# 线性预测自适应渲染（2015-TOG）

Dezeming Family

2021 年 5 月 12 日

DezemingFamily 系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列书，可以从我们的网站 [https://dezeming.top/] 找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

## 目录



## 一 基本介绍

本算法来自论文 [1]，我认为使用线性模型来拟合渲染结果是一个很不错的思路，因此我把这篇文章讲解一下。

本算法是为了降低蒙特卡洛（MC) 光线追踪（RT) 的噪声而设计的 [1]，它是基于重建（将渲染时得到的样本进行一定的处理，组合得到最终图像）的预测方法，可以去噪的同时保留物体的边缘信息（不会把边缘模糊）。

该算法构建多个稀疏的线性模型（稀疏是相对于图像空间来说的，即多个像素共同使用同一个参数的模型，因此比较“稀疏”），每个线性模型都有预测窗来根据几个样本预测真实结果（ground truth）。同时，该算法递归地估计不同的预测窗的线性预测带来的预测错误，并选择最优窗来最小化误差。

由于在模型的最优预测区间中，每个样本都可以预测多个像素，因此本算法在图像空间只需要比较稀疏的像素即可。不过用一个线性模型来预测多个像素很难，需要对区域进行误差分析，而不仅仅对单个像素进行误差分析。

经过验证，该算法可以用更少的重建时间并能得到更好的重建质量。

# 二　相关算法

自适应渲染是为了使用有限 MC 样本量得到更高质量的图像，一般有两种方法：多维度自适应采样和图像空间自适应采样。多维度方法在光线跟踪的路径中使用，缺点是每个样本都需要昂贵的计算（比如对光源采样多次）。图像空间方法利用了无噪声信息，比如 G-buffer(Ray 与物体相交的世界坐标、法向量、材质纹理以及几何 mesh 编号等)。

在多维度自适应渲染上，主要是基于路径空间做的；还有频率分析方法，可以重建景深、运动模糊等效果；重投影技术可以复用时序样本。但是这些算法都只支持有限的渲染效果（比如有的只支持渲染漫反射物体，有的只支持运动模糊，有的支持两三种效果等）。

图像空间自适应渲染，比如小波、高斯滤波器，利用输入信息（颜色、方差等）进行滤波。基于图像空间的方法很容易并行化，然而计算量大，很多都不适合高性能和实时计算（也有很多可以用于实时计算的）。

本文作者在 2014 年也发布了一篇论文，利用特征，使用线性模型近似渲染结果。通过把重建误差解耦为bias 和 variance来估计线性模型的误差，并对于不同的特征估计最优滤波器的带宽来最小化误差。但是需要昂贵的计算，同时模型是针对每个像素的。本论文算法用单个线性模型来同时重建多个像素，因此昂贵的误差估计只需要在稀疏的像素上执行，提升了渲染性能。

# 三　算法创新点

该算法在线性模型重建上很费时，但只需要少量像素样本就能计算得到重建模型，并预测其他像素样本上的值。创新点总结如下：

- 它使用了递归最小二乘法（recursive least squares, RLS）来迭代地构建多个线性模型。给定不同尺寸的预测窗后，递归地估计线性模型的参数。
- 设计递归的误差分析来估计线性模型带来的预测误差，根据误差分析选择最优的预测尺寸。
- 提供了自适应采样方法，基于评估预测的误差，在高噪声区域进行更多样本的采样。
- 预测方法在保持高重建质量的同时，只在稀疏的像素数上运行预测优化方法。

# 四　使用线性模型重建图像

## 预备知识

squared residuals 平方残余在线性拟合中特指估计值与真实值的差的平方。

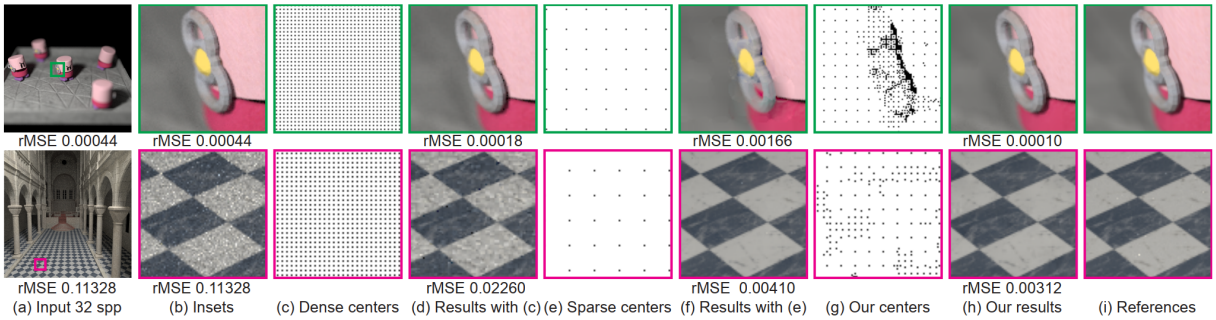
## 算法描述

本算法用 $y$ 表示 MC 方法获得的直接结果，来估计渲染的真实结果 $f(x)$，用下标 $i$ 表示第 i 个像素。

我们定义以像素 $c$ 为中心的滤波窗为 $\Omega_c^F$，同时还定义一个预测窗 $\Omega_c^P(k) \subseteq \Omega_c^F$，该预测窗里有 k 个像素。我们需要注意的是，滤波窗的尺寸是固定的，但是预测窗的大小可变（因为我们可能只需要里面一部分样本来进行预测）。下面公式表示中心像素与周边像素之间的梯度关系：

$$f(\boldsymbol{x}_i) \approx f(\boldsymbol{x}_c) + \nabla f(\boldsymbol{x}_c)^T(\boldsymbol{x}_i - \boldsymbol{x}_c) \tag{四.1}$$

因此我们计算出梯度以后就可以用它来估计周边像素值。当然我们并不知道 ground truth $f(x)$，也不知道梯度，但我们可以通过最小二乘来估计它们。我们结合图示来说明这个问题，首先需要注意的是，该算法的预测中心虽然是稀疏的，但不是均匀的，比如下图：




上图中 (a) 表示输入图像，(b) 表示输入图像某区域放大后的结果。(c) 和 (d) 表示使用稠密的预测中心生成的预测结果，(e) 和 (f) 表示稀疏的均匀预测中心预测的结果，(g) 和 (h) 表示本算法的稀疏非均匀预测中心得到的结果，可以看到，该算法自适应地选择预测像素中心的位置，且误差（rMSE）更小。

本算法的主要方法是在一个预测窗里使用单个线性模型来估计多个像素值的重建，而困难是寻找一个最优的局部估计窗来最小化预测误差，设 $\widehat{f}(\boldsymbol{x}_i)$ 是第 i 个像素的估计值：

$$\epsilon_c(k) = \frac{1}{k} \sum_{i \in \Omega_C^P(k)} (\widehat{f}(\boldsymbol{x}_i) - f(\boldsymbol{x}_i))^2 \tag{四.2}$$

- 合理构建预测窗的间隔（k 值）是一个挑战，基于 k 值的大小来构建相应的线性模型并不容易。同时，我们还得想办法估计预测误差，如何选择最优 k 值来最小化预测误差也在这个挑战之中。
- 另外，在计算多个线性模型后，最优预测大小 k 值仍然是未知的，因为实际值就是未知的，因此我们需要估计误差 $\widehat{\xi}_c(k)$ 和估计最优预测 k：$\widehat{k}_{opt}$ 的大小。

我们的主要算法就是来寻找这个估计最优预测窗 $\Omega_C^P(k_{opt})$，会在后面进行描述。

## 五　线性模型估计方法

最优预测窗大小 $\widehat{k}_{opt}$ 可以定义如下，其实就是选个最好的 k 值：

$$k_{opt} = argmin_k \xi_c(k) = argmin_k \frac{1}{k} \sum_{i \in \Omega_C^P(k)} (\widehat{f}(\boldsymbol{x}_i) - f(\boldsymbol{x}_i))^2 \tag{五.1}$$

估计方法分为两个方面，首先是一个迭代的估计过程，来估计以 k 为参数的线性模型；另外是一个递归的误差分析过程来计算 $\widehat{k}_{opt}$。这两个方面分两个小节来叙述。

### 5 1　递归构建线性模型

本算法把估计得到的系数作为一个向量 $\widehat{\beta}_c(k)$ 来估计真实系数 $\beta_c(k)$：

$$\widehat{\beta}_c(k) \equiv (\widehat{f}(\boldsymbol{x}_c), \nabla \widehat{f}(\boldsymbol{x}_c)) \tag{五.2}$$

我们把误差定义为矩阵的形式：

$$X_k \widehat{\beta}_c(k) = Y_k \tag{五.3}$$

$$Y_k = (y_1, y_2, \ldots, y_k)^T \tag{五.4}$$

$y_k$ 表示预测窗内 k 个像素的 MC 估计值；$X_k$ 是 $k \times (d+1)$ 维的矩阵，$d$ 是表示一个像素的特征向量 $x_i$ 的长度（注意这里的特征向量（feature vector）并不是矩阵分析里的特征向量（eigenvector），该向量表示图像每个像素的可用特征，例如 G-buffer 里的值）：

$$X_k = \begin{bmatrix} 1 & (\boldsymbol{x}_1 - \boldsymbol{x}_c)^T \\ 1 & (\boldsymbol{x}_2 - \boldsymbol{x}_c)^T \\ \ldots\ldots & \\ 1 & (\boldsymbol{x}_k - \boldsymbol{x}_c)^T \end{bmatrix} \tag{五.5}$$

由此可知，我们完全可以用最小二乘法（见 DezemingFamily 的《最小二乘法》）来求出 $\widehat{\beta}_c(k)$。但是这样存在问题，需要计算矩阵求逆非常耗时，而且我们并不知道预测窗大小 k，我们需要一个一个像素地加到预测窗里来进行估计（寻找最优 k 值），因此我们会使用递归最小二乘法（见 DezemingFamily 的《递归最小二乘法》）来解决这个问题。

关于递归最小二乘应用于这里的公式可以查看论文（与《递归最小二乘法》书的基本方法是一致的），并没有太多值得进一步解释的地方，我写这个小册子的意义也是为了更好的从图形学的角度来描述原理。

下一节我们介绍如何找到最优估计模型 $\widehat{\beta}_c(k_{opt})$。

## 5 2 递归估计预测误差

本节介绍如何在众多可能的 k 中选择最优预测窗大小 k，以及相应的最优线性模型。

对于选择的某个 k，我们应该估计预测误差 $\xi_c(k)$，本节提出了新的迭代的技术来预测以 k 为自变量的误差函数。以前有一些方法只能估计单个像素的预测误差，而本文要用一个模型去估计整个 k 大小的周边区域。

当我们有一个用 $(k-t)$ 个像素计算的线性模型 $\widehat{\beta}_c^T(k-t)$，当加入新的 $t$ 个样本后，计算预测误差。我们称这 $t$ 个样本叫测试集，$k-t$ 个样本叫训练集。

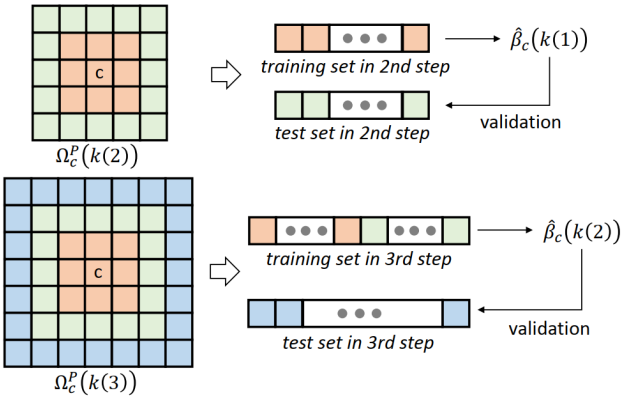
滤波窗（一定要明确滤波窗和预测窗，预测窗被包含于滤波窗）$\Omega_c^F$ 的大小设为 $(2R+1)\times(2R+1)$，我们就可以估计预测误差。k 的大小为：

$$k \in \{1^2, 3^2, 5^2, ..., (2R+1)^2\} \tag{五.6}$$

我们设 $k(r) \equiv (2r+1)^2$，表示预测窗大小的递增过程。

$$\widehat{\xi}_c(k(r)) = \frac{\widehat{\xi}_c^{acc}(k(r))}{(2r+1)^2} = \frac{\widehat{\xi}_c^{acc}(k(r-1)) + \Delta\widehat{\xi}_c^{acc}(k(r))}{(2r+1)^2} \tag{五.7}$$

这个过程可以用下图来进行表示，这是一个迭代的过程，用当前模型线性估计新像素的样本，然后再把新样本一起得到的线性模型来估计更新的一波样本。迭代完以后，找到其中误差最小的 k 即可（只要弄明白我们的目的是什么，则过程其实很简单）。

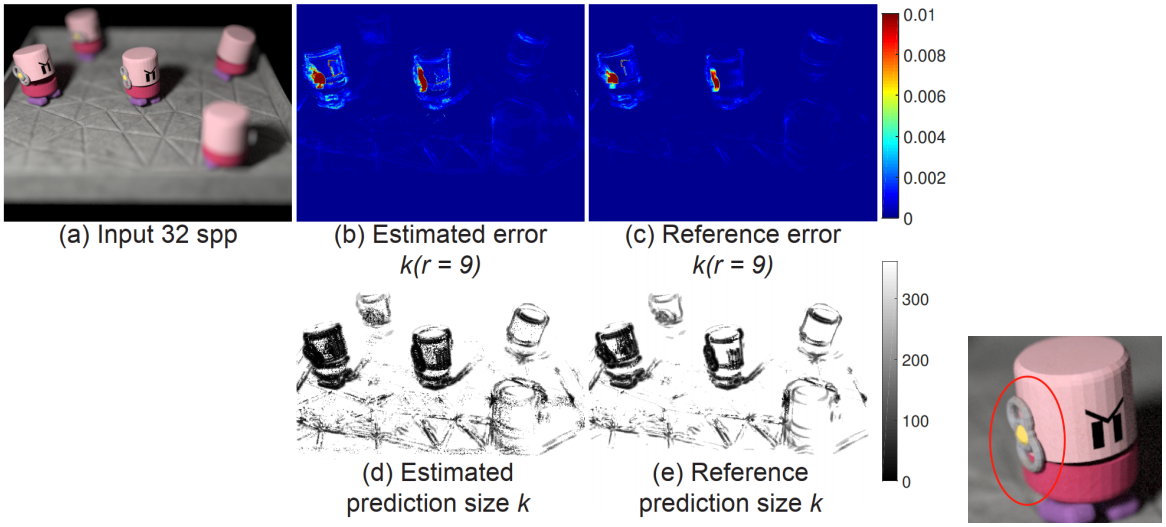



$\widehat{\xi}_c^{acc}(k(r-1))$ 表示从 $k(0)$ 一直到 $k(r-1)$ 的误差，以及新加入像素以后的误差 $\Delta\widehat{\xi}_c^{acc}(k(r))$。

$$\Delta\widehat{\xi}_c^{acc}(k(r)) = \sum_{i=1}^{8r}\left(\widehat{\beta}_c^T(k(r-1))\boldsymbol{z}_i - y_i\right)^2 \tag{五.8}$$

$$\boldsymbol{z}_i^T = \begin{bmatrix} 1 & (\boldsymbol{x}_1 - \boldsymbol{x}_c)^T \end{bmatrix} \tag{五.9}$$

下面我们稍微分析一下估计误差 $\xi\beta_c(k(r))$ 和参考图像误差 $\xi_c(k(r))$。注意参考图像在实际渲染中是得不到的，但是通过分析，我们可以认为本文的算法是有意义的。对于参考图像误差而言，我们选择的 k 值是用来最小化参考图和 MC 估计图之间的平方误差而选择的：




可以看到，我们估计的 $k(9)$ 误差与索引图和估计图的误差相比很相似，估计的预测窗大小和参考图像得到的预测窗大小也是一致的。也就是说，我们的模型预测到某些部位渲染的误差很大，实际上这些部位的误差就是很大（比如上图右边红圈圈起来的部位），因此我们需要在这些地方多采样一些样本。

这个算法的神奇之处就是，我们使用迭代的方法来估计误差，即我们的线性模型会在一个预测窗内不断估计新样本的误差，企图找到一个最小的误差。然后我们把这个最小的误差进行可视化，发现这个最小误差分布图与索引和 MC 估计图之间的误差是几乎一致的。

## 六　线性模型重建与自适应采样

现在，最优预测窗 k 的值我们已经知道怎么计算了，但是预测窗中心点 $c$ 的位置应该如何选取，以及如何在高误差区域进行更多的采样，将是我们现在需要介绍的内容。

### 6 1　迭代地构建线性模型

得到一个线性模型的计算复杂性为 $O(|\Omega_c^F|d^2)$，如果一共需要建立 L 个线性模型（L 个中心位置 $c$）则总复杂度就再乘以 L。

首先，我们建立比较粗粒度的中心点分布，例如一个滤波窗的中心位置。之后我们估计最优预测窗的大小。我们这么操作：设定一个步长 g，初始化为滤波窗宽度，然后每隔 g 放置一个中心点 $c$，并估计以 $c$ 为中心点的 $k_{opt}$ 大小。

然后，我们把 g 减少一半，看当前位置是否在上面预测到的模型的范围里，如果不在说明有光照急剧变化，因此我们就为该点 $c$ 创建一个新的线性模型。

我们不断缩小 g 值，直到所有像素都能被模型预测到为止。但是有的像素可以被多个线性模型预测到，因此我们需要把多个模型的预测结果取一个平均。

### 6 2　自适应采样

第一阶段，我们每个像素只采样 4 个样本即可。我们基于估计误差 $\hat{\xi}_c(\hat{k}_{opt})$ 为每个像素分配额外的光线。至于分配多少光线论文 [1] 中是给了一些说辞的，但我认为我们并不是要非常严格地去执行，我们可以基于误差和期望来给一个理论支撑，不过更多时候我们还是凭感觉走。

## 时序拓展

其实原理就是跟 TAA 一样，利用时序相关性，相当于添加了更多的样本进来。当然对于新样本肯定最好是让它的权重大于老样本的权重，因此，我们经常会加权。

# 七　总结

该论文的技术路线并不是很复杂，里面有些巧妙的地方，前面也介绍了。利用线性模型来估计光照是一个很好的思路，至于以后要如何做，论文叙述说要扩展到高维空间（例如透镜），以及对不同的特征使用不同的权重。

# 参考文献


[1] Moon B , Iglesias-Guitian J A , Yoon S E , et al. Adaptive Rendering with Linear Predictions[J]. Acm Transactions on Graphics, 2016, 35(4CD):121.1-121.11.

[2] MOON, B., CARR, N., AND YOON, S.-E. 2014. Adaptive rendering based on weighted local regression. ACM Trans. Graph. 33, 5, 170.