。

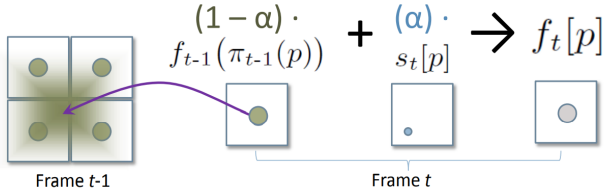
我们首先提供一个对于稳定视点和静态场景的抗锯齿方案，然后考虑更一般的任意场景。

3 1 静态场景和稳定视点

静态场景中，$a_t[p]$ 是随着时间逐步降低的，见公式 (5)，使得样本的不断累积时所有样本都具有均匀的权重。

3 2 移动的视点和动态场景

重投影到上一帧以后，采样得到上一帧的值，然后与当前帧融合：



3 2.1 可见性的改变

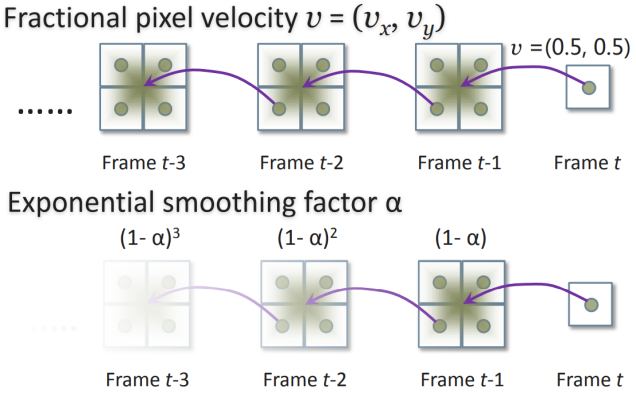
相邻帧之间的像素的可见性会变化，我们需要记录对当前帧每个像素的有效样本量。更精确地说，我们存储每个像素的方差减少因子 $N_t[p]$（其实这就是每个像素的有效样本量，公式 (3) 表明了方差减少和样本量之间的关系），使用该值来设置平滑因子 $\alpha_t[p]$。注意下式中由于 f_1 的方差大于 f_t 的方差，所以 N_t 一般都是大于 1 的：

$$N_t = \frac{Var(f_1)}{Var(f_t)} \tag{三.1}$$

当表面点第一次可见时，$\alpha_t[p]$ 和 $N_t[p]$ 都设为 1，之后按照公式 (6) 和公式 (7) 来逐帧改变。但正如稍后讨论的，场景运动时要求我们限制均匀加权样本的无限累积——即使对于在许多帧上仍然可见的像素也是如此。公式 (8) 是导出的新的 $N_t[p]$ 的更新规则见公式 (8)，后面还有对公式 (6) 的新形式。如果对 $\alpha_t[p]$ 的更新规则仍旧用 (6)，那么公式 (8) 其实就可以化简为公式 (7)。公式 (8) 的推导见附录（对于公式 (8)，其实就是探究方差的变化，即当先前样本权重 $1-\alpha$ 与当前样本权重 α 在每帧都变化时，某像素的方差减少的速度）。

3 2.2 对重采样带来的模糊进行建模

重投影得到的位置的值需要附近的值插值来得到，这个插值就是 resampling。随着一帧一帧地进行，重复地进行重采样其实就会增加对像素的样本贡献。临近区域样本的半径是随着时间增长的，见图 (2)，会导致模糊现象。图 (2) 我做一点解释：α 越大就表示当前帧的贡献越大，蓝框表示当前像素格子范围，假设对某点有贡献的样本会随着时间运动。Fractional pixel velocity 也就是一个像素在一帧的偏移（这个偏移是在一个格子内的）。



公式 (9) 意味着一个像素值是多个采样点的加权和。$n(t,p)$ 表示对于时间 t 像素 p 的样本量。一个像素的模糊度可以根据两个维度的平均加权方差来表示，即公式 (10)（$\delta_{t,i}[p]$ 表示由于 jitter 和重投影导致的偏置，也就是偏离像素中心的距离）。公式 (10) 的推导是借助了协方差矩阵，协方差矩阵的两个特征值表示最大最小方差，这里用平均值来表示模糊程度。尽管用闭式表达式对于任意场景来说并不够很好，但

可以提供一个理论分析的基础。其他的运动类型可以用平移运动来近似，每个像素的运动重采样计算的分数速度是（$\lfloor \pi_{t-1}(p) \rfloor$ 是向下取整函数）：

$$v = \pi_{t-1}(p) - \lfloor \pi_{t-1}(p) \rfloor \tag{三.2}$$

当使用公式 (4) 描述的常量平滑因子时，当 $t \to \infty$ 时期望的模糊方差会收敛于公式 (11)。

为什么会与 v_x 和 v_y 有关，是因为像素如果重投影以后不会偏移，而是正好处于像素中心，那么就永远不会因为对周围像素的插值造成模糊。那么模糊程度就仅仅与高斯滤波器有关（然而这是我们自己选择的图像重采样滤波器，哪怕场景和相机都不动，该项也是存在的，因此并不需要在意）。

图 (3) 中左边是实际渲染时计算的模糊方差，右边是根据公式 (11) 推导计算的模糊方差，可以看到它们几乎是差不多的。当分数速度不变时，α 小则模糊方差越大，这是因为历史帧的贡献也会变大，导致模糊方差增加；当 α 不变时，分数速度越高则模糊方差越大。

解决方案主要有两个（前面的推导主要也是为了导出当前的这个结果）：

- 增加历史帧分辨率（这样可以限制 v_x 和 v_y 的值，注意这两个值在 0 或者 1 时会减少模糊方差）。
- 当需要时，限制 α。

四　算法描述

我们的抗锯齿算法使用多个子像素缓冲区来限制模糊量，在未遮挡的像素处调整样本估计。

4 1　子像素 buffer

使用两倍于屏幕分辨率来渲染，意图降低公式 (11) 中的 $v(1-v)$。我们将估计值与每个像素的四个象限相关联，并以循环方式更新这些象限。这四个象限建立成 4 个独立的子像素 buffers$\{b_k\}$，每个子像素 buffer 的分辨率都是原屏幕分辨率。每个子像素的 x 和 y 坐标偏置都是 0.25，要么减 0.25，要么加 0.25。

注意，在没有场景运动的情况下，这四个子像素缓冲区有效地形成了更高分辨率的帧缓冲区。然而，在场景运动下，在先前帧中计算的子像素样本重新投射到偏移位置，如图 (4) 所示。图 (4) 中，在更新某一个子像素时，用 tent 滤波器对框内的像素样本进行滤波。当没有运动时，每个子像素中心对应的框内只有当前像素，因此子像素不会被模糊。当有运动时，框内就会有其他子像素，我们的方法选择最接近所需象限中心的样本来限制模糊量。图 (4) 演示的是如何更新子像素 buffers，得到的 $\tilde{b}_t$ 就是子像素值。

每帧中我们计算一个新的样本，然后更新子像素 buffer，然后用这些子像素 buffers 的值加权得到最终像素值。然后还需要更新有效样本量 $N_t[p]$。

图 (5) 表示了利用每帧新采样的样本来更新子像素 Buffer 和当前像素值的过程，当前像素值（渲染结果）存储在 f_t 表示的 framebuffer 里。这些过程主要就是涉及滤波和加权融合，这里使用的是双线性 tent 滤波器来更新子像素 buffer 的值；更新完子像素，然后再计算当前的像素值，见公式 (20)。之后再根据公式 (8) 来更新有效样本数 N_t。

4 2　限制模糊程度

其实就是限制 α 的值，让它不要太小，但是要在能容忍的模糊范围内尽可能足够小。

给定一个关于模糊量（样本分布方差）的阈值 τ_b 和速度 v，我们可能想去计算更小的平滑因子 $\alpha_{\tau_b}(v)$，使得在不超过模糊阈值的前提下达到更好的抗锯齿效果。

在更新子像素时，顺序并不一定是 0,1,2,3,0,1,2,3 这样，而是定义了一个新的顺序，防止 artifacts。此外，不是限定 σ^2，而是限定 $\sigma^2 + \mu^2$。这样就同时限定了模糊 (blur) 和偏移 (drift)。

选择合适的 τ_b 是很重要的，可以参考图 (8) 的结果。

五　考虑信号的变化

考虑当光照等因素变化时，需要消除时序伪影。

我们需要估计时序变化时，当前的着色结果与真实值之间的差别。这个残差见公式 (22)，S 表示真实的图像信号，由于正确的当前像素值我们是没有的（需要抖动采样很多当前样本才能得到，但我们只能采样一个样本），所以只能用当前采样值来估计残差。要注意这个当前采样值是有锯齿的，我们假设信号的时间变化均匀地影响表面的相邻区域，否则我们的策略将失效（见论文第 8 节）。

为了限制残差，导出了另一个公式 (26) 和公式 (27)，τ_ϵ 是我们自己选择的残差容忍度值。

六　启发性思考

本文其实就是在不同指标之间进行权衡：模糊度、锯齿、时序伪影。只不过作者从数学的角度推导了他们之间的关系，并且意图从数值上进行限定。当然我觉得子像素 buffer 确实是个有意义的建议，关于如何更新 α 和 N_t，这里有一个关键点，即公式 (3)，表明了方差和有效样本量之间的关系。而有效样本量不一定要求是整数，因为有的样本“没有那么有效”，因此需要一些计算和关系来得到过去样本的有效性。

技术的主要限制是它无法正确检测和消除阴影中的任意时序变化，例如快速移动的镜面反射高光和阴影边界，以及视差遮挡贴图。这些类型的影响无法通过我们的重投影框架准确预测，因为它们涉及相对于物体表面快速移动的信号。

本文可以说在如何有效利用过去帧样本上迈出了重要的一步，关于历史帧样本应该如何存储和有效利用，应该还是有不少值得深入挖掘的地方。

参考文献

[1] xxx