第六章

避免自相交快速有效的方法

概要

我们提供一种在光线追踪中避免自相交的解决方法，它比现有的一般方法更有效，仅需花费最小的开销，且无需调试参数。

6.1介绍

光线路径追踪的模拟从相机或光源开始，将光线与场景中的几何图形相交以建立路径。当物体与光线相交时，新的光线会在这些物体表面产生，继续它们的路径。理论上，这些新的光线不会再与同一个表面相交了，因为距离等于零的相交是被算法所排除的。但是实际上，在真实算法中所运用的有限精度浮点数经常导致误报相交的情况，这被称为自相交，它会带来图像上的失真，例如阴影失真（shadow acne），就是物体表面有时错误地给自己上阴影的现象。

避开上述问题最普遍的那些方法不能足够有效地适用于多种常见的情况，有时甚至还需要使用者基于每一个场景手动调试参数。另一类方法是对产生那些不精确数字的源头做完整的数学分析，这样可以更有效地避免这个问题。但是，它带来了极大的性能方面的开销，并且要求光线与表面相交算法的底层实现方式对其可见，这在一些软件API上，尤其是在硬件加速技术例如NVIDIA RTX上，是不可能的。

在这一章里我们提出一个相对有效的方法，它不需要任何参数调试，且只带来最小的开销，这使得它用于实时的应用场景和线下（非实时）的渲染都很合适。

6.2 方法

我们将计算一条新光线起始点的有效方法分为两步。第一，我们需要从光线追踪的结果中计算出一个离物体表面尽可能近的相交点，同时考虑浮点数运算带来的影响。第二，当下一条光线产生并开始它的路径时，我们必须采取措施避免它和同一个表面相交。6.2.2章节描述了现有方法的常见隐患，同时展示了我们的解决方案。

6.2.1 计算物体表面的相交点

直接沿原光线路径计算下一条光线的起始点，这样的方法经常受到有限精度数字的限制。即使计算相交点的那些方法在数学上都是相同的，但在实际中，选择最合适的方法是很重要的，因为它直接影响最后数字误差的大小。另外，每一种方法都有它自身的局限性。

通常，计算相交点坐标的方法是在光线方程式中代入命中距离（原光线起点与相交点间的距离）。详见图6-1。我们强烈反对这一步，因为计算出的新起点可能离相交物体表面所在的平面很远。对于本身离原光线起点很远的相交点尤其如此：由于浮点数是指数级的，可用浮点数表示的数字与真实的相交距离的差也会呈指数级增长。

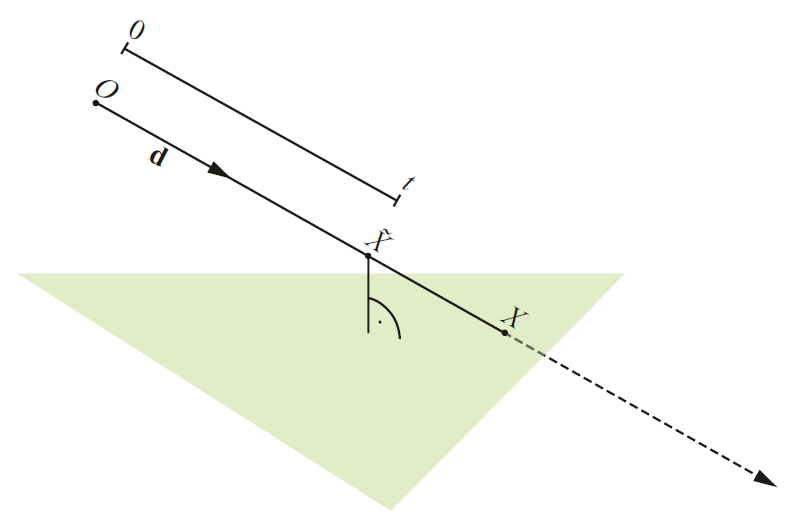


图6-1. 在光线方程式中代入相交距离t来计算光线与表面的相交点X。在这种情况下，由于t精确度不够而带来的误差大部分都会将计算出的相交点沿光线方向**d**移动——大多情况下，会远离三角形的平面。

（译者注：原文如此，但图中似乎不是沿**d**方向移动，不知我是否理解错误）

如果将上述方法换为基于表面参数化（surface parameterization）来计算上一条光线的相交点（如，使用在光线与图元（primitive）相交时计算出的质心坐标（barycentric coordinates）），下一条光线的起点可以处在离表面尽可能近的地方。详见图6-2。即使精度有限的计算依然会导致一些误差，但在使用表面参数化的时候这个误差带来的问题会相对小一些：当使用命中距离计算相交点位置时，有限精度带来的误差大多都会将计算出的相交点沿原光线的方向移动，那经常会是远离物体表面的方向（以及对避免自相交不利的方向，因为一些点会在表面的前面，而一些会在它的后面）。然而，当使用表面参数化时，运算误差大多会将下一条光线的起点沿平行于表面的方向移动——这意味着下一条光线可能会在离上一条光线稍有偏差的地方开始，但它永远是尽可能地靠近表面。使用表面参数化同时确保了新的光线起点与表面属性的一致性，例如插值明暗处理法向量（interpolated shading normals）和纹理坐标（texture coordinates），它们通常取决于表面参数化。

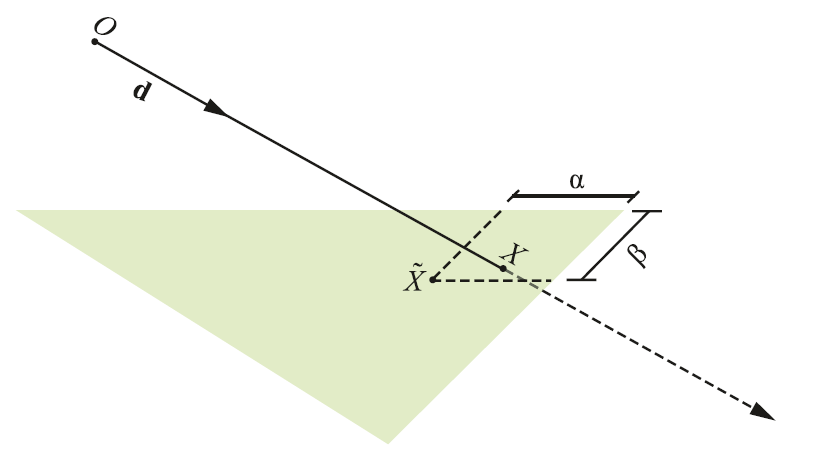


图6-2。用质心坐标（α，β）计算相交点X。在这种情况下，（α，β）的有限精度意味着计算出的相交点 可能不再在原光线上——但是它会离表面非常近。

6.2.2 避免自相交

将新的光线的起点“准确地”置于表面上通常依然会导致自相交[4]，因为计算出的起点到表面的距离不一定等于零。因此，仅仅将与表面距离为零的相交点排除掉还不够，还应该采取更有效的方法避免自相交。以下的分章节大致展示了一些常见的变通方案，以及每个方案的无法解决的情况。我们建议的方案在6.2.2.4章节中有介绍。

6.2.2.1 图元识别排除法

自相交经常可以通过用图元识别式排除同一个相交图元来避免。虽然这个方法不需要变量，与场景比例大小无关，并且不会略过邻近的几何图案，但它有两个明显的问题。第一，在共同边缘上或者共面几何物体上，以及与表面呈小角度的光线上的相交点仍然会引起自相交（见图6-3与6-4）。即使有附近物体的数据，也必须分辨那些产生凹凸形状的邻近表面。第二，它无法处理重复或者相互覆盖的表面。但是，一些渲染者仍然使用这个识别测试作为自相交解决方案的一部分[2]。

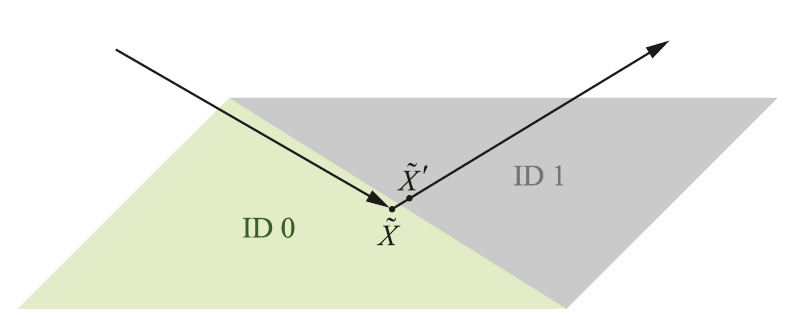


图6-3。如果上一个相交点在一个被共享的边缘上，或者与这个边缘非常接近，仅仅排除与 所在的图元ID 相符的图元表面相交点就无法起到作用。在这个例子中，处在ID为0的图元上。由于精度有限，下一个相交点会在ID为1的图元上被检测到，并且被认为是有效的，因为两者ID不同。

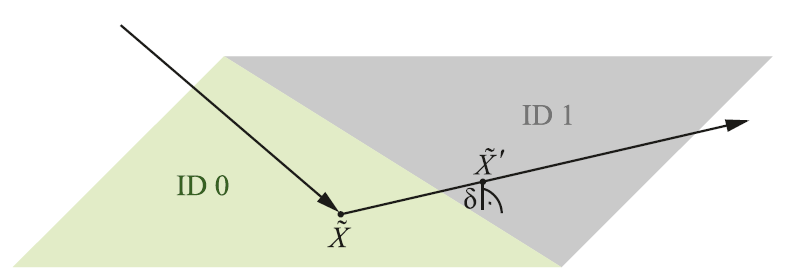


图6-4. 图元ID排除法在相交点处于平面或稍凸的表面上任何地方，且下一条光线为小角度光线时也无法起作用。同上，无效的相交点与其他图元表面间的距离δ会无限接近零，两个图元ID不同，因此这个无效的相交点会被认为是有效的。

另外，请注意图元识别排除法仅适用于平面物体的表面，因为非平面的表面会产生有效的自相交。

6.2.2.2 限制光线长度

除了排除距离为零的相交点，我们还可以为这个距离设置一个较小的最小值ε：tmin = ε > 0. 虽然这在性能上不会带来任何额外开销，但这个方法非常不稳定，因为ε值是取决于场景的，而且不适用于小角度相交，会导致自相交（图6-5）或略过相邻的表面（图6-6）。

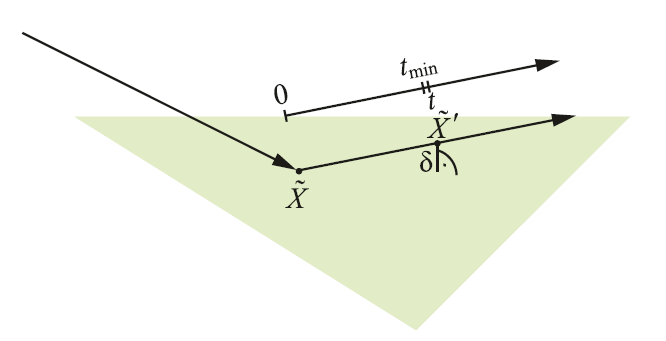


图6-5. 将tmin 设置为一个较小值 ε > 0并不能非常有效地避免自相交，尤其对与表面成小角度的光线。在这个例子中光线上的距离t大于tmin，但是由于精度有限，下一个（无效的）相交点与表面间的距离δ为零。

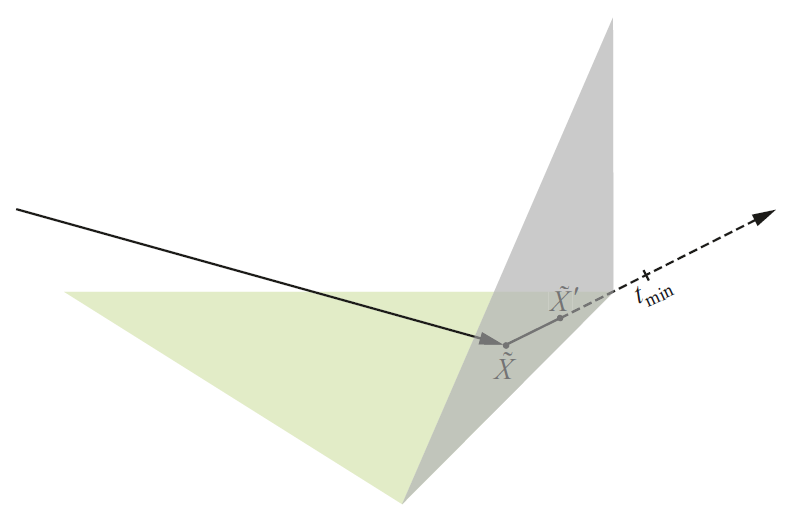


图6-6. 因为设置了tmin = ε > 0而略过一个有效的相交点，这种情况在转角处光线路径进出相邻物体时尤其常见。

6.2.2.3 沿平面法向量或上一条光线方向设置偏移量

沿平面法向量方向设置偏移量与设置最小值tmin = ε > 0的方法相似，并且有同样的缺陷，因为这个向量并不一定与表面垂直（由于插值，或凹凸和法向量贴图（bump or normal maps）计算时带来的变化）。

沿上一条光线方向平移新光线的起点同样会遇到上述问题。

6.2.2.4 沿几何法向量的适应性偏移

我们在前几节中看到，只有与物体表面垂直的几何法向量可以表示最小的偏移量，这个偏移量取决于相交点的距离，以此在不引入任何之前提过的缺陷的同时避免自相交。下一步我们将会着重介绍如何计算这个新光线起点的偏移量。

使用固定长度的ε作为偏移量并不适用于所有比例的场景，因此并非无需变量，也不会对离原光线起点距离不同的相交点都起作用。我们用质心坐标分析计算相交点时浮点数带来的误差，结果显示，相交点与物体表面所在的平面之间的距离和与原点（0,0,0）之间的距离成正比。同时，表面的大小也会影响误差，对于离原点（0,0,0）非常近的三角形，它甚至会成为决定性的因素。只使用沿光线方向的标准化向量可以排除光线长度带来的数学误差的影响。图6-7关于随机三角形的实验结果展示了这一特性：我们统计了计算出的相交点与一千万边缘长度在2-16至222之间的三角形的平均与最大距离。由于计算出的相交点可以处在真实平面的任意一边，一个有效的偏移量应当至少和最大距离一样大。

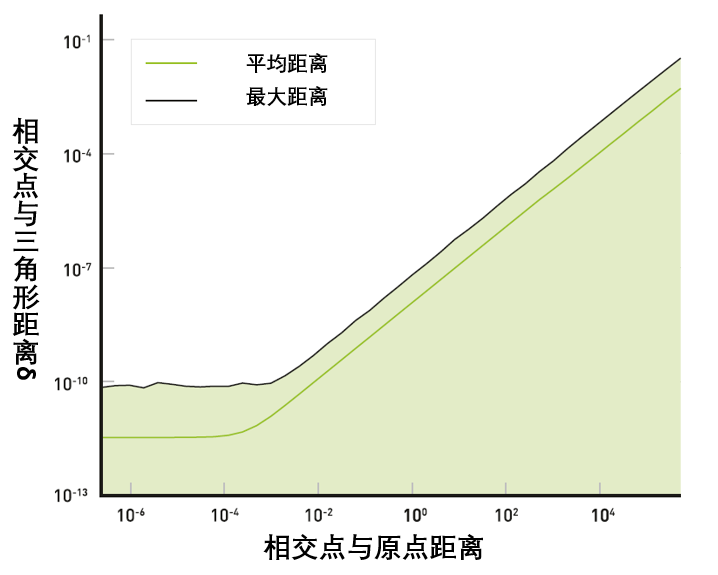
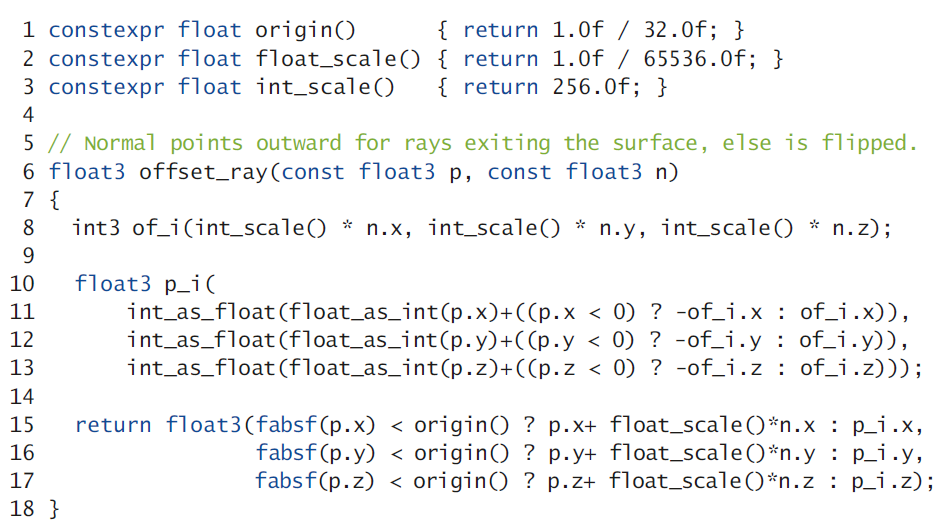


图6-7. 实验使用质心坐标模拟一千万个与原点距离不同的随机三角形，分析在三角形上的点到平面的平均与最大距离，由此得到示例6-1列出的常数的尺度。

为了隐式地解决相交点与原点间距离差异带来的问题，我们在整数代表的浮点数上用整数运算来计算光线起点沿几何法向量的偏移量。这样，这个偏移量就不会随尺度而改变，由此避免了不同距离上的自相交。

我们必须一一解决相交表面/物体与原点/零点太近的问题。光线方向上的浮点指数级与相交点上的指数级会有很大不同；因此，用固定整数ε作为偏移量来解决光线与平面相交时计算产生的数字误差不是一个合适的方法。因此，一个非常小的浮点常数ε被用于这种特殊情况，以避免引入额外的错误。示例6-1为这种方法的源代码。这些常数是基于图6-7选出的，它们包含了一个小的余量，用于处理在这个实验中没有覆盖到的更加极端的情况。



示例 6-1. 6.2.2.4章节中所描述方法的实际应用。

即使用我们的方法，在一些情况下沿几何法向量移动相交点依然会略过一个表面，例如图6-8中展示的裂缝。相似的失败样例可以被设计出来，并且在实际应用中确实可能会产生。然而，与之前所讨论的方法可能的失败样例相比，它们出现的可能性显著降低。



图6-8. 非常微小的几何细节，例如一个又深又窄的裂缝，没有任何一种已提出的解决方法适用于它。在这个例子中原相交点 在实际表面稍下方的位置。左图：限制光线长短可以避免一些自相交（如上方光线），但是可能对于另一些并不起作用（如下方光线）。右图：沿表面法向量的平移可能会将下一条光线的原点移动至邻近的物体中。

6.3 结语

我们提议的有效计算下一条光线路径原点的方法分为两个步骤。首先，使用表面参数化设置一个离相交物体表面所在的平面尽可能近的起始位置。然后，将相交点沿表面法向量远离表面的方向移动一个与场景比例大小无关的偏移量。我们全面的评估展示了这个方法在实践中足够有效，并且在任何渲染器中都可以简单地加入。它在十多年里一直是Iray渲染系统的一部分 [1] ，用于避免与三角形的自相交。

一些依然没有被解决的失败样例是罕见的特殊情况，但是我们应当注意到在物体变形中的大量缩放操作也会导致更大的偏移量（详细分析请见*Physically Based Rendering*（第三版）[3]）。这种情况会导致一个常见的质量问题，因为所有直接与间接的光源，都会被明显 “偏移”，这在邻近物体的光线反射中会变得尤其明显，甚至会产生失真。为了解决这个问题，我们建议将原点（0,0,0）周围的网格都以世界单位存储起来（译者注：原文storing all meshes in world units centered around the origin，不确定翻译是否正确）。另外，使用者应将照相机变形（camera transformation）中的转化与缩放提出来，将它们包括在物体实例的矩阵中。这些步骤使得我们的方法可以在所展示的实际应用场景中运行，并且避免了大偏移量带来的渲染失真。

由于用图元识别法在找到的相交点中排除平面图元并不会导致漏报，它可以作为一个快速而简单的附加测试，通常用于避免开始计算一次不必要的表面相交。

参考文献

[1] Keller, A., Wächter, C., Raab, M., Seibert, D., van Antwerpen, D., Korndörfer, J., and Kettner, L.

The Iray Light Transport Simulation and Rendering System. arXiv, https://arxiv.org/abs/1705.01263, 2017.

[2] Pharr, M. Special Issue On Production Rendering and Regular Papers. ACM Transactions on Graphics 37, 3 (2018).

[3] Pharr, M., Jakob, W., and Humphreys, G. Physically Based Rendering: From Theory to Implementation, third ed. Morgan Kaufmann, 2016.

[4] Woo, A., Pearce, A., and Ouellette, M. It’s Really Not a Rendering Bug, You See... IEEE Computer Graphics & Applications 16, 5 (Sept. 1996), 21–25.