辐射度是渲染过程计算出的最终量.到目前为止,我们一直在使用反射方程来计算它:

其中是从表面位置沿视图方向发出的辐射,是之上的方向的半球,是针对评估的BRDF和当前的进入方向,是从传入的辐射,是与之间的点积,负值截断为零.

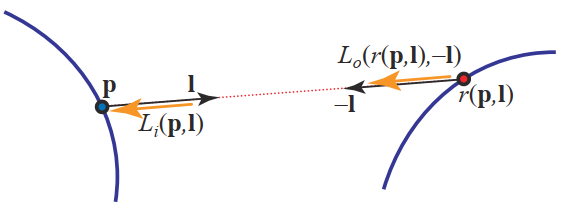
**11.1 渲染方程**

反射方程式是Kajiya于1986年提出的全渲染方程式的一种受限特例[846].渲染方程有不同的形式.我们将使用以下形式:

其中新元素为,它是从表面位置沿方向发出的辐射,并进行以下替换:

该项意味着从方向到位置的入射辐射等于在相反方向上从其他点的出辐射.在这种情况下,“其他点”由射线投射函数定义.此函数返回被沿方向投射的光线击中的第一个表面点的位置.参见图11.1.

**图11.1**: 着色表面位置,照明方向,射线投射函数和入射辐射,也表示为.



渲染方程式的含义很简单.为了着色表面位置,我们需要知道在视图方向上离开的出射辐射,它等于发射辐射加上反射辐射.在前面的章节中已经研究了光源的发射以及反射.甚至光线投射操作也不像看起来那样陌生.例如,Z缓冲区针对从眼睛投射到场景中的光线进行计算.

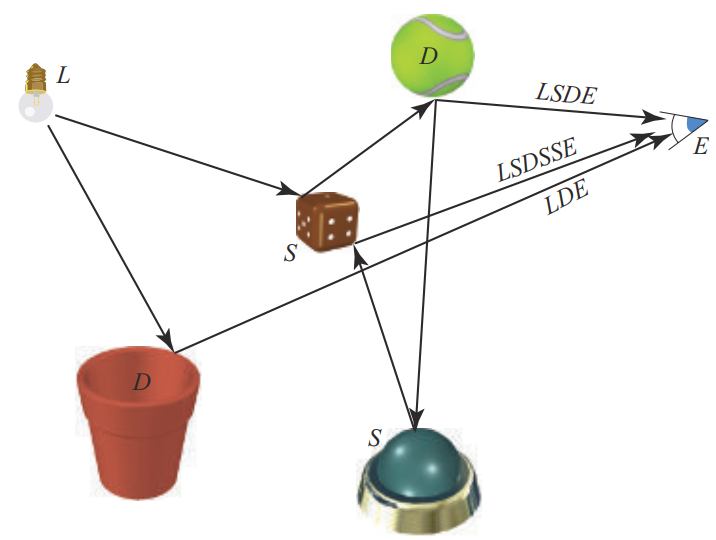
唯一的新项是,这明确表明了进入一个点的辐射必须从另一个点发出的事实.不幸的是,这是一个递归项.也就是说,它是通过对位置的输出辐射进行又一求和而得出的.这些又需要无限地从位置计算出辐射.现实世界可以实时令人惊讶计算所有这些.

我们凭直觉知道这一点,即光线照亮了一个场景,每次碰撞以及周围发生碰撞的光子都以各种方式被吸收,反射和折射.渲染方程很重要,因为它以简单的方程概述了所有可能的路径.

渲染方程的一个重要属性是,它相对于发出的光照是线性的.如果我们使灯光强度增加一倍,则着色的结果将变亮两倍.材料对每种光的响应也独立于其他光源.也就是说,一盏灯的存在不会影响另一盏灯与材料的相互作用.

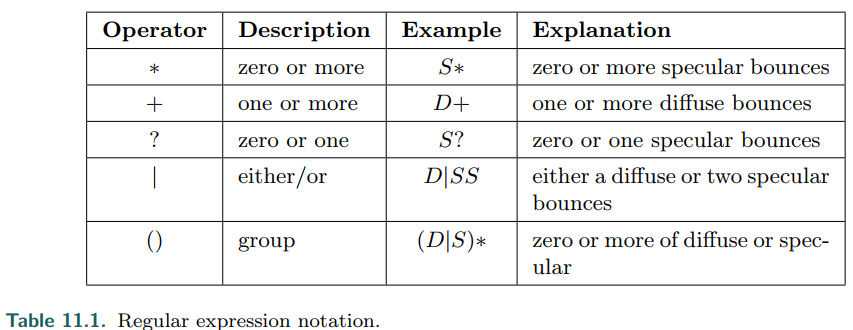
在实时渲染中,通常仅使用局部照明模型.仅需要可见点的表面数据即可计算照明,而这正是GPU可以最有效地提供的.对基元进行独立处理和栅格化,然后将其丢弃.在b点进行计算时,无法访问a点的照明计算结果.透明度,反射和阴影是全局照明算法的示例.他们使用来自被照明物体以外的其他物体的信息.这些效果大大有助于提高渲染图像的真实感,并提供有助于观看者理解空间关系的线索.同时,它们的模拟也很复杂,可能需要进行预计算或渲染多次计算以计算一些中间信息.

思考照明问题的一种方法是通过光子采取的路径.在局部照明模型中,光子从光传播到表面(忽略中间物体),然后到达眼睛.阴影技术考虑了这些中间物体的直接遮挡效果.环境贴图捕获从光源传播到远处物体的照明,然后将其应用于局部发亮的物体,该物体将这些光镜反射到眼睛.辐照度贴图还捕获了光对遥远物体的影响,这些物体在半球的每个方向上都已集成.将从所有这些对象反射的光进行加权和求和,以计算表面的照明,然后再由眼睛看到.



**图11.2**:一些路径及其等效的符号,当它们到达眼睛时.注意,图中显示了从网球延伸出两条路径.

以更正式的方式考虑光传输路径的不同类型和组合,有助于理解现有的各种算法.Heckbert [693]的符号表示法对于描述一种技术模拟的路径非常有用.光子沿其从光(L)到眼睛(E)的行程的每个相互作用都可以标记为漫反射(D)或镜面反射(S).可以通过添加其他表面类型来进一步进行分类,例如“有光泽”,即有光泽但不像镜子一样.参见图11.2.可以使用正则表达式来简要总结算法,以显示算法模拟的交互类型.有关基本符号的概述,请参见表11.1.



光子可以采取从光到眼睛的各种路径.最简单的路径是LE,眼睛直接看到光线.基本的缓冲区为L(D|S)E,或等效地为LDE|LSE.光子离开光源,到达漫反射或镜面表面,然后到达眼睛.请注意,在基本渲染系统中,点光源没有物理表示.赋予光几何形状将产生系统L(D|S)?E,然后光也可以直接进入眼睛.

如果将环境映射添加到渲染器,则紧凑表达式不太明显.尽管Heckbert的表示法是从光线到眼睛的,但通常可以更容易地建立反方向的表达方式.眼睛首先会看到镜面或漫反射表面(S|D)E.如果表面是镜面的,那么它还可以可选地反射渲染到环境贴图中的（远）镜面或漫射表面.因此,存在一条附加的电位路径((S|D)?S|D)E.要计算眼睛直接看到光的路径，请加上？.将此中心表达式设为可选,并用光本身上限:L((S|D)?S|D)？E.

该表达式可以扩展为LE|LSE|LDE|LSSE|LDSE,它分别显示所有可能的路径,或者更短的L(D|S）?S？E.每个人在理解关系和限制方面都有其用途.该符号的部分用途是表示算法效果并能够从中建立效果.例如,L(S|D)是在生成环境贴图时进行编码的内容,而SE是随后访问该贴图的结果.

渲染方程式本身可以用简单表达式L(D|S)\*E来概括,即,来自光源的光子在到达眼睛之前可以撞击零到几乎无限数量的漫反射或镜面表面.

全球照明研究侧重于计算沿其中某些路径的光传输的方法.在将其应用于实时渲染时,我们经常愿意牺牲一些质量或正确性来进行有效评估.两种最常见的策略是简化和预先计算.例如,我们可以假设所有光线在到达人眼之前都会发生散射,这种简化可以在某些环境下很好地发挥作用. 我们还可以离线预先计算一些有关对象间效果的信息,例如生成记录表面照明水平的纹理,然后实时仅执行依赖于这些存储值的基本计算.本章将展示如何使用这些策略实时实现各种全局照明效果的示例.

11.2 一般全局光照

前面的章节集中于解决反射方程的各种方法.我们假设入射辐射的分布一定,并分析了它如何影响着色.在本章中,我们介绍旨在解决完整渲染方程式的算法.两者之间的区别在于,前者忽略了辐射的来源,它只是简单地给出.后者明确指出:到达某一点的辐射是从其他点发出或反射的辐射.

解决完整渲染方程的算法可以生成令人惊叹的逼真的图像(图11.3).但是,这些方法对于实时应用而言在计算上过于昂贵.那么,为什么要讨论它们?第一个原因是,在静态或部分静态的场景中,此类算法可以作为预处理运行,将结果存储起来供以后在渲染期间使用.例如,这是游戏中的常见方法,我们将讨论此类系统的不同方面.

第二个原因是全局照明算法建立在严格的理论基础上.它们直接从渲染方程导出,并对其进行的近似进行仔细分析.在设计实时解决方案时,可以并且应该应用类似的推理.即使我们做出某些捷径,我们也应该知道后果是什么,正确的方法是什么.随着图形硬件变得越来越强大,我们将能够做出更少的妥协,并创建更接近正确物理结果的实时渲染图像.

求解渲染方程的两种常见方法是有限元方法和蒙特卡洛方法.光能传递是一种基于第一种方法的算法;各种形式的光线跟踪使用第二种.在这两种方法中,光线追踪更受欢迎.这主要是因为它可以在同一个框架内有效地处理一般的光传输,包括体积散射之类的影响.它还可以更轻松地规模化和并行化.

我们将简要介绍这两种方法,但有兴趣的读者应参阅涵盖出色解决非实时设置渲染方程细节的任何优秀书籍[400,1413].

11.2.1 光能传递

光能传递[566]是第一种计算机模拟技术,用于模拟漫反射表面之间的反射光.它从算法计算的数量中获得名称.以经典形式,光能传递可以计算区域光的相互反射和柔和阴影.已经有整本关于这种算法的书[76,275,1642],但是基本思想相对简单.光线在环境中反弹.打开灯,照明很快达到平衡.在这种稳定状态下,每个表面本身都可以视为光源.基本的辐射度算法做出简化的假设,即所有间接光都来自漫射表面.对于铺有大理石地板或墙壁上有大镜子的地方,此前提不起作用,但是对于许多建筑环境,这是一个合理的近似值.光能传递可以有效地进行无数次漫反射.使用本章开头介绍的符号,其光传输集为LD\*E.

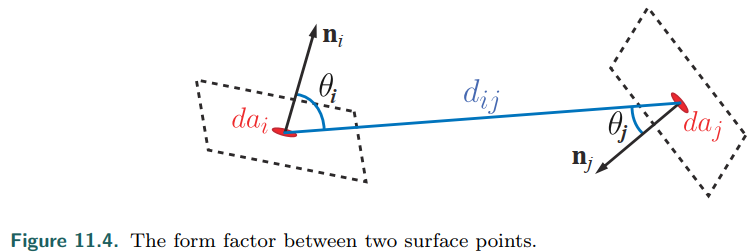
光能传递假设每个表面都由一些小块组成.对于这些较小的区域中的每个区域,它都会计算一个平均的光能传递值,因此这些色块必须足够小以捕获照明的所有细节(例如阴影边缘).但是,它们不需要使表面三角形一对一匹配,甚至不必大小一致.

从渲染方程开始,我们可以得出区域的辐射度等于

其中表示区域的辐射度,是辐射出射量,即区域发出的辐射度,而是次表面反照率(第9.3节).发射仅对于光源是非零的.是区域和之间的形状因数.形状因数定义为

其中是面片的面积,而是点和之间的可见度函数,如果它们之间没有任何阻挡光,则等于1,否则等于0.值和是两个区域法线与射线连接点和之间的角度.最后,是射线的长度.参见图11.4.

形状因数纯粹是几何项.离开区域的均匀扩散辐射能的一部分入射到区域上[399].两个区域的面积,距离和方向以及它们之间的任何表面都会影响其形状系数值.想象一下以计算机监视器为代表的区域.房间中的其他所有区域都将直接接收显示器发出的任何部分光.如果表面在显示器后面或无法“看到”显示器,则该分数可以为零.这些分数加起来等于一.辐射度算法的重要部分是准确确定场景中的成对区域之间的形状因子.



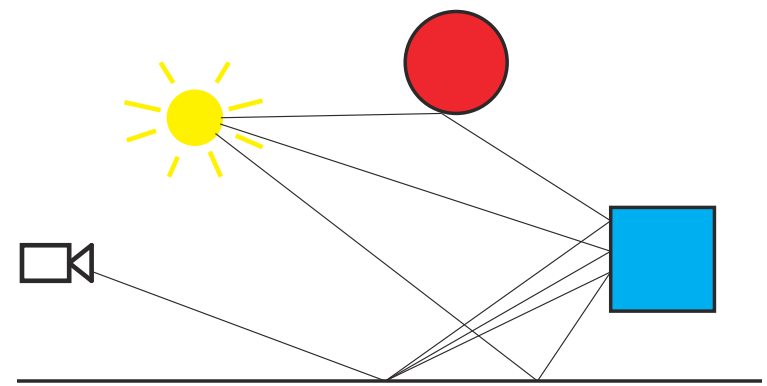
计算出形状因子后,所有区域的方程(方程11.4)被组合成一个线性系统.然后解决该系统,从而为每个区域生成光能传递值.随着区域数量的增加,由于高计算复杂度,减少这种矩阵的成本相当可观.

由于该算法的伸缩性较差且具有其他限制,因此经典的辐射度很少用于生成照明解决方案.但是,在现代实时全局照明系统中,预先计算形状因数并在运行时使用它们执行某种形式的光传播的想法仍然很流行.我们将在本章(第11.5.3节)中讨论这些方法.

11.2.2 光线追踪

射线投射是从某个位置发射射线以确定特定方向上的物体的过程.光线跟踪使用光线来确定各种场景元素之间的光传输.在最基本的形式中,光线是从相机通过像素网格射入场景的.对于每条射线,找到最接近的对象.然后,通过向每条光线发射光线并找出两者之间是否存在任何对象,来检查此交点是否在阴影中.不透明的物体会挡住光线;透明物体会使其衰减.可以从相交点生成其他射线.如果表面发亮,则会在反射方向上产生射线.该射线拾取第一个相交对象的颜色,然后对相交点进行阴影测试.还可以在透明固体物体的折射方向上生成射线,然后再次递归评估.这种基本的机制是如此简单,以至于功能射线追踪器已经写在了名片的背面[696].

经典的光线跟踪只能提供有限的一组效果:锐利的反射和折射以及硬阴影.但是,可以使用相同的基本原理来求解完整的渲染方程式.Kajiya[846]意识到,可以使用发射光线并评估它们携带多少光的机制来计算公式11.2中的积分.该方程是递归的,这意味着对于每条射线,我们都需要在不同的位置再次求积分.幸运的是,已经存在处理此问题的坚实数学基础.为在曼哈顿计划期间进行物理实验而开发的蒙特卡洛方法,是专门为解决此类问题而设计的.代替通过正交规则直接计算每个着色点中积分的值,而是在来自域的多个随机点处对被积进行评估.然后将这些值用于计算积分值的估计值.采样点越多,精度越高.该方法最重要的特性是只需要对被积分进行点评估.给定足够的时间,我们可以任意精度地计算积分.在渲染的上下文中,这正是光线跟踪所提供的.当我们发射光线时,我们对等式11.2中的被积点进行点采样.即使在交点处还有另一个积分需要评估,我们也不需要其最终值,我们可以再次对其进行点采样.当光线在整个场景中反弹时,会建立一条路径.沿着每条路径传播的光提供了对被积物的一种评估.此过程称为路径跟踪(图11.5).



**图11.5**.由路径跟踪算法生成的示例路径.所有这三个路径都通过胶片平面中的同一像素,并用于估计其亮度.图底部的地板具有很高的光泽度,可以在较小的立体角内反射光线.蓝框和红球是漫反射的,因此光线在相交点处围绕法线均匀散射.

跟踪路径是一个非常强大的概念.路径可用于渲染有光泽或漫反射的材料.使用它们,我们可以生成柔和的阴影并渲染带有腐蚀性效果的透明对象.将路径跟踪扩展到体积中的采样点(不仅是表面)后,它可以处理雾和次表面散射效果.路径跟踪的唯一缺点是实现高视觉保真度所需的计算复杂性.对于电影品质的图像,可能需要追踪数十亿条路径.这是因为我们从不计算积分的实际值,而仅计算其估计值.如果使用的路径太少,则这种近似将不精确,有时相当不精确.此外,即使对于彼此相邻的点,结果也可能有很大的不同,对于这些点,人们希望照明几乎相同.我们说这样的结果有很大的差异.在视觉上,这本身表现为图像中的噪点(图11.6).已经提出了许多方法来对抗这种影响而不追踪其他路径.一种流行的技术是重要性采样.想法是,可以通过在大多数光线来自的方向上发射更多的光线来大大减少差异.关于路径跟踪和相关方法的许多论文和书籍已经出版.Pharr等人[1413]对现代的基于离线光线跟踪的技术进行了很好的介绍.Veach[1815]为光传输算法的现代推理奠定了数学基础.在本章结尾的第11.7节中,我们将以交互速率讨论光线和路径追踪.

**11.3 环境遮挡** 2020年4月22日17点57分

上一节中介绍的一般全局照明算法在计算上非常昂贵.它们可以产生各种各样的复杂效果,但是可能要花费数小时才能生成图像.我们将以最简单但仍在视觉上令人信服的解决方案开始对实时替代方案的探索,并在本章中逐步构建更复杂的效果.

一种基本的全局照明效果是环境光遮蔽(AO).该技术是由Landis[974]在工业光与魔术公司(Industrial Light＆Magic）于2000年代初开发的,旨在提高电影《珍珠港》中计算机生成的飞机上使用的环境照明的质量.尽管效果的物理基础包括相当多的简化,但结果似乎令人惊讶.当照明缺乏方向性变化且无法展现物体细节时,此方法可廉价地提供有关形状的线索.

11.3.1 环境遮挡理论

可以从反射方程式直接得出环境光遮挡的理论背景.为了简单起见,我们将首先关注朗伯曲面.从这些表面发出的辐射辐射与表面辐照度成正比.辐照度是入射辐射的余弦加权积分.通常,它取决于表面位置和表面法线.再次,为简单起见,我们将假设所有辐射方向的入射辐射率都是常数.这将导致以下公式计算辐照度:

在可能的入射方向的半球上进行积分.在恒定均匀照明假设下,辐照度(以及因此的出射辐射度)不依赖于表面位置或法线,并且在整个对象上是恒定的.这导致平坦的外观.

公式11.6没有考虑任何可见性.某些方向可能会被对象的其他部分或场景中的其他对象所阻挡.这些方向具有不同的入射辐射,而不是.为简单起见,我们将假设从受阻方向入射的辐射为零.这将忽略所有可能会从场景中的其他物体反弹并最终从此类受阻挡方向到达点的光,但是这大大简化了推理过程.结果,我们得到了下面的方程式,该方程式首先由Cook和Torrance提出[285,286]:

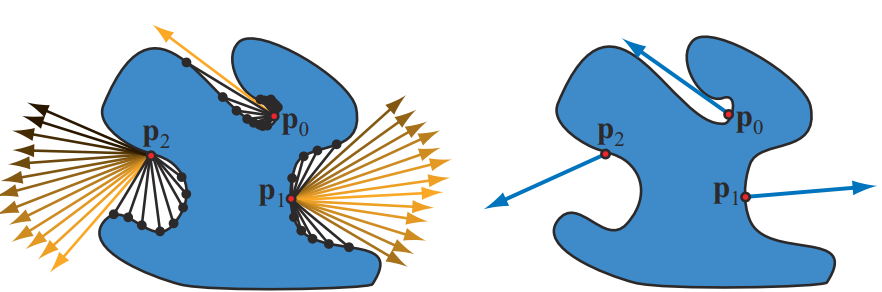
其中是可见性函数,如果从沿方向投射的光线被阻挡,则它的可见性等于0;否则,则为1.

可见度函数的标准化余弦加权积分称为环境光遮挡:

它表示未遮盖的半球的余弦加权百分比.值的范围从0(对于完全遮挡的曲面点)到1(对于没有遮挡的位置).值得注意的是,诸如球体或盒子之类的凸起物体不会对其自身造成遮挡.如果场景中没有其他对象,则凸形对象的各处遮挡值均为1.如果物体有任何凹陷,则在这些区域的遮挡将小于一个.

定义了后,存在遮挡的环境辐照度方程为

请注意,由于的变化,现在辐照度确实随表面位置而变化.如右图11.7所示,这将导致更为实际的结果.折痕处的表面位置会变暗,因为其值较低.比较图11.8中的表面位置和.表面取向也有影响,因为可见度函数在积分时由余弦因子加权.比较图左侧的和.两者都具有大约相同大小的未遮挡立体角,但是的大部分未遮挡区域都围绕其表面法线,因此,如箭头的亮度所示,余弦因数相对较高.相反,的大部分非遮挡区域不在表面法线的一侧,而余弦因子的值相应较低.因此,的值在较低.从这里开始,为简洁起见,我们将不再明确显示对表面位置的依赖性.



**图11.8**:在环境照明下的物体.显示了三个点(,和).在左侧,被遮挡的方向显示为在相交点(黑圈)处结束的黑射线.畅通的方向显示为箭头,并根据余弦因数进行着色,因此,更靠近表面法线的方向会更亮.在右侧,每个蓝色箭头显示平均未遮挡方向或弯曲法线.

除了之外,Landis[974]还计算平均无遮挡方向,称为弯曲法线.此方向向量计算为不遮挡光方向的余弦加权平均值:

积分的结果除以其自身的长度以产生归一化的结果.参见图11.8的右侧.在着色过程中,可以使用所得矢量代替几何法线,以提供更准确的结果,而不会产生额外的性能成本(第11.3.7节).

11.3.2 可见性和遮蔽

需要仔细定义用于计算环境遮挡因子(公式11.8)的可见性函数.对于诸如角色或车辆之类的物体,基于从方向上的表面位置投射的光线是否与同一物体的任何其他部分相交来定义是很简单的.但是,这不能解释其他附近物体的遮挡.通常,出于照明目的,可以假定将对象放置在平面上.通过将该平面包括在可见性计算中,可以实现更实际的遮挡.另一个好处是,物体对地面的遮挡可以用作接触阴影[974].

不幸的是,可见性函数方法对封闭的几何体无效.想象一下一个场景,其中包含一个封闭的房间,里面摆放着各种物品.所有表面的值为零,因为来自表面的所有光线都会撞击到物体上.尝试再现环境光遮挡的外观而不必模拟物理可见性的经验方法通常在这种场景下效果更好.这些方法中的某些方法受Miller的可访问性阴影[1211]概念的启发,该模型可模拟表面上的角落和缝隙如何捕获污垢或腐蚀.

Zhukov等人[1970]引入了遮蔽的思想,该思想通过用距离映射函数替换可见性函数来修改环境遮挡的计算:

与仅具有两个有效值(无遮挡时为1和有遮挡时为0)的不同,是基于光线在与曲面相交之前传播的距离的连续函数.对于大于指定距离的任意相交,或根本没有相交时,为1,相交时为0.超过的交叉路口不需要进行测试,这可以大大加快的计算.图11.9显示了环境光遮挡和环境光遮蔽之间的差异.请注意,用环境光遮挡渲染的图像要暗得多.这是因为即使在较远的距离也可以检测到交叉点,因此会影响的值.

尽管尝试以物理为由对其进行辩解,但遮蔽在物理上并不正确.但是,它通常会给出符合观众期望的合理结果.一个缺点是,必须手动设置的值才能获得令人满意的结果.在计算机图形学中通常会出现这种折衷,这种技术没有直接的物理基础,但是“在感觉上令人信服”.目标通常是可信的图像,因此可以很好地使用这些技术.也就是说,基于理论的方法的一些优点是它们可以自动工作,并且可以通过推理现实世界的工作方式进一步加以改进.

**11.3.3 相互反射** 2020年4月23日12点24分

即使环境光遮挡产生的结果在视觉上令人信服,但它们比完整的全局照明模拟产生的结果要暗.比较图11.10中的图像.

相互反射是造成环境光遮挡与整体全局照明之间差异的重要原因.公式11.8假设被阻挡方向的辐射为零,而实际上互反射将从这些方向引入非零辐射.与右侧的模型相比,图11.10左侧的模型的折痕和凹坑可以看到这种效果变暗.这种差异可以通过增加的值来解决.使用遮挡距离映射函数代替可见函数(第11.3.2节)也可以缓解此问题,因为对于遮挡方向,遮挡函数的值通常大于零.

由于需要解决递归问题,因此以更准确的方式跟踪相互反射非常昂贵.要为一个点着色,必须先为其他点着色,依此类推.计算的值比执行完整的全局照明计算要便宜得多,但是通常希望以某种形式包括这种缺失的光,以避免过暗.Stewart和Langer[1699]提出了一种廉价的方法,但出乎意料的是,准确地估计了相互反射.基于观察,对于在漫射照明下的朗伯场景,从给定位置可见的表面位置往往具有相似的辐射度.通过假定来自阻塞方向的辐射等于从当前着色点传出的辐射,递归被打破,可以找到一个解析表达式:

其中是次表面反照率或漫反射率.这等效于用新的因子代替环境阻塞因子:

该方程趋向于使周围的遮挡因子变亮,使其在视觉上更接近完整的全局照明解决方案(包括相互反射)的结果.效果高度取决的值.基本近似假设在着色点附近表面颜色相同,从而产生某种效果,类似于颜色渗色.Hoffman和Mitchell [755]使用这种方法用天光照亮地形.

Jimenez等人[835]提出了另一种的解决方案.它们对许多场景执行完整的脱机路径跟踪,每个场景均被均匀的白色无限远的环境图照亮,以获取适当考虑互反射的遮挡值.基于这些示例,它们拟合三次多项式,以近似函数,该函数从环境光遮挡值和次表面反照率映射到光遮挡值,该值被互反射光变亮.他们的方法还假定反照率是局部恒定的,并且可以基于给定点的反照率来得出入射反弹光的颜色.

**11.3.4 预计算环境光遮挡**

环境遮挡因子的计算可能很耗时,并且通常在渲染之前离线进行.预先计算任何与照明有关的信息(包括环境光遮挡)的过程通常称为烘焙.

预计算环境光遮挡的最常用方法是通过蒙特卡洛方法.投射光线并检查与场景的相交,并计算公式11.8.例如,假设我们选择围绕法线的均匀分布在半球上的N个随机方向,并在这些方向上跟踪光线.根据相交结果,我们评估可见性函数.然后可以将环境光遮挡计算为

在计算环境暗度时,可以将投射光线限制为最大距离,并且的值基于找到的相交距离.环境光遮挡或遮挡因子的计算包括余弦加权因子.尽管可以直接将其包括在内,如公式11.14所示,但结合重要性因子的更有效方法是通过重要性采样.不是将射线均匀地投射到半球上并对结果进行余弦加权,而是对射线方向的分布进行余弦加权.换句话说,光线更有可能被投射到更靠近表面法线的方向,因为来自这样的方向的结果可能更重要.这种采样方案称为Malley方法.

环境光遮蔽预计算可以在CPU或GPU上执行.在这两种情况下,都可以使用针对复杂几何体加速射线投射的库.最受欢迎的两个是用于CPU的Embree[1829]和用于GPU的OptiX[951].过去,GPU管道的结果(例如深度图[1412]或遮挡查询[493])也已用于计算环境遮挡.随着越来越多的通用射线投射解决方案在GPU上的普及,如今它们的使用已越来越少.大多数商业可用的建模和渲染软件包提供了一种用于预先计算环境光遮挡的选项.

遮挡数据对于对象上的每个点都是唯一的.它们通常存储在纹理,体积或网格顶点中.不管存储的信号类型如何,不同存储方法的特性和问题都是相似的.可以使用相同的方法来存储环境光遮挡,定向光遮挡或预先计算的照明,如第11.5.4节所述.

预计算数据还可以用于对物体彼此之间的环境光遮挡效果进行建模.Kontkanen和Laine [924,925]将物体在其周围的环境光遮挡效果存储在一个称为环境光遮挡场的立方体贴图中.他们使用二次多项式的倒数来建模环境光遮蔽值如何随距对象的距离而变化.其系数存储在立方体图中,以模拟遮挡的方向变化.在运行时,使用遮挡对象的距离和相对位置来获取适当的系数并重建遮挡值.

Malmer等人[1111]通过将环境光遮挡因子以及可选的弯曲法线存储在称为环境光遮挡体的三维网格中,以改进结果.由于直接从纹理读取而不是计算环境遮挡因子,因此计算要求较低.与Kontkanen和Laine的方法相比,存储的数据规模更少,而且两种方法的纹理分辨率都较低,因此总体存储要求相似.Hill[737]和Reed [1469]描述了Malmer等人的方法在商用游戏引擎中的实现.他们讨论了算法的各个实际方面以及有用的优化方法.两种方法都适用于刚性物体,但是它们可以扩展到具有少量运动部件的铰接物体,其中每个零件都被视为一个单独的物体.

无论选择哪种存储环境光遮挡值的方法,都需要注意我们正在处理连续信号.当我们从空间中的特定点发射光线,我们会进行采样,并且在着色之前从这些结果中插值一个值时,便会进行重构.信号处理领域的所有工具均可用于提高采样重建过程的质量.Kavan等人[875]提出了一种称为最小二乘烘焙的方法.遮挡信号在整个网格上均匀采样.接下来,得出顶点的值,以便最小二乘意义上最小化内插和采样顶点之间的总差.他们在将数据存储在顶点的上下文中专门讨论了该方法,但是可以使用相同的推理来得出要存储在纹理或体积中的值.

《命运》就是一个广受赞誉的游戏示例,该游戏使用预先计算的环境光遮挡作为其间接照明解决方案的基础(图11.11）.该游戏在两代游戏机硬件之间的过渡期间交付,需要一种解决方案在新平台上期望的高质量与旧平台在性能和内存使用方面的限制之间取得平衡.该游戏具有动态时间,因此任何预先计算的解决方案都必须正确考虑这一点.开发人员选择环境遮挡是因为其外观可信且成本低廉.由于环境光遮挡使可见度计算与照明分离,因此无论一天中的什么时间,都可以使用相同的预先计算的数据.Sloan等人[**1658**]描述了完整的系统,包括基于GPU的烘焙管道.

育碧公司的刺客信条[1692]和孤岛惊魂[1154]系列还使用一种预先计算的环境光遮挡形式来增强其间接照明解决方案.他们从上到下的角度渲染世界,并处理生成的深度图以计算大规模遮挡.基于相邻深度样本的分布,各种启发式方法用于估计值.通过将对象的世界空间位置投影到纹理空间,可以将生成的世界空间AO地图应用于所有对象.他们称此方法为世界AO. Swoboda [1728]也描述了类似的方法.

**11.3.5 环境光遮挡的动态计算**

对于静态场景,可以预先计算环境遮挡因子和弯曲法线.但是,对于物体移动或形状变化的场景,可以通过动态计算这些因素来获得更好的结果.这样做的方法可以分为在物体空间中操作和在屏幕空间中操作.

用于计算环境光遮挡的离线方法通常涉及从每个表面点向场景中投射大量光线,数十至数百条,并检查交叉点.这是一项昂贵的操作,实时方法侧重于近似或避免大量计算的方法.

Bunnell[210]通过将表面建模为放置在网格顶点处的圆盘形元素的集合来计算环境遮挡因子和弯曲法线.选择圆盘是因为可以通过分析计算一个圆盘与另一个圆盘的遮挡,从而避免了投射射线的需要.简单地将所有其他磁盘的阻塞因子相加会导致由于双重阴影而导致过暗的结果.也就是说,如果一个磁盘在另一个磁盘之后,则即使两个磁盘之间的距离更近,两个磁盘都将被视为遮挡了表面.Bunnell使用巧妙的两次通过方法来避免此问题.第一遍计算包括双重阴影的环境光遮挡.在第二遍中，每个磁盘的阻塞会因其与第一遍的阻塞而减少.这是一个近似值,但实际上它产生的结果令人信服.

计算每对元素之间的遮挡是阶操作,对于最简单的场景,这对于所有场景而言都太昂贵了.通过将简化的表示用于遥远曲面,可以降低成本.Bunnell构造元素的分层树,其中每个节点都是一个磁盘,代表该树中位于其下方的磁盘的聚合.在执行磁盘间遮挡计算时,更高级别的节点用于更远的表面.这样可以将计算减少到,这更加合理.Bunnell的技术效率很高,并且可以产生高质量的结果.例如,它被用于加勒比海盗电影的最终效果图[265].

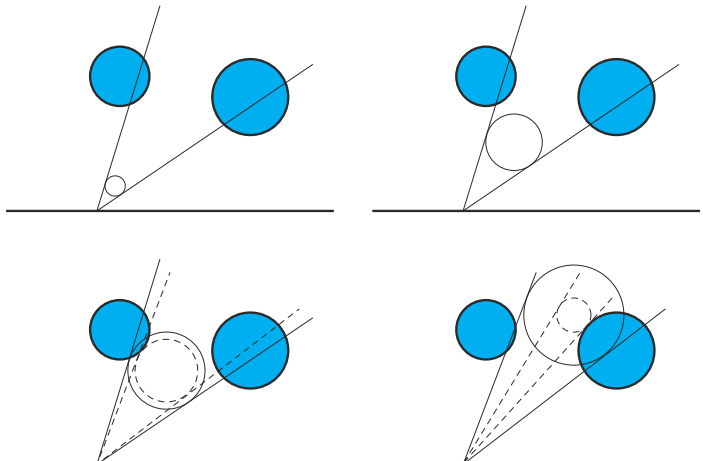
Hoberock[751]提出了对Bunnell算法的一些修改,这些修改以更高的计算费用提高了质量.他还提出了距离衰减因子,其得出的结果与Zhukov等人[1970]提出的遮蔽因子相似.

Evans[444]描述了一种基于有符号距离场(SDF)的动态环境遮挡近似方法.在该表示中,将对象嵌入三维网格中.网格中的每个位置都存储了距对象最近表面的距离.对于任何对象内部的点,此值均为负;对于所有对象外部的点,此值均为正.Evans使用体积纹理为场景创建并存储SDF.为了估计物体上某个位置的遮挡,他使用了一种启发式方法,该方法结合了沿法线逐渐远离表面的多个点处采样的值.如Qu´ılez [1450]所述,当用SDF解析表示(第17.3节)而不是存储在三维纹理中时,也可以使用相同的方法.尽管该方法非物理方法,但结果在视觉上令人愉悦.

Wright[1910]进一步扩展了使用有符号距离场进行环境光遮挡的功能.Wright没有使用临时启发式方法来生成遮挡值,而是执行了圆锥跟踪.圆锥体起源于要着色的位置,并测试了相交处,场景表示编码在距离场中.通过沿轴执行一组步骤并在每个步骤中检查SDF与半径增加的球体的交点,可以近似锥轨迹.如果到最近的遮挡物的距离(从SDF采样的值)小于球体的半径,则遮挡圆锥的那一部分(图11.12).跟踪单个圆锥体是不精确的,并且不允许合并余弦项.由于这些原因,赖特绘制了一组覆盖整个半球的视锥,以估计环境光的遮挡.为了提高视觉保真度,他的解决方案不仅为场景使用全局SDF,而且还使用局部SDF,它们代表单个对象或逻辑连接的对象集. Crassin[305]等人描述了类似的方法.在场景的体素表示的上下文中.他们使用稀疏体素八叉树(第13.10节)存储场景的体素化.他们的计算环境光遮挡的算法是一种更通用的方法,用于呈现完整的全局照明效果(第11.5.7节).

Ren等人[1482]将球的几何形状近似为球体的集合(图11.13).使用球谐函数表示单个球体遮挡的表面点的可见性函数.由一组球体遮挡的聚集可见性函数是各个球体可见性函数相乘的结果.不幸的是,计算球谐函数的乘积是昂贵的操作.他们的关键思想是求和各个球面谐波可见度函数的对数,并对结果求幂.这产生与乘以可见性函数相同的最终结果,但是球谐函数的求和要比乘运算便宜得多.本文表明,采用正确的近似值,对数和指数运算可以快速执行,从而实现整体加速. 该方法不仅计算环境光吸收系数,而且还计算由球谐函数表示的完整球面可见度函数(见10.3.2节).第一个(0阶)系数可以用作环境光吸收系数,接下来的三个(1阶)系数可以用来计算弯曲法线.高阶系数可用于阴影环境贴图或圆形光源.由于将几何近似为边界球,因此无法对折痕和其他小细节的遮挡进行建模.

Sloan等人[1655]在屏幕空间中执行Ren描述的可见性功能的累加.对于每个遮挡物,他们都考虑了一组距离其中心一定的世界空间距离内的像素.可以通过渲染球体并在着色器中执行距离测试或使用模板测试来实现此操作.对于所有受影响的屏幕区域,将适当的球形谐波值添加到屏幕外缓冲区.在累积了所有遮挡物的可见性之后,对缓冲区中的值求幂以获取每个屏幕像素的最终组合可见性函数.Hill[737]使用相同的方法,但是将球谐函数的可见度函数限制为仅二阶系数.按照这种假设,球形谐波乘积只是少数标量乘积,甚至可以由GPU的固定功能混合硬件执行.这使我们即使在性能有限的控制台硬件上也可以使用该方法.由于该方法使用低阶球谐函数,因此不能用于生成边界更明确的硬阴影,而只能生成大部分无方向的光遮挡.



**图11.12**:通过在场景几何体和半径增大的球体之间执行一系列相交来近似圆锥跟踪.球的大小对应于距轨迹起点给定距离的圆锥半径.在每个步骤中,锥角都会减小,以解决场景几何图形的遮挡问题.最终的遮挡因子被估计为被修剪的圆锥体所包围的立体角与原始圆锥体的立体角之比.

**11.3.6 屏幕空间方法**

物体空间方法的开销与场景复杂度成正比.但是,一些有关遮挡的信息可以纯粹从已经可用的屏幕空间数据中推导出,例如深度和法线.这种方法的成本是恒定的,与场景的详细程度无关,而仅与用于渲染的分辨率有关.(实际上,执行时间将取决于深度缓冲区或普通缓冲区中数据的分布,因为这种分散会影响遮挡计算逻辑使用GPU缓存的效率.)

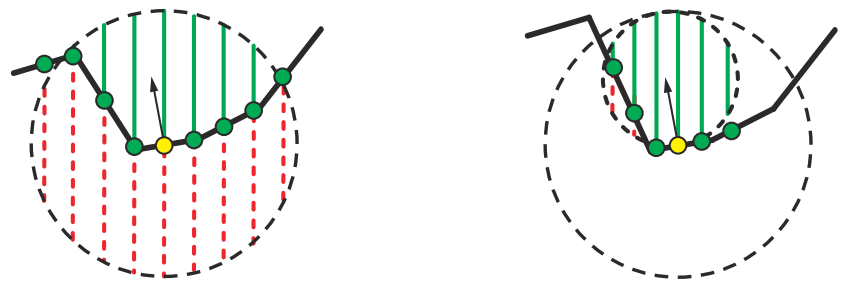
Crytek开发了一种在Crysis[1227]中使用的动态屏幕空间环境光遮挡(SSAO)方法.他们使用z缓冲区作为唯一输入,以全屏方式计算环境光遮挡.通过测试一组点相对于z缓冲区分布在像素位置周围的球体中,可以估算出每个像素的环境光吸收系数.的值是z缓冲区中相应值前面的样本数量的函数.通过的样本数量越少,值越低.参见图11.14.样本的权重随与像素距离的增加而减小,类似于遮蔽因子[1970].请注意,由于样本未按因子加权,因此导致的环境光遮挡是不正确的.不仅考虑表面位置上方的半球中的那些样本,还对所有样本都进行了计数和考虑在内.这种简化意味着,对不应该在表面以下的样本进行计数.这样做会使平坦的表面变暗,边缘比周围的环境明亮.尽管如此,结果通常在视觉上令人愉悦.参见图11.15.

Shanmugam和Arikan[1615]同时开发了一种类似的方法.他们在论文中描述了两种方法.一个可以从附近的小细节中产生精细的环境光遮挡.另一个会从较大的对象生成粗糙的环境光遮挡.将两者的结果合并以产生最终的环境光吸收系数.他们的精细环境光遮挡方法使用访问z缓冲区的全屏通道以及包含可见像素的表面法线的第二个缓冲区.对于每个着色像素,从z缓冲区中采样附近的像素.采样像素表示为球体,并考虑着色像素的法线,计算出着色像素的遮挡项.没有考虑双重阴影,因此结果有些暗.他们的粗糙遮挡方法类似于Ren等人的物体空间方法.在第456页上讨论的[1482]中,闭塞几何近似为球体的集合.但是,Shanmugam和Arikan使用屏幕对齐的广告牌覆盖每个遮挡球的“作用区域”,在屏幕空间中累积遮挡.与Ren等人的方法不同,粗遮蔽法也不考虑双阴影.

业界和学术界都迅速注意到这两种方法的极端简单性,并催生了许多后续工作.许多方法,例如Filion等人[471]在游戏Starcraft II中使用的方法.和McGuire等人[1174]的可扩展环境遮蔽,使用临时启发式方法来生成遮挡因子.这些类型的方法具有良好的性能特征,并公开了一些可以手动调整以实现所需艺术效果的参数.

其他方法旨在提供更原则的计算遮挡的方法.Loos和Sloan[1072]注意到Crytek的方法可以解释为蒙特卡洛积分.他们将计算值称为体积遮蔽,并将其定义为

其中是围绕该点的三维球面邻域,是距离映射函数,类似于公式11.11中的距离函数,是占用函数,如果不被占用则为0,否则为1.他们注意到函数对最终的视觉质量影响很小,因此使用常数函数.在此假设下,体积遮蔽是点附近的占用函数的积分.Crytek的方法随机采样三维邻域以评估积分.Loos和Sloan通过随机采样像素的屏幕空间邻域,在维度上以数值方式计算积分.z维以解析集成.如果该点的球面邻域不包含任何几何体,则积分等于射线与表示的球体之间的交点的长度.在存在几何体的情况下,深度缓冲区用作占位函数的近似值,并且仅在每个线段的未占用部分上计算积分.参见图11.16的左侧.该方法产生的质量结果与Crytek相当,但使用的样本较少,因为在其中一个维度上的积分是精确的.如果表面法线可用,则可以扩展该方法以将其考虑在内.在该版本中,线积分的评估被夹在评估点法线定义的平面上.



**图11.16**: 体积遮蔽(左)使用线积分估计点周围未占用体积的积分.体积环境光遮挡(右)也使用线积分,但是要计算与着色点相切的球的占有率,该球体会根据反射方程对余弦项进行建模.在这两种情况下,积分都是根据球的未占用体积(用实心绿色线标记)与球的总体积(未占用和已占体积的总和,用红色虚线标出)之比估算得出的.对于这两个图,相机都是从上方观看的.绿点表示从深度缓冲区读取的样本,黄点表示正在计算遮挡的样本.

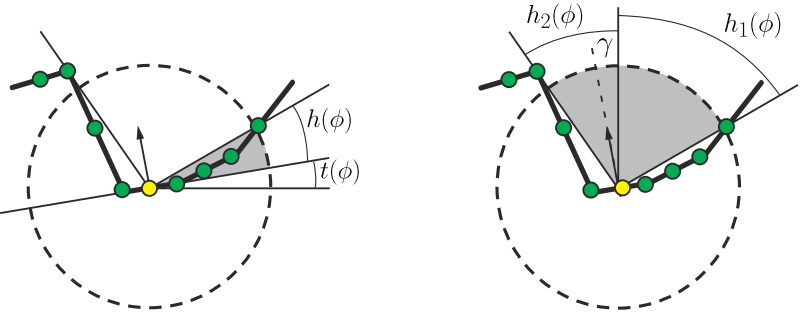
Szirmay-Kalos等人[1733]提出了另一种使用正常信息的屏幕空间方法,称为体积环境光遮挡.公式11.6在法线周围的半球上执行积分,并包含余弦项.他们提出,可以通过从被积数中去除余弦项并用余弦分布截断积分范围来近似这种类型的积分.这会将积分转换为一个球体而不是一个半球,球的半径为半球半径的一半,并沿法线移动,从而完全包围在半球中.就像在Loos和Sloan的方法中一样,通过随机采样像素邻域并在z维度上解析占用函数来计算其未占用部分的体积.参见图11.16的右侧.

Bavoil等人[119]提出了一种不同的方法来估计局部可见性.他们从Max[1145]的地平线制图技术中汲取了灵感.他们的方法称为基于地平线的环境光遮挡(HBAO),它假设z缓冲区中的数据表示连续的高度场.可以通过确定水平角(切线平面上方被邻域遮挡的最大角度)来估计点的可见性.就是说,给定一个点的方向,我们记录可见的最高物体的角度.如果我们忽略余弦项,则可以将环境遮挡因子计算为地平线上未遮挡部分的积分,或者将其减去地平线下遮挡部分的积分:

其中是切平面上方的水平角,是切平面与视图矢量之间的切角,是衰减函数.参见图11.17.项对积分进行归一化,以便结果在零和一之间.

通过在给定的定义地平线的点的距离上使用线性衰减,我们可以解析地计算内部积分:

通过采样多个方向并找到水平角,可以通过数值计算剩余积分.



**图11.17**: 基于水平的环境光遮挡(左)可找到切线平面上方的水平角并对其之间的不遮挡角进行积分.切平面和视图向量之间的角度表示为.Ground Truth的环境光遮挡(右)使用相同的水平角和,但也使用法线和视图向量之间的角度将余弦项纳入计算中.在两个图中,相机都从上方观看场景.这些图显示了横截面,并且水平角是(围绕视线方向的角度)的函数.绿点代表从深度缓冲区读取的样本.黄点是正在为其计算遮蔽的样本.

Jimenez等人[835]也将基于地平线的方法称为地面真实环境遮挡(GTAO).他们的目的是获得真实的结果,与从光线追踪获得的结果相匹配,假设唯一可用的信息是z缓冲区数据形成的高度场.基于地平线的环境光遮挡在其定义中不包括余弦项.它还增加了在公式11.8中不存在的ad hoc衰减,因此,即使接近射线追踪的结果,其结果也不相同.GTAO引入了丢失的余弦因子,去除了衰减函数,并在视图向量周围的参考帧中制定了闭塞积分.遮挡因子定义为

其中和是给定的左右水平角,而是法线和视图方向之间的角度.归一化项与HBAO不同,因为包含了余弦项.这使得开放半球积分为.公式中不包括余弦项,其积分应为.给定高度场假设,该公式与公式11.8完全匹配.参见图11.17.内部积分仍可以通过解析求解,因此仅需要对外部积分进行数值计算.通过对给定像素周围的多个方向进行采样,可以执行与HBAO中相同的集成.

基于水平方法的过程中最昂贵的部分是沿屏幕空间线对深度缓冲区进行采样,以确定水平角度.Timonen[1771]提出了一种专门旨在改善此步骤的性能特征的方法.他指出,用于估计给定方向的地平线角度的样本可以在屏幕空间中沿直线放置的像素之间大量重用.他将遮挡计算分为两个步骤.首先,他在整个z缓冲区上执行线迹.在走线的每个步骤中,他沿线移动时都会更新地平线角度,以解决规定的最大影响距离,并将此信息写到缓冲区中.为水平映射中使用的每个屏幕空间方向创建一个这样的缓冲区.缓冲区的大小不必与原始深度缓冲区的大小相同.它们的大小取决于线之间的间距以及沿线的台阶之间的距离,选择这些参数具有一定的灵活性.不同的设置会影响最终质量.

第二步是基于存储在缓冲区中的层位信息计算遮挡因子.Timonen使用HBAO(公式11.17)定义的遮挡因子,但也可以使用其他遮挡估计器,例如GTAO(公式11.18).

深度缓冲区并不是场景的完美表示,因为对于给定的方向,只有最接近的对象被记录下来,我们不知道其背后发生了什么.许多方法使用各种试探法来尝试推断有关可见对象厚度的一些信息.这些近似值在许多情况下就足够了,并且眼睛可以容忍误差.尽管有些方法使用多层深度来缓解该问题,但由于与渲染引擎的复杂集成和较高的运行时成本,它们从未获得更广泛的普及.

屏幕空间方法依赖于对z缓冲区重复采样以在给定点周围形成一些简化的几何模型.实验表明,要获得较高的视觉质量,需要多达数百个样本.但是,要用于交互式渲染,最多只能采样10到20个样本,甚至更少.Jimenez等人[835]报告说,要适合60 FPS游戏的性能预算,他们只能负担每个像素仅使用一个样本！为了弥合理论与实践之间的鸿沟,屏幕空间方法通常采用某种形式的空间抖动.在最常见的形式中,每个屏幕像素都使用略有不同的随机样本集,它们沿径向旋转或移动.在AO计算的主要阶段之后,将执行全屏过滤.联合双边过滤(第12.1.1节)用于避免在表面不连续处进行过滤并保留尖锐的边缘.它使用有关深度或法线的可用信息来限制过滤,以仅使用属于同一表面的样本.一些方法使用随机变化的采样模式和实验选择的过滤内核.其他人则使用固定大小的屏幕空间图案(例如4×4像素)重复样本集,并使用一个仅限于该邻域的过滤器.

环境光遮挡计算也经常随时间进行超采样[835,1660,1916].通常通过对每帧应用不同的采样模式并执行遮挡因子的指数平均来完成此过程.使用上一帧的z缓冲区,相机变换以及有关动态对象运动的信息,将前一帧的数据重新投影到当前视图.然后将其与当前帧结果混合.基于深度,法线或速度的启发式方法通常用于检测来自最后一帧的数据不可靠且应丢弃的情况(例如,因为已经出现了一些新物体).第5.4.2节在更笼统的背景下说明了时间超采样和抗混叠技术.时间过滤的成本很小,易于实现,尽管它并不总是完全可靠的,但实际上大多数问题并不明显.这主要是因为从不直接可视化环境光遮挡,它只是照明计算的输入之一.将这种效果与法线贴图,反照率纹理和直接照明相结合后,所有较小的伪影都将被掩盖并且不再可见.

11.3.7 环境光遮挡着色

即使我们在恒定,远距离照明的情况下得出环境光遮挡值,也可以将其应用于更复杂的照明场景.再次考虑反射方程:

上述公式包含可见性函数,如第11.3.1节所述.

如果我们要处理的是漫反射表面,则可以用朗伯BRDF代替,它等于次表面反照率,我们得到

我们可以将上面的等式改写为

如果我们使用公式11.8中的环境光遮挡定义,则以上内容简化为

其中

该公式为我们提供了有关着色的新视角.可以认为公式11.22中的积分是将方向滤波内核应用于入射辐射.滤波器在空间和方向上都以复杂的方式变化,但它具有两个重要的特性.首先,由于截断点积,它最多覆盖点处法线周围的半球.其次,由于分母中的归一化因子,其在半球上的积分等于1.

要进行着色,我们需要计算两个函数乘积的积分,即入射辐射度和滤波器函数.在某些情况下,可以以简化的方式描述滤波器,并在例如,当和都使用球谐函数表示时,成本相当低(第10.3.2节).处理该方程复杂度的另一种方法是用具有相似属性的简单滤波器近似滤波器.最常见的选择是归一化余弦内核:

当没有任何东西挡住入射光时,这种近似是准确的.它还涵盖了与我们近似的滤波器相同的角度范围.它完全忽略了可见性,但是等式11.22中仍然存在环境光遮挡项,因此在着色表面上会出现一些与可见性相关的变暗现象.

选择滤波内核,公式11.22变为

这意味着,以最简单的形式,可以通过计算辐照度并将其乘以环境光遮挡值来执行对环境光遮挡的着色.辐照度可以来自任何来源.例如,可以从辐照环境图上采样(第10.6节).该方法的准确性仅取决于近似滤波器代表正确滤波器的程度.对于在整个球面上平滑变化的照明,近似值可得出合理的结果.如果在所有可能的方向上都是恒定的,即好像场景是由代表照明的全白色环境图照亮的,那也是完全准确的.

此公式还使我们对为什么环境光遮挡对于点光源或小面积光源的可见性差了一个见解.它们仅在表面上具有较小的立体角(在精确照明无限小的情况下),可见度函数对照明积分的值具有重要影响.它以几乎二进制的方式控制光的贡献,即完全启用或禁用它.就像在公式11.25中所做的那样,忽略可见性是一个很大的近似值,通常不会产生预期的结果.阴影缺乏清晰度,也没有任何预期的方向性,也就是说,似乎不是由特定的灯光产生的.环境光遮挡不是为此类灯光的可见性建模的好选择.应改用其他方法,例如阴影贴图.但是,值得注意的是,有时使用小的局部光来模拟间接照明.在这种情况下,用环境光遮挡值来调节它们的贡献是合理的.

直到现在,我们都假定我们要对朗伯曲面进行着色.当处理更复杂的非常数BRDF时,不能像公式11.20那样从积分中拉出此函数.对于镜面材料,不仅取决于可见性和法线,还取决于观察方向.典型微面BRDF的波瓣在整个域中发生显着变化.用单一的预定形状近似它太粗糙而无法产生可信的结果.这就是为什么使用环境光遮挡进行阴影处理对于漫反射BRDF最有意义的原因.在接下来的部分中讨论的其他方法对于更复杂的材料模型更好.

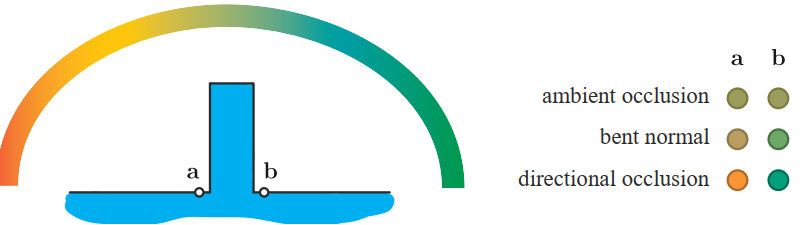
使用弯曲法线(请参见第448页的公式11.10)可以看作是更精确地逼近滤波器的一种方法.可见性项仍未出现在滤波器中,但其最大值与平均未遮挡方向匹配,这使它总体上近似于公式11.23.如果几何法线和弯曲法线不匹配,则使用后者将获得更准确的结果.Landis[**974**]不仅将其用于环境地图的着色,还用于某些直接照明,而不是常规的着色技术.

对于使用环境贴图进行着色,Pharr[**1412**]提出了一种替代方案,该方案使用GPU的纹理过滤硬件来动态执行过滤.滤波器K的形状是动态确定的.它的中心是弯曲法线的方向,其大小取决于的值.这与公式11.23中的原始滤波器提供了更加精确的匹配.

**11.4 定向遮挡**

即使仅使用环境光遮挡可以极大地提高图像的视觉质量,它仅仅是一个简化的模型.当处理甚至是大面积的光源,它对可见性的估计也很差,更不用说小的或精确光源.它还无法正确处理光泽BRDF或更复杂的照明设置.考虑一个由远距离天空光照亮的表面,整个天空的颜色从红色变为绿色.考虑到颜色,这可能表示地面被天空发出的光照亮,可能是在某个遥远的星球上.参见图11.18.即使周围的遮挡会使点和的照明变暗,但它们仍将被天空的红色和绿色部分照亮.使用弯曲的法线有助于减轻这种影响,但也不是完美的.我们之前介绍的简单模型不足以应对此类情况.一种解决方案是以某种更具表现力的方式描述可见性.

我们将着重于对整个球形或半球形可见性进行编码的方法,即描述哪些方向阻挡入射辐射的方法.尽管此信息可用于遮挡精确光源,但这不是其主要目的.针对那些特定类型的光源的方法(在第7章中进行了广泛讨论)能够获得更好的质量,因为它们只需要为光源的单个位置或方向编码可见性即可.此处描述的解决方案主要用于为大面积的灯光或环境照明提供遮挡,在这种情况下,生成的阴影很柔和,并且由于近似可见性而导致的伪影不明显.此外,这些方法还可用于在常规阴影技术不可行的情况下提供遮挡,例如用于凹凸贴图细节的自阴影化和用于阴影图没有足够分辨率的超大型场景的阴影.



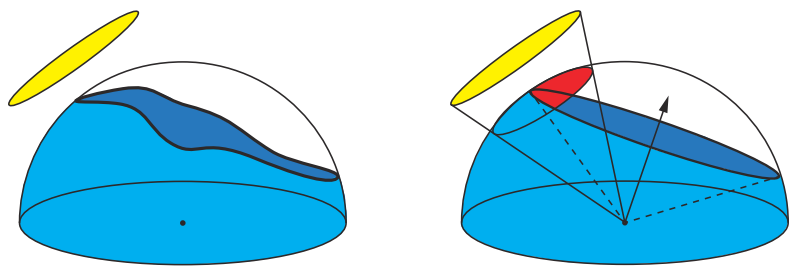
**图11.18**: 在复杂照明条件下,点a和b辐照度近似颜色.环境光遮挡不会对任何方向性进行建模,因此两个点的颜色相同.使用弯曲的法线可以将余弦波有效地移向天空的未被遮挡的部分,但是由于积分范围不受任何限制,因此不足以提供准确的结果.定向方法能够正确消除来自天空遮挡部分的照明.

**11.4.1预计算方向遮挡**

Max[**1145**]引入了水平映射的概念来描述高度场表面的自遮挡.在水平映射,对于表面上的每个点,针对一组方位角方向(例如八个)确定地平线的高度角:北,东北,东,东南,周围.

代替存储某些给定罗盘方向的水平角,整个未封闭的三维方向的集合可以建模为椭圆[**705**,**866**]或圆形[**1306**,**1307**]孔径.后一种技术称为环境光圈照明(图11.19).这些技术的存储要求比地平线图低,但是当一组未遮挡的方向与椭圆或圆形不相似时,可能会导致阴影不正确.例如,一个平坦的平面,高尖峰从该平面以规则的间隔突出,该平面应设置一个星形方向,该方向与该方案的映射不佳.

遮挡技术有很多变体.Wang等人[**1838**]使用球形符号距离函数(SSDF）表示可见性.它编码到球上被遮挡区域边界的有符号距离.第10.3节中讨论的任何球形或半球形基数也可以用于编码可见性[**582**,**632**,**805**,**1267**].就像环境光遮挡一样,方向可见性信息也可以存储在纹理,网格顶点或体积中[**1969**].



**图11.19**: 环境光照明用圆锥体近似着色点上方非遮挡区域的实际形状.左侧是黄色的区域光源,蓝色是表面位置的可见水平线.在右侧,水平线简化为一个圆,即圆锥的边缘,从表面位置向右上方突出,以虚线显示.然后通过将其光锥与光锥锥相交来估计该区域光的光锥,得出以红色显示的面积.

11.4.2 定向遮挡的动态计算 2020年4月24日09点25分

大多数生成环境光遮挡的方法也适用于生成方向可见性信息.Ren等人[**1482**]的球谐指数,以及Sloan等人[**1655**]的屏幕空间变体.以球形谐波矢量的形式生成可见性.如果使用多个SH波段,则这些方法本身会提供方向信息.使用更多的频段可以更精确地编码可见性.

圆锥跟踪方法,例如Crassin等人[**305**]和Wright[**1910**]的方法.为每条迹线提供一个遮挡值.出于质量原因,即使使用多个迹线也可以执行环境光遮挡估计,因此可用信息已经是定向的.如果需要特定方向的可见性，我们可以追踪更少的视锥细胞.

Iwanicki[**806**]也使用圆锥跟踪,但他将其限制在一个方向.结果被用来生成由动态角色投射到静态几何体上的柔和阴影,这些动态角色与一组球近似,类似于Ren等人[**1482**]和Sloan等人[**1655**].在此解决方案中,静态几何体的照明使用AHD编码存储(第10.3.3节).环境和定向组件的可见性可以独立处理.通过分析计算环境部分的遮挡.跟踪单个圆锥并与球相交,以计算方向分量的衰减因子.

许多屏幕空间方法也可以扩展为提供方向遮挡信息.克莱姆等人[904]使用z缓冲区数据来计算屏幕空间弯曲锥,实际上它是圆形孔径,非常类似于Oat和Sander[1307]离线预先计算的那些.在对像素的邻域进行采样时,它们会将未遮挡的方向相加.所得向量的长度可用于估计可见圆锥的顶角,其方向定义该圆锥的轴.Jimenez等人[835]基于视场角估计锥轴方向,并从环境遮挡因子得出角度.

**11.4.3 方向性遮挡着色**

编码定向遮挡的方法多种多样,我们无法为如何执行着色提供单一的处方.解决方案将取决于我们想要实现的特殊效果.

让我们再次考虑反射方程,在一个将入射辐射分为远处照明和可见度的版本中:

我们可以做的最简单的操作是使用可见性信号遮挡精确光艳.由于大多数编码可见性的方法都比较简单,因此结果的质量通常不尽人意,但这将使我们能够在一个基本示例上进行推理.此方法也可以用于传统的阴影方法由于分辨率不足而失败的情况,并且结果的精确性远不如实现任何形式的遮挡重要.这种情况的示例包括非常大的地形模型,或由凹凸贴图表示的较小表面细节.

按照第9.4节的讨论，在处理守时灯时,公式11.26变为

其中是从面对光的白色朗伯表面反射的辐射,而是朝向光的方向.我们可以将上述方程解释为计算材料对未遮挡光的响应,并将结果乘以可见性函数的值.如果光的方向落在地平线以下(使用水平贴图时),在可见性圆锥体外部(使用环境光圈照明时)或在SSDF的负区域中.则可见性函数等于零.因此光的任何贡献不应该考虑在内.值得一提的是,即使将可见性定义为二进制函数.(至少在大多数情况下成立.在某些情况下,我们希望可见性函数采用非零或一的值,但仍在该范围内.例如,当编码由半透明材料引起的遮挡时,我们可能要使用分数遮挡值.)许多表示形式也可以返回整个范围的值,而不仅仅是零或一个.这样的值表示部分遮挡.由于振铃,球谐或H基甚至可以重建负值.这些行为可能是不需要的,但仅仅是编码的固有属性.

我们可以对区域照明进行类似的推导.在这种情况下,在所有地方都等于零，但在光所包围的立体角内除外,等于该光源发出的辐射.我们将其称为,并假设它在光源的立体角上是恒定的.我们可以用光源的立体角积分域代替整个球面的积分域:

如果我们假设BRDF是恒定的,即我们正在处理Lambertian曲面,那么它也可以从积分下面拉出:

为了确定被遮挡的照明,我们需要计算可见度函数的积分,再乘以在光所包围的立体角上的余弦项.在某些情况下可以通过解析解来完成.Lambert[967]推导了一个公式来计算球形多边形上余弦的积分.如果区域光是多边形的,并且可以将其裁剪为可见性表示形式,则只需要使用Lambert公式即可得出精确的结果(图11.20).例如,当我们选择水平角度作为可见性表示形式时，这是可能的.但是,如果由于某种原因我们决定采用其他编码方式(例如弯曲圆锥体),则裁剪将产生圆形片段,因此我们将无法再使用Lambert公式.如果我们要使用非多边形区域光源,则同样适用.

另一种可能性是假设余弦项的值在整个积分域中是恒定的.如果区域光的大小很小,则此近似值非常精确.为简单起见,我们可以使用沿区域光的中心方向评估的余弦值.这给我们留下了可见度项在光的立体角上的积分.我们如何选择还取决于我们对可见性表示形式和区域光类型的选择.如果我们使用球形光和以弯曲圆锥体表示的可见性,则积分的值是可见性圆锥体与该光所对向的圆锥体的交点的立体角.可以通过分析来计算,如Oat和Sander[1307]所示.尽管确切的公式很复杂,但它们提供了一种在实际中效果很好的近似值.如果能见度是用球谐函数编码的,则积分也可以通过解析计算.

对于环境照明,我们不能限制积分范围,因为照明来自各个方向.我们需要找到一种方法,可以根据公式11.26计算全积分.让我们首先考虑Lambertian BRDF:

该方程的积分类型称为三元乘积积分.如果各个函数以某些方式表示(例如,以球谐函数或小波表示),则可以进行分析计算.不幸的是,这对于典型的实时应用来说太昂贵了,尽管已经证明这种解决方案可以在简单的设置中以交互帧速率运行[1270].

但是,由于余弦是函数之一,因此我们的情况稍微简单一些.我们可以将公式11.30写成

或

其中

和都是球形函数,就像和.首先,我们不尝试计算三元积积分,而是先将余弦乘以(方程11.31)或(方程11.32).这样做使积分只包含两个函数的乘积.尽管这看起来像是一个数学技巧,但它极大地简化了计算.如果使用正交标准(例如球谐函数)表示这些因子,则可以简单地计算双乘积积分.它是其系数向量的点积(第10.3.2节).

我们仍然需要计算和,但是因为它们涉及余弦,所以这比完全普通的情况要简单.如果我们用球谐函数表示函数,则余弦会投影到区域谐波(ZH),这是球谐函数的子集,其每个频带中只有一个系数为非零(10.3.2节).该投影的系数具有简单的解析公式[1656].SH和ZH的乘积比SH和另一个SH的乘积的计算效率要高得多.

如果我们决定先将余弦乘以(公式11.32),则可以离线进行处理,而只存储可见性.如Sloan等人[1651]所述,这是一种预先计算的辐射传递形式.(第11.5.3节).但是,以这种形式,我们不能对法线进行任何精细的修改,因为由法线控制的余弦项已经与可见性融合在一起.如果要对精细尺度的正常细节建模,可以先将余弦乘以(公式11.31).由于我们不预先知道法线方向,因此我们可以针对不同的法线预先计算此乘积[805],或者在运行时执行乘法[809].离线预先计算和余弦的乘积意味着依次限制照明的任何变化,并且允许照明在空间上变化将需要大量的内存.另一方面,在运行时计算乘法是昂贵的.Iwanicki和Sloan[809]描述了如何降低这一成本.可以以较低的粒度(在顶点上)计算乘积.将结果与余弦项进行卷积,投影到更简单的表示形式(AHD)上,然后使用每像素法线进行插值和重构.这种方法使他们可以在性能要求很高的60 FPS游戏中使用该方法。

克莱姆等人[904]提出了一种解决方案,用于以环境地图表示的照明和以视锥编码的可见性.他们使用不同大小的内核过滤环境图,这些内核代表不同圆锥孔的可见性和照明乘积的积分.它们将增加圆锥角的结果存储在纹理的mip级别中.这是可能的,因为大锥角的预滤波结果在整个球面上平滑变化,并且不需要以高角度分辨率存储.在预过滤期间,他们假设可见性圆锥体的方向与法线对齐,这是一个近似值,但实际上给出的结果合理.他们提供了有关此近似值如何影响最终质量的分析.如果我们要处理有光泽的BRDF和环境照明,

情况将更加复杂.我们不能再将BRDF从积分下方拉出,因为它不是恒定的.为了解决这个问题，格林等人[582]建议用一组球形高斯近似BRDF本身.这些是径向对称函数,可以仅用三个参数来紧凑表示:方向(或平均值),标准偏差和幅度.近似BRDF定义为球形高斯的总和:

其中是球形高斯波瓣,沿方向,具有锐度(见10.3.2节),并且在第个波瓣的振幅中.对于各向同性的BRDF,波瓣的形状仅取决于法线和视图方向之间的角度.近似值可以存储在一维查找表中并进行插值.

通过这种近似,我们可以将公式11.26写成

格林等人还假设可见性函数在每个球形高斯的整个支撑范围内是恒定的,这使他们可以将其从积分下方拉出.他们评估在波瓣中心方向上的可见性函数:

剩余的积分表示入射光与以给定方向和给定标准偏差定向的球形高斯卷积.这种卷积的结果可以预先计算并存储在环境图中,而较大的卷积则存储在较低的mip级别中.可见性是用低阶球谐函数编码的,但是也可以使用任何其他表示形式,因为它仅是点评估的.

Wang等人[1838]以类似的方式近似BRDF,但是以更精确的方式处理可见性.它们的表示使他们能够在可见性函数的支持下计算单个球形高斯积分.他们使用该值引入了一个新的球形高斯分布,其方向和标准偏差相同,但幅度不同.他们在照明计算中使用了这一新功能.

对于某些应用,此方法可能太昂贵.它需要从预过滤的环境贴图中获取多个样本,并且纹理采样在渲染过程中通常已经成为瓶颈.Jimenez等人[835]和El Garawany[414]提出了更简单的近似值.为了计算遮挡因子,它们用单个圆锥体表示整个BRDF瓣,而忽略了其对视角的依赖性,只考虑了诸如材料粗糙度之类的参数(图11.21).它们将可视性近似为圆锥形,并计算可视性与BRDF圆锥体相交的立体角,这与环境光圈照明非常相似.标量结果用于衰减照明.即使大大简化,结果还是可以相信的.

**11.5 漫反射全局照明** 2020年4月24日13点54分

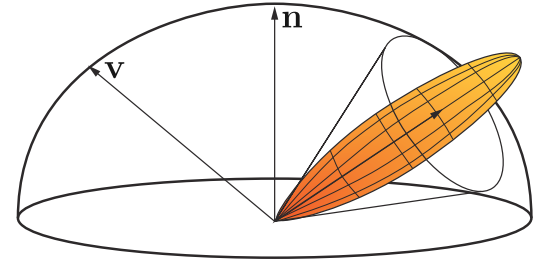
 下一部分将介绍各种方法,不仅可以实时模拟遮挡,还可以模拟全光线反弹.它们可以粗略地划分为算法,这些算法假定在到达眼睛之前,光会从漫反射或镜面反射回来.相应的光路可以写成L(D|S)\*DE或L(D|S)SE,其中许多方法都对较早的反弹类型施加了一些约束.第一组中的解决方案假定传入照明在着色点上方的半球上平滑变化，或者完全忽略该变化.第二组算法假定入射方向的变化率很高.他们依靠这样的事实,即照明将仅在相对较小的立体角内进入.由于这些巨大的限制,因此分别处理这两个组是有益的.在本节中,我们将介绍漫反射全局照明的方法,在下一部分中将讨论镜面反射,然后在最后一节中介绍有希望的统一方法.

图11.21: 为了计算遮挡,可以将光泽材料的镜面波瓣表示为圆锥体.如果将可见性近似为另一个圆锥体,则可以使用与环境光圈照明相同的方式将遮挡因子计算为两者的交点的立体角(图11.19).该图显示了用圆锥体表示BRDF瓣的一般原理,但仅作为示例.在实践中,为了产生合理的遮挡效果,圆锥体需要更宽.

11.5.1 表面预光照

光能传递和路径跟踪均设计用于离线渲染.尽管已努力在实时设置中使用它们,但结果仍然太不成熟,无法在生产中使用.当前,最常见的做法是使用它们来预先计算与照明有关的信息.昂贵的离线过程会提前运行,其结果将存储起来,然后在显示期间用于提供高质量的照明.如11.3.4节所述,以这种方式对静态场景进行预计算称为烘焙.

这种做法带有一定的限制.如果我们提前执行光照计算,则无法在运行时更改场景设置.所有场景几何,灯光和材质都需要保持不变.我们不能改变一天中的时间或在墙上打洞.在许多情况下,此限制是可以接受的折衷方案.建筑可视化可以假定用户仅在虚拟环境中走动.游戏也限制了玩家的动作.在此类应用中,我们可以将几何分为静态对象和动态对象.静态对象用于预计算过程,它们与照明完全交互.静止的墙壁投下阴影,静止的红色地毯反射红光.动态对象仅充当接收者.它们不会阻挡光线,也不会产生间接照明效果.在这种情况下,通常将动态几何图形限制为相对较小,因此可以忽略其对其余照明的影响,也可以使用其他技术对其进行建模,而质量损失最小.动态几何可以例如使用屏幕空间方法来生成遮挡.典型的动态对象集包括角色,装饰几何和车辆.

可以预先计算的最简单形式的照明信息是辐照度.对于平坦的朗伯表面以及表面颜色,它可以充分描述材料对照明的反应.由于照明源的效果与其他光源无关,因此可以在预先计算的辐照度之上添加动态光(图11.22).

1996年的Quake和1997年的Quake II是第一个利用预计算辐照度值的商业交互式应用程序.Quake正在预先计算静态光的直接贡献,主要是为了提高性能.Quake II还包含一个间接组件,使它成为第一款使用全局照明算法生成更逼真的照明的游戏.它使用了基于光能传递的算法,因为该技术非常适合在朗伯环境中计算辐照度.同样,时间的存储约束将照明限制为相对较低的分辨率,这与光能传递解决方案中常见的模糊低频阴影很好地匹配.

通常将预先计算的辐照度值与存储在一组单独纹理中的漫反射色或反照率贴图相乘.尽管理论上可以预先计算出出射量(辐照度乘以散布的颜色)并存储在一组纹理中,但是在大多数情况下,许多实际考虑都排除了该选项.彩色贴图的频率通常很高,它们使用各种平铺方式,并且它们的各个部分经常在整个模型中重复使用,以保持合理的内存使用率.辐照度值通常是低得多的频率,不能轻易重复使用.保持照明和表面颜色的分离会消耗更少的内存.如今,除了最严格的硬件平台外,很少使用预先计算的辐照度.由于按照定义,辐照度是针对给定法线方向计算的,因此我们无法使用法线贴图来提供高频细节.这也意味着只能针对平坦表面预先计算辐照度.如果需要在动态几何体上使用烘焙的照明,则需要其他方法来存储它.这些局限性促使人们寻找一种方法来存储带有方向分量的预先计算的照明.

11.5.2 定向表面预光照

要将预照明与法线贴图一起使用在Lambertian曲面上,我们需要一种表示辐照度如何随曲面法线变化的方法.为了给动态几何体提供间接照明,我们还需要为每个可能的表面方向提供其值.幸运的是,我们已经有了代表这些功能的工具.在10.3节中,我们介绍了根据法线方向确定照明的各种方法.这些包括针对函数域为半球形且球体下半部分的值无关紧要的特殊解决方案,不透明表面的情况也是如此.

最通用的方法是存储完整的球形辐照度信息,例如通过使用球形谐波.该方案由Good和Taylor[564]首先在加速光子映射的背景下提出,并由Shopf等人[1637]在实时设置中使用.在这两种情况下,定向辐照度都存储在纹理中.如果使用九个球谐系数(三阶SH),则质量很好,但是存储和带宽成本很高.仅使用四个系数(二阶SH)的成本较低,但会丢失许多细节,照明的对比度较小,法线贴图的发音不太明显.

Chen[257]使用Halo 3方法的一种变体,旨在降低成本获得三阶SH的质量.他从球形信号中提取出最主要的光,并将其分别存储为颜色和方向.残基使用二阶SH编码.这将系数的数量从27个减少到18个,而质量损失很小.Hu[780]描述了如何进一步压缩这些数据.Chen和Tatarchuk [258]提供了有关他们在生产中使用的基于GPU的烘烤管道的更多信息.

Habel等人[627]的H基础,是另一种替代解决方案.由于它仅编码半球形信号,因此,较少的系数可以提供与球形谐波相同的精度.只需六个系数即可获得与三阶SH相当的质量.由于仅针对半球定义了基础,因此我们需要在表面上具有一些局部坐标系以使其正确定向.通常,将由uv参数化产生的切线框架用于此目的.如果将H基分量存储在纹理中,则其分辨率应足够高以适应基础切线空间的变化.如果切线空间明显不同的多个三角形覆盖相同的纹理像素,则重构的信号将不精确.

球形谐波和H基的一个问题是它们会出现振铃(第10.6.1节).虽然预过滤可以减轻这种影响,但也能使照明变得平滑,这可能并不总是希望的.另外,就存储和计算而言,即使是较便宜的变体也具有相对较高的成本.在限制性更强的情况下,例如在低端平台上或为虚拟现实进行渲染时,此费用可能是禁止的.

成本是为什么简单替代方法仍然很受欢迎的原因.《半条命2》使用自定义的半球形基础(第10.3.3节),该基存储了三个颜色值,每个样本总共有九个系数.尽管很简单,但环境/高光/方向(AHD)基(第10.3.3节)也是一个受欢迎的选择.它已用于诸如《使命召唤》[809,998]系列和《最后生还者》[806]之类的游戏中.参见图11.23.

Crytek在《孤岛惊魂》[1227]游戏中使用了一个变体.Crytek表示由切线空间中的平均光线方向,平均光线颜色和标量方向性因子组成.最后一个值用于混合使用相同颜色的环境和方向分量.这样可以将每个样本的存储量减少到六个系数:三个用于颜色的值,两个用于方向的值和一个用于方向性因子的值.Unity引擎在其模式之一中也使用类似的方法[315].

这种类型的表示是非线性的,这意味着从技术上讲,在纹理像素或顶点之间线性内插单个组件在数学上是不正确的.如果主导光的方向快速变化,例如在阴影边界上,则可能在阴影中出现视觉伪像.尽管存在这些错误,但结果看起来还是令人愉悦的.由于周围环境和定向照明区域之间的对比度很高,因此通常会希望突出法线贴图的效果.另外,在计算BRDF的镜面响应时,可以使用方向分量,以提供一种低成本替代环境的低光泽度材质贴图.

在频谱的另一端是为高视觉质量而设计的方法.Neubelt和Pettineo[1268]在游戏《教团:1886》中使用纹理图存储了球形高斯系数.(图11.24).它们存储入射辐照度,而不是辐照度,它被投影到切线框架中定义的一组高斯波瓣(第10.3.2节).根据特定场景中照明的复杂性,它们使用五到九个瓣.为了产生漫反射响应,将球形高斯与沿曲面法线定向的余弦波瓣卷积.通过使高斯与BRDF高光波瓣卷积,该表示也足够精确,可以提供低光泽的高光效果.Pettineo详细描述了整个系统[1408].他还向能够烘焙和渲染不同照明表示形式的应用程序提供源代码.

如果我们需要有关任意方向的照明信息,而不仅仅是在表面上方的半球内(例如,为动态几何图形提供间接照明),则可以使用对完整球形信号进行编码的方法.球谐函数在这里很自然.当内存问题较少时,三阶SH(每个颜色通道有9个系数)是流行的选择.否则,将使用二阶(每个颜色通道四个系数,这些系数与RGBA纹理中的组件数量匹配,因此单个贴图可以存储一个颜色通道的系数).球形高斯也可以在完全球形的环境中工作,因为波瓣可以分布在整个球体或法线周围的半球中.但是,由于球形技术需要叶片覆盖的立体角是原来的两倍,因此我们可能需要使用更多的叶片来保持相同的质量.

如果我们想避免振铃,但又不能使用大量的波瓣,则环境立方[1193](第10.3.1节)是可行的选择.它由六个夹紧的cos2瓣组成,沿主轴方向.余弦波瓣仅覆盖一个半球,因为它们具有局部支持,这意味着它们仅在其球域的子集上具有非零值.因此,在重建过程中仅需要六个存储值中的三个可见瓣.这限制了照明计算的带宽成本.重建的质量类似于二阶球谐函数.

环境骰子[808](也见第10.3.1节)可用于获得比环境立方体更高的质量.此方案使用沿二十面体的顶点定向的十二个裂片,它们是cos2和cos4裂片的线性组合.重建过程中使用了十二个存储值中的六个.其质量可与三阶球形谐波媲美.这些和其他类似表示形式(例如,由三个cos2瓣和一个余弦瓣组成的基,扭曲以覆盖整个球体)已在许多商业上成功的游戏中使用,例如Half-Life 2 [1193],《使命召唤》系列[766,808],《孤岛惊魂3》[533],《全境封锁》[1694]和《刺客信条4:黑旗》[1911]等.

11.5.3 预计算辐射传输

尽管预计算的照明看起来很漂亮,但它本身也是静态的.几何形状或照明的任何更改都可能使整个解决方案无效.就像在现实世界中一样,打开窗帘(场景中几何体的局部变化)可能会使整个房间充满光线(照明的全局变化).在寻找允许某些类型的更改的解决方案上已经花费了很多研究工作.

如果我们假设场景的几何形状不会改变,仅改变照明,则可以预先计算光线与模型的交互方式.可以在某种程度上预先分析对象间的影响,例如相互反射或地下散射,并存储结果,而无需对实际辐射值进行操作.将传入的光照转换为整个场景中的辐射分布的描述的函数称为传输函数.对此进行预先计算的解决方案称为预计算传输或预计算辐射传输(PRT)方法.

与将光源完全脱机烘烤相比,这些技术确实具有明显的运行时成本.在屏幕上显示场景时,我们需要计算特定照明设置的辐射值.为此,将实际的直接光量“注入”到系统中,然后应用传递函数在整个场景中传播.一些方法假定这种直接照明来自环境地图.其他方案允许照明设置是任意的并以灵活的方式进行更改.

Sloan等人[1651]在图形中引入了预先计算的辐射传递的概念.他们用球谐函数来描述它,但是该方法不必使用SH.基本思想很简单.如果我们使用一些(最好是较低数量)的“构件”灯来描述直接照明,则可以预先计算其中的每一个如何照亮场景.想象一下一个房间,里面有三台计算机显示器,并假设每个显示器只能显示一种颜色,但是强度会有所不同.考虑每个屏幕的最大亮度等于一个标准化的“单位”亮度.我们可以独立地预先计算每个监视器对房间的影.可以使用第11.2节介绍的方法完成此过程.因为光传输是线性的,所以用所有三个监视器照亮场景的结果将等于直接或间接来自每个监视器的光的总和.每个监视器的照明不会影响其他解决方案,因此,如果将其中一个屏幕的亮度设置为一半,这样做只会改变其自身对总照明的贡献.

这使我们能够快速计算整个房间内的全反射光.我们采用每个预先计算的光源解决方案,将其乘以屏幕的实际亮度,然后对结果求和.我们可以打开和关闭显示器,使它们变亮或变暗,甚至改变它们的颜色,而要获得最终照明,需要做的就是这些乘数和加法运算(图11.25).

我们可写为

其中是在点的最终辐射,是来自屏幕的预计算的单位贡献,而是其当前亮度.该方程在数学上定义了一个向量空间,其中是该空间的基础向量.可以通过灯光贡献的线性组合来创建任何可能的灯光.

Sloan等人[1651]的原始PRT论文使用相同的推理,但是使用球谐函数表示无限远的照明环境.他们没有存储场景对监视器屏幕的响应,而是存储了由球谐函数基函数定义的分布对场景光的响应.通过对一定数量的SH波段执行此操作,它们可以渲染由任意照明环境照明的场景.他们将这种照明投射到球谐上,将每个结果系数乘以其各自的归一化“单位”贡献,然后将它们加在一起,就像我们对显示器所做的那样.

请注意,用于将光“注入”场景的基础的选择与用于表达最终照明的表示方式无关.例如,我们可以描述如何使用球谐函数照亮场景,但可以选择另一个基础来存储在任何给定点到达多少辐射.假设我们使用环境立方体进行存储.我们将计算从顶部到侧面的辐射量.这些方向的每个方向的转移将被单独存储,而不是作为代表总转移的单个标量值存储.

Sloan等人[1651]的PRT论文分析了两种情况.第一种是当接收方为表面的标量辐照度值时.为此,接收器必须是具有预定法线的完全漫反射表面,这意味着它不能将法线贴图用于精细比例的细节.传递函数采用输入照明的SH投影与预先计算的传递向量之间的点积形式,该点积在整个场景中会发生空间变化.

如果我们需要渲染非朗伯材料,或者允许进行法线贴图,则可以使用显示的第二种变体.在这种情况下,周围照明的SH投影将转换为给定点的入射辐射的SH投影.因为此操作为我们提供了整个球体(或半球,如果我们要处理静态不透明物体)的完整辐射度分布,则我们可以将其与任何BRDF适当地卷积.传递函数将SH向量映射到其他SH向量,并具有矩阵乘法的形式.无论在计算上还是在存储器方面,这种乘法运算都是昂贵的.如果我们对源和接收器都使用三阶SH,则需要为场景中的每个点存储9×9矩阵,并且这些数据仅用于单色传输.如果需要颜色,则需要三个这样的矩阵-每个点的存储量惊人.

一年后,Sloan等人[1652]解决了这个问题.与其直接存储转移向量或矩阵,不如使用主成分分析(PCA)技术分析它们的整个集合.可以将传递系数视为多维空间中的点(例如,在9×9矩阵的情况下为81维),但它们的集合在该空间中分布不均匀.它们形成维数较低的簇.这种聚类就像沿着一条线分布的三维点有效地全部存在于三维空间的一维子空间中一样.PCA可以有效地检测这种统计关系.一旦发现了子空间,就可以使用更少数量的坐标来表示点,因为我们可以将维度中的位置存储在子空间中.使用线类比,我们可以只存储其沿线的距离.而不必使用三个坐标存储点的完整位置.Sloan等人使用此方法可以将传递矩阵的维数从625维(25×25传递矩阵)减少到256维.尽管对于典型的实时应用来说,这仍然太高了,但是许多后来的光传输算法都采用PCA作为压缩数据的一种方式.

这种降维本质上是有损的.在极少数情况下,数据形成一个完美的子空间,但大多数情况下是近似的,因此将数据投影到其上会导致性能下降.为了提高质量,Sloan等人将转移矩阵集划分为多个簇,并分别对每个簇执行PCA.该过程还包括一个优化步骤,可确保在群集边界上不存在不连续性.还提出了允许对象有限变形的扩展,称为局部可变形预计算辐射传递(LDPRT)[1653].

PRT已在多种游戏中以各种形式使用.它在游戏玩法集中于一天中的时间和天气条件会动态变化的户外区域的游戏中特别受欢迎.《孤岛惊魂3》和《孤岛惊魂4》使用PRT,其中源基础为二阶SH,接收方基础为定制的四向基础[533,1154].《刺客信条4:黑旗》使用一种基本函数作为来源(太阳色),但会预先计算一天中不同时间的传输量.可以将这种表示解释为具有在时间维度而非方向上定义的源基础函数.接收者的依据与《孤岛惊魂》的标题相同.

SIGGRAPH 2005课程[870]中有关预计算辐射转移的课程很好地概述了该领域的研究.Lehtinen[1019,1020]提供了一个数学框架,可用于分析各种算法之间的差异并开发新算法. 原始的PRT方法假定周围光线无限远.尽管此模型可以很好地模拟室外场景的照明,但对于室内环境而言却过于严格.但是,正如我们前面提到的,该概念与照明的原始来源完全无关.Kristensen等人[941]描述了一种方法,其中PRT是针对散布在整个场景中的一组灯光计算的.这对应于具有大量的“源”基础函数.接下来,将灯光组合成簇,然后将接收几何体划分为多个区域,每个区域受不同的灯光子集影响.该过程导致传输数据的显着压缩.在运行时,通过对来自预先计算的集合中最接近的灯光的数据进行插值,可以近似估算由任意放置的灯光产生的照明 Gilabert和Stefanov [533]在游戏《孤岛惊魂3》中使用了这种方法来产生间接照明.这种方法的基本形式只能处理点光源.虽然可以扩展以支持其他类型,但是成本随每个灯的自由度数量呈指数增长.

至此讨论的PRT技术会预先计算一些元素的传输,然后将其用于建模灯光.另一类流行的方法是预先计算曲面之间的传递.在这种类型的系统中,实际的照明源变得无关紧要,可以使用任何光源,因为这些方法的输入是来自一组表面的出射辐射(如果该方法采用仅漫反射表面,则为其他相关量,例如辐照度).这些直接照明计算可以使用阴影(第7章),辐照环境贴图(第10.6节)或本章前面讨论的环境和方向遮挡方法.通过将任何表面的出射辐射设置为所需值,并将其变成面光源,也可以使所有表面都变得微不足道.

按照这些原则运行的最流行的系统是Geomerics的Enlighten(图11.26).尽管该算法的确切细节尚未完全公开披露,但大量的演讲和演示都准确地描述了该系统的原理[315,1101,1131,1435].

场景假定为朗伯风格,但仅用于光传输的目的.使用Heckbert的表示法,处理的路径集为LD\*(D|S)E,因为眼睛之前的最后一个曲面不需要仅是漫反射的.该系统定义了一组“源”元素和另一组“接收器”元素.源元素存在于曲面上,并共享它们的某些属性,例如漫反射色和法线.预处理步骤计算光如何在光源元件和接收器之间传输.此信息的确切形式取决于源元素是什么以及在接收器处收集照明的基础.以最简单的形式,源元素可以是点,然后我们有兴趣在接收位置生成辐照度.在这种情况下,传输系数只是源和接收方之间的相互可见性.在运行时,所有源元素的输出辐射都提供给系统.根据这些信息,我们可以使用预先计算的可见性和有关源和接收器的位置和方向的已知信息,对反射方程(方程11.1)进行数值积分.以这种方式,执行一次光反射.由于大多数间接照明来自第一次反射,因此仅执行一次反射就足以提供合理的照明.但是,我们可以使用此光并再次运行传播步骤,以生成第二个光反弹.这通常是在几帧的过程中完成的,其中一帧的输出用作下一帧的输入.

将点用作源元素将导致大量连接.为了提高性能,代表相似法线和颜色区域的点的群集也可以用作源集.在这种情况下,传输系数与辐射度算法(见11.2.1节)中看到的形状因数相同.请注意,尽管有相似之处,但该算法与经典的辐射度不同,因为它一次只计算一个反弹光,并且不涉及求解线性方程组.它借鉴了渐进辐射的思想[275,1642].在该系统中，单个贴片可以通过迭代过程确定从其他贴片接收多少能量.将辐射转移到接收位置的过程称为收集.

接收元件处的辐射可以以不同的形式收集.到接收元件的转移可以使用我们之前描述的任何定向基础.在这种情况下,单个系数成为值的向量,维数等于接收基础中的函数.使用定向表示执行收集时,结果与11.5.2节中描述的脱机解决方案相同,因此可以与法线贴图一起使用或提供低光泽的镜面反射.在许多变体中都使用相同的一般思想.为了节省内存,Sugden和Iwanicki [1721]使用SH传输系数,对其进行量化,然后将其间接存储为调色板中条目的索引.Jendersie等人[820]建立源补丁的层次结构,并在子级所占据的立体角太小时将对更高元素的引用存储在此树中.Stefanov[1694]引入了一个中间步骤,首先将表面元素的辐射传播到场景的体素化表示中,然后将其用作传输的源.

从某种意义上说，理想的表面分割成源斑块取决于接收器的位置.对于遥远的元素,将它们视为单独的实体会产生不必要的存储成本,但近距离查看时应将其单独对待.源补丁程序的层次结构在某种程度上缓解了此问题，但并未完全解决。可以为特定接收器组合的某些补丁可能相距足够远，以防止此类合并。 Silvennoinen和Lehtinen [1644]提出了解决该问题的新方法。他们的方法不是显式创建源补丁，而是为每个接收位置生成不同的源补丁集。将对象渲染到散布在场景周围的稀疏环境地图集。每个图都投影到球谐函数，而这个低频版本“虚拟地”投影回环境。接收点记录他们可以看到多少投影，并且此过程分别针对每个发送者的SH基础功能完成。这样做会根据来自环境探测器和接收器点的可见性信息为每个接收器创建不同的源元素集。由于源基础是从投影到SH的环境贴图生成的，因此自然会组合更远的曲面。为了选择要使用的探针，接收器使用启发式的方法来支持附近的探针，这使接收器以相似的比例“看到”环境。为了限制必须存储的数据量，使用群集PCA压缩传输信息。 Lehtinen等人描述了另一种形式的预计算转移。 [1021]。在这种方法中，源元素和接收元素都不存在于网格上，而是体积的，可以在三维空间中的任何位置进行查询。这种形式便于在静态和动态几何之间提供照明一致性，但是该方法的计算成本相当高。 Loos等。 [1073]在具有不同配置的侧壁的模块化单位单元内进行预计算转移。然后，将多个此类单元进行缝合和扭曲以近似场景的几何形状。辐射首先通过预先计算的模型传播到充当界面的单元边界，然后传播到相邻单元。该方法足够快，即使在移动平台上也可以有效运行，但是对于要求更高的应用程序而言，其质量可能并不足够。

11.5.4 存储法

无论我们是要使用完整的预先计算的照明还是要预先计算传输信息并允许对照明进行一些更改，都必须以某种形式存储结果数据。必须使用GPU友好格式。光照贴图是存储预先计算的光照的最常用方法之一。这些是存储预先计算的信息的纹理。尽管有时使用诸如辐照度图之类的术语来表示存储的特定类型的数据，但是使用术语光图来共同描述所有这些数据。在运行时，将使用GPU的内置纹理机制。值通常是双线性过滤的，对于某些表示形式，可能并不完全正确。例如，当使用AHD表示时，滤波后的D（方向）分量在插值后将不再是单位长度，因此需要重新归一化。使用插值法还意味着，如果我们直接在采样点进行计算，则A（环境）和H（高亮）将不完全相同。也就是说，即使表示形式是非线性的，结果通常看起来也可以接受。

在大多数情况下，光照贴图不使用mipmapping，这通常是不需要的，因为与典型的反照率贴图或法线贴图相比，光照贴图的分辨率较小。即使在高质量的应用中，单个光照贴图纹理像素也可以覆盖至少约20×20厘米（通常更大）的区域。使用这种大小的纹理像素，几乎不需要额外的mip级别。为了将光照存储在纹理中，对象需要提供唯一的参数化。在将漫反射色纹理映射到模型上时，通常可以使网格的不同部分使用纹理的相同区域，尤其是在模型使用常规重复图案进行纹理化的情况下。重用光照贴图最多是困难的。照明对于网格上的每个点都是唯一的，因此每个三角形都需要在光照贴图上占据其自己的唯一区域。创建参数化的过程始于将网格划分为较小的块。这可以使用某些启发式方法自动完成[1036]，也可以在创作工具中手动完成。通常，使用已经存在的其他纹理映射拆分。接下来，分别对每个块进行参数化，以确保其各个部分在纹理空间中不重叠[1057、1617]。纹理空间中的最终元素称为图表或外壳。最后，所有图表都打包到一个通用纹理中（图11.27）。必须注意确保不仅图表不重叠，而且它们的过滤范围也必须分开。呈现给定图表时（双线性过滤访问四个相邻纹理像素）可以访问的所有纹理像素都应标记为已使用，因此没有其他图表与它们重叠。否则，图表之间可能会发生渗色，而其中一个图表上的灯光可能会在另一图表上可见。尽管对于光映射系统来说，在光映射表之间提供用户控制的“装订线”量以用于间隔非常普遍，但这种分隔不是必需的。可以使用一组特殊的规则，通过在光照图空间中对其进行栅格化来自动确定图表的正确过滤范围。参见图11.28。如果以此方式栅格化的炮弹不重叠，我们可以保证不会发生渗漏。避免流血是mipmapping很少用于光照贴图的另一个原因。图表过滤足迹需要在所有mip级别上保持分开，这将导致壳之间的间距过大。将图表最佳地打包到纹理中是一个NP完全问题，这意味着没有已知的算法可以生成具有多项式复杂度的理想解决方案。由于实时应用程序可能在单个纹理中包含成千上万个图表，因此所有现实世界的解决方案都使用微调的启发式方法和经过精心优化的代码来快速生成打包[183，233，1036]。如果稍后对光照贴图进行块压缩（第6.2.6节），则为提高压缩质量，可能会向打包程序添加其他约束，以确保单个块仅包含相似值。光照贴图的一个常见问题是接缝（图11.29）。由于将网格划分为图表，并且每个网格都单独进行参数设置，因此无法确保沿拆分边缘的照明在两侧完全相同。这表现为视觉上的不连续。如果手动拆分网格，则可以通过将网格拆分到不直接可见的区域来避免此问题。但是，这样做很费力，并且在自动生成参数化时无法应用。 Iwanicki [806]在最终的光照贴图上执行后处理，该过程会修改沿分裂边缘的纹素，以最小化两侧插值之间的差异。刘和弗格森等。 [1058]强制通过等式约束沿边缘匹配的插值，并求解最能保持平滑度的纹素值。另一种方法是在创建参数化和打包图表时考虑到此约束。雷等。 [1467]显示了如何将保存网格的参数化用于创建不受接缝伪影影响的光照贴图。

预先计算的照明也可以存储在网格的顶点处。缺点是照明的质量取决于网格的细分程度。由于此决定通常是在创作的早期阶段做出的，因此很难确保网格上有足够的顶点以在所有预期的照明情况下看起来都不错。另外，镶嵌细分可能很昂贵。如果网格精细细分，则照明信号将被过采样。如果使用存储光照的定向方法，则需要通过GPU在顶点之间插入整个表示，并将其传递到像素着色器阶段以执行光照计算。在顶点着色器和像素着色器之间传递如此多的参数非常少见，并且会产生无法优化现代GPU的工作负载，从而导致效率低下和性能降低。由于所有这些原因，很少使用在顶点上存储预先计算的光照。即使在曲面上需要有关入射辐射的信息（第14章中讨论的进行体积渲染时除外），我们也可以按体积进行预先计算和存储。这样做，可以在空间中的任意点查询照明，从而为预计算阶段场景中不存在的对象提供照明。但是请注意，这些对象将无法正确反射或遮挡照明。 Greger等。 [594]提出了辐照量，它代表了具有辐照环境图的稀疏空间采样的五维（三维和二维）辐照函数。也就是说，空间中存在一个三维网格，并且在每个网格点处都有一个辐照环境贴图。动态对象根据最接近的贴图对辐照度值进行插值。 Greger等。对于空间采样，使用两级自适应网格，但是也可以使用其他体积数据结构，例如八叉树[1304、1305]。

在原始辐照量中，Greger等人在每个采样点处以较小的纹理存储了辐照度，但是无法在GPU上有效过滤此表示。 如今，体积照明数据通常存储在三维纹理中，因此可以使用GPU的加速过滤对体积进行采样。 采样点处辐照度函数的最常见表示形式包括:

二阶和三阶的球谐（SH），更常见，因为单个颜色通道所需的四个系数可以方便地打包为典型纹理格式的四个通道.

球形高斯.

环境立方体或环境骰子.

即使在技术上能够表示球形辐照度，AHD编码也会产生分散注意力的伪影。如果使用SH，则球谐梯度[54]可以进一步提高质量。上述所有表示已成功用于许多游戏中[766，808，1193，1268，1643]。 Evans [444]描述了LittleBigPlanet中用于辐照量的技巧。代替完整的辐照度图表示，平均辐照度存储在每个点上。根据辐照场的梯度，即场变化最快的方向，可以计算出近似的方向性因子。代替显式地计算梯度，梯度和表面法线n之间的点积是通过获取两个辐照场样本来计算的，一个在表面点p，另一个在向n方向稍微偏移的点，然后减去一个从另一个。这种近似表示的动机是，LittleBigPlanet中的辐照量是动态计算的。辐照量也可以用于为静态表面提供照明。这样做的优点是不必为光照图提供单独的参数设置。该技术也不会产生接缝。静态和动态对象都可以使用相同的表示形式，从而使两种类型的几何图形之间的照明保持一致。体积表示法很方便用于延迟着色（第20.1节），在该方法中，所有照明都可以单次执行。主要缺点是内存消耗。光照贴图使用的内存量与分辨率的平方成正比。对于常规的体积结构，它随立方体一起增长。因此，将较低的分辨率用于网格体积表示。适应性的分层形式的照明体积具有更好的特性，但与光图相比，它们仍存储更多的数据。它们也比具有规则间距的网格要慢，因为额外的间接调用会在着色器代码中创建负载依赖关系，这可能导致停顿并降低执行速度。在体积结构中存储表面照明有些棘手。有时具有不同照明特性的多个表面可以占据同一体素，因此不清楚应存储哪些数据。从此类体素采样时，照明经常不正确。这种情况尤其经常发生在明亮的室外和黑暗的室内之间的墙壁附近，并导致外面的黑暗斑块或里面的明亮斑块。对此的解决方法是使体素大小足够小，以至于永远不会跨越这些边界，但是由于所需的数据量，这通常是不切实际的。解决该问题的最常见方法是沿法线移动采样位置一定量，或调整插值过程中使用的三线性混合权重。这通常是不完善的，可能需要手动调整几何形状以掩盖问题。 Hooker [766]在辐照量上增加了额外的剪切平面，这将它们的影响限制在凸多面体的内部。 Kontkanen和Laine [926]讨论了各种减少出血的策略。

保持照明的体积结构不必是规则的。一种流行的选择是将其存储在不规则的点云中，然后将其连接以形成Delaunay四面体化（图11.30）。 Cupisz [316]推广了这种方法。要查找照明，我们首先找到采样位置所在的四面体。这是一个迭代过程，可能会有些昂贵。我们遍历网格，在相邻单元之间移动。查找点相对于当前四面体角的重心坐标用于选择下一步要访问的邻居（图11.31）。因为典型的场景可能包含数千个存储照明的位置，所以此过程可能很耗时。为了加快速度，我们可以在上一帧中记录用于查找的四面体（如果可能），或者使用简单的体数据结构为场景中的任意点提供良好的“起始四面体”。找到合适的四面体后，使用已经可用的重心坐标对存储在其四角的照明进行插值。 GPU不会加速此操作，但是它只需要四个值进行插值，而不是网格上三线性插值所需的八个值。

可以手动[134，316]或自动[809，1812]放置照明预先计算和存储的位置。当它们探测（或采样）照明信号时，它们通常被称为照明探头或照明探头。该术语不应与“光探头”（第10.4.2节）混淆，“光探头”是环境地图中记录的远处照明。从四面体网格采样的照明质量高度依赖于该网格的结构，而不仅取决于探针的整体密度。如果它们分布不均匀，则生成的网格可以包含生成视觉伪像的细长的四面体。如果用手放置探针，可以轻松纠正问题，但这仍然是手动过程。四面体的结构与场景几何的结构无关，因此，如果处理不当，灯光将被插在墙壁上并产生出血伪像，就像辐照量一样。在手动放置探头的情况下，可能要求用户插入其他探头以阻止这种情况的发生。当使用自动放置探针时，可以将某种形式的可见性信息添加到探针或四面体中，以将其影响限制在仅相关区域[809、1184、1812]。通常，对静态和动态几何体使用不同的光照存储方法。例如，静态网格物体可以使用光照贴图，而动态对象可以从体积结构中获取光照信息。尽管很流行，但该方案会在不同类型的几何外观之间造成不一致。这些差异中的一些可以通过正则化消除，后者是对表示中的照明信息进行平均。烘焙照明时，仅在真正有效时才需要计算其值。网格常常是不完美的。某些顶点可以放置在几何图形内部，或者网格的某些部分可以自相交。如果我们在这些有缺陷的位置计算入射辐射，结果将是不正确的。它们会导致不正确的阴影阴影导致不必要的变暗或渗色。 Kontkanen和Laine [926]以及Iwanicki和Sloan [809]讨论了可用于丢弃无效样本的不同启发式方法。环境和方向遮挡信号共享漫射照明的许多空间特征。如第11.3.4节所述，上述所有方法也可以用于存储它们。

11.5.5 动态漫反射全局照明

即使预先计算的照明可以产生令人印象深刻的效果，但其主要优点还是主要缺点-需要预先计算。这样的离线过程可能很漫长。在典型的游戏关卡中，灯光烘烤要花费多个小时并不罕见。由于照明计算需要花费很长时间，因此通常会迫使艺术家同时在多个级别上工作，以免在等待烘烤完成时停机。反过来，这通常会导致用于渲染的资源过多，并使烘焙花费更长的时间。该循环会严重影响生产率并导致挫败感。在某些情况下，甚至无法预先计算照明，因为几何图形会在运行时更改或由用户在某种程度上创建。已经开发了几种方法来模拟动态环境中的全局照明。它们不需要任何预处理，或者准备阶段足够快以至于每帧都可以执行。在全动态环境下模拟全局照明的最早方法之一是基于“即时辐射度” [879]。尽管名称如此，但该方法与光能传递算法几乎没有共同点。在其中，光线从光源向外投射。对于光线照射的每个位置，都会放置一个光线，表示该表面元素的间接照明。这些光源称为虚拟点光源（VPL）。利用这个想法，Tabellion和Lamorlette [1734]开发了一种在《怪物史莱克2》的制作过程中使用的方法，该方法执行场景表面的直接光照遍历并将结果存储在纹理中。然后，在渲染过程中，该方法跟踪光线并使用缓存的光照来创建一反射间接照明。 Tabellion和Lamorlette表明，在许多情况下，一次弹跳足以产生可信的结果。这是一种离线方法，但它启发了Dachsbacher和Stamminger [321]提出的一种称为反射阴影贴图（RSM）的方法。与常规阴影贴图（第7.4节）相似，反射阴影贴图是从灯光的角度渲染的。除了深度以外，它们还存储有关可见表面的其他信息，例如反射率，法线和直接照明（磁通量）。执行最终着色时，将RSM的纹理像素视为点光源，以提供间接反射的单次反射。因为典型的RSM包含成千上万个像素，所以使用重要性驱动的启发式算法只能选择其中的一个子集。 Dachsbacher和Stamminger [322]随后展示了如何通过逆转过程来优化该方法。与其从每个阴影点的RSM中选择相关纹理，不如根据整个RSM创建一定数量的灯光并将其散布在屏幕空间中（第13.9节）。

该方法的主要缺点是它不能为间接照明提供遮挡。虽然这是一个很大的近似值，但结果看起来似乎合理，并且对于许多应用程序都是可接受的。为了获得高质量的结果并在光移动过程中保持时间稳定性，需要创建大量的间接光。如果创建的文件太少，它们将在重新生成RSM时迅速改变其位置，并导致闪烁的伪像。另一方面，从性能的角度来看，过多的间接照明是一项挑战。 Xu [1938]描述了如何在《神秘海域4》游戏中实现该方法。为保持性能约束，他每像素使用少量（16）灯，但在几帧中循环使用不同的灯组并过滤结果。时间上的（图11.32）。已经提出了不同的方法来解决间接闭塞的缺乏。莱恩等。 [962]使用双抛物面阴影贴图作为间接光照，但将它们逐步添加到场景中，因此在任何单帧中仅渲染了少数阴影贴图。 Ritschel等。 [1498]使用场景的简化的基于点的表示来渲染大量不完善的阴影贴图。这样的地图很小，直接使用时会有很多缺陷，但是经过简单的过滤后，就可以提供足够的保真度来为间接照明提供适当的遮挡效果。某些游戏使用了与这些解决方案相关的方法。灰尘514呈现了自顶向下的视图，在需要时最多具有四个独立的图层[1110]。这些生成的纹理用于执行间接照明的收集，非常类似于Tabellion和Lamorlette的方法。在Kite演示中，使用类似的方法从地形提供间接照明，展示了虚幻引擎[60]。

11.5.6 光传播体积

辐射传递理论是对电磁辐射如何通过介质传播进行建模的一般方法。它说明了散射，发射和吸收。即使实时图形努力显示所有这些效果，除了最简单的情况之外，用于这些模拟的方法的数量级太高，无法直接应用于渲染。但是，已证明该领域中使用的某些技术在实时图形中很有用。由卡普兰扬[854]提出的光传播体积（LPV），从辐射传递中的离散纵坐标方法中汲取了灵感。用他的方法，将场景离散为三维单元格的规则网格。每个单元将保持流经它的辐射的方向分布。他对这些数据使用二阶球谐函数。第一步，将光照注入包含直接光照表面的单元。可以访问反射阴影图以找到这些像元，但是也可以使用任何其他方法。注入的照明是从照明表面反射的辐射。这样，它围绕着法线形成了一个背向表面的分布，并从材质的颜色中获取了颜色。接下来，照明被传播。每个单元分析其邻居的辐射场。然后，它修改自己的分布以说明从各个方向到达的辐射。在单个步骤中，辐射仅在单个单元格的距离上传播。需要多次迭代才能进一步分发它（图11.33）。该方法的重要优点是，它为每个单元生成一个完整的辐射场。这意味着即使使用二阶球谐函数时光泽BRDF的反射质量会很低，我们也可以使用任意BRDF进行着色。卡普兰（Kaplanyan）展示了同时具有漫反射和反射表面的示例。

为了允许光在更大距离上传播，并增加卷所覆盖的区域，同时保持合理的内存使用，Kaplanyan和Dachsbacher [855]开发了该方法的级联变体。他们不使用具有相同大小的单元格的单个体积，而是使用其中的一组逐渐增大的单元格，彼此嵌套。照明被注入所有级别并独立传播。在查找期间，他们选择给定位置可用的最详细级别。最初的实现并未考虑间接照明的任何遮挡。经过修订的方法使用了反射阴影贴图的深度信息以及相机位置的深度缓冲区，以将有关遮光剂的信息添加到体积中。该信息不完整，但是在预处理过程中也可以将场景体素化，因此可以使用更精确的表示。该方法存在其他体积方法的问题，其中最大的是出血。不幸的是，提高网格分辨率以修复它会导致其他问题。当使用较小的像元大小时，需要更多的迭代才能在相同的世界空间距离上传播光，这使该方法的成本大大增加。在网格分辨率和性能之间找到平衡并非易事。该方法还存在混叠问题。网格的有限分辨率与辐射的粗略方向表示相结合，会导致信号在相邻单元之间移动时质量下降。在多次迭代之后，空间伪像（例如对角条纹）可能会出现在解决方案中。这些问题中的一些可以通过在传播遍历之后执行空间滤波来消除。

11.5.7 基于体素的方法

由Crassin [304]引入，体素圆锥跟踪全局照明（VXGI）也基于体素化场景表示。几何体本身以稀疏体素八叉树的形式存储，如第13.10节所述。关键概念是这种结构提供了场景的类似于mipmap的表示形式，因此可以快速测试一定空间的遮挡。体素还包含有关从其表示的几何结构反射的光量的信息。它以定向形式存储，因为辐射在六个主要方向上被反射。使用反射阴影贴图，直接照明首先注入八叉树的最低级别。然后将其向上传播到层次结构中。八叉树用于估计入射辐射。理想情况下，我们将跟踪射线以获取来自特定方向的辐射的估计值。但是，这样做需要很多光线，因此，将这些光线的整个束近似用沿其平均方向跟踪的圆锥近似，只返回一个值。精确测试圆锥体与八叉树的相交并不是一件容易的事，因此，通过沿着圆锥体轴对树进行一系列查找，可以近似进行此操作。每次查找都会读取树的级别，该级别的节点大小与给定点处圆锥体的横截面相对应。查找提供了在圆锥体原点方向上反射的已过滤辐射率，以及几何所占查找足迹的百分比。该信息用于以类似于alpha混合的方式衰减来自后续点的照明。跟踪整个圆锥的遮挡。在每个步骤中，它都会减少，以解决当前样本被几何图形占用的百分比。累积辐射率时，首先将其乘以组合的遮挡因子（图11.34）。此策略无法检测到由于多个部分遮挡而导致的完全遮挡，但是结果仍然令人相信。

为了计算漫射照明，将跟踪多个圆锥体。 生成和转换的数量是性能和精度之间的折衷。 跟踪更多的圆锥体可以提供更高质量的结果，但要花费更多的时间。 假定余弦项在整个圆锥上是恒定的，因此该项可以从反射率方程积分中分解出来。 这样做使漫射照明的计算就像计算圆锥轨迹返回的值的加权和一样简单。 如Mittring [1229]所述，该方法是在Unreal Engine的原型版本中实现的。 他给出了开发人员需要进行的一些优化，以使其能够在完整渲染管道中运行。 这些改进包括以较低的分辨率执行迹线并在空间上分布视锥。 进行此过程是为了使每个像素仅跟踪一个圆锥体。 通过对屏幕空间中的结果进行过滤，可以获得漫反射的全部辐射。

使用稀疏八叉树存储照明的主要问题是查找成本高。查找包含给定位置的叶节点对应于一系列内存查找，并与确定要遍历的子树的简单逻辑交错。典型的内存读取可能需要数百个周期。 GPU试图通过并行执行多组着色器线程（扭曲或波前）来隐藏此延迟（第3章）。即使在任何给定时间只有一个组执行ALU操作，当它需要等待内存读取时，另一组也会代替它。可以同时激活的经纱数量由不同的因素决定，但所有因素都与单个组使用的资源量有关（第23.3节）。遍历分层数据结构时，大部分时间都花在等待从内存中取出下一个节点。但是，在此等待期间将执行的其他扭曲也很可能也会执行内存读取。由于与内存访问次数相比，ALU的工作量很少，并且由于飞行中的扭曲总数有限，因此所有组都在等待内存且不执行任何实际工作的情况很常见。具有大量的停滞经纱会导致性能欠佳，并且已经开发出了减轻这些低效率的方法。迈凯轮[1190]用一系列的三维纹理替换八叉树，就像级联的光传播体积[855]（第11.5.6节）。它们具有相同的尺寸，但覆盖范围逐渐扩大。这样，仅通过常规纹理查找即可完成数据的读取-无需进行依赖读取。纹理中存储的数据与稀疏体素八叉树中的数据相同。它们包含六个方向的反照率，占用率和反射光信息。由于级联的位置随相机的移动而变化，因此，对象会不断地进出高分辨率区域。由于内存限制，不可能始终将这些体素化的版本保留在所有位置，因此在需要时可以按需对它们进行体素化。迈凯轮还描述了许多优化措施，使该技术适用于30 FPS游戏《明天的孩子》（图11.35）。

11.5.8 屏幕空间方法

就像屏幕空间的环境光遮挡（第11.3.6节）一样，仅使用存储在屏幕位置的表面值即可模拟某些漫反射全局照明效果[1499]。这些方法没有SSAO流行，主要是因为有限的可用数据量导致的伪像更加明显。诸如渗色之类的效果是强直射光照亮了颜色相当恒定的大区域的结果。这样的表面通常不可能完全适合视图。这种情况使反射光的数量在很大程度上取决于当前取景，并且会随着相机的移动而波动。出于这个原因，屏幕空间方法仅用于在超出原始算法可实现的分辨率的情况下，以小规模增强一些其他解决方案。游戏Quantum Break [1643]中使用了这种类型的系统。辐照量用于模拟大规模全局照明效果，并且屏幕空间解决方案可在有限距离内提供反射光。

11.5.9 其它方法

Bunnell的计算环境光遮挡的方法[210]（第11.3.5节）还允许动态计算全局照明效果。 通过存储有关每个磁盘的反射辐射率的信息来增强场景的基于点的表示（第11.3.5节）。 在收集步骤中，不仅可以收集遮挡，还可以在每个收集位置构造一个完整的入射辐射函数。 就像环境光遮挡一样，必须执行后续步骤以消除来自被遮挡的磁盘的光线。

**11.6 镜面全局照明**

前面各节中介绍的方法主要是为了模拟漫反射全局照明而定制的.现在,我们将研究可用于呈现与视图相关的效果的各种方法.对于有光泽的材料,镜面瓣比用于漫射照明的余弦瓣要紧得多.如果我们要显示一种极高光泽的材料,即具有薄的镜面瓣,则需要一种可以提供此类高频细节的辐射表示.另一方面,这些条件还意味着对反射方程的评估仅需要从有限的立体角入射的照明,这不同于反射整个半球照明的Lambertian BRDF.这与漫反射材料施加的要求完全不同.这些特征解释了为什么需要进行不同的权衡以实时传递这种效果.

存储入射辐射的方法可用于传递依赖于视图的粗略效果.当使用AHD编码或HL2基时,我们可以计算镜面响应,就好像照明来自来自编码方向(或三个方向,在HL2基础的情况下)到达的定向光一样.这种方法的确从间接照明中获得了一些镜面反射高光,但是它们并不十分精确.对于AHD编码,使用此想法尤其成问题,因为AHD编码的方向分量可能会在短距离内急剧变化.该变化导致镜面高光以不自然的方式变形.可以通过在空间上过滤方向来减少伪像[806].如果相邻三角形之间的切线空间快速变化,则在使用HL2基时会出现类似的问题.

也可以通过以更高的精度表示入射光来减少伪影.Neubelt和Pettineo在游戏《教团:1886》 [1268]使用球形高斯波瓣来表示入射辐射.为了渲染镜面效果,他们使用了Xu等人[1940]的方法.他开发了对典型微面BRDF的镜面反射的有效近似(第9.8节).如果照明用一组球形高斯表示,并且菲涅耳项和掩蔽阴影函数在它们的支撑上被假定为常数,则反射方程可以近似为

其中是表示入射辐射的第个球面高斯,是组合菲涅耳和遮蔽阴影函数的因子,是NDF.徐等人 引入各向异性球形高斯(ASG),他们将其用于NDF建模.如公式11.37所示,它们还为计算SG和ASG乘积的积分提供了有效的近似值.

Neubelt和Pettineo使用9到12个高斯波瓣来表示照明,这使它们只能对中等光泽的材质进行建模.他们能够使用这种方法来表示大多数游戏照明,因为游戏发生在19世纪的伦敦,而且高度抛光的材料,玻璃和反射表面很少见.

11.6.1 局部化环境图

到目前为止讨论的方法不足以令人信服地渲染抛光材料。对于这些技术，辐射场太粗糙，无法精确地编码入射辐射的细节，这会使反射看起来很暗淡。如果在同一材质上使用，则产生的结果也与分析光的镜面高光不一致。一种解决方案是使用更多的球形高斯分布或更高阶的SH来获取我们需要的细节。这是可能的，但是我们很快会遇到性能问题：SH和SG都具有全球支持。每个基本函数在整个球体上都不为零，这意味着我们需要所有这些函数来评估给定方向上的照明。这样做比使用锐利反射所需的基函数少，因此代价非常昂贵，因为我们将需要数千个。以漫反射照明通常使用的分辨率存储这么多的数据也是不可能的。用于实时设置全局照明的镜面反射组件的最流行解决方案是局部环境图。他们解决了我们先前的两个问题。传入的辐射度表示为环境图，因此仅需要几个值即可评估辐射度。它们还稀疏地分布在整个场景中，因此以入射辐射的空间精度来换取增加的角分辨率。这种在场景中特定点绘制的环境图通常称为反射探测器。有关示例，请参见图11.36。环境贴图非常适合渲染完美的反射，即镜面反射间接照明。已经开发出了许多使用纹理的方法来传递各种各样的镜面效果（第10.5节）。所有这些都可以与局部环境图一起使用，以对间接照明渲染镜面反射.

将环境图与空间中的特定点相关联的首批出版物之一是Half-Life 2 [1193，1222]。在他们的系统中，艺术家首先在整个场景中放置采样位置。在预处理步骤中，从每个位置绘制一个立方体贴图。然后，对象将最近位置的结果用作镜面光照计算中入射辐射的表示。可能会发生相邻对象使用不同的环境贴图的情况，这会导致视觉上的不匹配，但是美术师可以手动覆盖多维数据集贴图的自动分配。如果对象很小，并且从其中心渲染了环境贴图（在隐藏对象之后使其不出现在纹理中），则结果将非常精确。不幸的是，这种情况很少见。通常，同一反射探针可用于多个对象，有时具有很大的空间范围。镜面曲面的位置离环境贴图中心越远，结果与实际情况的差异就越大。 Brennan [194]和Bjorke [155]提出了一种解决这个问题的方法。他们假设入射光不是来自无限远的周围球体，而是假定其来自具有有限大小的球体，并且半径是用户定义的。查找入射辐射时，该方向不直接用于索引环境图，而是被视为源自评估的表面位置并与该球体相交的射线。接下来，计算一个新方向，即从环境地图的中心到相交位置的一个方向。该向量用作查找方向。参见图11.37。该过程具有将环境图“固定”在空间中的效果。这样做通常称为视差校正。相同的方法可以与其他基本元素一起使用，例如box [958]。用于射线相交的形状通常称为反射代理。使用的代理对象应代表渲染到环境贴图中的几何图形的一般形状和大小。尽管通常不可能，但是如果它们完全匹配（例如，当使用一个盒子表示一个矩形房间时），该方法将提供完美的局部反射。该技术在游戏中获得了极大的普及。它易于实现，运行时速度很快，并且可以在正向和延迟渲染方案中使用。艺术家可以直接控制外观和内存使用。如果某些区域需要更精确的照明，则可以放置更多的反射探针并更好地适合代理。如果用于存储环境图的内存过多，则很容易删除探针。当使用有光泽的材料时，阴影点与具有代理形状的相交之间的距离可用于确定使用哪个级别的预过滤环境图（图11.38）。这样做可以模拟我们远离阴影点时BRDF瓣的增长足迹。

当多个探针覆盖相同区域时，可以建立有关如何组合它们的直观规则。例如，探针可以具有用户设置的优先级参数，该参数使具有较高值的​​探针优先于较低的探针，或者它们可以平滑地相互融合。不幸的是，该方法的简单性导致各种伪像。反射代理很少会完全匹配基础几何。这使得反射在某些区域以不自然的方式延伸。这主要是针对高反射率的抛光材料的问题。此外，渲染到环境地图中的反射对象的BRDF会从地图的位置进行评估。访问环境贴图的表面位置将不具有这些对象的完全相同的视图，因此纹理的存储结果并不完全正确。代理也会导致（有时是严重的）漏光。通常，查找会从环境图的明亮区域返回值，因为简化的射线投射会遗漏应引起遮挡的局部几何形状。有时可以通过使用定向遮挡方法来缓解此问题（第11.4节）。缓解此问题的另一种流行策略是使用预先计算的漫射照明，该照明通常以更高的分辨率存储。首先将环境贴图中的值除以渲染该位置处的平均漫射光照。这样做可以有效地消除环境图的平滑分散影响，仅保留高频分量。进行阴影处理时，在阴影位置[384，999]，反射将与漫射照明相乘。这样做可以部分缓解反射探针的空间精度不足。已经开发出使用反射探针捕获的几何形状的更复杂表示的解决方案。 Szirmay-Kalos等。 [1730]为每个反射探针存储一个深度图，并在查找时对其进行射线追踪。这样可以产生更准确的结果，但要额外付费。 McGuire等。 [1184]提出了一种更有效的将光线跟踪到探测器的深度缓冲区的方法。他们的系统存储多个探针。如果最初选择的探针没有足够的信息来可靠地确定命中位置，则选择一个后备探针，并使用新的深度数据继续跟踪。

使用有光泽的BRDF时，通常会对环境图进行预过滤，并且每个mipmap都将入射辐射率与逐渐增大的内核卷积在一起。 预过滤步骤假定该内核是径向对称的（第10.5节）。 但是，使用视差校正时，BRDF凸角在反射代理形状上的覆盖区会根据阴影点的位置而变化。 这样做会使预过滤略有不正确。 Pesce和Iwanicki分析了该问题的不同方面，并讨论了可能的解决方案[807，1395]。 反射代理不必是封闭的凸形。 也可以使用简单的平面矩形来代替或增加具有高质量细节的框或球体代理[1228，1640]。

11.6.2 环境地图的动态更新

使用局部反射探针需要绘制和过滤每个环境图。这项工作通常是脱机完成的，但是在某些情况下可能需要在运行时进行。如果一天中的时间在变化的开放世界游戏中，或者动态生成世界几何图形时，离线处理所有这些地图可能会花费很长时间，并影响生产率。在极端情况下，当需要许多变体时，甚至不可能将所有变体存储在磁盘上。实际上，一些游戏会在运行时渲染反射探针。需要仔细调整此类系统，以免对性能产生重大影响。除琐碎的情况外，不可能每帧都重新渲染所有可见的探针，因为现代游戏中的典型帧可以使用数十甚至数百个探针。幸运的是，这不是必需的。我们很少要求反射探针始终准确地描绘其周围的所有几何形状。通常，我们确实希望它们对一天中的时间变化做出适当的反应，但是我们可以通过其他方式（例如稍后介绍的屏幕空间方法）（11.6.5节）来近似动态几何的反射。这些假设使我们能够在加载时进行一些探查，而其余的探查则一次一次地进行。即使当我们确实希望在反射探针中渲染动态几何图形时，我们几乎可以肯定可以承受以较低帧速率更新探针的费用。我们可以定义要花费多少帧时间来渲染反射探针，并在每帧中仅更新固定数量的反射探针。根据每个探针到摄像机的距离，自上次更新以来的时间以及类似因素的启发式方法可以确定更新顺序。在时间预算特别小的情况下，我们甚至可以将单个环境图的渲染划分为多个帧。例如，我们可以在每帧中仅渲染立方体贴图的一个面。

脱机执行卷积时，通常使用高质量的过滤。这种过滤涉及多次采样输入纹理，这在高帧速率下是不可能提供的。 Colbert and Kˇriv´anek [279]开发了一种方法，该方法使用重要性抽样，以相对较低的样本数量（约64个数量级）实现可比的过滤质量。为了消除大部分噪声，他们从具有完整mip链的立方体贴图采样，并使用试探法确定每个样本应读取哪个mip级别。他们的方法是快速，运行时环境图预过滤的流行选择[960，1154]。 Manson和Sloan [1120]用基本函数构造了所需的过滤内核。必须在优化过程中获得用于构造特定内核的确切系数，但是对于给定的形状，它仅发生一次。卷积分两个阶段执行。首先，对环境图进行降采样，并使用一个简单的内核同时对其进行过滤。接下来，将来自生成的Mip链的样本合并以构建最终的环境图。为了限制照明过程中使用的带宽以及内存使用情况，压缩生成的纹理是有益的。 Narkowicz [1259]描述了一种将高动态范围反射探针压缩为BC6H格式（第6.2.6节）的有效方法，该方法能够存储半精度浮点值。渲染复杂的场景，甚至一次渲染一个立方体贴图，对于CPU来说可能太昂贵了。一种解决方案是为离线环境映射准备G缓冲区，并仅计算（更少的CPU需求）光照和卷积[384，1154]。如果需要，我们甚至可以在预生成的G缓冲区顶部渲染动态几何。

11.6.3 基于体素的方法

在性能最受限制的情况下，本地化环境地图是一个很好的解决方案。但是，它们的质量常常不能令人满意。在实践中，必须使用变通方法来掩盖由于探针的空间密度不足或代理过于粗略而无法近似实际几何形状而导致的问题。当每帧有更多时间可用时，可以使用更复杂的方法。体素锥体跟踪（在稀疏八叉树[307]和级联版本[1190]（第11.5.7节）中）也可以用于镜面反射分量。该方法针对存储在稀疏体素八叉树中的场景表示执行圆锥跟踪。一条圆锥形轨迹仅提供一个值，代表来自圆锥形对角的平均辐射度。对于漫射照明，我们需要跟踪多个圆锥体，因为仅使用单个圆锥体是不准确的。将圆锥形描迹用于光面材料要有效得多。在镜面照明的情况下，BRDF瓣较窄，仅需考虑来自小立体角的辐射。我们不再需要追踪多个圆锥体；在许多情况下，仅一个就足够了。仅在较粗糙的材料上的镜面效果可能需要跟踪多个圆锥，但由于此类反射模糊，因此对于这些情况，通常只需要使用局部反射探针就可以了，而根本不跟踪圆锥。在光谱的另一端是高度抛光的材料。它们的镜面反射几乎像镜子一样。这使圆锥变薄，类似于单射线。通过这样精确的跟踪，底层场景表示的体素性质在反射中可能会很明显。它会显示由体素化过程产生的立方体，而不是多边形几何体。在实践中，这种伪影很少出现问题，因为几乎从未直接看到反射。它的贡献被纹理修饰，通常会掩盖任何缺陷。当需要完美的镜面反射时，可以使用其他方法以较低的运行时成本提供它们。

11.6.4平面反射

另一种选择是重用场景的常规表示并重新渲染它以创建反射图像。如果反射表面的数量有限且它们是平面的，我们可以使用常规的GPU渲染管道来创建从此类表面反射的场景图像。这些图像不仅可以提供准确的镜面反射，还可以通过对每个图像进行一些额外的处理来呈现合理的光泽效果。理想的反射器遵循反射定律，该定律指出入射角等于反射角。即，入射射线与法线之间的角度等于反射射线与法线之间的角度。参见图11.39。该图还显示了反射对象的“图像”。由于反射定律，物体的反射图像只是物体本身，通过平面进行物理反射。就是说，我们可以通过反射器跟踪入射光线并到达同一点，但要在反射对象上，而不是跟随反射光线。

这引出了一个原理，即可以通过创建对象的副本，将其转换为反射位置并从那里进行渲染来呈现反射。为了获得正确的照明，就位置和方向而言，光源也必须在平面上反射[1314]。一种等效的方法是改为通过镜子将观众的位置和方向反射到反射镜的另一侧。可以通过简单修改投影矩阵来实现这种反射。位于反射器平面远端（即后面）的物体不应被反射。这个问题可以通过使用反射镜的平面方程来解决。最简单的方法是在像素着色器中定义一个裁剪平面。放置剪切平面，使其与反射器[654]的平面重合。在渲染反射场景时使用此裁剪平面将裁剪掉与视点在同一侧的所有反射几何体，即最初位于镜子后面的所有对象。

11.6.5 屏幕空间方法

就像环境光遮挡和整体散射一样，某些镜面反射效果只能在屏幕空间中计算。 由于镜面波瓣的清晰度，这样做比扩散情况下的精度稍高。 仅从围绕反射视图矢量的有限立体角而不是整个半球需要有关辐射率的信息，因此屏幕数据更可能包含该信息。 Sousa等人首先提出了这种方法。 [1678]，同时被其他开发人员发现。 整个方法系列称为屏幕空间反射（SSR）。

给定要着色的点的位置，视图矢量和法线，我们可以沿着通过法线反射的视图矢量跟踪光线，测试与深度缓冲区的交点。通过沿射线迭代移动，将位置投影到屏幕空间并从该位置获取z缓冲区深度来完成此测试。如果光线上的点距离相机比深度缓冲区表示的几何形状更远，则意味着光线在几何形状内并且检测到碰撞。然后可以从颜色缓冲器中读取相应的值，以获得从跟踪方向入射的辐射值。该方法假定射线照射的表面是朗伯面，但是这种情况是许多方法所共有的近似值，在实践中几乎没有约束。可以在世界空间中以统一的步长追踪射线。该方法相当粗糙，因此当检测到命中时，可以执行优化遍历。在有限的距离内，可以使用二进制搜索来精确定位相交位置。 McGuire和Mara [1179]注意到，由于透视投影，以统一的世界空间间隔步进会在屏幕空间中沿着射线创建采样点的不均匀分布。靠近相机的光线部分采样不足，因此可能会丢失某些命中事件。较远的像素会被过度采样，因此多次读取同一深度缓冲区像素，从而产生不必要的内存流量和冗余计算。他们建议改为使用数字差分分析仪（DDA）在屏幕空间中执行射线行进，该方法可用于对线进行栅格化。首先，将要追踪的光线的起点和终点都投影到屏幕空间。沿着这条线的每个像素都会依次检查，以确保一致的精度。这种方法的一个结果是，相交测试不需要为每个像素完全重建视图空间深度。视图空间深度的倒数，即在典型透视投影情况下存储在z缓冲区中的值，在屏幕空间中呈线性变化。这意味着我们可以在实际走线之前针对屏幕空间的x和y坐标计算其导数，然后使用简单的线性插值来获取沿屏幕空间段的任何位置的值。计算值可以直接与深度缓冲区的数据进行比较。屏幕空间反射的基本形式仅跟踪一条光线，并且只能提供镜面反射。但是，完全镜面反射的表面很少见。在基于物理的现代渲染管线中，更经常需要进行光泽反射，并且SSR也可以用于渲染这些反射。

在简单的临时方法[1589，1812]中，反射仍沿反射方向用单一射线跟踪。 结果存储在屏幕外缓冲区中，并在后续步骤中进行处理。 应用了一系列过滤内核，这些内核通常与对缓冲区的下采样结合以创建一组反射缓冲区，每个反射缓冲区的模糊程度不同。 在计算照明时，BRDF瓣的宽度决定了对哪个反射缓冲区进行采样。 即使经常选择滤波器的形状来匹配BRDF瓣的形状，这样做仍然只是粗略的近似值，因为执行屏幕空间滤波时不会考虑不连续性，表面方向以及其他对精度至关重要的因素。 结果。 在末尾添加自定义试探法，以使光泽的屏幕空间反射在视觉上与其他来源的镜面反射匹配。 即使是近似值，结果还是令人信服的。

Stachowiak [1684]以更原则的方式解决了这个问题。计算屏幕空间反射是光线跟踪的一种形式，就像光线跟踪一样，它可以用于执行适当的蒙特卡洛积分。他不仅使用反射的视线方向，还使用BRDF的重要性采样并随机拍摄光线。由于性能的限制，以一半的分辨率进行跟踪，并且每个像素（一到四个之间）跟踪少量的光线。光线太少，无法产生无噪声的图像，因此相交结果在相邻像素之间共享。假设对于某些范围内的像素，可以将局部可见性视为相同。如果从点p0沿d0方向射出的光线与点i0处的场景相交，则可以假定，如果从点p1沿d1方向射出的光线也穿过i0，那么它也会命中i0的几何体而且之前没有任何交叉点。这样，我们就可以通过适当地修改其对邻居积分的贡献来使用该射线，而无需实际跟踪它。从形式上讲，从相邻像素发射的光线的方向在相对于当前像素的BRDF的概率分布函数进行计算时将具有不同的概率。为了进一步增加光线的有效数量，对结果进行时间过滤。通过离线执行积分的场景无关部分并将其存储在由BRDF参数索引的查找表中，还可以减少最终积分的方差。如果在屏幕空间中可以获得反射射线的所有必需信息，那么这些策略使我们能够获得精确的，无噪声的结果，接近于路径跟踪的真实图像（图11.40）。在屏幕空间中追踪光线通常很昂贵。它包括重复采样深度缓冲区（可能多次），并对查找结果执行一些操作。由于读取相当不连贯，因此缓存利用率可能很低，导致在着色器执行期间由于等待内存事务完成而导致长时间的停顿。在使实现尽可能快的过程中，需要格外小心。屏幕空间反射通常以降低的分辨率计算[1684，1812]，并且使用时间滤波来弥补降低的质量。 Uludag [1798]描述了一种使用分层深度缓冲区（第19.7.2节）来加速跟踪的优化。首先，创建一个层次结构。深度缓冲区在每个方向的每个方向上以2的倍数逐步降采样。较高级别的像素在较低级别的四个对应像素之间存储最小深度值。接下来，通过层次结构执行跟踪。如果在给定的步骤中，射线没有撞击存储在其通过的单元中的几何体，则将其前进到单元的边界，并在下一步中使用较低分辨率的缓冲区。如果射线在当前单元格中遇到击中，则将其前进到击中位置，并在下一步中使用更高分辨率的缓冲区。当记录了最高分辨率缓冲区的命中时，跟踪终止（图11.41）。

该方案特别适合长迹，因为它可以确保不会遗漏任何特征，同时可以使光线以较大的增量前进。 它也可以很好地访问缓存，因为深度缓冲区不是在随机的，远处读取的，而是在本地附近读取的。 Grenier [599]提出了许多实施该方法的实用技巧。

其他人则避免完全追踪光线。 Drobot [384]重用了与反射代理相交的位置，并从那里查找屏幕空间的辐射度。 Cichocki [266]假设使用平面反射器，而不是追踪光线，而是颠倒了这一过程，并进行了全屏扫描，其中每个像素将其值写入应反射的位置。就像其他屏幕空间方法一样，由于可用数据有限，反射也会受到伪影的影响。通常，反射光线会在记录到击中之前离开屏幕区域，或者击中没有可用照明信息的几何图形的背面。此类情况需要妥善处理，因为即使对于相邻像素，迹线的有效性也常常不同。空间过滤器可用于部分填充跟踪缓冲区[1812、1913]中的间隙。 SSR的另一个问题是在深度缓冲区中缺少有关对象厚度的信息。因为仅存储一个值，所以当光线在深度数据描述的曲面后面时，无法判断光线是否撞击任何东西。 Cupisz [315]讨论了各种低成本的方法来减轻由于不知道深度缓冲区中对象的厚度而引起的假象。马拉等人。 [1123]描述了深层G缓冲区，该缓冲区存储多层数据，因此具有有关表面和环境的更多信息。屏幕空间反射是提供一组特定效果的好工具，例如附近对象在大多数平面上的局部反射。它们极大地提高了实时镜面照明的质量，但没有提供完整的解决方案。本章中描述的不同方法通常相互叠加，以提供一个完整而强大的系统。屏幕空间反射充当第一层。如果无法提供准确的结果，则使用局部反射探针作为后备。如果没有探针在给定区域中应用，则使用全局默认探针[1812]。这种类型的设置提供了一种获得合理的间接镜面反射贡献的一致且强大的方法，这对于获得可信的外观尤其重要。

11.7 统一方法

到目前为止介绍的方法可以组合成能够渲染精美图像的相干系统。但是，它们缺乏路径跟踪的优雅和概念上的简单性。渲染方程式的每个方面都以不同的方式处理，每个方面都有各种折衷。即使最终图像看起来很逼真，但在许多情况下这些方法都失败了，幻觉也破灭了。由于这些原因，实时路径跟踪已成为大量研究工作的重点。用路径跟踪渲染质量可接受的图像所需的计算量远远超过了快速CPU的能力，因此改用GPU。它们的极高速度和计算单元的灵活性使其成为此任务的理想选择。实时路径跟踪的应用包括架构演练和电影渲染的预可视化。对于这些用例，可以接受较低且变化的帧速率。当相机静止不动时，可以使用渐进式细化（第13.2节）之类的技术来改善图像质量。高端系统可以期望使用多个GPU。相反，游戏需要以最终质量渲染帧，并且需要在时间预算内一致地进行渲染。 GPU可能还需要执行渲染本身以外的任务。例如，诸如粒子模拟之类的系统通常被卸载到GPU，以释放一些CPU处理能力。所有这些元素的结合使得路径追踪对于当今的渲染游戏来说是不切实际的。图形社区中有一句谚语：“射线追踪是未来的技术，而且永远都是！”这个讽刺意味着这个问题是如此的复杂，以至于即使硬件速度和算法都有了所有的进步，仍然总是会有更有效的方式来处理渲染管线的特定部分。付出额外的成本并仅使用射线投射（包括用于主要可见性）可能很难证明是合理的。当前有相当大的道理，因为GPU从未被设计为执行有效的光线跟踪。他们的主要目标一直是对三角形进行栅格化，并且他们在此任务上变得非常出色。虽然可以将光线跟踪映射到GPU，但是当前的解决方案没有来自固定功能硬件的任何直接支持。始终很难用在GPU的计算单元上运行的有效软件解决方案来始终击败硬件栅格化。

较合理，较不纯粹的方法是对在光栅化框架内难以处理的效果使用路径跟踪方法。栅格化从相机可见的三角形，但不依赖于近似的反射代理或不完整的屏幕空间信息，而是跟踪路径以计算反射。与其尝试模拟具有临时模糊的区域光阴影，不如将射线跟踪到源并计算正确的遮挡。发挥GPU的优势，针对无法在硬件中有效处理的元素使用更通用的解决方案。这样的系统仍然有些拼凑，缺少路径跟踪的简单性，但是实时渲染一直都是折衷方案。如果必须在几毫秒内放弃一些优雅，这是正确的选择-帧速率是不可协商的。尽管我们也许永远无法将实时渲染称为“已解决的问题”，但更多地使用路径跟踪将有助于使理论和实践紧密结合。随着GPU每天变得越来越快，这种混合解决方案甚至应在不久的将来适用于最苛刻的应用程序。基于这些原理构建的系统的初始示例已经开始出现[1548]。

光线跟踪系统依赖于加速方案，例如使用边界体积层次结构（BVH）来加速可见性测试。有关此主题的更多信息，请参见第19.1.1节。 BVH的幼稚实现无法很好地映射到GPU。如第3章所述，GPU本身执行线程组，称为扭曲或波阵面。扭曲以锁定步骤进行处理，每个线程执行相同的操作。如果某些线程不执行代码的特定部分，则会暂时禁用它们。因此，GPU代码的编写应尽量减少同一波阵面内线程之间的分歧流控制。假设每个线程处理一条光线。该方案通常导致线程之间的巨大差异。不同的射线将执行遍历代码的分支，并沿途与不同的边界体积相交。有些光线将比其他光线更早地完成树遍历。这种行为使我们脱离了理想状态，因为理想状态中，扭曲中的所有线程都在使用GPU的计算功能。为了消除这些效率低下的问题，已经开发出了遍历方法，该方法可以最小化发散和重新使用早完成的线程[15，16，1947]。每个像素可能需要追踪数百或数千条光线，以生成高质量的图像。即使有了最佳的BVH，高效的树遍历算法和快速的GPU，对于今天（除了最简单的场景）而言，都无法实时做到这一点。在可用的性能限制内我们可以生成的图像非常嘈杂，不适合显示。但是，可以使用降噪算法对其进行处理，以生成大部分无噪声的图像。请参见图11.42，以及第1044页的图24.2。近来，该领域取得了令人瞩目的进步，并且已经开发了算法，该算法可以通过跟踪生成的输入创建视觉上接近于高质量，路径跟踪参考的图像，即使是单个跟踪每个像素的路径[95、200、247、1124、1563]。 2014年，PowerVR宣布了他们的Wizard GPU [1158]。除了典型的功能之外，它还包含在硬件中构造和遍历加速结构的单元（第23.11节）。该系统证明了既有兴趣又有能力定制固定功能单元以加速射线投射。看到未来将是多么令人兴奋！