北京郵電大學

本 科 毕 业 设 计（论文）

题目：蒙特卡洛光线跟踪的实现与优化

姓　　名　兰　毅

学　　院　计算机学院

专　　业　计算机科学与技术

班　　级　2012211307

学　　号　2012211279

班内序号　7

指导教师　王文成

2016 年 6 月

蒙特卡洛光线跟踪的实现与优化

摘　要

基于蒙特卡洛积分技术的整体光照计算是合成高质量、物理真实感图像的主要途径，其中的一个关键环节就是所谓的像素采样，即如何在图像平面的每个像素内抽取适当的位置实施整体光照计算、进而得到生成图像。本文针对常用的随机像素采样导致生成图像随机噪声大的缺点，提出一个新的采样算法——基于面片空间聚类探测场景复杂度的自适应性采样。

在真实感绘制过程中，场景表面的光照情况通常与表面材质、表面法向及其相对视点与光源位置等几何材质因素相关，在几何复杂的区域，光照情况变化很快，这些区域因采样数目不足生成的图像伴随大量噪点，而在几何平坦区域，只需要少量采样即能很好地重建出质量较高的图像结果。如果能预先知道场景的几何和材质信息，预测光照情况复杂的场景区域，并在采样阶段为之分配更多采样数目，就能在同等采样数目下加速生成图像的收敛，提升图像的视觉质量。

基于这样的考虑，我们深入研究了自适应采样生成算法，针对现有算法在采样阶段对表面法向等场景三维信息利用的不足，提出了将场景面片按材质相同、法向相似、空间位置相似聚类，并利用空间包围盒帮助像素在绘制时快速探测场景区域复杂度，并将其引入到蒙特卡洛直接光照的自适应采样中。

采用新的采样算法，使得合成图像过程中某些像素的蒙特卡洛整体光照计算收敛速度较快、即只需少量采样数目既能得到较准确的光照值，而对于另外一些像素则需较大的采样和计算量才能计算出准确值。采用普通方法时，对所有像素都采用统一的采样量，当总体采样量较小时生成图像随机噪声较大，但总体采样量较大时又会导致计算资源的浪费。新算法在采样时，对每个像素探测其成像面片的聚类的空间尺度，以此来评价该像素的成像区域处三维场景复杂度，并按复杂度高低分配像素所需求的采样数目，从而在给定平均采样量的情况下，达到在整幅图像上合理分配采样量的目的。

关键词　蒙特卡洛绘制　自适应采样　空间聚类

Implementation and Optimization of Monte Carlo Ray Tracing Method

ABSTRACT

The method based on Monte Carlo integration is the main technology to generate the high quality, physical realistic image, and the first important key in it is called pixel sampling, means how to get the proper positions in every pixel of the image scene to compute the global illumination effect and then give the generated image. This paper give a new sampling algorithm, spatial clustering based adaptive sampling, to overcome the usual more noise points in image generated by random sampling algorithm.

Illuminations are corresponding to material, surface as well as other geometry situation of the surface. On the geometry-complex area where illuminations change rapidly, large amount of noises arise due to the deficiency of samples. While on the flat area, only a limited number of samples are needed to reconstruct. If geometry and material information is acquired and predict the complex scene areas, we may attribute more samples to these areas and converge the image to become visibly stratifying faster with same sampling rate.

Based on this fact, we have deep study of adaptive sampling algorithms. Aimed at existing algorithms that lacks the exploit of 3D information in the scene, we proposed a new method that firstly clustering meshes with same material, similar normal and similar spatial position. Bounding box are used to quickly detect complexity and helped adaptive sampling of Monte Carlo direct lighting.

This new method of sampling makes some pixels of an image converge faster than the others. This means that they only need fewer samples to get precise result than the others. In usual way, all of the pixel take a uniform number of pixels, consequently when the total number of samples is small the image has more random noise，and the number is big lead to computing waste. The new algorithm detect the spatial scale of cluster of the aiming mesh as the complexity criterion and distribute more samples to more complex pixels, so that it can get more reasonable distribution of samples in image when the total computation is fixed.

Key Words Monte Carlo Rendering Adaptive sampling spatial cluster

目　　录

第1章　绪论

1.1　引言

现代计算机图形学领域中包括的内容很多，其中一个日渐重要的部分是所谓的整体光照计算。

在现实世界中，可能存在多个不同颜色及强度的光源，各种物体对于光的不同的反应也因各自不同的材质而有所不同，场景中物体可能呈现出反射、折射、高光等各种光现象，整体光照计算正是在这种情况下产生的。

整体光照渲染一直是研究的热点之一。截至今日，有许多整体光照算法提出，其中得到肯定的有 1980 年提出的光线跟踪算法(Ray Tracing)，1984 年提出的辐射度算法，1986 年 Kajiya 提出的路径跟踪算法(Path Tracing)[1]，1996 年 Jensen 提出的光子映射算法(Photon Mapping)[2][3]。

对于复杂场景的处理，基于蒙特卡洛路径跟踪算法是处理这种复杂性的最佳选择，但是这种技术最大的问题是积分估计的方差，在绘制的图像中表现为噪声。自适应采样是减少生成图像噪声的一种采样方法。选择合适的衡量采样像素质量的标准，使得自适应采样法能够发挥出自身的特点。迄今为止，人们提出了许多自适应采样方法。

自适应性采样算法的基本原则就是，根据待计算像素的光照的收敛速度——计算难易程度来决定要在像素上投入的采样量，自动分配合理的采样量给每个像素。在场景中收敛速度比较慢的区域如物体边界，阴影边界投入比较大的采样量，而在收敛速度比较快的区域仍然采用比较小的采样数量。这样，既提高了成像质量，又减小了计算开销的增加。

1.2　自适应采样问题的定义

如今，蒙特卡洛方法被广泛认为是最实用性的真实感图像合成方法。即便朴素的蒙特卡洛绘制算法也拥有一系列优点：首先，蒙特卡洛算法具备一致性，具体意味着随采样路径数增加，估算图像将收敛到正解；其次，部分蒙特卡洛算法也具备无偏性（例如路径追踪），即，估算图像的期望值或者说采样的期望值对应与正解，估算错误仅由误差组成。最后，蒙特卡洛方法能够应用于实际中绝大多数的场景配置。

但另一方面，因为仅有限数量的随机光径被采样，所有的蒙特卡洛方法都遭受图像噪声污染，本质是估算像素颜色值的误差。不幸的是，要获得没有明显噪声、视觉上满意的结果常常需花上数分钟或数小时的计算时间。因而为了降噪或加速蒙特卡洛收敛，

研究人员近年来提许多类策略，例如各类路径采样策略（重要性采样、双边技术、Metropolis 采样），统计学技术（低差异采样序列、采样强度预测），信号处理技术（频率分析、非线性滤波）。特别地，自适应采样与滤波技术自适应采样与重建技术在最近再次兴起，以其通用且重建质量高、计算高效、易于实现等特点引人关注。

绘制误差的大小通常在图像局部区域间变化巨大。经调研，现有自适应采样技术大多基于已有采样，根据已有采样局部误差大小去进一步分配采样，而不是直接采样直到一个提前确定的目标采样强度。初始采样或者说已有采样的低采样率不足以获得低噪可靠的结果，因此也给局部误差的估计引入了大量噪声。目前并没有哪种方法能完全避免初始采样噪声对绘制结果的污染和对加快收敛速度的阻碍。考虑到有大量未被利用的场景信息，我们希望能做到直接为图像区域设定局部自适应的采样强度，从而在绘制中减少误差引入和加速结果收敛。

1.3 本文研究的创新与贡献

本文将重点自适应采样技术在图形学邻域内的真实感图形绘制中的应用。首先我们将介绍蒙特卡洛全局光照算法的基础知识与相关理论背景，接着我们介绍了多种用于加速蒙特卡洛绘制收敛的采样与重建策略，最后我们进一步研究已有的自适应采样策略，针对已有方法对高噪声的初始绘制结果的依赖和对场景三维空间几何信息利用的不足，提出了一种基于空间聚类的场景局部复杂度表达策略，预计算自适应采样强度指导绘制过程中各局部的采样工作。同样在进行了大量的实验工作后，也取得了一些有意义的、创新性的成果。

1.4 论文其余部分安排

本文组织如下：

- 一章介绍了整体光照模型的意义，以及它存在的一些问题，并介绍了新算法所解决的问题以及我们提出新算法的基本思路。
- 第二章主要介绍了光线跟踪算法的基本过程，并阐述了蒙特卡洛等相关理论背景，同时介绍了多种经典的自适应采样与重建策略。
- 第三章从挖掘场景三维几何信息的考虑出发，我们提出了基于空间聚类的方法对有相似光照情况的面片进行划分聚类，并利用空间包围盒快速表达局部的几何复杂度和采样需求程度，综合多种因素计算确定出局部采样率。
- 第四章展示了实验结果。
- 第五章总结并给出未来工作，可以将我们的空间聚类自适应采样方法结合目前基于重建滤波图像误差估计的后验方法，进一步加快这类后验方法的收敛速度。

第2章　相关算法综述

2.1　光线跟踪算法

光线跟踪算法是典型的整体光照模型。当光线碰到一个物体表面的时候，可能产生三种新的类型的光线：反射、折射与阴影。光滑的物体表面将光线按照镜像反射的方向反射出去，然后这个光线与场景中的物体相交，最近的相交物体就是反射中看到的物体。在透明物质中传输的光线以类似的方式传播，但是在进入或者离开一种物质的时候会发生折射。为了避免跟踪场景中的所有光线，人们使用阴影光线来测试光线是否可以照射到物体表面。光线照射到物体表面上的某些点上，如果这些点面向光线，那么就跟踪这段交点与光源之间的光线。如果在表面与光源之间是不透明的物体，那么这个表面就位于阴影之中，光线无法照射。 这种新层次的光线计算使得光线跟踪图像更加真实。

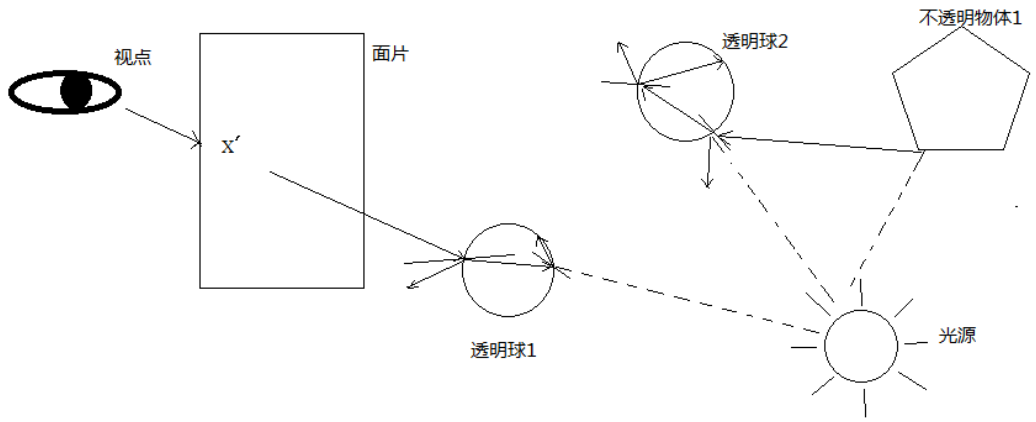

图 2-1 光线跟踪算法图示[22]

光线跟踪算法的基本思想如图 1 所示，图中包含以两个透明球和一个不透明物体的场景为例，对该算法进行分析。对于屏幕上的每个象素，跟踪一条从视点出发经过该象素的光线，求出与环境中物体的交点。在交点处光线分为两支，再分别沿镜面反射方向和透明体的折射方向进行跟踪，形成一个递归的跟踪过程。建立一棵递归的光线跟踪树以记录光路情况。光线每经过一次反射或折射，由物体材质决定的反射、折射系数都会使其强度衰减。渲染方程描述了能量从光源发射，经过各种材质表面的多次反射，折射，最后进入人眼的这一过程。在面片上某点 X7 沿方向 8 出射的辐射度可以定义如下：

$$L(x' \to \theta) = L_e(x' \to \theta) + \int_{\Omega_x} f_r(x', \psi \leftrightarrow \theta) L(x' \leftarrow \psi) \cos(N_{x'}, \psi) d\omega_\psi$$

式（2-1）

其中Ω_x是点 x 的所有半球方向集合，$L_e(x \to \theta)$是点 x 沿 θ 方向发射的辐射度，f_r是双向散射分布函数 BSDF，$L(x \leftarrow \psi)$是点 x 从ψ方向得到的入射辐射度，$cos(N_x, \psi)$是点 x 法向量和入射方向ψ的夹角余弦。

在实际的算法进行过程中，出现以下情况光线跟踪中止：

1．光线没有和场景中任何物体相交。

2．光线遇到了背景。

3．光线在经过许多次反射和折射以后，发生的衰减使得该光线对原象 素光亮度的贡献小于给定的阈值。

4．光线反射或折射次数即跟踪深度大于设定值。

光线跟踪的阴影处理也很简单，只需从光线与物体的交点处向光源发出一条测试光线，就可以确定是否有其他物体遮挡了该光源(对于透明的遮挡物体需进一步处理光强的衰减)，从而模拟出场景中软阴影和焦散的效果。

光线跟踪很自然地解决了环境中所有物体之间的消隐、阴影、镜面反射和折射等问题，能够生成十分逼真的图形，而且算法的实现也相对简单。但是，作为一种递归算法其计算量十分巨大。减少光线与物体之间求交运算的计算量和提高 求交效率是提高光线跟踪效率的关键，常用的方法有：包围盒、层次结构及区域分割等技术。

光线跟踪是一个典型的采样过程，各个屏幕象素的亮度值都是分别计算的，因而会产生走样，而算法本身的计算量使得传统的加大采样频率的反走样技术难以实用。象素细分是一种适用于光线跟踪的反走样技术。

与像素细分不同，Cook、Porter 和 Carpenter 提出的分布式光线跟踪是一种随 机采样的方法，在交点处镜面反射方向和折射方向所夹的立体角内，按照一定的分布函数同时跟踪若干根光线，然后进行加权平均。Cook 等人还提出了利用分布式随机采样技术模拟半影、景深和运动模糊等效果的方法。

光线跟踪的另一个问题是，光线都是从视点发出的，阴影测试光线则需另外处理，因而无法处理间接的反射或折射光源，例如镜子或透镜对光源所产生的作用就难以模拟。为解决这一问题，可以从光源和视点出发对光线进行双向跟踪。

2.2 蒙特卡洛方法

2.2.1 蒙特卡洛积分

蒙特卡洛方法(Monte Carlo 方法)，也称统计模拟方法，是二十世纪四十年代提出的一种以概率统计理论为指导的一类重要的数值计算方法。Monte Carlo 方法利用随机采样，使用随机数(或更常见的伪随机数)来解决很多计算问题。

在积分计算中，有些积分是极其复杂的以至于用一般方法来计算是不可想象的，此时可以用蒙特卡洛方法来计算这些积分。

考虑有以下积分式：

$$I = \int_{x \in A} f(x) dx$$

式（2-2）

公式(2-1)中，A 为积分区域，x 是其中任意选取的点，若被积函数f可以分解成$f(x) = g(x)p(x)$，当 x 在积分域 A 上有某种概率分布，$p(x)$是其概率密度分布函数，$0 \leqslant p(x) \leqslant 1$，$\int_{x \in A} f(x)\, dx = 1$，此时(2-1)式可以写为：

$$I = \int_{x \in A} g(x)\, p(x) dx$$

式（2-3）

公式(2-2)实际上计算的是当函数$g(x)$有概率密度函数$p(x)$时的数学期望，即$E(g(x))$。对于$E(g(x))$，可以采用抽取随机点来近似的计算。

$$I = E(g(x)) \approx \frac{1}{N} \sum_{i=1}^{N} g(x_i)$$

式（2-4）

x_i是在 A 中随机选择的采样点，当$N \to \infty$采样量趋向于无穷时，运用蒙特卡洛方法计算积分能够得到(2-1)式精确的积分值。

$$I = E(g(x)) = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} g(x_i)$$

式（2-5）

也许对于被积函数 f，分解是不明显的复杂的，但是同样能够按照这种思路，稍加变换计算其积分值，人为的选择一个概率密度函数$p(x)$，将(2-2)式改写为：

$$\int_{x \in A} f(x) dx = \int_{x \in A} \frac{f(x)}{p(x)} p(x) dx$$

式（2-6）

将$\frac{f(x)}{p(x)}$视为公式(2-2)中的被积函数，则(2-5)式可以根据蒙特卡洛计算，

$$I = E\left(\frac{f(x_i)}{p(x_i)}\right) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

式（2-7）

所以无论被积函数如何，都可以借助于蒙特卡洛方法，把复杂的积分问题，转换成为相对简单的通过随机采样来近似计算的函数数学期望问题。

可以证明，当选择的概率密度函数 $p(x)$，与被积函数的形状越接近，所计算出来的积分值越精确，所需要的采样点越少，在极限情况下，当概率密度函数 $p(x)$有$p(x) = \frac{f(x)}{I}$时，只需要一个采样点就可以得到 I 的精确值。

正是因为，的难于计算，所以我们才会采用蒙特卡洛方法计算积分，所以也许无法为，找到很好的 $p(x)$，但是可以证明，随着采样量的上升，计算误差 σ 是缩小的，误差与采样量之间的关系为公式(2-7)。

$$\sigma \propto \frac{1}{\sqrt{N}}$$

式（2-7）

蒙特卡洛中的各种推导就不在这里再做说明了，请参考这些有关文献[4][5]

2.2.2 蒙特卡洛全局光照

如今，蒙特卡洛方法被广泛认为是最实用性的真实感图像合成方法。Kajiya 在 1986 年提出了渲染方程 [1]，将图像绘制问题描述为对所有光径的积分；连接图像传感器上点到光源上任一点的光线路径定义为“光径”。通过随机地采样光径同时累积这些采样的图像颜色贡献值，蒙特卡洛方法完成对该积分的估算。

即便朴素的蒙特卡洛绘制算法也拥有一系列优点：首先，蒙特卡洛算法具备一致性，具体意味着随采样路径数增加，估算图像将收敛到正解；其次，部分蒙特卡洛算法也具备无偏性（例如路径追踪），即，估算图像的期望值或者说采样的期望值对应与正解，估算错误仅由误差组成。最后，蒙特卡洛方法能够应用于实际中绝大多数的场景配置。

将蒙特卡洛方法应用到全局光照领域，是全局光照领域的一个新的突破。通过结合蒙特卡洛方法，可以建立从视点到光源的随机游动链为基础的蒙特卡洛光径跟踪算法来逐像素地生成图像。

其中，蒙特卡洛光径跟踪是所有蒙特卡洛全局光照方法的基本框架。蒙特卡洛光径跟踪的核心环节在于采样穿过像素的光径样本，像素值即为所有光径样本光照值的均值。

图 2-2 示意了像素的一个光径样本。光径的建立过程是：在像素区域内随机地采样一个点%，以视点出发产生一条射线穿过y_0射进场景，计算该光线与第一个物体的交点y_1，以y_1为起点、根据y_1所在物体表面的散射特性随机产生一个方向而形成一条射线，再求出另一个交点y_2，上述过程反复进行，在任一交点处，光径均可以一定的概率t_i而终止。整个光径的光照值$< L(y_0 \leftarrow y_1) >$按照蒙特卡洛积分可以得到：$< L(y_0 \leftarrow y_1) >=$

$< L_{dir}(y_0 \leftarrow y_1) > +(\frac{1}{1-t_1}) < L(y_1 \leftarrow y_2) >$。其中$< L_{dir}(y_0 \leftarrow y_1) >$是光源经$y_1$。发生散射而到达$y_0$处的光照值。上式中$< L(y_1 \leftarrow y_2) >$的计算完全可按$< L(y_0 \leftarrow y_1) >$的计算得到。

蒙特卡洛光径跟踪计算简单，具有普适性，适于计算其它确定性数值方法无法计算的复杂场景光照，可以模拟软阴影、泅色和焦散等各种全局光照效果。当场景中存在非常复杂的反射模型需要渲染的时候，蒙特卡洛方法更为适用。但是，由基于蒙特卡洛的全局光照算法中，当每个像素的采样量不足时，生成图像存在有大量噪声。

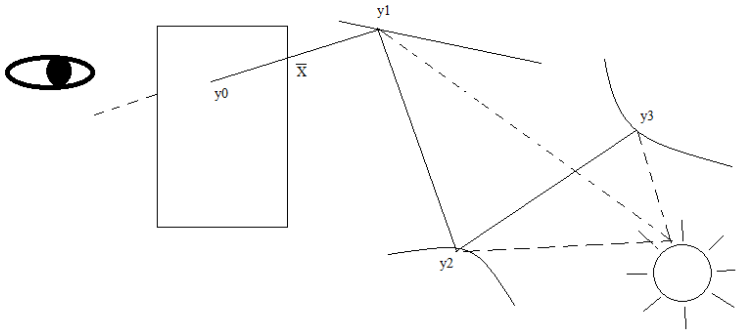

图 2-2 蒙特卡洛全局光照[19]

2.3 自适应采样

蒙特卡洛方法的最大缺点是计算时间很长，并且受到单个光子运动的统计特性和物理模型精度的限制，在给定的时间内光子到达成像平面接收机的概率很小，因此为了得到可靠的统计结果，需要跟踪大量的光子轨迹。当采样量不足时，采用该方法生成的图像会存在有大量的噪声。自适应采样方法是减少生成图像噪声的一种方法。

在不应用自适应采样的绘制算法中，我们通常给每个像素生产固定数目的采样。由于所生成图像并不是每个区域都需要相同数目的采样来收敛（如图 2-3），那么如果我们每个图像区域分配其足够收敛的采样数，我们就能显著地加速这个收敛过程。这就是自适应采样算法做的事：

1. 为每个像素分配固定数目的采样并计算采样光照贡献
2. 基于已生成的采样估算图像的误差水平
3. 依据估算的误差分配额外的采样并计算采样光照贡献
4. 终止条件未满足时，跳回步骤 2
5. 从生成采样重建绘制结果图像

终止条件通常由使用者设置，可以是时间限制或总采样数量限制。但有一点重要说明：目前的自适应采样方法值其实在考虑蒙特卡洛采样空间的一个低维的子集，当分布

额外采样（光径）时，其实只是每条新光径在图像平面上的那一个顶点是依据误差估计值确定分布位置的，光径的其余部分生成时和误差估计无关。

图 2-1 一个绘制的例子，中间由非自适应采样算法绘制，右边由自适应采样算法绘制[19]

方法分类：

我们可以将现有自适应采样方法划分为四类，每一组使用不一样的误差估计策略或额外采样分布策略。

2.3.1 基础方法

基础方法是最初也是最简易关于自适应采样的思考。这些方法预测每个像素误差然后给误差最高的像素分配额外采样。这类方法之间的主要区别就是估算误差的方法各异。

最常用的误差估计方法有

1. 采样方差是最直接的误差估计。Kajiya 在他提出的真实感图像生成里的第一个自适应采样方法就用了这个作为误差估计[1]。
2. Mitchell 第一个使用了考虑人眼视觉反馈的误差估计，具体地为 contrast metric 对比度度量[7][8]。对比度度量是个非线性度量函数，它独立在红绿蓝三个分量上计算，然后依据人眼对不同颜色分量的敏感程度加权组合。
3. 由 Rigan 开始将概率理论引入像素误差的估算中[8]。Rigan 的方法主要思想是不仅是比较采样方差，而是要比较他们整个分布。具体来说就是对每个像素我们获得采样颜色值的分布，将其与均匀分布比较并计算一个距离值。一个像素内采样分布的距离值越大，误差越高。同时，为度量两个概率分布间的距离，可以使用各类 f 散度，如 Kullback-Leibler divergence。

2.3.2 频域分析方法

频率分析方法不直接在像素上操作，而是利用图像变换如离散余弦变换、离散小波变换等将像素颜色值变换到频域。当图像像素被保存为频域基函数的系数，误差在每个系数上预测，额外的采样分配到整个图像平面，误差最高的系数获得大部分新采样。

Bolin and Meyer 最早提出在自适应采样中使用变换方法[9]。该方法在 8X8 图像块内使用离散余弦变换，并通过只保留块内采样数一半个数的系数有效地消除走样（奈奎斯特采样定理）。估计误差使用了一个简单的人眼视觉系统模型。

近期的频率分析方法使用比 DCT 更有效率、且能在全图使用而不仅仅在块内使用的离散小波变换[10]。估算误差时，对比度度量经过改变，使得仅使用局部 DWT 变换即可从采样分布为每个小波基系数计算度量。

2.3.3 滤波方法

滤波方法是使用像素滤波器作为自适应采样算法不可分割的一部分。

第一个最有代表性的此类算法由 Rousselle 提出[15]，误差估计并不直接从采样值计算，而是从滤波后的像素值计算。该算法的滤波器候选库有 5 个核大小不同的高斯滤波器，算法每个滤波器施加到每个像素上，选取结果最优的滤波器，最大限度地减小均方误差值。均方误差值不能被直接计算（无法获取参考图像），但能被用来估算出有较小采样误差的最优滤波器。这样迭代整个过程，我们能轻易地为每个像素选取最优滤波器。一旦滤波器选取，新的采样数被分配到有着最高均方误差的像素上。随后额外的采样根据像素处滤波器概率密度函数分配到像素及其周围像素。

Li 等用 SURE 方法改进了均方误差估计[16]。更优越的误差估计改善了滤波方法的性能。

更近的滤波方法例如 Rosselle 提出的[17]，使用了稍微不同的方法。该方法将采样集均分给两个图像 A 与 B，两个图像的差异及其采样被接下来估算像素误差。算法使用了 NL-means 滤波器。并且，A 图像滤波器权值从 B 图的误差估计获得，B 的也反之。额外采样数的分配不仅与误差估计也与滤波器权值有关。最终 A 与 B 平均整合入最终图像。

由于能够预确定自适应采样值而不用由颜色值迭代计算，我们的方法可以对比此类算法的初始采样阶段。由于初始采样是使用随机分布，且我们的算法在采样率较低时效果较好，因此在将来考虑在已有滤波方法结合我们的自适应采用方法，进一步加速滤波方法的收敛。

2.3.4 多维度方法

多维度方法与之前几类方法的巨大区分处在于：多维度方法在考虑图像平面采样点的选取之外，还将更多维度上的采样考虑进入，目的是更好地拟合如动态模糊、景深等绘制效果。例如当我们打算绘制一张有动态模糊的场景，不仅在图像平面也在时间区间生成采样位置。

一部分多维度方法考虑的维度有限，仅特别地针对有限数量的绘制效果设计[11][12]。然而也有能处理任意多维度方法在研究中。Hachisuka 等通过为高维空间建立 KD 树并在树节点放置采样[13]。除了计算像素误差以外，每个节点的误差也被估算出来，额外采样将分配给误差最大的节点。

第3章　算法的设计

3.1　算法流程总结

- 用自适应场景尺度的均匀网格剖分场景
- 依据空间与几何相似性聚类同网格内面片
- 为面片聚类建立包围盒，利用包围盒的大小快速估算聚类区域的尺度
- 由成像平面上像素投射光线，计算出像素的自适应采样率

用流程图表示为：

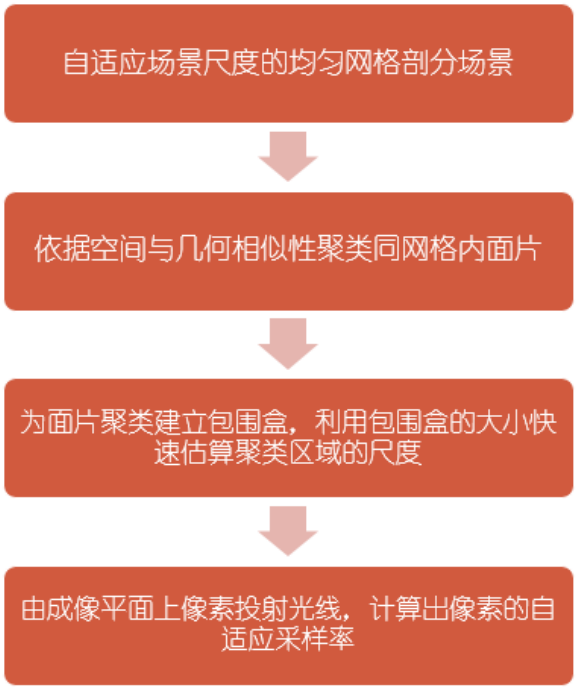

图 3-1 算法流程

3.2　可能遇到的问题

3.2.1　投射光线走样

让我们回顾确定采样率计算算法的第二步：

- 投射光线 *Ray* 与场景面片相交，所交面片所在聚类即是覆盖像素 *P* 的聚类。

如果对一个像素 *p*，我们将单根投射光线 *ray* 相交的聚类确定为覆盖该像素 *p* 的聚

类，其实是认为最多仅有一个聚类投影到像素区域 P。而很明显，事实上会有多个聚类覆盖一个像素区域的情形，如图 3-2 所示。

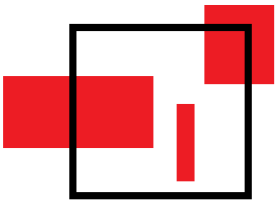

图 3-22 走样示意

这就有错误计算采样率的可能：多个聚类的情形仅被当成某一个聚类处理。

我们可以分情况讨论这种误差：

设聚类被覆盖面积为 *Scc*，聚类总投影面积为 *Sc*，像素面积为 *Sp*，聚类总采样率为 *SamplingRateC*，要估算的像素采样率为 *SamplingRateP*

a) 被一个聚类覆盖：

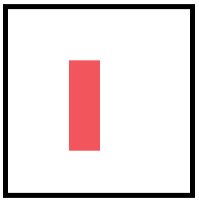

图 3-3 一个聚类完全在像素区域范围内

对图 3-3 所示情况：采样率应该估算为$\left(\frac{Scc}{Sc}\right) * samplingRateC$，由**公式**估算的采样率结果为$(\frac{\mathrm{Sp}}{\mathrm{Sc}}) * \mathrm{SamplingRateC}$。由于 *Scc*=*Sc*<<*Sp*，公式估算的采样率是实际估算采样率的数倍之多。

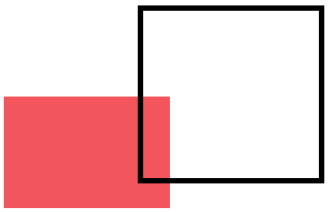

图 3-4 一个聚类仅部分覆盖像素区域

对图 3-4 所示情况：采样率应为$(\frac{Scc}{Sc}) * samplingRateC$，由**公式**估算的采样率结果为$(\frac{Sp}{Sc}) * SamplingRateC$。$Scc < Sp$公式估算的采样率比实际估算采样率大。

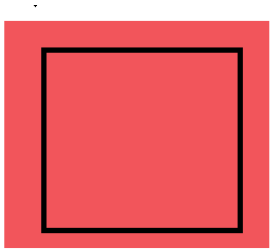

图 3-5 一个聚类能覆盖整个像素区域

对图 3-5 所示情况：采样率应为$(\frac{Scc}{Sc}) * SamplingRateC$，公式估算的采样率结果为$(\frac{Sp}{Sc}) * SamplingRateC$。$Scc = Sp$，公式估算等于实际值。

b) 多个聚类完全在像素区域范围内

图 3-63 多个聚类完全在像素区域范围内

如图 3-6 所示,有 N 个聚类完全在像素区域内的聚类,采样率应该为$\left(\frac{Sc1}{Sc1} + \frac{Sc2}{Sc2} + \cdots + \frac{ScN}{ScN}\right) * samplingrateC = N * samplingrateC$，单条光线的方法估算结果为$\frac{Sp}{Scx} * samplingrateC$。因为$Sp > (Sc1 + Sc2 + \cdots + ScN)$，公式估算的采样率比实际值大了。

c) 多个聚类仅部分覆盖像素区域

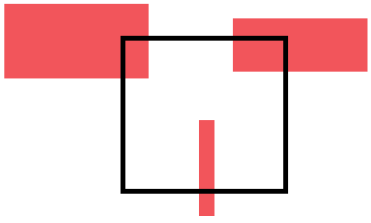

图 3-7 多个聚类仅部分覆盖像素区域

如图 3-7 所示,有 N 个聚类完全在像素区域内的聚类，采样率应该为$(\frac{Scc1}{Sc1} + \frac{Scc2}{Sc2} + \cdots + \frac{SccN}{ScN}) * samplingrateC$，单条光线的方法估算结果为$\frac{\mathrm{Sp}}{\mathrm{Scx}} * \mathrm{samplingrateC}$. 因为$Sp > (Scc1 + Scc2 + \cdots + SccN)$，公式估算的采样率还是比实际值大了。

d) 多个聚类能覆盖整个像素区域

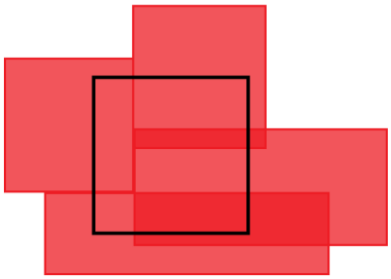

图 3-8 多个聚类能覆盖整个像素区域

如图 3-8 所示,有 N 个聚类完全在像素区域内的聚类，采样率应该为$(\frac{Scc1}{Sc1}+\frac{Scc2}{Sc2}+\cdots+\frac{SccN}{ScN})*samplingrateC$，单条光线的方法估算结果为$\frac{Sp}{Scx}*samplingrateC$. 因为$Sp=(Scc1+Scc2+\cdots+SccN)$, 当$Sc1=Sc2=\cdots=ScN=Scx$时，易得$(\frac{Scc1}{Sc1}+\frac{Scc2}{Sc2}+\cdots+\frac{SccN}{ScN})=\frac{Sp}{Scx}$，此时公式估算采样率等于实际采样率。我们又能得出：这种情形下的采样率计算误差仅与各聚类包围盒的大小差异有关，差异大带来的误差大。而实际中这种差异通常会是较小的，所以此情形下采样率计算误差在可接受范围内。

e) 总结投射光线有限带来的采样率计算误差：

可以总结出，在像素面积未被聚类投影覆盖满时，公式估算的采样率都偏大了很多；而在像素被完全覆盖时，如 a 中下图和 d 中情形，误差很小或仅与各聚类尺度差异性有关联，且聚类间大小差异通常在一定范围内，所以带来的误差较为可接受。

实际场景情况里，大部分像素也正好是 a 中下图和 d 中情形。由于设置的空间聚类尺度通常要比一个像素区域大很多，以及实际场景面片连通性很高，离散排列的聚类很少，出现半覆盖的像素的可能性也就很小。

这些原因使得像素面积被各聚类投影完全覆盖满的情形占绝大多数，我们的公式误差实际在可接受范围内。再考虑到如果投射更多条光线探测聚类带来大量的求交运算，衡量效率性能比后我们决定只用单条光线投影获得可接受误差的采样率计算。

3.2.2 聚类包围盒估算带来的误差

包围盒投影面积代替面片面积投影大大加速了聚类区域尺度的估算，然而也带来潜在的的估算误差。

首先，包围盒比所包面片的空间范围大，通常投影到成像平面的面积也更大，因此包围盒的引入使得像素采样率通常“变小”。但这大多还在可接受范围内：这种空间范围的增大对所有聚类是几乎同等的。

更为引人关注的是，由于场景表面的曲率特性，我们的面片聚类算法有可能将法向相似但并不连通反而彼此远离的两组面片聚类在一起。尽管这两组面片必定在一个均匀网格范围内，误差不至于巨大，但其包围盒仍然比我们希望的空间尺度大了数倍，导致此区域采样率的下降。想象细小离散分布的聚类，本来需要高采样率采样这个快速变化的区域，可是这几个聚类被一个更大更粗糙的聚类合并，这块区域就被看作更为平坦的区域而低速率地采样了。

要解决第二个问题，只有从我们的空间聚类算法入手。原有的空间聚类算法实际上空间相似性并不鲁棒，不能保证聚类区域的连通。我们想到两种方法改进：

a) 直接计算面片投影面积
b) 考虑连通性的空间聚类

a 方法并不能直接解决空间聚类误差的问题，但可以大大削减包围盒带来的成倍的采样率低估。增加的时间复杂度在 *O(N)*级别，*N* 为场景面片数。

b 方法考虑连通性的空间聚类思路如下:

在三维空间中保证聚类区域连通，我们需要获取场景面片的邻接信息。在读入场景文件时保存所有点上的面片（共点面片）的序号，保存在文件中。为网格内面片聚类阶段，维护一个已聚类面片序号数组和一个未聚类面片序号数组，随机选取一个面片开始聚类，在未聚类面片数组中搜寻候选面片，若满足几何相似性，再比较该面片是否与聚类内已有面片共点，仅共点满足才将其加入聚类，将该面片移除未聚类数组而加入已聚类数组，否则继续在未聚类数组里寻找，每次找到一个方可重头遍历，否则到数组头没找到终止遍历，该聚类确定下来，再在未聚类面片里随机挑选下一个聚类的主面片。直到未聚类面片数为 0，该网格聚类完成。这个算法的复杂度在 *O(NlogNlogN)*级别。

两个解决方案的运算效率都不高，尤其第二种，虽然更加根本地解决空间聚类问题，但面对大场景必然耗费巨大时间构建聚类。第一种可以考虑替代已有的包围盒方法，会在今后通过实验比较两者效果。

3.3 空间聚类面片

3.3.1 由基于空间聚类增强 Lightcuts 的绘制算法想到的

基于空间聚类增强 Lightcuts 的绘制算法则利用了场景三维空间连贯性[14]，根据几何位置和法向，对场景空间中与绘制光线相交的各成像点进行在线的逐步聚类，并为每个类动态地更新初始光源割，以作为初始位置来为该类中各成像点搜索其相应的光源割。

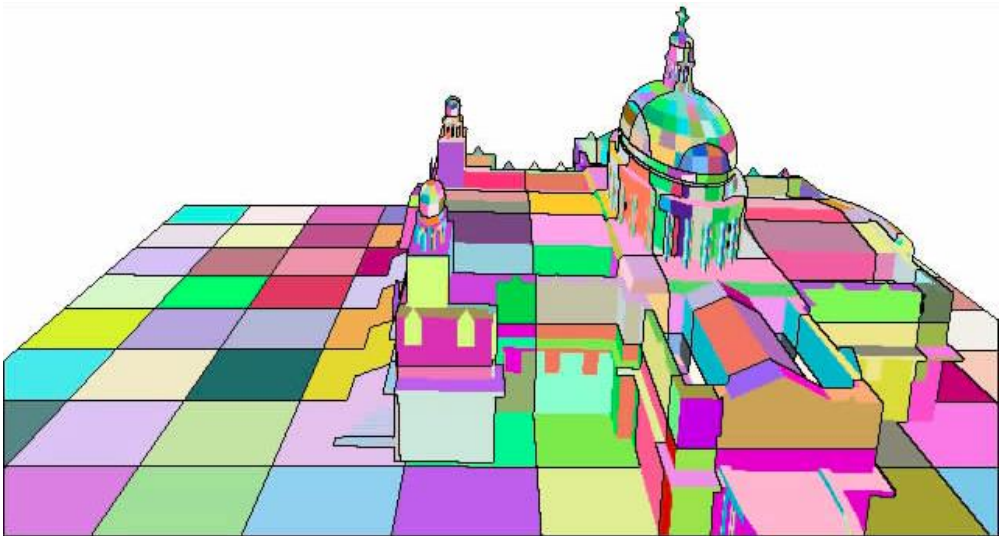

图 3-10 空间聚类结果[14]

由聚类结果图 3-10 可以看出，材质相同、法向相似且在同一个网格范围内的面片被组织到同一聚类中，同一聚类中的面片面对的光源和场景相似，概率反射函数也相似，一般将返回相近的光照情况。

换个角度，从理想采样率的局部性考虑，场景中曲率变化快的几何部分如教堂弧顶，聚类划分更为细致，直观地看聚类体积小；而曲率变化平坦的部分聚类体积大。这给了我们一个朴素的印象：绘制前的空间预聚类可以探测区分场景中几何复杂与几何平坦的区域，单位体积或单位投影面积内聚类个数越多，说明该区域场景复杂度高，几何曲率变化快，因而区域内光照情况变化快。对光照情况变化快的区域应当分配更多的随机采样以保证颜色值收敛，避免噪声点和像素走样；反之，聚类个数很少，很可能表明该处表面较平坦（如图 3-3 中场景里的地面，聚类体积即原均匀网格大小），光照情况也相似，只需要较低的采样率就能较好地重建原颜色值，后置滤波器也能利用这种空间连续性。

3.3.2 空间聚类

在绘制之前，空间预聚类过程将场景面片按照材质相同、法向相近、空间位置相近的原则聚类组织，以为之后计算面向场景几何复杂度的自适应采样提供依据。

由空间聚类增强 lightcuts 算法启发，我们联想到：几何相似，材质相同、法向相近、空间距离近的面片有着相近的光照情况，它们的颜色采样性质也“相似”。同态的采样我们只需要一定数量就能重建该区域的光照情况。比如，可以为每个相似性聚类分配固定数目的采样，无论其空间体积大小。

换个角度思考，曲率变化频率高、几何复杂的区域，为了重建光照情况需要高采样率(奈奎斯特定理)，基于空间聚类的自适应采样算法能利用预聚类探测场景的几何复杂程度，并按复杂程度分配采样。场景复杂的区域，大量聚类分割与表达此区域的面片，

每个聚类的体积小较小而区域内聚类数量与密度高，需要为每个相似性聚类分配一定数量的采样来更好地重建其面片的光照情况。表现为即几何复杂区域聚类数量多，分配的采样率也高。由此作为我们先验的自适应采样率计算的依据。

第一步：均匀网格剖分

当场景面片被读入后，我们对所有面片进行空间聚类。实现空间聚类时，先使用均匀网格剖分场景，使得被聚类面片将在一定空间距离内，同时降低空间聚类计算的复杂度。

具体实现时，网格剖分算法简单地将三维空间划分为均匀网格，每个网格维持一个 Voxel 结构，遍历场景内所有面片，所有被面片包围盒覆盖的 Voxel 均更新其维持的面片队列。

我们希望网格数量尽量正比于体素数。那么对每个场景网格密度如何确定呢？设投影到第 i 坐标轴的网格数量为 *nVoxel[i]*，三维空间网格分布即 *nVoxel[1]*nVoxel[2]*nVoxel[3]*。如果第 i 坐标轴方向场景宽度为 *Length[i]*,又设沿坐标轴单位长度投影到的正方形网格数即网格密度为 *VoxelsPerUnit*，于是有计算关系：

$$nVoxels[i] = VoxelsPerUnit * Length[i] \qquad \text{式（3-1）}$$

考虑尽量令三维空间场景面片数正相关于均匀网格数量，*VoxelsPerUnit* 应该正相关于面片数量的 1/3 次幂，于是我们设计如下计算公式：

$$VoxelsPerUnit = K * [(nPrims)^{\frac{1}{3}}] * (\frac{1}{MaxLength}) \qquad \text{式（3-2）}$$

其中 *nPrims* 代表场景面片数，*MaxLength* 表示场景最宽方向长度，K 控制正比例系数，*VoxelsPerUnit* 随其线性增长。不同的 *K* 值可以影响网格密度

具体代码如下：

```
void makeGrid( vector<Reference<Primitive> > &primitives,
    bool refineImmediately, Clustertree *CTree) {
    int nVoxels[3];
    BBox bounds;
    Vector width, invWidth;
    Voxel **voxels;
    MemoryArena voxelArena;
    // Initialize _primitives_ with primitives for grid
    // Compute bounds and choose grid resolution
    for (uint32_t i = 0; i < primitives.size(); ++i)
        bounds = Union(bounds, primitives[i]->WorldBound());
    Vector delta = bounds.pMax - bounds.pMin;

    // Find _voxelsPerUnitDist_ for grid
    int maxAxis = bounds.MaximumExtent();
    float invMaxWidth = 1.f / delta[maxAxis];
    Assert(invMaxWidth > 0.f);
    float cubeRoot = 9.f * powf(float(primitives.size()), 1.f / 3.f);//3X
    float voxelsPerUnitDist = cubeRoot * invMaxWidth;
    for (int axis = 0; axis < 3; ++axis) {
        nVoxels[axis] = Round2Int(delta[axis] * voxelsPerUnitDist);
        nVoxels[axis] = Clamp(nVoxels[axis], 1, 32);//ly:控制网格密度//Sam
    }

    // Compute voxel widths and allocate voxels
    for (int axis = 0; axis < 3; ++axis) {
        width[axis] = delta[axis] / nVoxels[axis];
        invWidth[axis] = (width[axis] == 0.f) ? 0.f : 1.f / width[axis];
    }
    int nv = nVoxels[0] * nVoxels[1] * nVoxels[2];
    voxels = AllocAligned<Voxel *>(nv);
    memset(voxels, 0, nv * sizeof(Voxel *));

    // Add primitives to grid voxels
    for (uint32_t i = 0; i < primitives.size(); ++i) {
        // Find voxel extent of primitive
        BBox pb = primitives[i]->WorldBound();
        int vmin[3], vmax[3];
        for (int axis = 0; axis < 3; ++axis) {
            vmin[axis] = Clamp(Float2Int((pb.pMin[axis] - bounds.pMin[axis]) *
                invWidth[axis]), 0, nVoxels[axis] - 1);
            //posToVoxel(pb.pMin, axis);
            vmax[axis] = Clamp(Float2Int((pb.pMax[axis] - bounds.pMin[axis]) *
                invWidth[axis]), 0, nVoxels[axis] - 1); //posToVoxel(pb.pMax, a
        }

        // Add primitive to overlapping voxels
        for (int z = vmin[2]; z <= vmax[2]; ++z)
            for (int y = vmin[1]; y <= vmax[1]; ++y)
                for (int x = vmin[0]; x <= vmax[0]; ++x) {
                    int o = z*nVoxels[0] * nVoxels[1] + y*nVoxels[0] + x;

                    if (!voxels[o]) {
                        // Allocate new voxel and store primitive in it
                        voxels[o] = voxelArena.Alloc<Voxel>();
                        *voxels[o] = Voxel(primitives[i], o);//ly:
                    }
                    else {
                        // Add primitive to already-allocated voxel
                        voxels[o]->AddPrimitive(primitives[i]);
                    }
                }
    }//ly:finish constructing voxels
    CTree->Constructor(voxels, nv);
    // Create reader-writer mutex for grid
    for (int i = 0; i < nVoxels[0] * nVoxels[1] * nVoxels[2]; ++i)
        if (voxels[i]) voxels[i]->~Voxel();
    FreeAligned(voxels);
}
```

图 3-11 算法实现

K 值的选取

在我们的 teapot 场景中，通过调节 k 值，不同的网格密度得到的网格分割结果可视化：

当 K=3 时，网格密度 *VoxelsPerUnit* = 0.197：

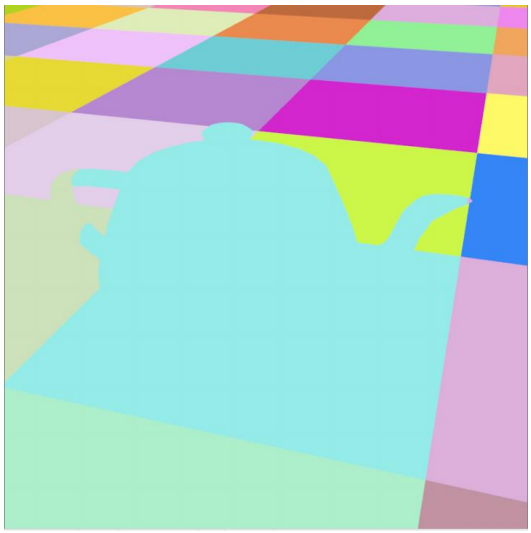

图 3-12 k 值为 3 时网格分割

因为均匀网格对空间分布不均场景的局限性，绝大多数面片都击中在一个网格内，并没有完全发挥出网格区分面片空间距离的作用，同时必须在一个网格内遍历大量面片来建立场景聚类，聚类效率也较低。

当 K=9 时，网格密度 *VoxelsPerUnit* = 0.590：

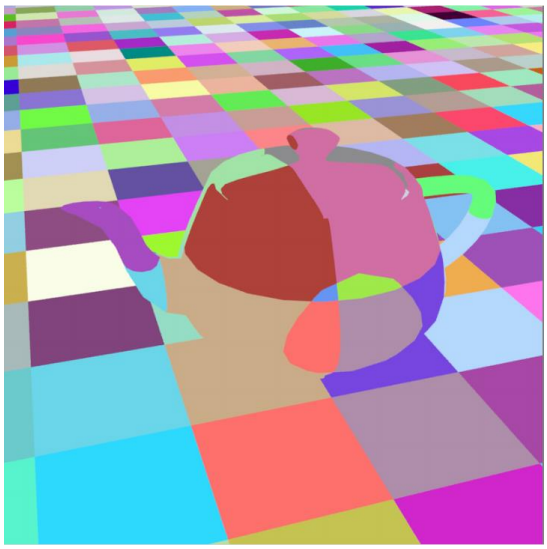

图 3-13 k 值为 9 时网格分割

此时茶壶的几个主要区域已经能被不同网格区分开，每个网格内的面片都有大致相似的取向，即能区分不同物体又能保证法向连续，同时聚类计算也较为有效率。

当 K=30 时，网格密度 *VoxelsPerUnit* = 1.968：

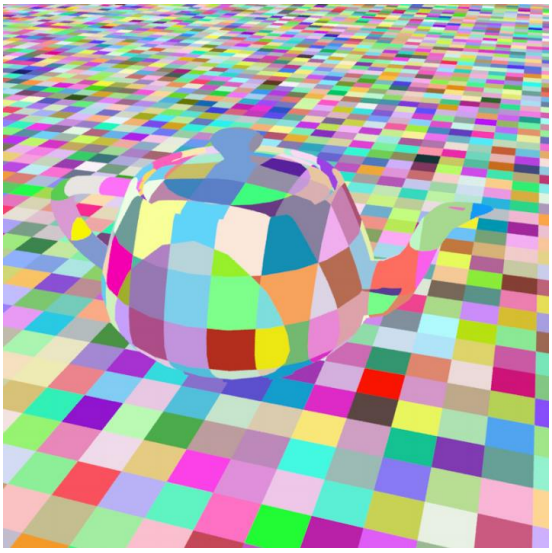

图 3-14 k 值为 30 时网格分割

更大的网格密度使得空间划分更为理想，在下一步空间聚类时，减少了因为曲率的波动性而导致的相隔较远面片聚为一类的可能性。同时 teapot 场景较小，网格划分带来的额外开销并不显著。

考虑到有可能更好的聚类结果，我们为 Teapot 场景设置 K=30 作为网格密度控制参数。但对更大型的场景来说过多的网格增加了网格划分的时间，通常达数分钟，而且在场景物体、面片分布更为均匀的场景中，选取适中的参数也能保证良好的划分结果。实验中的其他几个场景通常会选取参数 k=9 以平衡空间划分效果和计算效率。

第二步：网格内面片聚类

我们的方法空间聚类的原则是材质相同、法向相近、空间位置相近。

就网格内的一个面片而言，它将首先与网格范围内已存在的聚类的代表面片比较，如果它与代表面片的材质相同、几何相似，它就被归入该聚类；否则，没有聚类存在或没有聚类可归属时，它就自成一个新聚类并作为其代表面片。具体实现中，如果一个面片与代表面片材质相同、法向夹角小于 A，它将被归入聚类。

聚类算法代码如下：

```
1    void Constructor(Voxel** voxels, int nVoxel){
2        for (int i = 0; i < nVoxel; i++)
3            if (voxels[i])
4                voxels[i]->FindClusters(this);//voxel之间聚类互斥，不共享
5    }
6
7
```

图 3-15(a) 算法的实现

```
1  void Voxel::FindClusters(Clustertree *CTree){//using Clustertree::AddOneClu
2      vector <gridCluster> ClustersInGrid;
3      if (this->primitives.size() == 0)
4          return;
5
6      for (uint32_t i = 0; i < primitives.size(); ++i) {//对所有面片遍历
7          Reference<Primitive> &prim = primitives[i];
8          if (ClustersInGrid.size()>0){//存在聚类
9              bool hasClustered = false;
10             //贪心遍历
11             for (uint32_t i = 0; i < ClustersInGrid.size(); ++i)
12             {
13                 if (ClustersInGrid[i].IsInCluster(prim)){//调节的关键函数//
14                     ClustersInGrid[i].addIntoCluster(prim);
15                     //如果Cluster还是-1，可能是拷贝复制，Reference<>并不使
16                     hasClustered = true;
17                     break;//一旦进入一个聚类即终止，尽量不同时在多个聚类内
18                 }
19             }
20             //遍历完发现无现有聚类匹配
21             if (hasClustered == false)
22                 ClustersInGrid.push_back(gridCluster(prim, VoxelID));
23
24         }
25         else{ //暂未存在聚类
26             ClustersInGrid.push_back(gridCluster(prim,VoxelID));
27         }
28     }//ClustersInGrid构造完成
29
30     for (uint32_t i = 0; i < ClustersInGrid.size(); i++)
31     {
32         CTree->AddOneCluster(ClustersInGrid[i]);
33     }
34     return;
35 }
36
```

图 3-15(b) 算法的实现

我们对类内面片间法向夹角阈值 A 的取值作出了实验，希望尽可能地在聚类精度和效率间平衡，同时获得符合场景几何特征的自适应采样率分配结果。

我们分别试验了 A=30°，A=10°，A=5°的聚类情形，最终选取 A=10°作为我们的聚类判定条件

当 *A=30°*、*A=10°*、*A=5°* 时，聚类情况如图 3-9、3-10、图 3-11 所示

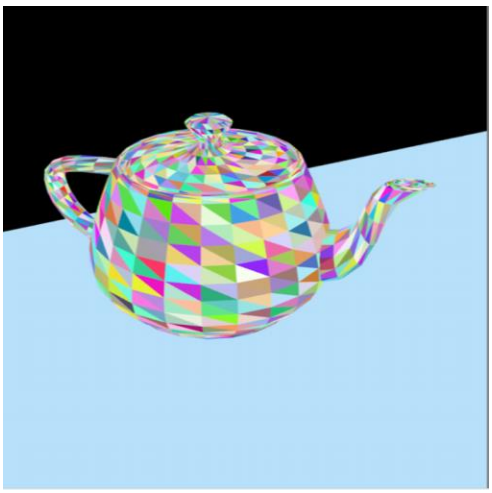

图 3-16 当 A=5° 时

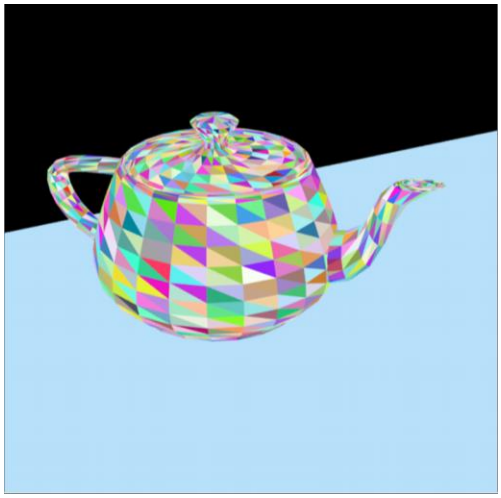

图 3-47 当 A=5° 时

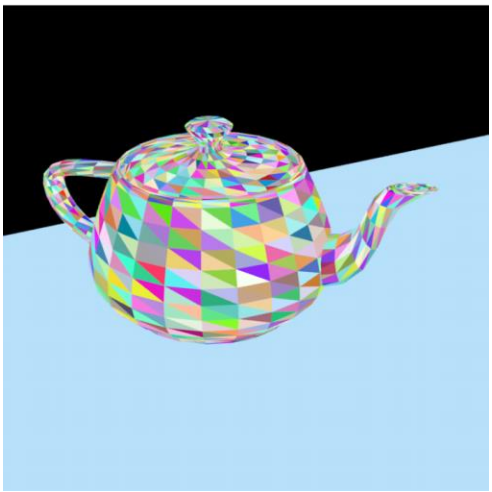

图 3-58 当 A=5° 时

对于 Teapot 场景而言，由于场景面片数较少，面片较为粗糙，面片之间法向几何相似性很差，所以 A 的多个取值对聚类的影响并不明显。这是面片不够精细带来的问题。

而在地形场景中，*A=10°* 与 *A=30°* 聚类结果差异如下:(左图是 10° 聚类结果，右图是 30° 聚类结果)

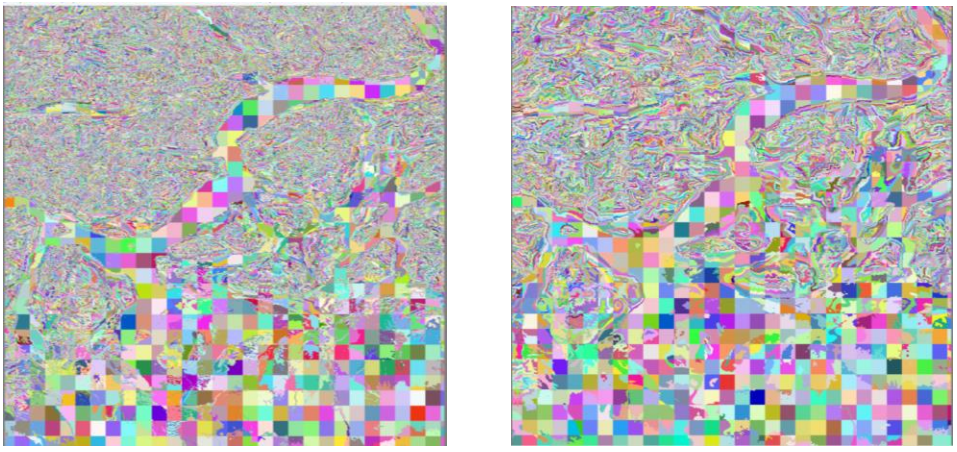

图 3-69

A=10° 聚类能更好地反映精细的几何特征，更有可能更精确地分布采样，且时间开销、内存开销也在可接受范围内，所以对地形场景我们选取 *A=10°* 作为聚类条件。

更多考虑：根据空间连通性对聚类再分割

离散的区域由于法向相似而被聚类在一起，影响了聚类的可靠性，我们需要根据连通性将这样的区域区分开，第 3.2.3.2 节会具体讨论如何进行考虑连通性的空间聚类。

3.4 自适应采样率的计算

现在已有场景面片按照几何相似性聚类，在较邻近的空间范围内，法向相似的面片反射的光照情况也相似，可以以较恒定数量采样每个聚类的所有面片获知这一“局部平坦”区域的光照情况。那么对被一个聚类覆盖的所有像素点来讲，它们应该平均地分担该聚类的总采样数量。换句话讲：一个像素覆盖的聚类区域占该聚类区域总大小的比例，应该正比于该像素采样率占该聚类总采样数的比例。又因我们希望聚类总采样数对每个聚类恒定，所以像素采样率能由像素覆盖空间区域占聚类区域的比例获得。

$$f(\frac{SP}{SC}) = \frac{SamplingRateP}{SamplingRateC} \qquad \text{式（3-3）}$$

其中 *SP* 表示像素 *P* 的区域大小，*SC* 表示像素 *P* 对应的聚类 *C* 的区域大小，这里区域大小都用二维面积衡量，*SamplingRateP* 表示像素 P 采样率，*SamplingRateC* 表示像素 *P* 对应的聚类 *C* 的总采样率，函数 f 是某正相关的映射关系。像素 P 的区域面积可以看作常数值 1，*SamplingRateP* 恒定，所以欲估算像素 P 处采样率，只要获知像素 *P* 对应的聚类 *C* 的区域面积。

$$SamplingRateP = ClusterSamplingRate * f(\frac{1}{SC}) \qquad \text{式（3-4）}$$

我们简述计算采样率的算法步骤：

- 遍历成像平面所有像素，对像素 *P*，向场景投射光线 *Ray*
- 投射光线 *Ray* 与场景面片相交，所交面片所在聚类即是覆盖像素 *P* 的聚类
- 估算像素 *P* 所对应的聚类的区域尺度
- 利用对应聚类区域尺度计算像素 *P* 处采样率

3.4.1 利用包围盒快速估算聚类区域尺度

要反映像素区域占整个聚类区域的比例，应考虑将聚类区域投影到成像平面。一个直观的想法是对聚类所有面片求总的投影面积。这很有趣可以避免很多问题但是计算开销也是比较大。注意只是比较大，大约只是增加 O(N), N 是场景面片数。更快速的估算方法是同每个聚类的轴对齐包围盒代表其空间尺度。轴对齐包围盒利用聚类内面片坐标右上点和左上点表示面片数组的空间范围，直接将聚类空间看作轴对齐的长方体，长方体的投影面积就能够很简单地计算出来。以包围盒代替真实投影面积加快了计算效率，也会带来很多精度误差，在 3.2 节进行了详细讨论。

计算包围盒的投影面积，用包围盒三个面投影到视点面积之和即求得，这涉及到计算视点光线与包围盒各面夹角正弦值。且各面与坐标平面平行，各与坐标轴垂直，夹角正弦值由正则化后的投影光线 xyz 坐标值可求得。

包围盒估算聚类区域体积的算法如下：

```
Vector _dir = Normalize(cast_ray.d);
float sin_XoY = abs(_dir.z);
float sin_XoZ = abs(_dir.y);
float sin_YoZ = abs(_dir.x);

Vector _l = boundbox.pMax - boundbox.pMin;
float S_XoY = (_l.x*_l.y);
float S_XoZ = (_l.x*_l.z);
float S_YoZ = (_l.y*_l.z);

float S = S_XoY*sin_XoY + S_XoZ*sin_XoZ + S_YoZ*sin_YoZ;//包围盒相对视点
```

图 3-20 算法实现

3.4.2 基于场景复杂度的自适应采样率

在基于场景复杂度的自适应采样率计算阶段，每个像素的最低采样速率之外，我们为整个视平面绘制预留一定的采样预算，即，每个像素的预留采样率乘以像素数，所有像素根据其“采样重要性”从该采样预算中自适应获取一部分额外采样率。采样重要性的计算将在本节表述。

如图 3-14，我们首先为每个像素向场景中投射一条视点光线，光线所交面片的所属聚类将决定采样率的计算。我们为聚类估算其投影到视平面的面积以比较像素面积占其

的比例，从而推导像素的采样重要性。额外的，我们为聚类主法向分配一部分权值——法向与视线夹角过大的可见面片需要多分配采样。再其次，考虑投影光线交点距离，为距离远的面片多分配些采样。

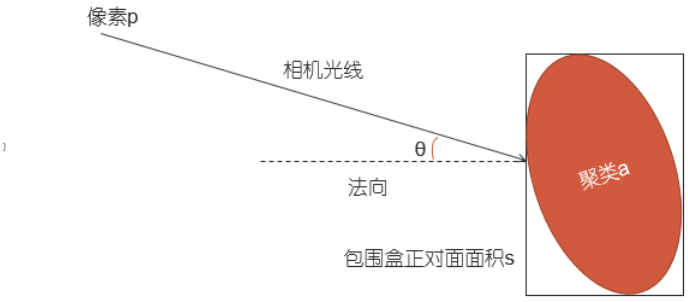

图 3-21 光线投射图

于是我们的自适应采样公式总结如下：

$$\mathrm{SamplingRateP} = \mathrm{ClusterSamplingRate} * \mathrm{f}(\frac{1}{\mathrm{sc}}) \qquad \text{式（3-5）}$$

自适应采样率分布结果的实例可视化如图 3-15：

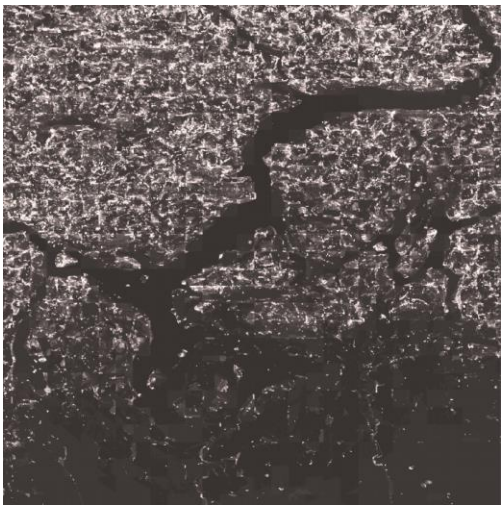

图 3-22 自适应采样率计算实例

第4章　实验分析

4.1　测试环境

表 4-1 测试环境

测试环境	
硬件	Intel Core i7-4600U 2.10GHz
	8.0GB DDR3
软件	Windows8.1
	VS2013
	PBRT

4.2　场景选择与误差评测标准

我们使用了大量场景比较与评测我们的自适应采样方法与已有采样方法的质量、效率差异。本章主要展示了地形和茶壶两个场景的实验图像。两个场景均采用直接光照绘制，地形场景包含大量面片，同时存在几何复杂区域和几何平坦区域；茶壶场景包括高光材质和高曲率表面，以及纹理贴图。两个场景能展示我们的算法相对已有蒙特卡洛采样方法的优越性，也能反映算法的不足之处，对未来工作提出思路。两个场景的分辨率分别为 1024X1024，800X800，采用路径追踪算法，并且以每个像素采样 10000 个样本点绘制的场景结果作为评测误差时的参考图像，如图所示。

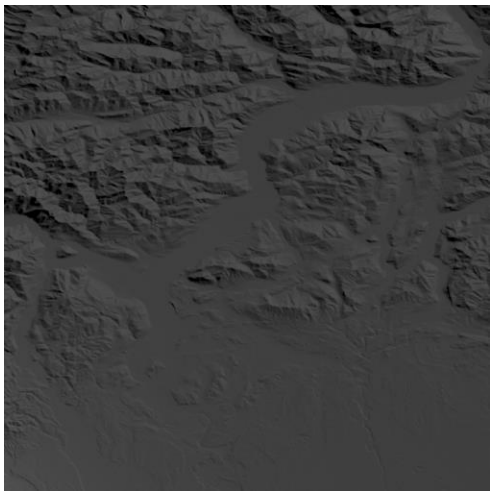

图 4-1 “地形”参考图

图 4-2 “茶壶”参考图

我们采用的误差评价标准是

$$\mathrm{MSE}(p) = \mathrm{E}[(\hat{u}(p) - f(p))^2] \qquad \text{式（4-1）}$$

其中，$\hat{u}(p)$和$f(p)$分别代表绘制图像和参考图像的像素值。MSE 计算了图像估计值与图像参考值之差平方的期望值，能够反映绘制图像的误差水平

4.3 绘制结果比较

我们将基于空间聚类的自适应采样方法与以下方法做了对比：

- LD: 低差异采样
- Mitchell: 基于人眼感知系统误差评价的自适应采样方法，是最常被使用的自适应基础采样方法之一。

滤波器均选取了核为 0.5、宽度为 8 像素的高斯滤波器。特别地，我们为空间聚类方法修改了滤波器使之更多地考虑同聚类像素的贡献值，从而利用聚类特性更高效地滤波。

对于每种方法，我们都设置最小采样数和最大采样数分别为 4spp 和 128spp，实验通过对比相同采样率下各算法的绘制质量和误差水平估计进行，并给出了不同自适应采样方法的采样率强度分布图。

4.3.1 地形场景

图 4-3，图 4-4，图 4-5 显示的是平均采样率为 16 时不同方法生成的地形场景的图像。从图中可看出我们的方法明显好过差异性采样，在特别的区域也要比 mitchell 方法好。尤其在几何复杂的“山地”区域，有较明显的体现。

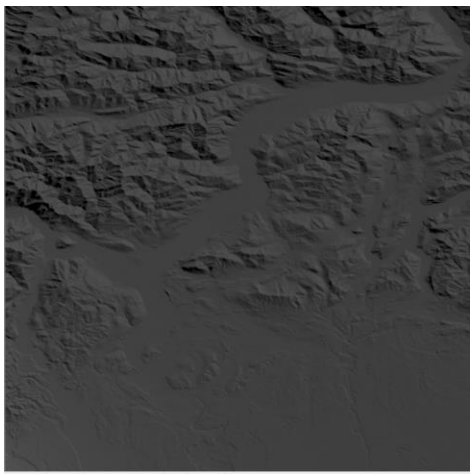

图 4-3 低差异性采样 16spp

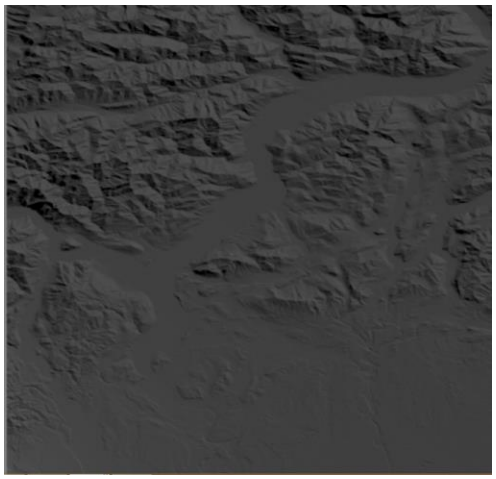

图 4-4 mitchell 16spp

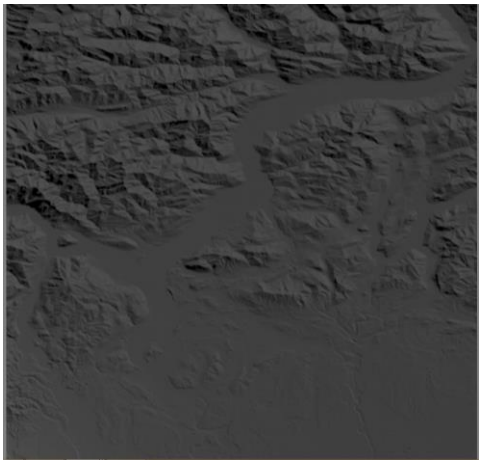

图 4-5 我们的方法 16.21spp

我们的方法总计算时间 80s，聚类时间开销 2s，投射光线计算采样率时间开销 35s，路径追踪时间开销 43s。

图 4-6 与图 4-7 特意在在几何复杂区域将我们的方法与 mitchell 方法做了对比。

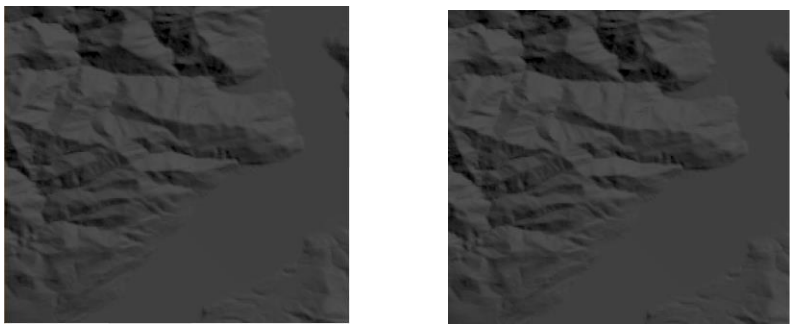

图 4-6 图左为 mitchell 算法 图右为我们的方法

在平坦区域，我们的方法只用了很少的采样就获得了和 mitchell 方法一样的结果。而在山地区域，由于在这些几何复杂区域分配了更多的采样数目，走样明显地减少了不少。

图 4-7、图 4-8、图 4-9 显示的是平均采样率为 64 时不同方法生成的“地形”场景的图像。可以看出，加大采样量后，差异性采样在几何复杂的“山地”区域噪声减少了很多，但在地貌的阴影部分仍然存在走样。对比来看，mitchell 和我们的方法都各有显著的改善。

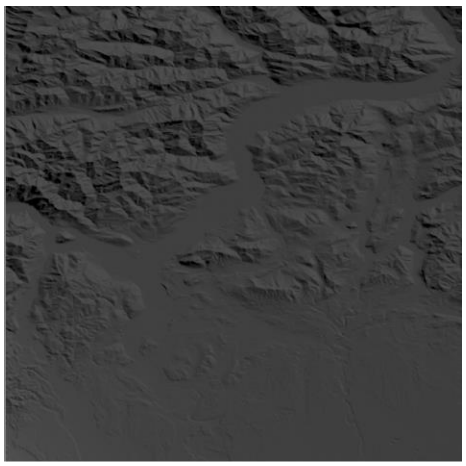

图 4-7 低差异性采样 64spp

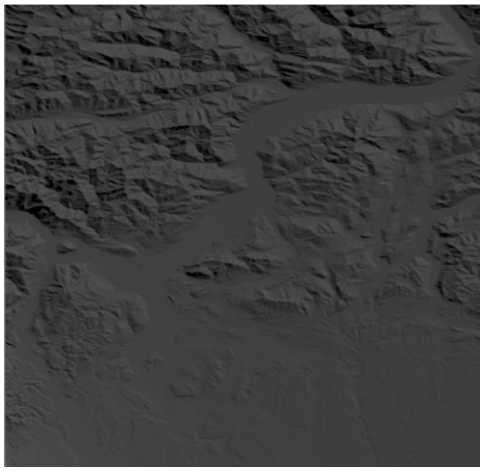

图 4-8 mitchell 67spp

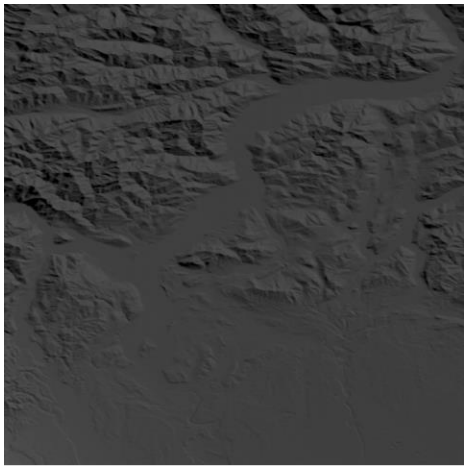

图 4-9 我们的方法 58spp

我们的方法总计算时间 182s，聚类时间开销 2s，投射光线计算采样率时间开销 35s，路径追踪时间开销 145s。

图 4-10 是 mitchell 算法与我们的方法在红色线框内的细节图，可以更清晰的看出，相比较采样率为 16 时的结果，地貌复杂区域内，噪声的收敛速度比较快。

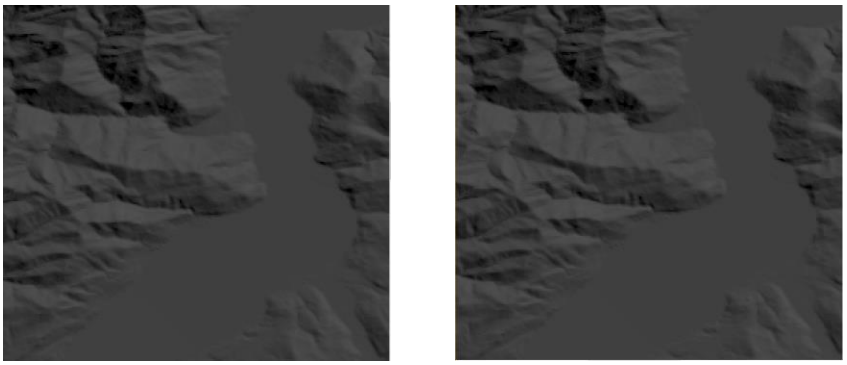

图 4-10 图左为 mitchell 算法 图右为我们的方法

4.3.2 茶壶场景

“茶壶”场景设置中既有高光材质也有复杂的纹理贴图，对验证我们算法在多种光照条件下的适用性是个很好的场景。

图 4-11，图 4-12，图 4-13 显示的是平均采样率为 16 时不同方法生成的“茶壶”场景的图像。由于高光材质反射概率模型光路复杂，所以低采样率时直接绘制所得图像噪点很大。但是对比低差异性采样和 mitchell 方法，我们可以看出，尤其在高光材质的“茶壶”部分，很多区域的噪声被降低了。因为我们在几何复杂的区域分配了更多的采样，而几何复杂的区域在复杂反射概率模型下采样的误差进一步加大，所以需要在这些地方收敛大量噪点。我们的算法较好地拟合了这一情况。

图 4-11 低差异性采样 8spp

图 4-12 mitchell 8spp

图 4-13 我们的方法 7.85spp

我们的方法总计算时间 36.1s，聚类时间开销 0.5s，投射光线计算采样率时间开销 15.6s，路径追踪时间开销 20s。

图 4-14 是 mitchell 算法与我们的方法在红色线框内的细节图，对比 mitchell 方法可

以清晰地看出，茶壶柄、茶壶嘴等处曲率变换较快，mitchell 等已有自适应采样方法并没有很高效地处理好这样的区域，而我们的方法因为探测到这些位置的曲率变化，为复杂区域分配了更多采样，高误差像素的更好地被收敛。

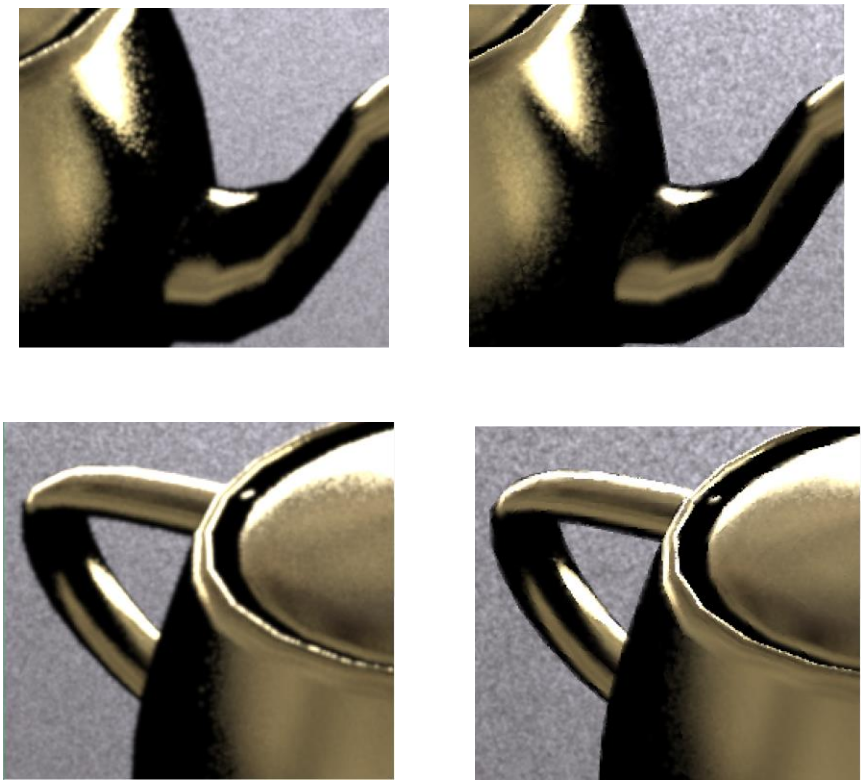

图 4-14 左边为 mitchell 方法 右边为我们的方法

同时，我们发现，由于地板是法向平坦的一块面片，且在较低采样率下我们给平坦区域分配很少的采样，未使用滤波器时，如图 4-14 所示，8spp 采样率下我们的算法在地板区域留存了很多的噪点。通过使用滤波器对同聚类像素滤波，在一个更大的空间区域内计算采样贡献重建图像值，我们的方法对噪点区域的改善明显。

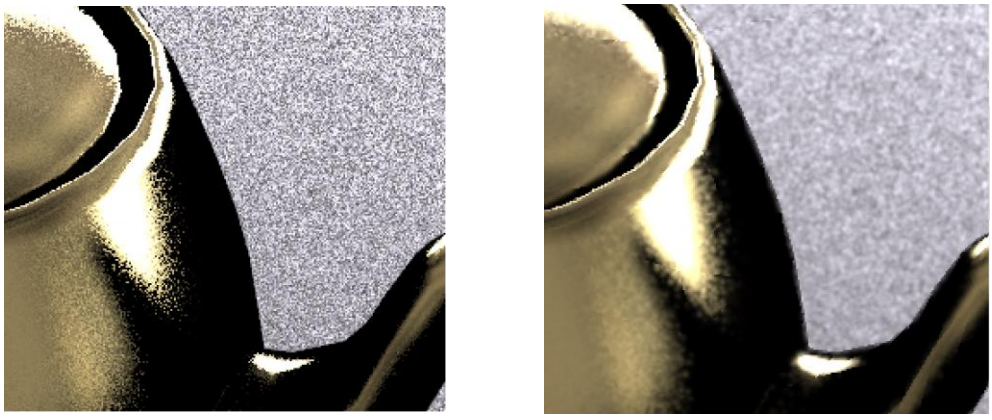

图 4-15 左图为滤波前我们的方法，右图为滤波后我们的方法

图 4-15、图 4-16、图 4-17 显示的是平均采样率为 64 时不同方法生成的“茶壶”场景的图像。可以看出，加大采样量后，各个方法的噪声都减少了很多，茶壶的把柄和壶嘴

也没了明显的走样，远处的地板走样程度减轻。

同时，在曲率复杂的区域我们的方法仍然表面更好，说明随采样率上升我们的方法还有着更快的收敛速度。

图 4-16 低差异性 64spp

图 4-17mitchell 64spp

图 4-18 我们的方法 57.45spp

我们的方法总计算时间 195.4s，聚类时间开销 0.4s，投射光线计算采样率时间开销 15s，路径追踪时间开销 180s。

图 4-18 是 mitchell 算法与我们的方法在红色线框内的细节图，增加采样后，茶壶柄、茶壶嘴等处曲率变换较快的地方，mitchell 方法的走样已减轻，但高光区域等反射模型复杂、采样误差大的区域，mitchell 方法仍没很好地收敛，反映其在这些区域的采样数目不足，而我们的方法在高光区域绘制的更为光滑低噪，高误差像素的更好地被收敛。

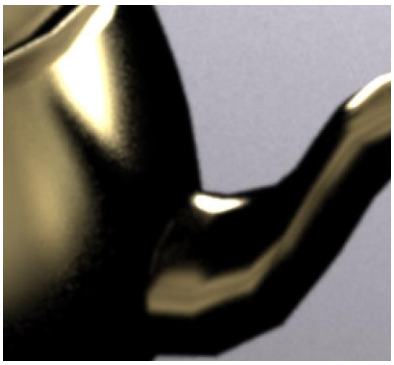


图 4-19 左边为 mitchell 方法 右边为我们的方法


图 4-20 左边为 mitchell 方法 右边为我们的方法

4.3.3 MSE 分析

均方误差 MSE 计算了图像估计值与图像参考值之差平方的期望值，能够反映绘制图像的误差水平。我们统计了茶壶场景下，平均采样率为 8，16，32，64 时各算法绘制图像的均方误差值。各采样率下不同自适应算法参数设置如表 4-2，表 4-3 所示。其中 mitchell 自适应采样的采样参数设置如表 4-2 所示。

表 4-2 mitchell 方法采样参数设置

	平均采样率 8	平均采样率 16	平均采样率 32	平均采样率 64
初始采样率	4	4	12	16
追加采样率	6	16	24	54

我们的空间聚类方法自适应采样的参数设置如表 4-3 所示。

表 4-3 基于空间聚类方法采样参数设置

	平均采样率 8	平均采样率 16	平均采样率 32	平均采样率 64
最低采样率	6	12	24	56
最高采样率	16	32	48	80

为获得较小的绘制误差水平，各算法参数都经衡量选择了较优的参数设置

各算法均方误差值比较

经统计，各算法在不同采样率下的误差水平估计如下表所示：

表 4-4 各采样率下 MSE 比较(高斯滤波)

MSE	平均采样率 8	平均采样率 16	平均采样率 32	平均采样率 64
低差异性采样	174.8	132.5	117.4	109.8
Mitchell 自适应采样	197.5	192.7	151.7	139.9
基于空间聚类自适应采样	153.3	96.08	69.75	60.49

更低的 MSE 值代表更小的绘制误差水平。从 MSE 值上能很清楚的体现基于空间聚类的采样算法比低差异性采样有很大的进步，比起 mitchell 自适应采样方法我们也表现得更加好。我们还计算了两种自适应采样方法相对低差异采样时 MSE 的下降比如下表所示（负值代表 MSE 水平相对低差异采样上升）：

自适应采样方法均方误差值的下降比

表 4-5 mitchell 方法与我们的方法 MSE 下降比比较

MSE 下降比	平均采样率 8	平均采样率 16	平均采样率 32	平均采样率 64
Mitchell 自适应采样	-12.5%	-49.2%	-29.1%	-26.3%
基于空间聚类自适应采样	12.6%	27.2%	40.6%	54.5%

由均方误差值的下降比，对比我们可以更清晰地看出基于空间聚类方法较基于对比度的 mitchell 方法的优越性。

值得一提的是，在低采样率下，Mitchell 方法的均方误差值甚至高于低差异性采样，这是要因为 Mitchell 方法在平坦地区如地板分配的采样数目不够，低采样率下该区域采样误差比均匀采样率的低差异采样更为明显，而滤波器又无法在更大尺度上较好地重建平坦区域，导致了这些区域噪声水平很高。而在茶壶场景中这样的平坦区域又有很多，于是导致 Mitchell 方法的 MSE 甚至大大低于均匀采样率的低差异性采样。

误差水平收敛速度

我们将随着采样率升高、各算法均方误差变化趋势统计如图 4-21。最下方的折线是我们的方法，中间的折线是低差异性采样，最上方是 mitchell 方法。由折线图斜率可以

看出，在实验的采样率区间内，随着采样数上升，我们的方法均方误差收敛速度也是最快的，这也达到了我们加速噪声收敛速度的目的。

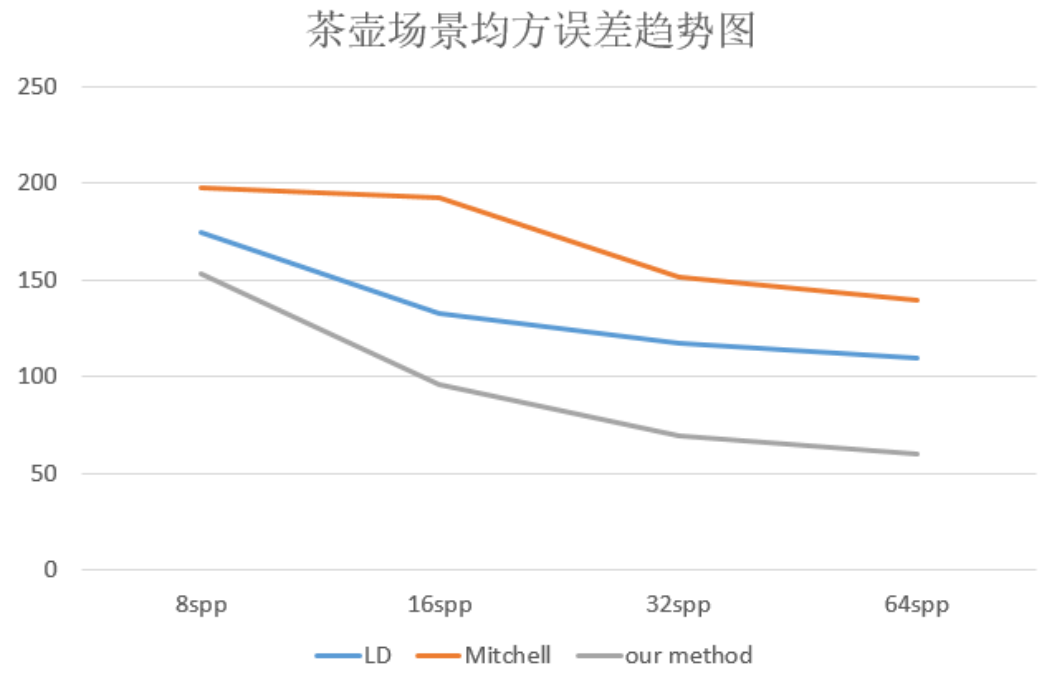

图 4-15 各个算法均方误差变化趋势图

滤波器对误差水平的影响

除此之外，为了分析滤波器对噪声收敛的影响，我们分别统计随机采样下和空间聚类方法未滤波时均方误差水平估计如下：

表 4-6 各采样率下随机采样与空间聚类方法未滤波 MSE 比较

MSE	平均采样率 8	平均采样率 16	平均采样率 32	平均采样率 64
随机采样	834.3	337.6	140.2	50.6
基于空间聚类自适应采样	1124.0	458.1	204.2	56.11

通过对比表 4-4，我们发现，滤波器在采样率较低时对误差水平的减小作用巨大，但随着采样率提升，有无滤波器误差水平变得十分接近。当采样率上升到 64 左右时，滤波器甚至使得误差水平上升。这是因为滤波器在重建采样值时将邻域像素采样共同计算贡献值，引入了偏差，而采样率上升后，滤波前的图像就已有较低的误差水平，强度较高的滤波器将尖锐变化的区域与平坦区域共同对待滤波，视觉上表现为过模糊，所以误差水平也提高了。但基于空间聚类的自适应采样方法滤波时因为利用聚类信息，能更好地降低滤波器带来的偏差。反映在均方误差值上也体现为高采样率下滤波后误差水平较滤波前升高有限。

4.4 采样强度分布图

采样强度分布图用合成图像的每个像素采样率表示，采样率越大颜色越偏向白色，采样率越小越偏向黑色。以下给出的采样强度分布图我们均采用平均采样量为 64 的图像。

4.4.1 地形场景

理论上分析，在几何复杂等区域光照情况变化快，比较难以处理的，因此采样率应该集中在这些区域。从实验结果分析，如图 4-23 所示，我们的方法在图像上部的山地区域处采样更多，河流和右下平坦区域只按最低采样速率采样，复杂和平坦区域区别分明，充分说明了我们自适应采样的合理性。而 mitchell 方法在平坦区域分配了更多采样，在几何复杂处的采样量相对就要少很多。

图 4-22 mithcell 方法

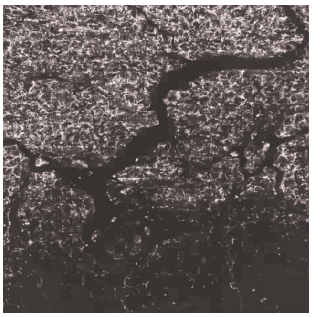

图 4-23 我们的方法

4.4.2 茶壶场景

这个茶壶场景，壶柄、壶嘴等曲率变化较大的地方以及物体边缘是比较难处理的区域，理应分配更多的采样量。从实验结果分析，相比较 mitchell 方法（图 4-24），其在这些区域的采样并不能完全覆盖曲率较大的部分和物体边缘，绘制效果与我们的方法有差距。而我们的方法（如图 4-25）在这些区域采样更多，体现了我们方法的台理性。

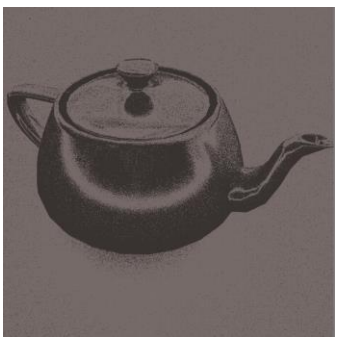

图 4-24 mitchell 方法

图 4-25 我们的方法

第5章　结论与展望

5.1　结论

整体光照是现代图形学的一个重要分支，在很多领域都有广阔的应用前景，如电影特技，辅助设计，虚拟现实等等。

整体光照的目的是使得计算机能够通过算法模型模拟真实的环境，结合 3D 造型等技术，得到逼真的生成图像。

蒙特卡洛路径追踪算法是其中一种模型，有着很多优点，诸如对多种光现象的支持，但是它本身的算法决定了它的一些不足，那就是因为蒙特卡洛方法导致的收敛速度的问题。

为了使得图像上每个像素都能够尽量收敛于真实值，需要对大量的采样，这种做法导致了巨大的计算开销的浪费。

为了减小这种浪费，基于直接获取场景区域表面几何复杂度的初衷，我们将几何相似、光照材质相同、空间相近的表面聚类，按空间聚类分配采样数目，实现对几何复杂区域敏感的自适应采样率分配，加速蒙特卡洛绘制的收敛。更重要的，采样方法直接获取场景信息而不需要初始采样的步骤，不依赖噪声很大的初始绘制结果，为集成到各类已有的蒙特卡洛采样方法中提供了可能。

实验证明，无论是在小采样量，还是在大采样量的情况下，这种算法能有效地减少计算开销的浪费，在同样计算量的比较下，可以发现，新算法所生成的图像的质量更令人满意，生成的图像更接近真实图像，在温度图中，可以看到采样量合理的分配在了各个像素之上。

5.2　展望

未来的工作可以从以下几个方面进一步深入：

- 提出新的聚类尺度到像素采样率的映射函数。
- 结合基于图像重建误差估计的自适应采样方法。（这类方法迭代地进行自适应采样，但是初始采样仍然采用随机分布方式，其在复杂区域存在大量噪声，将我们的方法应用其初始采样过程，将进一步加速这类迭代方法的收敛速度）。
- 考虑光源位置、场景分布等改进我们的自适应采样方法，使能更好地应用于直接与间接光照计算。

参考文献

[1] Kajiya, James T. "The rendering equation." ACM Siggraph Computer Graphics. Vol. 20. No. 4. ACM, 1986.

[2] Jensen, Henrik Wann. "Global illumination using photon maps." Rendering Techniques' 96. Springer Vienna, 1996. 21-30.

[3] Jensen, Henrik Wann. Realistic image synthesis using photon mapping. AK Peters, Ltd., 2001.

[4] Hammersley, John. Monte carlo methods. Springer Science & Business Media, 2013.

[5] Yakowitz, Sidney J. Computational probability and simulation. Reading, Massachusetts: Addison-Wesley, 1977.

[6]Mitchell, Don P. "Generating antialiased images at low sampling densities." ACM SIGGRAPH Computer Graphics. Vol. 21. No. 4. ACM, 1987.

[7] Mitchell, Don P. "Spectrally optimal sampling for distribution ray tracing." ACM SIGGRAPH Computer Graphics 25.4 (1991): 157-164.

[8] Rigau, Jaume, Miquel Feixas, and Mateu Sbert. "Refinement criteria based on f-divergences." Rendering Techniques. 2003.

[9] Bolin, Mark R., and Gary W. Meyer. "A perceptually based adaptive sampling algorithm." Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM, 1998.

[10] Overbeck, Ryan S., Craig Donner, and Ravi Ramamoorthi. "Adaptive wavelet rendering." ACM Trans. Graph. 28.5 (2009): 140-1.

[11] Soler, Cyril, et al. "Fourier depth of field." ACM Transactions on Graphics (TOG) 28.2 (2009): 18.

[12] Lehtinen, Jaakko, et al. "Temporal light field reconstruction for rendering distribution effects." ACM Transactions on Graphics (TOG). Vol. 30. No. 4. ACM, 2011.

[13] Hachisuka, Toshiya, et al. "Multidimensional adaptive sampling and reconstruction for ray tracing." ACM Transactions on Graphics (TOG). Vol. 27. No. 3. ACM, 2008.

[14] Wang, Guangwei, Guofu Xie, and Wencheng Wang. "Efficient search of lightcuts by spatial clustering." SIGGRAPH Asia 2011 Sketches. ACM, 2011.

[15] Rousselle, Fabrice, Claude Knaus, and Matthias Zwicker. "Adaptive sampling and reconstruction using greedy error minimization." ACM Transactions on Graphics (TOG). Vol. 30. No. 6. ACM, 2011.

[16] Li, Tzu-Mao, Yu-Ting Wu, and Yung-Yu Chuang. "SURE-based optimization for adaptive sampling and reconstruction." ACM Transactions on Graphics (TOG) 31.6 (2012): 194.

[17] Rousselle, Fabrice, Claude Knaus, and Matthias Zwicker. "Adaptive rendering with non-local means filtering." ACM Transactions on Graphics (TOG) 31.6 (2012): 195.

[18] Pharr, Matt, and Greg Humphreys. Physically based rendering: From theory to implementation. Morgan Kaufmann, 2004.

[19] Šik, M. "A Survey of Adaptive Sampling in Realistic Image Synthesis."

[20] Zwicker, Matthias, et al. "Recent advances in adaptive sampling and reconstruction for monte carlo rendering." Computer Graphics Forum. Vol. 34. No. 2. 2015.

[21] 徐庆, et al. "基于模糊不确定性的自适应采样." 计算机辅助设计与图形学学报 20.6 (2008): 689-699.

[22] CUI Chengxun．GH—distance based adaptive sampling algorithm[D]．Tianjin：Tianjin University．2009(in Chinese)．[崔承勋. 基于 GH-distance 的自适应性采样算法 [D]. Diss. 天津大学, 2009.]

致　　谢

在本课题的研究和论文写作期间，我得到了老师和同学们的热心帮助，在此向他们表示诚挚的谢意。

本论文从选题到最后定稿成文，中国科学院软件研究所的王文成老师一直给予了悉心指导，王文成老师那种严谨求实的作风，孜孜不倦的开拓精神和敬业精神令我深受启迪和教益，谨向我的指导老师王文成老师致以深深的谢意。

感谢本校计算机院张海涛老师，从中期到论文成稿一直给了我许多珍贵的建议。

感谢王光伟学长对我的耐心指导。学长不仅在科研工作中为我作出榜样，同时也给予我学习和职业生涯规划的有益建议。

最后，我还要向所有曾经帮助过我的同学和朋友们致敬。你们的鼓励和帮助是我永远前进的动力，真心的谢谢你们。